Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 2590940, 11 pages
https://doi.org/10.1155/2022/2590940

Research Article

Option Pricing Model Combining Ensemble Learning Methods and Network Learning Structure

Miao Wang,[1] Yunfeng Zhang,[1] Chao Qin,[2] Peipei Liu,[1] and Qiuyue Zhang[3]

[1]*School of Computer Science and Technology, Shandong University of Finance and Economics, Jinan 250000, China*
[2]*College of Marine Life Sciences, Ocean University of China, Qingdao 266000, China*
[3]*School of Management Science and Engineering, Shandong University of Finance and Economics, Jinan 250000, China*

Correspondence should be addressed to Yunfeng Zhang; yfzhang@sdufe.edu.cn

Received 30 July 2022; Revised 26 September 2022; Accepted 15 October 2022; Published 25 October 2022

Academic Editor: Rajesh Kaluri

Option pricing based on data-driven methods is a challenging task that has attracted much attention recently. There are mainly two types of methods that have been widely used, respectively, the neural network method and the ensemble learning method. The option pricing model based on the neural network has high complexity, and a large number of hyper-parameters will be generated during training, resulting in difficult model adjustment. Furthermore, a lot of training data are needed. The option pricing model based on ensemble learning is not ideal for data feature extraction, because each calculation of the ensemble learning method is mainly to reduce the final residual. Therefore, this paper adopts a learning framework that embeds the modular ensemble learning methods into the network learning structure, and an option pricing model based on deep ensemble learning is proposed. The model is mainly composed of two parts: features reorganization based on random forest, used to calculate the importance of features, combined with the original data as training input; the multilayer ensemble data training structure is based on network learning structure and embeds two ensemble learning methods as network modules, and it also designs a stop algorithm to automatically determine the number of layers. This enables the model to retain the effect of data feature extraction and adapt to small and medium data sets without generating many hyper-parameters. Moreover, in order to make the model fully absorb the advantages of the two ensemble learning methods, we adopt cross-training for data. From the experimental results, it can be concluded that compared with the current optimal method, the prediction performance of the proposed model is improved by 36% in the root mean square error (RMSE), which proves the superiority of the proposed model from the quantitative direction.

1. Introduction

Since the option as a stock derivative officially appeared in the market, options have always been a hot topic in financial markets. As a kind of stock derivative, options also have risk similar to stocks, and in the process of trading may also face huge losses. Thus, how to reasonably avoid the risk has become the early option market that has been plagued by people's problems. To solve this problem, the option pricing model is proposed.

In 1973, Black and Schole [1] first proposed an option pricing model, the Black–Schole model. The Black–Schole model is the first parametric model proposed to predict the price of options based on strict conditional assumptions in economics. Therefore, the model cannot perfectly fit the changing financial market, and there is also a certain error between the predicted value and the real value in the market. So, scholars have made further research by relaxing the strict economic assumptions in the model [2–4].

The parameter model attempts to find some specific laws from the option data obtained in the market for the prediction of future option prices and tries to convert these laws into formulas in economics and mathematics. Some information signals that can be directly obtained from the market, such as stock price, execution price, maturity time, and volatility, are taken as the input of the formula and the price forecast of the options as the output of the formula [5]. However, for the real market that is changing all the time, the sensitivity of the parameter model to the change in the market has not reached people's expectations; that is to say,

the prediction error of the parameter model is not as small as expected compared with the actual market data [6]. In order to make the prediction results of the model more attached to the data in the real market, people began to introduce the data-driven model [7], compared with the parameter model, the data-driven model is generally constructed based on the machine learning method. These models solve the prediction problem of option price in a data-driven way. Data-driven model is to find the connection between data from a large number of historical data, through which to predict the option price.

2. Motivation

The advantage of the data-driven model is that as long as the training data is enough, the regression model trained by the machine learning method can well summarize the situation outside the training data samples [5]. Therefore, the data-driven model can well predict the price of options and even surpass the formula derived from economic principles. The earliest data-driven model introduced into option pricing is the neural network. Starting from the basic neural network method, with the deepening of research, deep neural network (DNN), artificial neural networks (ANN), hybrid neural network, and other methods with higher complexity and better effect are also applied to the field of option pricing. The advantage of these neural network methods is that they can well extract the feature of the original training data, yet neural networks usually require a lot of data to train. At the same time, the neural network model is generally a complex model, and there are many hidden layers and a large number of super parameters, which makes the model parameter adjustment difficult [8]. Therefore, in recent years, the research on the option pricing model is not simply limited to the neural network and various optimization and improvement of the neural network. The traditional ensemble learning method has been introduced into the field of option pricing. In essence, ensemble learning is a method that combines the results of multiple basic learners to make decisions [9]. Compared with the neural network, ensemble learning can be trained on small data sets. At the same time, the ensemble learning method does not have a large number of super parameters, and the debugging of parameters is more convenient. Unfortunately, the ensemble learning method is not as good as the neural networks in feature extraction of original training data. If the advantages of these two methods can be combined and the shortcomings of each other can be complemented, the prediction effect of the model will be significantly improved. Some people [8] believe that the learning advantage of a neural network lies in their layer-by-layer processing of the original data characteristics. Therefore, a deep forest method is proposed. The deep forest adopts a cascade hierarchical structure like the neural network. Moreover, random forest and completely random forest are used as the processing units of each layer. This not only retains the layer-by-layer processing of neural network but also takes advantages of random forest with fewer super parameters and trained on small data sets. The deep forest method has been applied in many fields such as medical image, image classification, and multilabel learning and has achieved ideal results [10–12].

2.1. Contribution. Inspired by deep forest, we introduce this idea to the regression problem of option pricing, and a new option pricing model based on deep ensemble learning is proposed. The model is mainly composed of two parts: features reorganization is based on random forest, used to calculate the importance of features, combined with the original data as training input; the multilayer ensemble data training structure is based on network learning structure and embeds two ensemble learning methods as network modules and designs a stop algorithm to automatically determine the number of layers, which can produce a small number of hyper-parameters and adapt to small and medium data sets, and with good data feature extraction effect. For the input and output of data, the output of each layer is spliced with the original input data to form the input of the next layer. In the meantime, in order to make the model fully absorb the advantages of the two ensemble learning methods, influenced by the idea of mutation in genetic algorithm, we adopt cross-training for data, so that the data are trained in different methods in the adjacent two layers. To briefly summarize the contributions, we have the following:

(1) A new option pricing model based on deep ensemble learning is proposed. This model introduces the idea of deep ensemble learning into the field of option pricing, adopts a learning framework that embeds the modular ensemble learning into the network learning structure, and encompasses two subprocesses, namely, importance extraction and multilayer ensemble.

(2) The multilayer ensemble structure is based on network learning structure and embeds two ensemble learning methods as network modules and designs a stop algorithm to automatically determine the number of layers, which can produce a small number of hyper-parameters and adapt to small and medium data sets, and with good data feature extraction effect. The structure also adopts cross-training for data in order to make the model fully absorb the advantages of the two ensemble learning methods.

(3) A novel features reorganization module which can calculate the importance of features by using the processing results of random forests on the original data is designed. The feature importance matrix is taken as the weight matrix multiplied by the original data as the training data.

2.2. Organization. The rest of the paper is organized as follows: Section 2 explains the background of the parameter model and nonparametric model in the option pricing field. In Section 3, the principle of proposing a model is introduced. In Section 4, the experimental results of the proposed model and classical parametric and nonparametric models are discussed. Finally, Section 5 concludes the paper.

3. Related Work

3.1. Parametric Model. The earliest proposed option pricing model is the B-S model, which was proposed by Black and Schole [1] in 1973. There are some unrealistic assumptions in the B-S model, such as the assumption of stock price returns and the assumption of market-implied volatility are not in line with the actual market [13]. Therefore, the B-S model cannot perfectly fit the actual market, and there is a certain error between the predicted value and the real value. So, scholars have done further research by relaxing the assumption of the B-S model.

Heston [14] assumes that volatility follows a random diffusion process and proposes the Heston model. Merton [15] believes that the price of options should be the sum of a continuous process and a traditional discrete jump process. So the Poisson distribution satisfying the random process is added to the model, and the jump-diffusion model is proposed. Kou [16] found that the prices of some options are different from the traditional discrete jump process, and the probabilities of bipolar jumps are different or show multilevel jump changes. Therefore, a double-exponential jump-diffusion model is proposed. Bollerslev [17] regards the volatility variable of options as a discrete random process of change, unlike the assumption that the volatility in the B-S model is a constant, which proposes the GARCH model. There are also some models breaking the assumption of constant volatility, such as the random volatility model and the random volatility jump model [18]. Some scholars have broken the assumption that the risk-free interest rate is constantly replaced the risk-free interest rate with a variable short-term interest rate, proposed a stochastic interest rate model [19, 20], or replaced the initial constant risk-free interest rate with a weighted average of multiple interest rates, and proposed an "interest rate affine" model, Duffee [21].

These parameter models improve the prediction results of option prices to some extent, but they are still looking for some specific laws from the market and trying to convert them into formulas in economics and mathematics, so they are still subject to some economic and statistical assumptions. Different from the traditional parametric model, our proposed model is based on data-driven option pricing to avoid some assumptions that affect the effectiveness of the model.

3.2. Data-Driven Model. Computer scientists have also used neural networks to solve the problem of option pricing for a long time [7, 22, 23]. Option pricing can be seen as a standard regression problem, and many methods in machine learning can be applied to the field of option pricing.

Many people have been studying how to better apply neural network (NN) to option pricing over the years [24–28]. Bennell [29] used an artificial neural network (ANN) to predict the option price; PC Andreou [30] tried to build a new model by combining ANN and parameter model in addition to using ANN alone; Lajbcygier and Connor [31] proposed a hybrid neural network for trading by applying a guided method; Culkin [32] used deep neural networks to construct an option pricing model. These network models make people pay more attention to the development of big data, and big data have many other aspects that are closely related to people, such as big data analysis of health care can better protect the health [33]. The standard neural network consists of many simple connection processors called neurons, each of which produces a series of real-valued activations. The input neurons are activated by the sensor of the sensing environment, and other neurons are activated by the weighted connection from the previous active neurons. These neural networks are designed to mathematically simulate how the human brain works by receiving a wide range of stimuli and then parsing them by learning the neuron layer that associates input and output [34]. The application field of deep learning is very extensive, and the field of auxiliary diagnosis is also a recent research hotspot. Some scholars have constructed a detection model for sentiment analysis of mental disorder based on attention-based deep learning and fuzzy classification [35]. These neural network methods can well extract data features because they can process the original data layer by layer. Meanwhile, because the neural network methods are black boxes, the processing of each layer is invisible, resulting in a large number of super parameters, and the parameter adjustment of the model is very difficult. At the same time, because the neural network method requires a large number of data for training, the effect on small data sets may not be ideal. Therefore, the traditional ensemble learning method is introduced into the field of option pricing. Similarly, ensemble learning methods have a wide range of applications, such as diabetic retinopathy classification model based on ensemble learning [36] and software cost analysis model based on multiobjective optimization [37]. Some ensemble learning methods also have good applications in financial-related fields [38]. Codru [39] constructed multiple options pricing models by using the ensemble learning methods such as random forests, XGBoost, and LightGBM and conducted prediction experiments on the actual market data. Ensemble learning is a general term for the methods that combine multiple basic learners to make decisions, which is usually used to supervise machine learning tasks. A basic learner is an algorithm that takes a set of labelled examples as input and generates models that generalize these examples (such as classifiers or regressions). The main premise of ensemble learning is that by combining multiple models, the error of a single basic learner is likely to be compensated by other basic learners, so the overall prediction performance of the ensemble will be better than that of a single basic learner [40]. For each basic learner, it can complete training on small data sets as well, and there is no need for a large number of hyperparameters in training. Therefore, the ensemble learning method has the advantages of fewer hyperparameters and adapting to small data sets. Although ensemble learning has many advantages, it is inferior to the neural networks in data feature extraction.

Our proposed model processes data layer by layer to ensure the effect of data feature extraction, and each layer is a set of ensemble learning algorithms, that is, an ensemble of

the ensemble. In order to maintain the diversity of integration, we choose two ensemble learning methods at each level. As is well known, diversity is the key to overall construction [8]. In terms of the selection of specific methods, according to the actual test results of various ensemble learning algorithms in the field of option pricing [39], we choose two methods with the best performance, namely, XGBoost [41] and LightGBM [42], which retains the advantages of fewer hyperparameters of integrated learning and adapting to small data sets and obtains better results than single neural network or single ensemble learning methods.

3.3. Proposed Method. As presented in Section 2.2, there are recent studies showing promising results in option pricing using neural network methods and ensemble learning methods. Thus, this paper aims to retain the advantages of the neural network method in feature extraction, and at the same time, it has the advantages of ensemble learning fewer hyperparameters and adapting to small data sets. Our methodology is presented in Figure 1. In this section, we first introduce the overall architecture of our proposed framework and then discuss details of the two main modules: (1) the features reorganization for obtaining feature weight and (2) the multilayer ensemble structure are used to training data.

3.4. Overall Architecture. Input data are first prepared, which consists of stock prices, execution prices, maturity times, and implied volatility. First, the original input data are processed by random forests, and the output vector and the trained forest model can be obtained. Based on this, the importance of features can be calculated, and the weight matrix of features can be obtained. In the multilayer integration structure, each layer receives the feature information processed by the previous layer, and the processing results of this layer are spliced with the input vector to the next layer. The last layer is the output layer, and the input data of the output layer will no longer splice the original data, but the average value of the prediction results obtained by each processing module of the front layer is output to obtain the final prediction results. The calculation process is as follows:

$$\widehat{y} = \frac{y_1 + y_2 + \cdots + y_n}{n}, \tag{1}$$

where $\widehat{y}$ represents the prediction result of the final output, $y_1, y_2, \ldots, y_n$ represents the output value obtained by the processing module in each layer, and n represents the number of processing modules in each layer.

3.5. Features Reorganization. We obtain the weight matrix of features through the process of features reorganization and combined it with original data as training input in the model. First, the original input data are processed by random forest, and the output vector and the trained forest are obtained. Then, based on this, the importance of the feature is calculated, and the weight matrix of the feature is obtained. The weight matrix is multiplied by the original input feature as the input of multilayer integrated training. The process of calculating the importance of features is as follows:

$$\begin{aligned} n_k &= w_k * G_k - w_{\text{left}} * G_{\text{left}} - w_{\text{right}} * G_{\text{right}}, \\ f_i &= \frac{\sum_{j \in \text{nodess spilt on feature } i} n_j}{\sum_{k \in \text{all nodes}} n_k}, \\ f_{ni} &= \frac{f_i}{\sum_{j \in \text{all features}} f_j}, \end{aligned} \tag{2}$$

where n_k is the importance of a node, w_k, w_{left}, and w_{right} are the ratio of the number of training samples to the total number of training samples in the node and its left and right subnodes, respectively, and G_k, G_{left}, and G_{right} are the Gini impurity of the node and its left and right nodes, respectively.

3.6. Multilayer Ensemble Structure. In this section, we introduce the multilayer ensemble structure in three parts, namely, the layer-by-layer training strategy, two ensemble learning algorithms, and the cross-training and stop algorithm.

3.6.1. The Layer-by-Layer Training Strategy. The multilayered structure is based on the hierarchical structure of deep neural networks. The inner the network layer of the deep neural network can be divided into the input layer, hidden layer, and output layer according to different positions. Generally, the first layer is the input layer, the middle layers are hidden layers, and the last layer is the output layer. The layers are fully connected; that is, any neuron in layer i must be connected to any neuron in layer $i + 1$.

Although DNN looks complex, it is very similar to sensors in small local models, i.e., a linear relationship $z = \sum_{i=1}^{m} w_i x_i + b$ plus an activation function $\sigma(z)$.

Since there are many parameters and layers of DNN, the definitions of bias b and linear coefficient w need certain rules. Definition of bias b: the bias corresponding to the third neuron in the second layer is defined as b_3^2. Definition of linear coefficient w: the linear coefficient from the fourth neuron in the second layer to the second neuron in the third layer is defined as w_{24}^3. Among them, the upper marker 2 represents the number of layers, and the lower marker 3 represents the index of neurons where bias exists. Note that the input layer has no w parameter, bias parameter b.

Assuming that the activation function we choose is $\sigma(z)$, the output value of the hidden layer and the output layer is a. For the output a_1^2, a_2^2, a_3^2 of the second layer, we have

$$\begin{aligned} a_1^2 &= \sigma\left(z_1^2\right) = \sigma\left(w_{11}^2 x_1 + w_{12}^2 x_2 + w_{13}^2 x_3 + b_1^2\right), \\ a_2^2 &= \sigma\left(z_2^2\right) = \sigma\left(w_{21}^2 x_1 + w_{22}^2 x_2 + w_{23}^2 x_3 + b_2^2\right), \\ a_3^2 &= \sigma\left(z_3^2\right) = \sigma\left(w_{31}^2 x_1 + w_{32}^2 x_2 + w_{33}^2 x_3 + b_3^2\right). \end{aligned} \tag{3}$$

For output a_1^3 of the third layer, we have

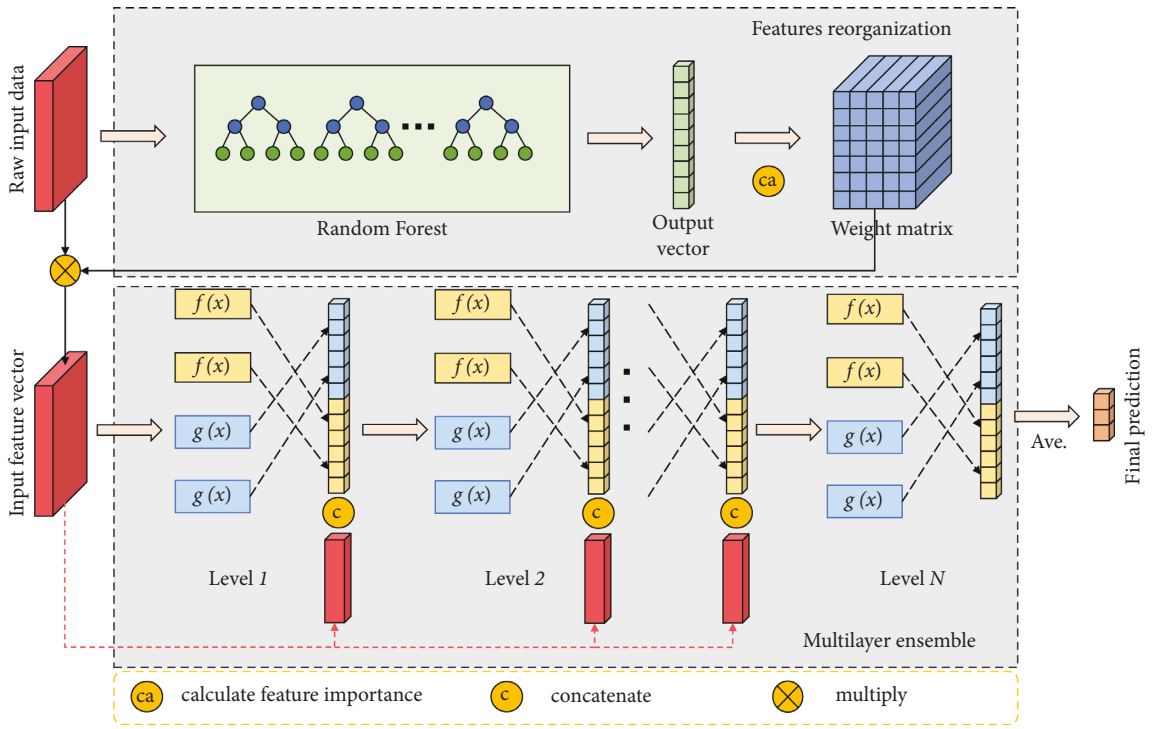

FIGURE 1: Illustration of the deep ensemble model framework. The model consists of two parts, namely, features reorganization module and multilayer ensemble module. In the multilayer ensemble module, the output of each layer is spliced with the original input data as the input of the next layer; the specific number of layers N is determined by the stopping algorithm $\partial(n)$, when $\partial(n) \geq 0$, layers stop growing automatically, N determined, and $f(x)$ and $g(x)$ represent two different ensemble learning methods, and x represents historical data from day 1 to day i, i.e., $x = [x_1, x_2, \ldots x_i]$.

$$a_1^3 = \sigma\left(z_1^3\right) = \sigma\left(w_{11}^3 a_1^2 + w_{12}^3 a_2^2 + w_{13}^3 a_3^2 + b_1^3\right). \tag{4}$$

Generalizing the above example, assuming that there is m neuron in the $l-1$ layer, for the output a_j^l of the j neuron in the l layer, we have

$$a_j^l = \sigma\left(z_j^l\right) = \sigma\left(\sum_{k-1}^{k-1} w_{jk}^l a_k^{l-1} + b_j^l\right). \tag{5}$$

If $l = 2$, then for a_k^1 is the input layer x_k.

From the above, it can be seen that using the algebraic method to express the output one by one is more complex, and if the matrix rule is relatively simple. Assuming that there are m neurons in the $l-1$ layer and n neurons in the l layer, the linear coefficient w of the l layer constitutes a matrix W^l of $n \times m$, the bias b of the l layer constitutes a vector b^l of $n \times 1$, the output a of the $l-1$ layer constitutes a vector a^{l-1} of $m \times 1$, the linear output z of the l layer before activation constitutes a vector z^l of $n \times 1$, and the output a of the l layer constitutes a vector a^l of $n \times 1$. Then, expressed by the matrix method, the output of the l layer is

$$a^l = \sigma\left(z^l\right) = \sigma\left(W^l a^{l-1} + b^l\right). \tag{6}$$

3.6.2. Two Ensemble Learning Algorithms. Each level is an ensemble of ensemble learning methods, i.e., an ensemble of ensembles. Here, we include two different types of ensemble learning methods to encourage diversity.

The core idea of $f(x)$ can be expressed in three steps. The first step is to continuously add a tree, that is, to continuously split the features to generate a new tree, and each time to add a tree is actually to learn a new function, so as to fit the residual of the previous prediction. The second step is to get k trees when we complete the training, and then, we need to get a predicted sample score. Specifically, according to the characteristics of this sample, each tree will fall to a corresponding leaf node, so that each leaf node will correspond to a score. The third step is to add the scores corresponding to each tree we get, which is our predicted value for the sample. Assuming that we use $\hat{y}$ to represent the predicted value:

$$\begin{aligned} &\hat{y} = \varnothing(x_i) = \sum_{k=1}^{K} f_k(x_i), \\ &\text{where } F = \left\{f(x) = \omega_{q(x)}\right\}\left(\text{q: } R^m \longrightarrow \text{T}, \quad \omega \in R^T\right), \end{aligned} \tag{7}$$

where $\omega_{q(x)}$ is the fraction of leaf node q, F corresponds to the set of all k regression trees, and $f(x)$ represents one of all regression trees.

Obviously, our goal is to make the current prediction value $\hat{y}_i$ as close as possible to the real value y_i, and improve the adaptability of the algorithm to the data outside the training sample as much as possible. Therefore, from a mathematical point of view, this is a problem of finding the optimal value. We regard the objective function as the sum

of the loss function and the regularization term; then, the objective function can be expressed as

$$L(\varnothing) = \sum_i l(\hat{y}_i - y_i) + \sum_k \Omega(f_k). \tag{8}$$

It can be seen from the formula that the objective function is divided into two parts, the left side $\sum_i l(\hat{y}_i - y_i)$ is the loss function, and the role is to reveal the training error, that is, the gap between the predicted score and the real score. The right side $\sum_k \Omega(f_k)$ is a regularization term to define the complexity of the objective function. For formula (8), $\hat{y}_i$ is the output of the entire cumulative model, and the regularization term $\sum_k \Omega(f_k)$ is a function that represents the complexity of the tree in the model. The smaller the value of the regularization term is, the lower the complexity of the tree is, and the stronger the generalization ability of the model is. The specific formula is expressed as

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|\omega\|^2, \tag{9}$$

where T represents the number of leaf nodes, γ represents the parameters used to control leaf nodes, so that the leaf node T is as few as possible, ω represents the fraction of leaf nodes, and λ is used to control the fraction of leaf nodes. The role of γ and λ is to minimize the prediction error and prevent overfitting.

Specifically, i in the first part of the objective function represents the prediction error of the i sample, and $l(\hat{y}_i - y_i)$ represents the prediction error of the i sample. Our goal is, of course, that the smaller the error is, the better. Previously, it was said that the first method needed to accumulate the scores of multiple trees to get the final prediction score. In the process of iterative implementation, each iteration was based on the existing tree, adding a tree to fit the residual between the prediction results of the previous tree and the true value. We need to select a f in each iteration to minimize our objective function. The objective function for the entire first method process can now be expressed as

$$\text{Obj}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + \text{canstant}. \tag{10}$$

Similarly, l in formula (10) represents the loss function, and $\Omega(f_t)$ represents the regularization term.

$g(x)$ is a method proposed to improve some defects of the traditional boosting method. The traditional boosting algorithm needs to scan all the samples for each feature to select the best segmentation point, so the traditional boosting method will be time-consuming. Therefore, with the development of the traditional boosting method, it has been unable to meet the current needs in efficiency and scalability. In order to solve the high-latitude mass sample data that need to be processed now, $g(x)$ uses two methods, one is GOSS (gradient-based one-side sampling) method, and the GOSS method does not use all the sample points to calculate the gradient, but sample to calculate the gradient; the other is the EFB (exclusive feature bundling) method. The EFB method does not scan and calculate all the features when searching for the best segmentation point but binds some features together to reduce the dimension of the features and then finds the best segmentation point, which will greatly reduce the consumption in the process of searching for the best segmentation point. These two methods can reduce the time complexity of processing samples but do not lose accuracy.

GOSS method: in the calculation of information gain, generally, the sample points with a large gradient play a major role in the calculation; that is to say, these sample points with a large gradient can contribute more information gain, so the main idea of GOSS method is to retain these sample points with large gradient when the sample is downsampled, ignored a part of the remaining sample points with small gradient, and randomly sampled these sample points in proportion, which can not only save the processing time but also maximize the accuracy of information gain assessment.

EFB method: the traditional boosting method will not only conduct data sampling but also conduct feature sampling, mainly to further reduce the training speed of the model. The second method also has feature sampling, but this feature sampling is not the same as the traditional feature sampling method. It binds mutually exclusive features to reduce the dimension of features so that there will be less consumption for feature sampling. Usually, the high-latitude data in our application are basically sparse data, which makes it possible to reduce the number of valid features by designing an almost lossless method, especially in a sparse feature space where there are many mutually exclusive features, which allows us to bind mutually exclusive features stably together to form a new feature, so as to reduce the feature dimension.

The combination of mutually exclusive features in $g(x)$ uses the histogram algorithm. The basic idea of the histogram algorithm is to discrete the continuous eigenvalues into k integers and construct a histogram with width k. According to the discrete value as the index in the histogram of the cumulative statistics, when traversing the data once, the histogram accumulated the required statistics, and then according to the discrete value in the histogram, traverse to find the optimal segmentation point.

3.6.3. The Cross-Training and Stop Algorithm. Both $f(x)$ and $g(x)$ are very mature ensemble learning algorithms, which are widely used in various scenarios. In comparison, $f(x)$ adopts the greedy process to calculate each feature to find the optimal segmentation point with better accuracy, and $g(x)$ adopts the decision tree growth strategy of selecting a leaf node with the largest splitting gain in each leaf node layer to split with higher accuracy. To fully absorb the different advantages of the two methods in training, inspired by the idea of mutation in genetic algorithms, we exchange the output data of each layer and input the results

of the two methods into the next layer after splicing with the original data, so that the adjacent two layers are trained by different methods.

At the same time, the layers of our model are adaptive, and after adding a new layer, the performance of the whole model will be estimated on the verification set. If there is no obvious improvement, the number of layers will no longer increase, and the training process will terminate. For the performance evaluation criteria, we choose the root mean square error of prediction which is more suitable for the options field, not the accuracy commonly used in the classification field. The multilayered structure of the whole model can adaptively determine the number of layers by $\partial(n)$, which also makes the module applicable to different scales of training data. The calculation process of $\partial(n)$ is as follows:

$$\partial(n) = r(n) - r(n-1), \tag{11}$$

where $r(n)$ represents the root mean square error of prediction with n layer, and $r(n-1)$ represents the root mean square error of prediction with $n-1$ layer. Stop increasing layers when $\partial(n) \geq 0$.

4. Experiments

4.1. Data. The data used in our experiment are daily data on the KOSPI200 option market for the period from 2 June 2009 to 7 November 2019. KOSPI200 option is based on KOSPI200 index. The KOSPI200 index is a weighted total market price index of 200 blue-chip stocks listed in the Korean stock market. In order to avoid the synchronization caused by the trading effect, we use the closing price of the KOSPI call option as price data. We use 2064 (80%) samples as the training data set, and the remaining 516 (20%) samples as the test data set. Input vector X consists of stock price S, strike price K, maturity time $T-t$, and volatility σ. Here, we use σ available in the market, which is given according to the general implied volatility formula. We exclude the interest rate γ from the input vector because it changes very little in the period of KOSPI200 data.

The evaluation criteria of pricing results are measured by root mean square error (RMSE). The errors were calculated according to moneyness and the duration of the contract. According to the duration of the contract, it can be divided into short term (<1 month), medium term (1 – 3 months), and long term (>3 months), as well as according to the moneyness, it can be divided into Deep In The Money($S/K < 0.8$)), In The Money(S⁄(K∈(0.8,0.96))), At The Money(S⁄(K∈(0.96,1.04))), Out of The Money(-S⁄(K∈(1.04,1.2))), Deep Out of The Money($S/K > 1.2$)).

5. Results

This section discusses the experimental results of the model proposed in Section 3. We selected several comparative experimental models for analysis, including two parameter models, respectively, the Black–Scholes model (1973) [1] and the Heston model (1993) [14]. At the same time, there are three machine learning models, namely DNN [43], XGBoost [41], and LightGBM [42]. To emphasize the prediction power of our model, the comparison will be made with the error of these parametric models and machine learning models.

Table 1: Pricing error (RMSE) for 3 different models.

	M1	M2	M3
Input	S/K, t	S, K, t	S, K, t, σ
Output	C/K	C	C
BS	2.808	2.808	2.808
Heston	2.831	2.831	2.831
DNN	0.027	0.028	0.030
XGBoost	0.033	0.024	0.027
LightGBM	0.026	0.025	0.025
Ours	**0.022**	**0.023**	**0.016**

In order to find the most suitable feature space, three models are compared, and each model has different input and output. The experimental results are shown in Table 1. Here, S represents the KOSPI200 stock price, K is the strike price, t is the expiration time of the option contract, and σ is the implied volatility available in the market. The roughened number represents the minimum pricing error of each model.

The input and output of Model 1 are the same as those proposed by Hutchinson (1994) [7]. In order to reduce dimensionality, he assumes that the evaluation function is homogeneous of degree one in S and K, respectively, $f(S_t, K, \ldots)/K = f(S_t/K, 1, \cdots)$. This approach has been intensively used in the literature [44, 45] with good reported results. In this case, nonparametric models do outperform parametric models.

Model 2 uses the same input and output as model 1, but the difference is that model 2 is not a homogeneous assumption. For the next models, implied volatility has been considered. Like Model 1, the performance of parametric models in Model 2 and Model 3 is not as good as that of nonparametric models. In the nonparametric model, the proposed model is better than the separate DNN, XGBoost, or LightGBM models. It can also be seen that the prediction error is the smallest when using the input and output of model 3.

After determining the input and output, we performed ablation experiments on the proposed model. Ma is a model without cross-training, Mb is a model with only $f(x)$ ensemble learning method, Mc is a model with only $g(x)$ ensemble learning method, and Md is our model. Table 2 shows the results of ablation experiments, and we used root mean square error and mean absolute error as the evaluation indexes of the ablation experiment. It can be seen from Table 2 that the prediction accuracy of the model trained by only $f(x)$ method or only $g(x)$ method is lower than that of the proposed model. Similarly, the prediction accuracy of the model without cross-training method is also lower than that of the proposed model. This proves from the experimental point of view that the cross-training and two methods of training data used in the proposed model are effective.

Based on the input and output of model 3, we do more detailed experiments in terms of moneyness and time until

Table 2: Ablation experiment for 4 different models.

	Ma	Mb	Mc	Md
RMSE	0.023	0.025	0.020	**0.016**
MAE	0.0064	0.0067	0.0063	**0.0062**

Table 3: Out-of-sample RMSE for Model 3 in terms of moneyness and time to maturity.

	Model	All	DITM	ITM	ATM	OTM	DOTM
ALL	BS	2.808	4.997	1.674	3.440	4.585	2.898
	Heston	2.831	4.121	1.887	3.446	4.482	2.238
	DNN	0.030	0.398	0.051	0.063	0.322	0.202
	XGBoost	0.027	0.288	0.020	0.048	0.133	0.215
	LightGBM	0.025	0.281	0.026	0.043	0.161	0.214
	Ours	**0.016**	**0.280**	**0.019**	**0.027**	**0.086**	**0.090**
Short term	BS	2.538	4.930	2.560	4.632	4.359	2.385
	Heston	2.607	4.101	2.969	4.135	4.257	2.097
	DNN	0.062	0.386	0.234	0.346	0.349	0.340
	XGBoost	0.042	0.302	**0.022**	0.260	0.181	**0.133**
	LightGBM	0.041	0.255	0.027	0.211	0.177	**0.133**
	Ours	**0.025**	**0.248**	0.023	**0.113**	**0.117**	**0.133**
Medium term	BS	2.500	4.113	2.497	1.945	2.789	2.317
	Heston	2.811	4.864	2.793	1.942	2.177	1.980
	DNN	0.062	0.144	0.285	0.302	0.386	0.179
	XGBoost	0.047	0.076	**0.149**	**0.092**	0.289	0.038
	LightGBM	0.043	0.072	**0.149**	0.093	0.238	0.034
	Ours	**0.034**	**0.036**	**0.149**	**0.092**	**0.178**	**0.020**
Long term	BS	2.248	—	3.234	3.336	4.160	3.116
	Heston	2.158	—	3.602	3.306	4.265	2.719
	DNN	0.273	—	0.536	0.514	0.773	0.397
	XGBoost	**0.140**	—	0.385	0.389	0.423	0.463
	LightGBM	0.152	—	0.371	0.385	0.456	0.465
	Ours	0.151	—	**0.311**	**0.366**	**0.411**	**0.282**

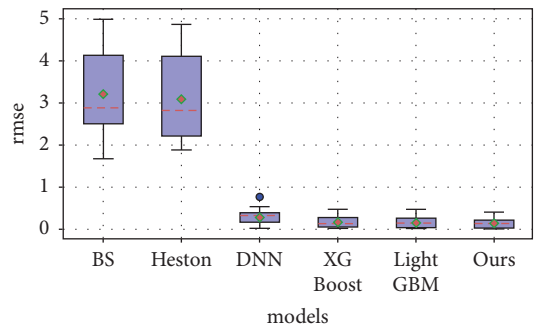

Figure 2: Boxplot of pricing error for the out-of-sample period. The body of the candle represents the distribution of errors between 25% and 75% of the data, and the wick represents the maximum and minimum recorded error. The results are shown for Model 3. Blue dots represent outliers.

maturity. The pricing error is shown in Table 3, and we can see in detail the prediction results of various methods under different moneyness or time until maturity. From the results, even in the case of different moneyness and time until maturity, the prediction effect of all nonparametric models is better than that of parametric models. In the nonparametric model, the prediction accuracy of the two ensemble learning methods is higher than that of DNN. At the same time, it can be seen that the prediction accuracy of our method is better than the existing methods in most cases.

Figure 2 presents the boxplot of pricing errors. The main body of the box represents the error distribution between the first and third quantiles. The center of the box represents the maximum and minimum record errors. As you can see, the error distribution centers of all nonparametric models are basically near 0. The error

Table 4: Pricing errors for 7 nonoverlapping periods.

Month	BS	Heston	DNN	XGBoost	LightGBM	Ours
1	2.812	2.391	0.200	0.123	0.123	**0.110**
2	2.570	2.191	0.328	0.149	0.125	**0.124**
3	2.617	2.862	0.309	0.192	**0.185**	0.187
4	2.686	2.167	0.318	0.162	0.168	**0.160**
5	2.384	2.191	0.270	0.157	0.151	**0.146**
6	2.527	2.680	0.321	0.155	0.155	**0.149**
7	2.205	1.967	0.290	0.194	0.174	**0.157**

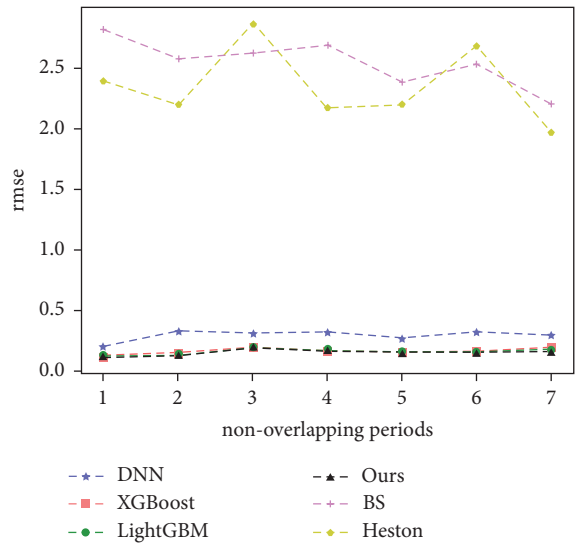

Figure 3: Comparison of error in out-of-sample prediction.

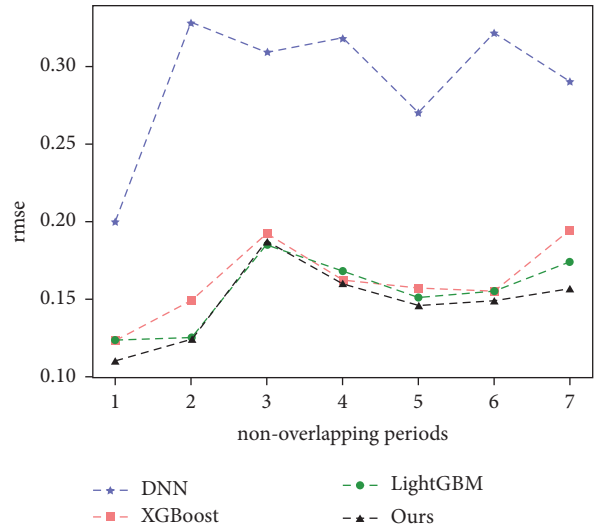

Figure 4: Comparison of error in out-of-sample prediction of nonparametric models.

distribution center of the proposed model is closer to 0, and the error distribution center of the DNN model is farther away from 0. The error distribution center of the parametric model is between 2 and 3, farther away from 0. This also shows that the overall error of all nonparametric models is relatively small, and the proposed model is more accurate in prediction accuracy.

For the robustness of the model, another prediction method is used for the experiment. Use 1-year data as a training set, and then use the next month's data as a test set. Taking one month as a sliding window, seven nonoverlapping periods were tested. The results can be seen in Table 4. Similarly, the nonparametric model has excellent prediction ability. In most cases, the proposed model has higher prediction accuracy than the nonparametric model. However, the differences between nonparametric and parametric models have been diminished. An explication could be the smaller train set compared with the analysis in Table 3, because the more training data of most nonparametric models, the more accurate the prediction results.

Figure 3 shows the visualization results of the seven-month test data. It can be seen that the prediction error of all nonparametric models is much smaller than that of parametric models. Figure 4 shows the error comparison of each nonparametric model more accurately. This also proves that the proposed model has better prediction accuracy.

6. Conclusion

In this study, a new model based on deep ensemble learning was developed for option pricing. The model applies the idea of the deep ensemble to the regression problem of option pricing and encompasses two subprocesses, namely, importance extraction and multilayer ensemble. The performance of the model was experimentally verified, and the results were evaluated from many aspects. A comprehensive comparative study ensures that the model is superior to other models in different measures. Therefore, the model based on deep integration learning is used as a skilled tool for option pricing.

The limitation of the current work is that although the model has achieved good results on option data with certain exercise time, the pricing of option data with uncertain exercise time is still a challenging problem. In terms of future work, we will consider how to improve the pricing power of the model for different types of option data.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant no. 61972227), in part by the Natural Science Foundation of Shandong Province (Grant nos. ZR2020KF015 and ZR2020MA036), in part by the Key Research and Development Plan of Shandong Province (Grant no. 2019GSF109112), and in part by the College Youth Innovation Technology Support Program of Shandong Universities (Grant no. 2020KJN007).

References

[1] F. Black and M. S. Scholes, "The pricing of options and corporate liabilities," *Journal of Political Economy*, vol. 81, no. 3, pp. 637–654, 1973.

[2] C. Menn and S. T. Rachev, "A garch option pricing model with α-stable innovations," *European Journal of Operational Research*, vol. 163, no. 1, pp. 201–209, 2005.

[3] Y. S. Kim, S. T. Rachev, M. L. Bianchi, and F. J. Fabozzi, "Financial market models with lévy processes and time-varying volatility," *Journal of Banking & Finance*, vol. 32, no. 7, pp. 1363–1378, 2008.

[4] G. Lachtermacher and L. A. R. Gaspar, "Neural networks in derivative securities pricing forecasting in brazilian capital markets," in *Proceedings of the 3rd International Conference on Neural Networks in the Capital Markets*, pp. 92–97, World Scientific, London, UK, October 1996.

[5] Y. Yang, Y. Zheng, and T. Hospedales, "Gated neural networks for option pricing: rationality by design," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[6] S. P. Das and S. Padhy, "A new hybrid parametric and machine learning model with homogeneity hint for european-style index option pricing," *Neural Computing & Applications*, vol. 28, no. 12, pp. 4061–4077, 2017.

[7] J. M. Hutchinson, A. W. Lo, and T. Poggio, "A nonparametric approach to pricing and hedging derivative securities via learning networks," *The Journal of Finance*, vol. 49, no. 3, pp. 851–889, 1994.

[8] Z. H. Zhou and J. Feng, "Deep forest," 2017, https://arxiv.org/abs/1702.08835?context=cs.

[9] R. Polikar, "Ensemble learning," in *Ensemble machine learning*, pp. 1–34, Springer, Berlin, Germany, 2012.

[10] L. Sun, Z. Mo, F. Yan et al., "Adaptive feature selection guided deep forest for covid-19 classification with chest ct," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 10, pp. 2798–2805, 2020.

[11] Z. Chen, X. Sun, and L. Shen, "An effective tumor classification with deep forest and self-training," *IEEE Access*, vol. 9, pp. 100944–100950, 2021.

[12] L. Yang, Xi-Z. Wu, Y. Jiang, and Z.-H. Zhou, "Multi-label learning with deep forest," 2019, https://arxiv.org/abs/1911.06557.

[13] H. Park, N. Kim, and J. Lee, "Parametric models and non-parametric machine learning models for predicting option prices: empirical comparison study over kospi 200 index options," *Expert Systems with Applications*, vol. 41, no. 11, pp. 5227–5237, 2014.

[14] S. L. Heston, "A closed-form solution for options with stochastic volatility with applications to bond and currency options," *Review of Financial Studies*, vol. 6, no. 2, pp. 327–343, 1993.

[15] R. C. Merton, "Option pricing when underlying stock returns are discontinuous," *Journal of Financial Economics*, vol. 3, no. 1-2, pp. 125–144, 1976.

[16] S. G. Kou and H. Wang, "Option pricing under a double exponential jump diffusion model," *Management Science*, vol. 50, no. 9, pp. 1178–1192, 2004.

[17] T. P. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of Econometrics*, vol. 31, no. 3, pp. 307–327, 1986.

[18] D. S. Bates, "Jumps and stochastic volatility: exchange rate processes implicit in deutsche mark options," *Review of Financial Studies*, vol. 9, no. 1, pp. 69–107, 1996.

[19] J. Orlin Grabbe, "The pricing of call and put options on foreign exchange," *Journal of International Money and Finance*, vol. 2, no. 3, pp. 239–253, 1983.

[20] J. E. Hilliard, J. Madura, and A. L. Tucker, "Currency option pricing with stochastic domestic and foreign interest rates," *Journal of Financial and Quantitative Analysis*, vol. 26, no. 2, pp. 139–151, 1991.

[21] G. R. Duffee, "Term premia and interest rate forecasts in affine models," *The Journal of Finance*, vol. 57, no. 1, pp. 405–443, 2002.

[22] D. L. Kelly and J. Shorish, *Valuing and Hedging American Put Options Using Neural Networks*, Carnegie Mellon University, Santa Barbara, CA, 1994.

[23] C. Boek, L. Paul, M. Palaniswami, and A. Flitman, "A hybrid neural network approach to the pricing of options," in *Proceedings of ICNN'95-International Conference on Neural Networks*vol. 2, , pp. 813–817, IEEE, 1995.

[24] J. Cao, J. Chen, and J. Hull, "A neural network approach to understanding implied volatility movements," *Quantitative Finance*, vol. 20, no. 9, pp. 1405–1413, 2020.

[25] F. Zhou and K. M. George, "Application of machine learning: an analysis of asian options pricing using neural network," in *Proceedings of the 2017 IEEE 14th International Conference on e-Business Engineering (ICEBE)*, pp. 142–149, IEEE, Shanghai, China, November 2017.

[26] M. Malliaris and L. Salchenberger, "Beating the best: a neural network challenges the black-scholes formula," in *Proceedings of the 9th IEEE Conference on Artificial Intelligence for Applications*, pp. 445–449, IEEE, Orlando, FL, USA, March 1993.

[27] D. Ackerer, N. Tagasovska, and T. Vatter, "Deep smoothing of the implied volatility surface," in *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, Canada USA, December 2020.

[28] J. Ruf and W. Wang, "Hedging with linear regressions and neural networks," *Journal of Business & Economic Statistics*, vol. 40, no. 4, pp. 1442–1454, 2021.

[29] J. Bennell and C. Sutcliffe, "Black-Scholes versus artificial neural networks in pricing FTSE 100 options," *Intelligent Systems in Accounting, Finance and Management*, vol. 12, no. 4, pp. 243–260, 2004.

[30] P. C. Andreou, C. Charalambous, and S. H. Martzoukos, "Pricing and trading european options by combining artificial neural networks and parametric models with implied parameters," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1415–1433, 2008.

[31] P. R. Lajbcygier and J. T. Connor, "Improved option pricing using artificial neural networks and bootstrap methods," *International Journal of Neural Systems*, vol. 08, no. 04, pp. 457–471, 1997.

[32] R. Culkin and S. R. Das, "Machine learning in finance: the case of deep learning for option pricing," *Journal of Investment Management*, vol. 15, no. 4, pp. 92–100, 2017.

[33] A. Priyanka, M. Parimala, K. Sudheer, R. Kaluri, K. Lakshmanna, and M. Praveen Kumar Reddy, "BIG data based on healthcare analysis using IOT devices," *IOP Conference Series: Materials Science and Engineering*, vol. 263, p. 042059, 2017.

[34] J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[35] U. Ahmed, R. H. Jhaveri, G. Srivastava, and J. C. W. Lin, "Explainable deep attention active learning for sentimental analytics of mental disorder," *Transactions on Asian and Low-Resource Language Information Processing*, New York, NY, USA, 2022.

[36] G. T. Reddy, S. Bhattacharya, S. S. Ramakrishnan et al., "An ensemble based machine learning model for diabetic retinopathy classification," in *Proceedings of the 2020 international conference on emerging trends in information technology and engineering (ic-ETITE)*, Vellore, India, February 2020.

[37] S. P. Singh, G. Dhiman, P. Tiwari, and R. H. Jhaveri, "A soft computing based multi-objective optimization approach for automatic prediction of software cost models," *Applied Soft Computing*, vol. 113, 2021.

[38] C. Qin, Y. Zhang, F. Bao, C. Zhang, P. Liu, and P. Liu, "XGBoost optimized by adaptive particle swarm optimization for credit scoring," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–18, 2021.

[39] C. F. Ivaşcu, "Option pricing using machine learning," *Expert Systems with Applications*, vol. 163, Article ID 113799, 2021.

[40] O. Sagi and L. Rokach, "Ensemble learning: a survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[41] T. Chen and C. Guestrin, "Xgboost: a scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, San Francisco, CA, USA, August 2016.

[42] K. Guolin, M. Qi, T. Finley et al., "Lightgbm: a highly efficient gradient boosting decision tree," *Advances in Neural Information Processing Systems*, vol. 30, pp. 3146–3154, 2017.

[43] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[44] H. Amilon, "A neural network versus black–scholes: a comparison of pricing and hedging performances," *Journal of Forecasting*, vol. 22, no. 4, pp. 317–335, 2003.

[45] S.-Ho Yang and J. Lee, "Predicting a distribution of implied volatilities for option pricing," *Expert Systems with Applications*, vol. 38, no. 3, pp. 1702–1708, 2011.