Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 3, No. 1, 2023

Research on CSI 300 Stock Index Price Prediction Based On EMD-XGBoost

Yu Wang, Li Guo, Yanrui Zhang, Xinyue Ma

Dalian Polytechnic University, Dalian, China

Abstract: The combination of artificial intelligence techniques and quantitative investment has given birth to various types of price prediction models based on machine learning algorithms. In this study, we verify the applicability of machine learning fused with statistical method models through the EMD-XGBoost model for stock price prediction. In the modeling process, specific solutions are proposed for overfitting problems that arise. The stock prediction model of machine learning fused with statistical learning was constructed from an empirical perspective, and an XGBoost algorithm model based on empirical modal decomposition was proposed. The data set selected for the experiment was the closing price of the CSI 300 index, and the model was judged by four indicators:mean absolute error, mean error, and root mean square error, etc. The method used for the experiment was the EMD-XGBoost network model, which had the following advantages: first, combining the empirical modal decomposition method with the XGBoost model is conducive to mining the time series data for Second, the decomposition of the CSI 300 index data by the empirical modal decomposition method is helpful to improve the accuracy of the XGBoost model for time series data prediction. The experiments show that the EMD-XGBoost model outperforms the single ARIMA or LSTM network model as well as the EMD-LSTM network model in terms of mean absolute error, mean error, and root mean square error.

Keywords: XGBoost Algorithm; EMD-XGBoost model; LSTM; Empirical Modal Decomposition.

1. Introduction

In the financial field, stock price prediction has always been a high concern for investors or institutional companies, but due to the dynamic and nonlinear characteristics of stock price time-series data, how to accurately predict stock prices is still a relatively challenging task. Compared with traditional mathematical models, deep neural networks can represent the actual complex nonlinear problems more carefully and efficiently by pre-training the data layer by layer to get the primary features of each layer; recurrent neural networks introduce the concept of time series in the network structure, which makes them excellent in the processing of time series data; however, due to the problem of long-time dependence, Hochreiter and Schmidhuber have made RNN network unit for structural improvement, and then proposed a long and short-term memory model, adding a gate control unit to solve the problems of gradient disappearance and lack of long-term data memory capability, making the neural network can really deal with long-distance temporal information effectively[4]. XGBoost is an improved algorithm proposed by Dr. Tianqi Chen based on the gradient boosting decision tree algorithm.XGBoost uses parallelization to improve its operation speed, and introduces the second-order bias of the loss function for more general prediction effect[5]. By combining the ensemble empirical modal decomposition (EEMD) with the XGBoost algorithm, Yi Jing constructed a combined EEMD-XGBoost model and used the model to predict the daily closing price of the SZSE Composite Index, and analyzed and optimized the model[6]. Jiawei Gu proposed a combined stock price prediction model of XGBoost-ESN and optimized the parameters of XGBoost model and echo state network model (ESN) using lattice search method[7].

2. Model Theoretical Basis

2.1. XGBoost

2.1.1. Sub-section Headings

The XGBoost model uses the greedy algorithm and quadratic optimization to ensure that the predicted value of each leaf node of the decision tree is the optimal solution by building K regression trees, using cross-validation to select the best parameters, and adding regularization to prevent overfitting; it has the advantages of high efficiency, good results, ability to handle large-scale data, and support for custom loss functions.

The objective function of XGBoost consists of two parts: the loss function and the canonical term. The loss function is a representation of how well the model fits the data. Usually, its first-order derivative is used to point out the direction of gradient decline. XGBoost also calculates its second-order derivative, which further takes into account the trend of gradient change and fits faster and with higher accuracy; the regular term is used to control the complexity of the model. The more leaf nodes, the larger the model, which not only takes a long time to operate, but also over-fits after exceeding a certain limit, leading to the degradation of the classification effect. the regular term of XGBoost is a penalty mechanism, and the more the number of leaf nodes, the stronger the penalty is, thus limiting their number.

XGBoost belongs to the Boost integrated learning method, which applies serial base learners, where the learning target of the kth learner is the residual of the previous k-1 learners and the target output, and the final learner is represented as follows:

$$y^{(k)} = \sum_{k=1}^{t} f_k(x), k = 1, 2, \cdots, t \qquad (1)$$

where represents the base regression tree numbered k. Thus,

for input $\{(x_1, y_1),(x_2, y_2),\cdots,(x_n, y_n)\}$, the objective function we learn when learning the kth base learner is expressed as follows:

$$Obj^{(k)} = \sum_{i=1}^{n} l\left(y_i, y_i^{k-1} + f_{(k)}(x_i)\right) \quad (2)$$

where, y_i is the real input of the ith data, y_i^{k-1} is the integrated output of the first k-1 learners for the i data,, and $f_k(\cdot)$ is the kth learner to be learned.

The objective function of XGBoost will add a regularization term, and the decision tree will be pruned at a later stage to prevent overfitting, and the objective function after adding the regularization term is as follows:

$$Obj^{(k)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{k-1} + f_{(k)}(x_i)\right) + \sum_{j=1}^{k} \Omega(f_{(j)}) \quad (3)$$

The regularization term is represented as follows:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{t=1}^{T} w_t^2 \quad (4)$$

where T is the total number of leaf nodes of the base regression tree, w_t is the output value of the tth leaf node of the base regression tree, and γ、λ is the coefficient of the regularization term, which is a hyperparameter.

When training the kth base regression tree, the regularization term of the first k-1 base regression trees is a constant, and extract them individually into C(a constant), then the objective function becomes as follows:

$$Obj^{(k)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{k-1}\right) + f_{(k)}(x_i) + \gamma T + \frac{1}{2}\lambda \sum_{t=1}^{T} w_t^2 + C \quad (5)$$

where, y_i and y_i^{k-1} are constants, so the loss is a function of $f_k(x_i)$, The quadratic Taylor expansion of the loss function for $f_k(x_i) = 0$ is as follows:

$$f_k(x_i) = g_i f_k(x_i) + \frac{1}{2} h_i f_k(x_i)^2 \quad (6)$$

where,

$$g_i = \frac{\partial l(y_i, y_i^{k-1})}{\partial y_i^{k-1}} = \partial y_i^{k-1} l(y_i, y_i^{k-1}),$$
$$h_i = \frac{\partial^2 l(y_i, y_i^{k-1})}{\partial (y_i^{k-1})^2} = \partial^2 y_i^{k-1} l(y_i, y_i^{k-1}). \quad (7)$$

The loss function introduced above is brought back to the objective function and simplified to obtain Equation (8):

$$Obj^{(k)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{k-1}\right) + g_i f_{(k)}(x_i) + h_i f_{(k)}(x_i)^2 + \gamma T + \frac{1}{2}\lambda \sum_{t=1}^{T} w_t^2 + C \quad (8)$$

where, g_i and h_i are constants, and $f_{(k)}(x_i) = w$, so we change the summation order, combine $f_{(k)}(x_i)$ and w to obtain the new objective function as follows:

$$Obj^{(k)} = \sum_{t=1}^{T} \left(\sum_{f_{(k)}(x_i)=w} g_i \right) w_t + \frac{1}{2} \left(\sum_{f_{(k)}(x_i)=w} h_i + \lambda \right) w_t^2 + \gamma T \quad (9)$$

Define $G_t = \sum_{f_{(k)}(x_i)=w} g_i, H_t = \sum_{f_{(k)}(x_i)=w} h_i$, then the objective function further becomes equation (10):

$$Obj^{(k)} = \sum_{t=1}^{T} G_t w_t + \frac{1}{2}(H_t + \lambda) w_t^2 + \gamma T \quad (10)$$

This is an independent quadratic function of T with respect to w_t. Taking the smallest value of each quadratic function, i.e.

$$w_t = -\frac{G_t}{H_t + \lambda} (H_t + \lambda > 0)$$
$$\sum_{t=1}^{T} \left(-\frac{G_t^2}{2(H_t + \lambda)}\right) + \gamma T \quad (11)$$

The method of determining the base regression tree structure is mainly to recursively determine whether a leaf node is suitable to be extended. For a particular leaf node t_x that we want to extend, calculate the value of its objective function before extension:

$$obj_1 = \sum_{t=1}^{T} \left(-\frac{G_t^2}{2(H_t + \lambda)}\right) + \gamma T \quad (12)$$

Iterate through all possible values of all features and calculate the objective function value after each value is extended, t_x Split the two new leaf nodes t_1 and t_2:

$$obj_2 = \sum_{t=1}^{T+1} \left(-\frac{G_t^2}{2(H_t + \lambda)}\right) + \gamma(T+1) \quad (13)$$

The difference between the two represents the information gain after segmentation:

$$obj_1 - obj_2 = \frac{1}{2}\left(\frac{G_{t_1}^2}{(H_{t_1} + \lambda)} + \frac{G_{t_2}^2}{(H_{t_2} + \lambda)} - \frac{G_{t_x}^2}{(H_{t_x} + \lambda)}\right) - \gamma \quad (14)$$

The segmentation with the largest information gain is the optimal solution for this leaf node. The lower limit of information gain can be set to restrict the tree from growing too deep, while over-fitting can be prevented by setting the upper limit of the maximum depth of the tree.

2.2. EMD

The empirical modal decomposition algorithm idea[8] is that any time series consists of an intrinsic modal function (IMF) as well as a residual term. Obtain these IMF components with initial signal characteristics, the following approach can be followed.

(1) In the first step, all the extreme and minimal value points on that time series are found.

(2) Then, the extreme value envelopes $x_{\max}(t)$ and $x_{\min}(t)$ of the time series can be obtained by fitting all the extreme and extreme minima of the time series using the cubic spline interpolation function, respectively.

(3) The mean value of the envelope $m_1(t)$ is calculated from the extreme value envelopes $x_{\max}(t)$ and $x_{\min}(t)$, i.e.

$$m_1(t) = \frac{x_{\max}(t) + x_{\min}(t)}{2} \quad (15)$$

(4) The original time series x(t) is subtracted from the mean envelope $m_1(t)$ to obtain the first component $h_1(t)$ without lower order frequencies:

$$h_1(t) = s(t) - m_1(t) \quad (16)$$

(5) In general, the components $h_1(t)$ are largely unstable, so it is necessary to check whether $h_1(t)$ meets the IMF criteria and whether it is necessary to revert to step (1) and use $h_1(t)$ as the initial series and pick again, i.e.

$$\begin{aligned} h_2(t) &= h_1(t) - m_2(t) \\ &\cdots\cdots \\ h_k(t) &= h_{k-1}(t) - m_k(t) \end{aligned} \tag{17}$$

(6) Subtracting IMF1 from the original time series s(t) again yields a residual quantity signal $r_1(t)$ without higher order frequencies, denoted as follows:

$$r_1(t) = s(t) - IMF_1 \tag{18}$$

(7) Take $r_1(t)$ as a new round of the sequence, and again follow the procedures from step 1 to step 6 to obtain $r_2(t)$. Follow such steps until when the nth residual quantity $r_n(t)$ has and has only one extreme value point, at which time there are no more redundant IMF components to extract, then the whole EMD decomposition process will be able to stop.

Finally, we can express the initial time series s(t) as the sum of all IMF components and one residual quantity, i.e.

$$s(t) = \sum_{i=1}^{n} IMF_i(t) + r_n(t) \tag{19}$$

2.3. ARIMA

ARIMA (Auto regressive Integrated Moving Average) model, full name autoregressive differential moving average model, is a linear time series forecasting method jointly proposed by Box (Box) and Jenkins (jenkins) in the 1970s, and is also a model commonly used for time series forecasting[9].

ARIMA (p,d,q), where AR is the autoregressive and p is the number of autoregressive terms; MA is the moving average and q is the number of moving average terms; and d is the number of differences needed to make a non-stationary time series into a stationary time series. The ARIMA model is expressed as follows:

$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-1} + \varepsilon_t + \sum_{i=1}^{p} \theta_i \varepsilon_{t-1} \tag{20}$$

where y_t and ε_t are the actual value and random error of time period t; p,q are the orders of autoregressive and moving average, and ε_t is the sequence of random disturbance terms. Among them, the model order (p,q) is the key aspect of ARIMA model construction, which determines the accuracy of model prediction.

2.4. LSTM

LSTM is a special kind of recurrent neural network, it avoids the problem of gradient disappearance and gradient explosion caused by traditional recurrent neural networks by carefully designing the "gate" structure, and can effectively learn the long-term dependence relationship[10]. Therefore, the LSTM model with memory function shows a strong advantage in dealing with time series prediction and classification problems.

The LSTM is composed of multiple isomorphic cells, and the structure is able to store information in long time by updating the internal state, A indicates that 3 cells have the same cell structure. Each cell consists of 4 main elements: input gate, forgetting gate, output gate and cell state.

The first step in the LSTM is to decide the discard and retention of information through the forgetting gate, which has the following structure:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{21}$$

The second step in the LSTM is to determine the new information being stored in the cell state, and the algorithm expression is as follows:

$$\begin{aligned} i_t &= \sigma(W_t \cdot [h_{t-1}, x_t] + b_t) \\ \tilde{C}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \end{aligned} \tag{22}$$

The third step in the LSTM is to update the cell state by updating it to , with the following expression:

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t \tag{23}$$

The fourth step in the LSTM is to output the information, which needs to be filtered before outputting, and its algorithm expression is as follows:

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t \tanh(C_t) \end{aligned} \tag{24}$$

Where, x is the input vector of the LSTM cell; h is the output vector of the cell; f,i,o denote the forgetting gate, input guys, and output gate, respectively; C denotes the cell state; the subscript t denotes the current moment; σ, tanh are the activation functions of sigmoid,tanh, respectively; W and b denote the weight and bias matrix, respectively.

The key to the LSTM is the cell state C, which will keep the cell state in memory at moment t and is regulated by the forgetting gate f_t and the input gate i_t .The role of the forgetting gate is to let the cell remember or forget its previous state C_{t-1} , the role of the input gate is to allow or prevent the incoming signal from updating the cell state, and the role of the output gate is to control the cell state C input and transmission to the next cell. The internal structure of the LSTM cell is composed of multiple perceptrons, and the most common training method is the backpropagation method.

3. Model Construction and Effect Evaluation

In this paper, the opening price, closing price, maximum price, minimum price, volume, and turnover of the CSI 300 stock index futures data from January 4, 2002 to November 30, 2022 are selected as the data set, and the 5073-trading data are divided into the training and testing sets in a ratio of about 9.5:0.5. The window length of the training data in this paper is chosen as the closing price series, and the default lag order of 25 calculated by the ADF test AIC minimum criterion, i.e., the opening price, closing price, high price, low price, volume, and turnover of the past 25 days are used as input features, and the closing price of the next 1 day is used as the label for model training.

Since the input indicators such as opening price, closing price and volume have different quantiles, the difference in order of magnitude will adversely affect the convergence of the prediction model, so it is necessary to normalize each input indicator in the original data, with reference to the commonly used normalization process as follows:

$$x_i = \frac{X_i - \min(X_i)}{\max(X_i) - \min(X_i)} \quad (25)$$

Where, x_i is the input data after normalization, $\max(X_i)$ and $\min(X_i)$ represents the maximum and minimum values of the index, respectively.

In order to keep the input data and the output data in the same scale, and to evaluate the model prediction ability more realistically and objectively by using the model evaluation index, the output data need to be denormalized, with reference to the common denormalization processing as follows:

$$Y_i = y_i \times (\max(X_i) - \min(X_i)) + \min(X_i) \quad (26)$$

Where, Y_i is the closing price forecast after inverse normalization and y_i is the closing price forecast from the model before normalization.

Model evaluation metrics referring to the experience of the time-series data regression algorithm forecasting study, R^2, mean square error (MSE), mean absolute error (MAE), and mean absolute percentage error (MAPE) were selected as the model evaluation metrics:

$$R^2 = \sum_{i=1}^{n} (Y_i - \bar{Y}_i)^2 \Big/ (Y_i - \bar{Y}_i)^2 \quad (27)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (Y_i - Y_i)^2} \quad (28)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \left| Y_i - Y_i \right| \quad (29)$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} \left| \frac{Y_i - Y_i}{Y_i} \right| \times 100\% \quad (30)$$

In Equation (22) - Equation (25), Y_i is the closing price forecast data, Y_i is the closing price real data; R^2 indicates the ratio of the forecast value to the real value, the larger the value, the better the model performance; MAE,MSE,MAPE indicates the forecast deviation from the real value, the smaller the value, the better the model performance.

3.1. ARIMA Model Performance

In the experiment, the parameter p is set to 3; the integration factor q is set to 1; and d is set to 2. In constructing the ARIMA prediction model, the ARIMA model is trained on the given training set using the statsmodels package in Python, and after the ARIMA model is trained, it is then used to fit the test set data. the predicted Shanghai-Shenzhen 300 index closing price and the predicted results are shown below.

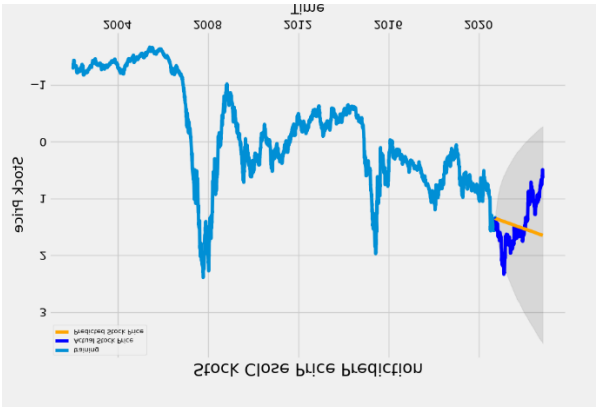

Figure 1. The prediction effect of ARIMA model for CSI 300 index

3.2. LSTM Model Performance

In building the LSTM prediction model, the regression prediction is implemented using the LSTM function that comes with the Keras module in Python. The model contains an input layer, an LSTM layer, a fully connected layer, and an output layer. The batch_size is setted to 100, 50 iterations are performed, and the model is optimized using the Adam optimizer. The prediction results of the model on the test set are shown in Figure 2.

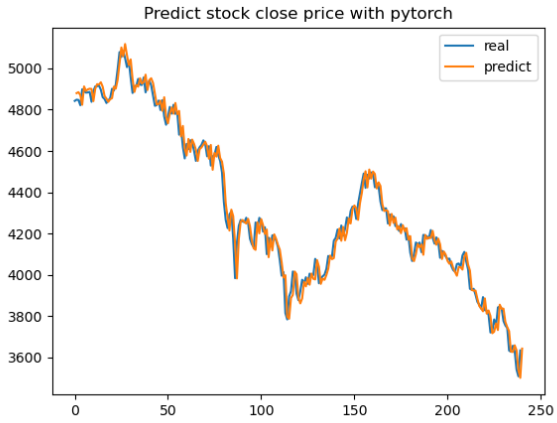

Figure 2. The prediction effect of LSTM model of CSI 300 index

3.3. XGBoost model performance

To construct the XGBoost prediction model, regression prediction using the XGBRegressor function of the XGBoost module in Python to call the reference, setting the penalty term coefficient gamma to 200, the maximum depth of sub-decision trees max_depth to 7, the number of sub-decision trees n_estimators to 90, and the random sampling ratio subsample is 0.3. The prediction effect of the model on the test set is shown in Figure 3.

Figure 3. The prediction effect of XGBoost model for CSI 300 index

3.4. Optimization of empirical modal decomposition method

The empirical modal decomposition algorithm must first extract some of the extreme points of the initial signal, and this property coincides with the instability of a large number of local extreme values caused by the financial markets being in volatility for a long period of time. Therefore, it is very suitable to use EMD algorithm to study and analyze financial data. The empirical modal decomposition of the CSI 300 index data is obtained in Figure 4 as follows.

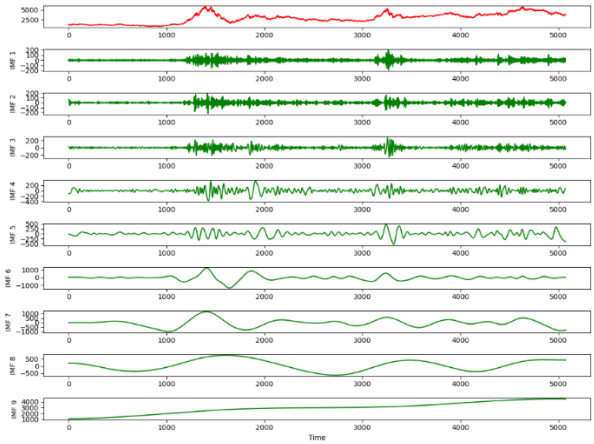

Figure 4. Plot of empirical modal decomposition data

The performance of the LSTM and XGBoost models on the test set after optimization using the EMD method is shown in Figures 4 to 5.

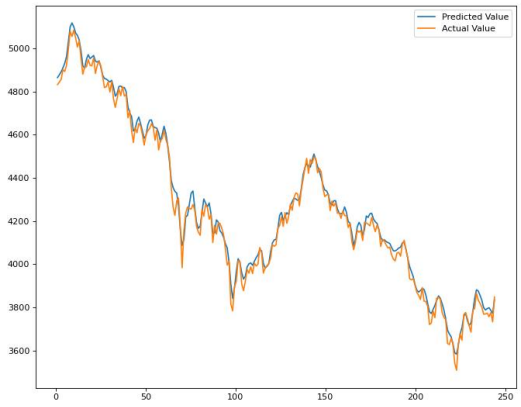

Figure 5. Prediction effect of LSTM model after EMD optimization

Figure 6. Prediction effect of XGBoost model after EMD optimization

3.5. Analysis of evaluation indicators

After training the LSTM and XGBoost algorithm models optimally using the empirical modal decomposition method, the evaluation metrics are compared with the unoptimized evaluation metrics as shown in the following figure.

Table 1. Comparison of evaluation indicators

Algorithm model	R^2	*RMSE*	*MAE*	*MAPE*
ARIMA	0.425	4615.230	2436.5628	0.3385
LSTM	0.921	53.0540	41.0113	0.0096
XGBoost	0.963	52.690	40.4740	0.0096
EMD-LSTM	0.975	1.3234	**1.3188**	0.0030
EMD-XGBoost	**0.982**	**1.1562**	1.3251	**0.0018**

According to Table 1, it can be seen that the prediction effect of LSTM and XGBoost models before optimization is significantly better than that of ARIMA model, and the prediction effect of the two algorithms is more balanced after optimization, while EMD optimization has the most obvious improvement for the prediction effect of XGBoost algorithm. After EMD optimization of XGBoost algorithm, MSE is reduced by 97.80%, MAE is reduced by 96.73% and 81.25% for MAPE. It can be seen that the optimized EMD-XGBoost model has the closest prediction value to the true value and the highest prediction accuracy.

4. Conclusion

In this paper, the XGBoost machine learning algorithm is used to predict the closing price of the CSI 300 stock index futures contract, and the feasibility of the machine learning algorithm for financial time series data prediction is verified. By comparing the prediction effect before and after optimization of empirical modal decomposition methods, the availability of empirical modal decomposition for the improvement of prediction effect of machine learning algorithms is verified. The results show that LSTM and XGBoost can achieve accurate prediction of financial time-series data, and the empirical modal decomposition method can significantly improve the prediction accuracy of XGBoost algorithm.

References

[1] Dezhi Yang. Design and Implementation of Stock Prediction System Based on Nonlinear Combination [D]. Liaoning: Dalian University of Technology,2009. DOI:10.7666/ d.y1606651.

[2] Yilin Ma. Research on Stock Price Prediction and Portfolio Based on Deep Neural Network [D]. Jiangsu: Southeast University,2020.

[3] Yang LIU. Stock Price Prediction Based on Neural Network [D]. Shanxi: Shanxi University of Finance and Economics,2018. (in Chinese)

[4] FELIX A. GERS, JURGEN SCHMIDHUBER, FRED CUMMINS. Learning to Forget: Continual Prediction with LSTM[J]. Neural computation,2000,12(10):2451-2471.

[5] Tianqi Chen,CarlosGuestrin.XGBoost:A Scalable Tree Boosting System. [J].CoRR, 2016, abs / 1603.02754 (abs / 1603.02754).

[6] Jing Yi. Research on Stock Market Analysis and Prediction Based on EEMD and XGBoost Algorithm [D]. Shandong: Shandong University,2020.

[7] Jiawei GU. XGBoost-ESN combination model stock price prediction method [J]. Journal of Mudanjiang Normal University (Natural Science Edition), 2022(1):1-5. DOI: 10.3969/j.issn.1003-6180.2022.01.002.

[8] WenBo Wang. Chinese stock market Prediction Based on EMD and Neural Network [J]. Systems Engineering Theory & Practice,2010,30(6):1027-1033. (in Chinese)

[9] JAMES C. MCKEOWN, KENNETH S. LOREK. A COMPARATIVE ANALYSIS OF THE PREDICTIVE ABILITY OF ADAPTIVE FORECASTING, RE-ESTIMATION, AND RE-IDENTIFICATION USING BOX-JENKINS TIME-SERIES ANALYSIS ON QUARTERLY EARNINGS DATA[J]. Decision Sciences,1978,9(4):658-672. DOI:10.1111/j.1540-5915.1978.tb00752.x.

[10] Stock closing price prediction based on sentiment analysis and STM[J].Neuralcomputing&applications,2020,32(13):9713-9729. DOI:10.1007/s00521-019-04504-2.