International Journal for Housing Science and Its Applications
Publish August 10, 2025. Volume 46, Issue 4 Pages 3099-3107

https://doi.org/10.70517/ijhsa464256

Research on the optimization model of smart contract execution efficiency based on integrated learning in the context of digital economy

Pan Li[1,2,*], Xu Song[3] and Hui Yuan[3]
[1] The Business School, Anyang Normal University, Anyang, Henan, 455000, China
[2] School of Software Engineering, Anyang Normal University, Anyang, Henan, 455000, China
[3] Anyang Water Conservancy Project Operation Support Center, Anyang, Henan, 455000, China
Corresponding authors: (e-mail: wodelunwen2024@126.com).

Abstract In the era of digital economy, smart contract as the core technology of blockchain application is facing the bottleneck of execution efficiency. This study proposes an optimization model based on integrated learning for the smart contract execution efficiency optimization problem. The study compares the performance of single algorithms such as logistic regression, decision tree, SVM and integrated learning methods such as GBDT and Stacking in the task of smart contract execution efficiency optimization. The experiment constructs a weighted Stacking integrated learning model, which makes full use of the advantages of each classifier by assigning different weights to the primary learners. The experimental results show that the weighted Stacking model outperforms the single algorithm model in all evaluation indexes, with an accuracy of 83.15%, an F1 value of 0.8496, which is 3.79% higher than that of the best-performing single model CatBoost, and an AUC value of 0.9178, which is 0.97% higher than that of CatBoost. Confusion matrix analysis shows that the model successfully predicts 5445 executive users and 400 non-executive users with a low misclassification rate. The study proves that the smart contract execution efficiency optimization strategy based on integrated learning can effectively improve contract execution performance, reduce resource consumption, significantly improve user experience, and provide efficient and stable technical support for blockchain applications.

Index Terms Smart contract, Execution efficiency, Integrated learning, Stacking model, Weighting algorithm, Machine learning

I. Introduction

With the development of digital technologies, especially the Internet, big data, artificial intelligence, and blockchain, the digital economy has become a key driver of global economic growth [1], [2]. The application of these technologies not only accelerates the process of economic globalization, but also promotes the formation of a new international economic cooperation model [3]. International economic cooperation in the era of digital economy shows new features and trends, such as the rapid growth of digital trade, the global expansion of digital platform economy, and the application of blockchain and other technologies in international trade and finance [4]-[6]. These changes have put forward new requirements for the development of global economic cooperation frameworks.

Smart contract technology plays an important role in promoting the efficiency of international trade and is revolutionizing traditional international trade processes and mechanisms by providing decentralized, transparent, and tamper-proof transaction records [7], [8]. Smart contracts are computer programs that automatically execute, control, or record the terms of a contract, and are decentralized, trusted and shared program codes deployed on a blockchain [9]. Smart contracts enable contract terms to be automatically executed when preset conditions are met, reducing intermediary involvement, lowering transaction costs, and can shorten the transaction cycle, and this automatic execution feature ensures the transparency and efficiency of the transaction process, and reduces delays and disputes in the contract execution process [10]-[13]. However, the low transaction execution efficiency of smart contracts also leads to a large number of transaction items being blocked on the blockchain, which also means that these transactions are not confirmed in a short period of time, affecting the real-time nature of transactions [14]-[16]. Therefore, how to take advantage of the current multi-core processors and clusters, so that smart contracts can be executed in parallel, which can increase the concurrency of the system to a certain extent, improve the throughput of the system, and shorten the confirmation time of transactions [17], [18].

The development of blockchain technology promotes the wide application of smart contracts, which as a kind of automatically executed computer program, can realize the automated execution of contract terms without the participation of centralized institutions. However, smart contracts face the problem of execution inefficiency in actual operation, which seriously limits the large-scale application of blockchain technology. The execution efficiency of smart contracts directly affects the transaction processing speed, system throughput, and user experience, so how to optimize the execution efficiency of smart contracts has become a focus of attention for both academia and industry. Existing research mainly starts from smart contract code optimization, consensus mechanism improvement, and blockchain architecture reconfiguration, etc., but most of the methods suffer from the defects of poor adaptability and limited generalization ability. Aiming at the characteristics of the complex and changing operating environment of smart contracts, a single algorithmic model is often difficult to meet the execution efficiency requirements under diverse scenarios. Machine learning methods, especially integration learning technology, have shown great potential in the field of complex system optimization by virtue of their powerful learning ability and generalization ability. By combining the decision results of multiple base learners, integrated learning can make up for the deficiencies of a single model and improve the overall prediction performance. In the smart contract execution efficiency optimization problem, integrated learning is expected to achieve a more accurate and efficient optimization scheme by combining the advantages of multiple algorithms. At present, in the field of smart contracts, the application research of integrated learning methods is still in its infancy, the relevant theory and practice are obviously insufficient, and the applicability and performance differences of different integrated learning methods in the task of smart contract execution efficiency optimization have not yet been systematically studied. Based on the background of digital economy, this study focuses on the problem of smart contract execution efficiency optimization and explores the application prospect of integrated learning methods in this field. A single algorithm model including logistic regression, decision tree, and support vector machine (SVM) is first constructed to analyze its performance characteristics and limitations in the smart contract execution efficiency optimization task. On this basis, the study focuses on the Stacking integrated learning method, which gives full play to the advantages of each base classifier by introducing a weighting mechanism and assigning different weights according to the performance differences of the primary learners. The study employs several evaluation metrics, including accuracy, precision, recall, F1 value and AUC value, to make a comprehensive comparison of different models and visualize the prediction performance of each model through a confusion matrix. Through experimental verification, this study aims to construct an efficient and stable smart contract execution efficiency optimization model to provide theoretical support and technical guidance for improving the performance of blockchain systems, and at the same time to expand the research path for the application of integrated learning in the blockchain field.

II. Smart contract execution efficiency optimization model

II. A. Single Algorithm

II. A. 1) Logistic regression

Logistic regression [19] is a very commonly used machine learning algorithm for classification, which makes it possible to predict the chance of an event occurring by fitting all the data to a logit function that is small. At the same time, logistic regression and linear regression also have a very hot correlation, that is, about the appearance of the sample points in the multi-dimensional space, we need to use a linear combination of features to fit the distribution and trajectory of the points in the space.

The core idea of logistic regression is to assume that the output of linear regression is continuous, but the range of values cannot be restricted. So we map this result value to a probability value in the interval (0, 1), i.e. the sigmoid function [20], as follows:

$$g(x)=\frac{1}{1+e^{-z}} \tag{1}$$

The main advantages of the logistic regression model are that the output is probabilistically significant and in (0, 1). Training is fast and takes up very low memory. Interpretability is very strong.

The disadvantages of the model are: it is not possible to present the relationship between the individual features clearly. Poor performance when the feature space is large. It is also prone to underfitting.

II. A. 2) Tree model

Decision trees [21] are commonly used to solve models for classification as well as regression, usually using a set of rules to categorize rows of numbers into a supervised machine learning tree structure. The principle of the decision tree algorithm is to train classification judgments on features in a set of data with different features and classification labels. A tree-like classification model is formed at the end of training. Usually consists of a root node and multiple leaf nodes, the tree model structure is shown in Figure 1.

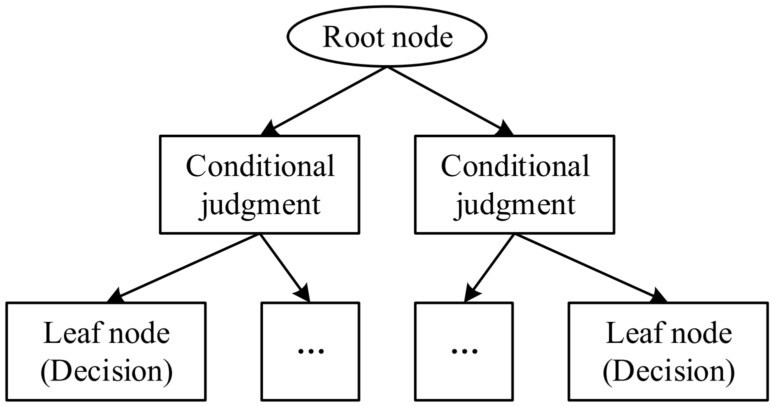


Figure 1: Tree Model Structure Diagram

There are many algorithms for decision tree construction, and the commonly used ones are ID3 algorithm and C5.0.

The core of ID3 algorithm is to select the characteristics on all nodes in the decision tree according to the corresponding message gain principle and recursively form the decision tree. Specifically: starting from the root node, by estimating the message gain of the characteristics of the various possibilities on the nodes, and selecting the characteristics with the highest message gain to become the characteristics of the child nodes, and then according to the difference of the characteristics of the value of the node became the node. The above approach is then recursively re-scheduled for the child nodes and a decision tree is created. Until each point has little or no characteristics to choose from for the signal gain and obtains to the decision tree.ID3 is equivalent to probabilistic model selection by the great likelihood method.

The common methods used to calculate the information gain are:

$$E(X)=-\sum_{i=1}^{n}p_i\log_2(p_i) \tag{2}$$

Let the variables $X=\{x_1,x_2...x_n\}$, then its probability in the set is $P=\{p_1,p_2...p_n\}$.

At the same time, there are two ending preconditions for recursion in the process of recursively constructing a decision tree: the first ending condition is that each class label is identical, and then one can go back to the class labels directly. The second ending condition is that all the features are used but still not able to classify the data into groups containing only unique classes, which means that the construction of the decision tree is not successful. But the data features are not sufficient, which means that the data dimension is not sufficient, due to the second termination condition and can not simply return to the unique class label, all of them have to select the class with the highest number of features as the return value. The learning process of decision tree can be roughly divided into three steps:

(1) feature selection: in many features to select a single feature value and defined as the current node division criteria, and because of the quantitative evaluation of each feature selection criteria vary, which gives rise to a variety of different decision tree algorithms.

(2) Decision Tree Generation: Generate child nodes in top-down order according to the set feature evaluation criteria and end at the moment when the division cannot be continued.

(3) Pruning: Since overfitting occurs in decision trees, they must be pruned to reduce the size of the tree structure and thus reduce overfitting.

C5.0 is a classic decision tree algorithm, which is an improvement and extension of the ID3 algorithm. The algorithm employs a heuristic approach to constructing decision trees, aiming to generate smaller, faster and more accurate models. During the construction process, C5.0 first preprocesses the data, including dealing with missing values and outliers. The purpose of this preprocessing is to ensure the quality and performance of the decision tree and improve the generalization ability of the model.C5.0 uses information gain, information gain ratio and other indicators to select the optimal features for node partitioning, and introduces pessimistic pruning technology to make the feature selection more accurate.

The algorithm adopts a top-down greedy strategy to recursively select the optimal features at each node until the stopping condition is satisfied. In the process of node division, C5.0 adopts an enhanced pruning strategy to prevent overfitting. After completing the construction and pruning of the decision tree, C5.0 can be used for classification and prediction tasks.

The main advantages of the decision tree model are: the amount of operation is small compared to other models, so the calculation is rapid. It is easy to be transformed into classification rules, and the next division can be decided independently according to the division requirements along the way. Easy to grasp, can get the main influence on the classification results of the characteristics. Reused to deal with data sets with large amounts of data.

The main drawbacks of decision tree models are how branching is handled incorrectly and can cause overfitting. Associations between attributes are also often not considered in the model.

II. A. 3) SVM Support Vector Machines

Support Vector Machine (SVM) [22] has been applied in the field of machine learning as well as data mining, through extensive sample data analysis in the modeling complexity and learning level to find the optimal compromise reached to reduce the real risk to enhance the technical power of the modeling applications.SVM algorithms have a better advantage in dealing with small sample data, high-dimensional data, and nonlinearly differentiable data under the model identification.

The basic idea of the statistical concept is to use the empirical risk minimization theory to study and model the ability to choose between, and finally compromise the actual expected business risk is minimized. The core of statistical theory is then VC-dimensional structure and theoretical risk minimization, that is, confidence risk, empirical risk, and theoretical risk minimization.

The main idea of SVM is to find a hyperplane that can separate the two types of data, and make the distance from the two types of data points to the hyperplane as far as possible, this distance is called the interval. The goal of interval maximization is to make this interval as large as possible. In the process of interval maximization, only part of the data points play a key role in determining the hyperplane, known as the support vectors, these support vectors determine the location of the hyperplane.An important feature of SVM is that it can be used to use kernel functions to map the input space to a higher-dimensional feature space, thus making the nonlinear problem into a linearly divisible or approximately divisible problem. Commonly used kernel functions include linear kernel, polynomial kernel, radial basis function kernel and so on. Support vector machines have good generalization performance, especially when dealing with high-dimensional data. Support Vector Machines are effective in preventing overfitting, which means that it does not rely excessively on training data, but adapts well to new and unseen data, which makes SVMs a powerful tool for dealing with complex datasets. Another advantage of support vector machines is their robustness to outliers. Since SVM's decisions are based on support vectors, i.e., some data points that are closest to the hyperplane, it is insensitive to data points that are far from the decision boundary, so SVM performs well when dealing with data that contains noise.

The goal of SVM is to find a hyperplane so that the data of the second class is as far away from the hyperplane as possible in order to classify the new data more accurately, even though the classifier is more robust.

The equation for this hyperplane can be expressed as (T in w^T stands for transpose):

$$w^t x + b = 0 \tag{3}$$

The length of any sample point in the sample space to the hyperplane at this point, that is:

$$r = \frac{|w^T x + b|}{|w|} \tag{4}$$

Let all the sample data be successfully divided by the hyperplane, i.e:

$$\begin{cases} w^T x_i + b \ge +1, y_i = +1 \\ w^T x_i + b \le -1, y_i = -1 \end{cases} \tag{5}$$

where y_i denotes the type of the sample.

After first obtaining the above minimum case, and thus w and b, a classification hyperplane is derived, which maximizes the geometric spacing, and is then used for segmentation.

II. B.Integrated Learning Algorithm

II. B. 1) Stacking Integrated Learning Approach

Stacking integrated learning [23] is a method commonly used in big data competitions, due to its good learning and generalization ability, it is often used by competition participants and achieved good results, and it is considered to be one of the most effective integrated learning methods.Stacking integrated learning consists of two main phases, the first phase is mainly used to create a new training set and a test set, and the second phase is to build a model based on the new training set and use the test set to test the classification effect. The second phase is used to build a model based on the new training set and test the classification effect with the test set. Before starting the first phase, the original dataset needs to be divided into training and test sets. In the first stage, n different primary learners can be used to train and fit the Train, and these primary learners can be learners with different parameters but the same algorithm, or learners with different algorithms, so as to ensure the diversity in the first stage. The process is shown in Figure 2.

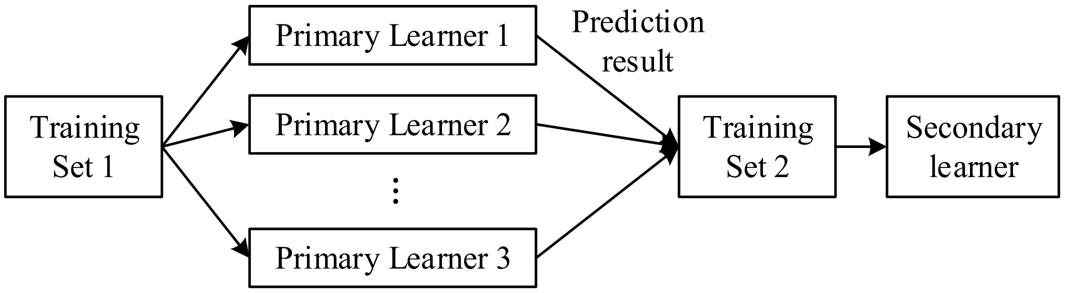


Figure 2: Basic Structure Diagram of Stacking

In the first stage n of the training process of a primary learner, K -fold cross-validation is used. For a single primary learner, it is to randomly divide the training set into K sets of samples with equal number of samples, and use only $K-1$ of them to build the model for each training, and then use the remaining set of samples as the test set, denoted as Train_test, and put it into the already built classification model for prediction. Repeat this operation K times to get K classification models, and combine the test results of K classification models to get a set of prediction results for the training set, which will be used as the training set Train2 for the second stage of learning, and will also be put into the model obtained from each $K-1$ fold training to do the prediction, and get the prediction for the test set Test's K predictions and average these predictions to obtain the test set in the second stage, which will be denoted as Test2.

The classification results obtained from each primary learner in the first stage are integrated and then secondary learning is performed. In the second stage, this paper chooses logistic regression as the secondary learner in Stacking, uses the new training set obtained in the first stage for training, and then puts Test2 into the trained classification model to get the final prediction results.

Stacking integrated learning has a very powerful function, the main feature of Stacking integrated learning is that it integrates the features obtained from the training of each primary learner in the first stage, absorbs the advantages of different models in classification prediction, and obtains better classification results than a single classification model. The addition of cross-validation to the first stage effectively avoids overfitting of the model, which in turn improves the generalization ability of the combined model obtained by Stacking integrated learning.

Although the Stacking integrated learning method has multiple advantages, there are also two places in the method that need attention:

1. In the training of the second layer of the learner can not use the first layer of the learner has been used in the training data.
2. Stacking integration method is prone to overfitting phenomenon, in order to avoid this situation, the second layer of the learner can use a simple classification model.
3. In order to achieve better classification results, diversity should be considered as much as possible when setting up the first layer learner, such as using different classification models.

II. B. 2) Empowerment Methods for Weighted Stacking Integration Learning

Since the performance of the models in the primary learner varies, each classification model gives different information to the second layer learner, in order to better get the effective information in the primary learner, this paper sets different weights for each learner based on the good or bad performance of each learner in the first layer. The assignment process is as follows:

1. Construct a classifier $h_i(x)$ in the first layer learner.
2. Calculate the error rate err_i for the default class of this classifier with the following formula:

$$err_i = \left(\sum_{j=1}^{n_i} \frac{n_j}{N}\right) / K \tag{6}$$

Where, $i = 1, 2, ..., n$, denotes the number of primary learners in Stacking integration learning. n_j is the number of samples that are actually default classes in the j-fold cross-validation of the ith learner, but are misclassified as non-default classes. N denotes the total number of samples in the training set that are actually default classes. K denotes the number of folds of cross-validation used in the model, and in this paper, five folds of cross-validation are used.

Error rate is defined as the percentage of the number of samples that actually belong to a certain class but are judged by the model to be in another class versus the total number of samples in that class, so the error rate of the default class here refers to the percentage of the number of samples that actually belong to the default class but are judged to be non-defaulted by the classification model versus the total number of defaulted samples. Since the actual project, more attention is paid to the default class, rather than the non-default class, so do not use the error rate of non-default or the overall error rate.

3. Since this paper mainly considers the recognition ability of the default class, the weights of the primary learner are set to be calculated as follows:

$$\beta_i = \frac{1 - err_i}{err_i} \tag{7}$$

where β_i is the learning weight of the ith learner.

4. In order to make the sum of the weights of the primary learners to be 1, the following transformation is made:

$$w_i = \frac{\beta_i}{\sum \beta_i} \tag{8}$$

where w_i denotes the weight occupied by the second stage of Stacking integration learning for the ith learner.

III. Effectiveness of Smart Contract Execution Efficiency Optimization Model

III. A. Comparison of Smart Contract Execution Efficiency Optimization Models

As shown in Table 1, the confusion matrix intuitively shows the prediction performance of each model, among which the models with the least positive samples misjudged as negative samples and negative samples as positive samples are stacking, with 10 and 31 models, followed by random forests, with 42 and 60 samples, respectively, and the model logistic regression models with the most positive samples and negative samples as positive samples are 145 and 133, respectively.

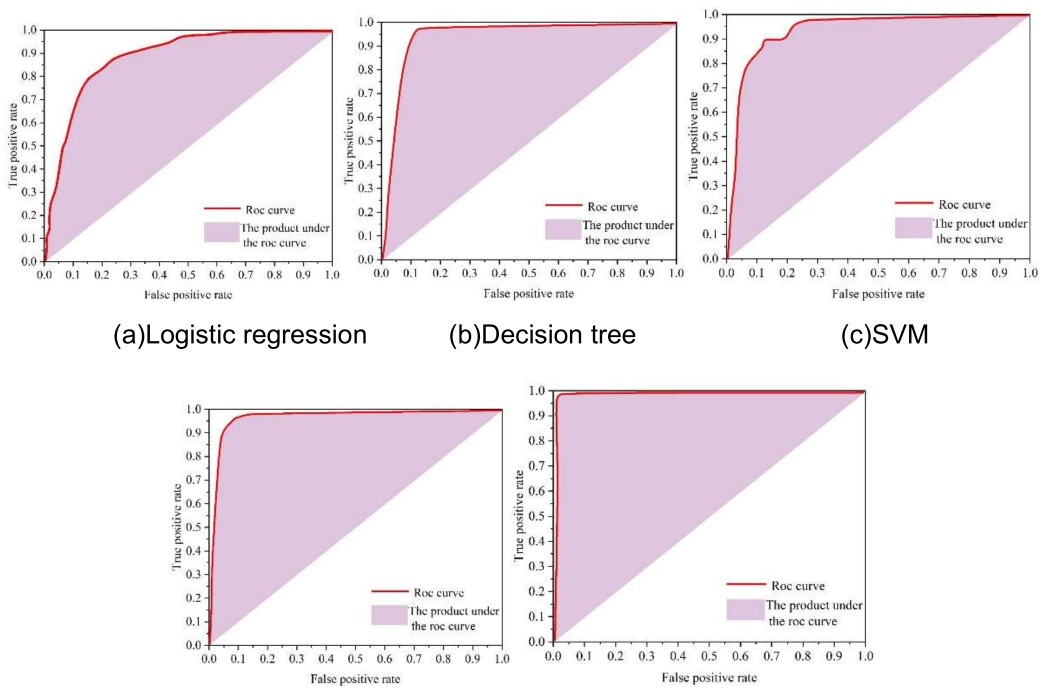
Table 1: Comparison of the model confusion matrix

	Confusion matrix		Prediction category	
			Positive class	Negative class
Single algorithm	Logical regression real category	Positive class	642	145
		Negative class	133	680
	The real category of the decision tree	Positive class	781	44
		Negative class	70	705
	SVM real category	Positive class	700	120
		Negative class	80	700
Integrated learning	GBDT real category	Positive class	785	42
		Negative class	60	713
	Stacking real category	Positive class	821	10
		Negative class	31	738

Then we compare the model evaluation indexes derived from the confusion matrix, the comparison of the evaluation indexes of each model is shown in Table 2, and the ROC curves of each model are shown in Fig. 3, and Figs. (a) to (e) are the ROC curves of logistic regression, decision tree, SVM, GBDT, and Stacking, respectively. Among the five models, Stacking has the highest accuracy, precision, recall, and F1 value, which are all above 0.9500, and the model with slightly worse prediction performance is logistic regression, but the indicators are all above 0.8000, which still has better prediction performance. Among the single-algorithm models, the decision tree has the best performance, and all five model evaluation indicators are in the leading position among the single-algorithm models, and the prediction performance is comparable to that of GBDT of integrated learning. Overall, the integrated learning model outperforms the single-algorithm model in all metrics.

Table 2: Comparison of model evaluation indicators

Model		Accuracy rate	Precision rate	Recall rate	F1 value	AUC
Single algorithm	Logistic regression	0.8137	0.8274	0.8005	0.8059	0.8845
	Decision tree	0.9268	0.9155	0.9417	0.9366	0.9545
	SVM	0.8669	0.8880	0.8465	0.8550	0.9430
Integrated learning	GBDT	0.9314	0.9222	0.9439	0.9396	0.9766
	Stacking	0.9703	0.9562	0.9878	0.9805	0.9934



(a)Logistic regression (b)Decision tree (c)SVM

(d)GBDT (e)Stacking

Figure 3: Roc curve

III. B. Prediction results of the weighted Stacking model

The data is imported into the Stacking model, and Table 3 shows the confusion matrix obtained by the Stacking model on the test set, the correctly predicted non-executing users are 400, 60 executing users are not detected, and 80 users are misclassified as non-executing users, the number of misclassified users by the Stacking model is less, which shows the Stacking model has a better performance on the smart contract execution efficiency optimization task with better performance.

Table 3: The confusion matrix of the stacking prediction model

True result	Predictive result	
	Execute	Nonexecution
Actual retention number	5445	80
Actual loss of users	60	400

Fig. 4 shows the ROC curve of the model on the test set with an AUC value of 0.9289, which shows an improvement in the generalization ability of the model.

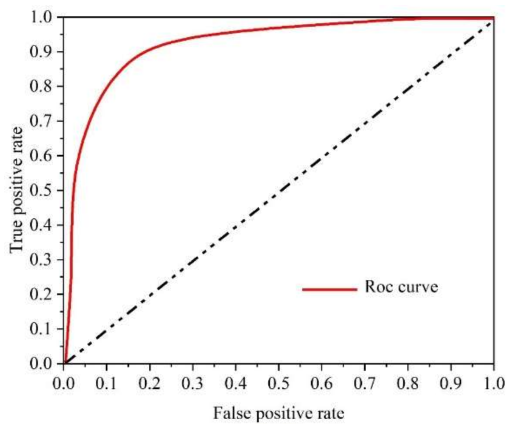


Figure 4: The roc curve of the stacking model

Table 4 shows the comparison of the model evaluation metrics values of Stacking model with the other classification models, from this table it can be seen that the highest value of accuracy rate among the individual classifiers is the SVM model and the accuracy rate of Stacking model is improved by 5.08% as compared to it.The highest value of F1 is the CatBoost model and the Stacking model is improved by 3.79% as compared to it. The AUC value of Stacking model is improved by 0.97% compared to the best performing CatBoost model.

The strategy of optimizing the execution efficiency of smart contracts based on the Stacking model significantly improves the performance of data classification, and the accuracy performance is improved by 83.15%, which means that the response time of the database is greatly shortened in practical applications, and the user experience is significantly improved. The optimized smart contract execution efficiency is faster and consumes less resources, which saves a lot of computing resources and time costs for users. Users can obtain the required information more quickly and make decisions more efficiently. In summary, the performance optimization strategy for smart contract execution efficiency is very successful.

Table 4: The stacking prediction model is compared to a single model

Model	Accuracy rate	Recall rate	F1 value	AUC
Logistic	0.7283	0.8069	0.7663	0.8748
KNN	0.6005	0.8194	0.6930	0.8106
SVM	0.7913	0.8247	0.8078	0.9019
RandomForest	0.7811	0.8176	0.7977	0.8871
XGBoost	0.7305	0.9032	0.8085	0.9055
LightGBM	0.7513	0.8841	0.8123	0.9069
CatBoost	0.7911	0.8475	0.8186	0.9090
Stacking	0.8315	0.8695	0.8496	0.9178

IV. Conclusion

The research on smart contract execution efficiency optimization model in this study has achieved remarkable results, and the experimental results have fully proved the excellent performance of the integrated learning method in this field. Comparative analysis of the experiments shows that the integrated learning model outperforms the single algorithm model in all evaluation indexes. The accuracy of the weighted Stacking model reaches 83.15%, with an F1 value of 0.8496 and an AUC value as high as 0.9178, which is significantly improved over the single model. Confusion matrix analysis shows that the model is able to accurately identify 5445 executive and 400 non-executive users with a low misclassification rate, reflecting a strong classification ability. Among the single algorithms, the SVM model performed the best with an accuracy rate of 79.13%, but the weighted Stacking model outperformed it with an accuracy rate of 5.08%. Data analysis shows that the introduction of the weighting mechanism effectively improves the model performance, and by assigning different weights to the primary learners, it fully utilizes the advantages of each classifier to obtain more accurate prediction results. The weighted Stacking model not only theoretically verifies the feasibility of integrated learning in smart contract efficiency optimization, but also shows good results in practical applications, which can effectively reduce the resource consumption during smart contract execution, increase the response speed and improve the user experience. Future work can further explore the application of more types of integrated learning methods in smart contract optimization and combine them with deep learning technology to construct a more efficient smart contract execution efficiency optimization model.

References

[1] Ertz, M., & Boily, É. (2019). The rise of the digital economy: Thoughts on blockchain technology and cryptocurrencies for the collaborative economy. International Journal of Innovation Studies, 3(4), 84-93.

[2] Catalini, C. (2017). How blockchain technology will impact the digital economy. Blockchains Smart Contracts Internet Things, 4, 2292-2303.

[3] Viriyasitavat, W., Da Xu, L., Bi, Z., & Pungpapong, V. (2019). Blockchain and internet of things for modern business process in digital economy—the state of the art. IEEE transactions on computational social systems, 6(6), 1420-1432.

[4] Dorofeyev, M., Kosov, M., Ponkratov, V., Masterov, A., Karaev, A., & Vasyunina, M. (2018). Trends and prospects for the development of blockchain and cryptocurrencies in the digital economy. European Research Studies, 21(3), 429-445.

[5] Babkin Alexander, V., Burkaltseva Diana, D., Pshenichnikov Wladislav, W., & Tyulin Andrei, S. (2017). Cryptocurrency and blockchain-technology in digital economy: development genesis. π-Economy, 67(5), 9-22.

[6] Pryanikov, M. M., & Chugunov, A. V. (2017). Blockchain as the communication basis for the digital economy development: advantages and problems. International journal of open information technologies, 5(6), 49-55.

[7] Sinitsyn, S. A., Diakonova, M. O., & Chursina, T. I. (2021). Smart Contracts in the Digital Economy: Contractual Regulation and Dispute Resolution. In Smart Technologies for the Digitisation of Industry: Entrepreneurial Environment (pp. 155-164). Singapore: Springer Singapore.

[8] Shah, M., & Samra, R. (2020). Smart contracts for the digital economy. Journal of Securities Operations & Custody, 12(1), 82-87.

[9] Glazkova, I., Kozioł-Kaczorek, D., & Shmatko, S. (2018). Smart contracts as a new technology in the digital economy. Problems of World Agriculture/Problemy Rolnictwa Światowego, 18(4), 146-151.

[10] Fiorentino, S., & Bartolucci, S. (2021). Blockchain-based smart contracts as new governance tools for the sharing economy. Cities, 117, 103325.

[11] Dash, B., Ansari, M. F., Sharma, P., & Swayamsiddha, S. (2022). Future ready banking with smart contracts-CBDC and impact on the Indian economy. International Journal of Network Security and Its Applications, 14(5).

[12] Unal, D., Hammoudeh, M., & Kiraz, M. S. (2020). Policy specification and verification for blockchain and smart contracts in 5G networks. ICT Express, 6(1), 43-47.

[13] Kumar, N. M., & Chopra, S. S. (2022). Leveraging blockchain and smart contract technologies to overcome circular economy implementation challenges. Sustainability, 14(15), 9492.

[14] Zou, W., Lo, D., Kochhar, P. S., Le, X. B. D., Xia, X., Feng, Y., ... & Xu, B. (2019). Smart contract development: Challenges and opportunities. IEEE transactions on software engineering, 47(10), 2084-2106.

[15] Hu, W., Fan, Z., & Gao, Y. (2019). Research on smart contract optimization method on blockchain. IT Professional, 21(5), 33-38.

[16] Hu, B., Zhang, Z., Liu, J., Liu, Y., Yin, J., Lu, R., & Lin, X. (2021). A comprehensive survey on smart contract construction and execution: paradigms, tools, and systems. Patterns, 2(2).

[17] Dolgui, A., Ivanov, D., Potryasaev, S., Sokolov, B., Ivanova, M., & Werner, F. (2020). Blockchain-oriented dynamic modelling of smart contract design and execution in the supply chain. International Journal of Production Research, 58(7), 2184-2199.

[18] Albert, E., Gordillo, P., Hernández-Cerezo, A., Rubio, A., & Schett, M. A. (2022). Super-optimization of smart contracts. ACM Transactions on Software Engineering and Methodology (TOSEM), 31(4), 1-29.

[19] Nathaniel von der Embse,Sonja Winter,Wes Bonifay,Stephen Kilgus,Carly Oddleifson,Katie Eklund... & Joseph Latimer. (2025). Reconciling discrepant universal screening data to improve decision-making: A Bayesian logistic regression approach. Journal of School Psychology,110,101461-101461.

[20] Shreya Venkatesh,R. Sindhu & V. Arunachalam. (2025). Hardware efficient approximate sigmoid activation function for classifying features around zero. Integration,103,102421-102421.

[21] Yutao He,Jiamin Guo,Huan Ping,MingLiang Zhu & Sujith Mangalathu. (2025). A robust force-finding framework for tensegrity structures using gradient-boosting decision trees and Latin hypercube sampling. Structures,76,108863-108863.

[22] Weijie Pan,Qianru Chen,Menghua Wu,Yue Sun,Ming Li,Shumei Wang... & Jiang Meng. (2025). Identifying of Anticoagulant Ingredients From Moutan Cortex Based on Spectrum-Effect Relationship Analysis Combined With GRA, PLS, and SVM Algorithms.. Biomedical chromatography : BMC,39(5),e70060.

[23] Peng Li,Yaozhao Li,Shaobo Ding,Bin Wei,Xuyong Yang,Daobin Yang & Guo Chen. (2025). Control of molecular stacking and film morphology of phosphomolybdic acid hole transporting layer via processing solvent for high-efficiency organic solar cells. Chemical Engineering Journal,512,162517-162517.