Proceedings of the 38th Chinese Control Conference
July 27-30, 2019, Guangzhou, China

Stock Selection based on Extreme Gradient Boosting

Xiaoyun Zhang[1], Wanyi Chen[1]

1. College of Artificial Intelligence, Nankai University, Tianjin 300350
E-mail: xiaoyunzhang@mail.nankai.edu.cn, wychen@nankai.edu.cn

Abstract: In this paper, we established a multi-factor stock selection model based on Extreme Gradient Boosting (XGBoost) to beat the benchmark. We used accounting indicators, valuation indicators, emotions and technical indicators, 62 in total, as the feature space of the XGBoost classifier, and attempted to identify stocks from CSI 300 Index that are likely to outperform the market by having exceptional returns. The cumulative return of the equally weighted stocks selected through XGBoost over 32 months is 134%, which is higher than the benchmark 28%. At the same time, we compared the other two classifiers, Logistic Regression (LR) and Support Vector Machines (SVM) and found that the XGBoost delivered a highest AUC and a better result in final cumulative returns over the other two algorithms in stock classification, which indicating that the XGBoost algorithm is reasonable for the complex nonlinear stock markets.

Key Words: Quantitative investment, Stock Selection, Multi-factor, XGBoost, Machine learning

1 Introduction

An effective quantitative stock selection strategy can greatly alleviate the problem of investors information overload, overcome the shortcomings of human greed and fear in subjective investment, and improve investors' investment performance. According to the Efficient Market Hypothesis (EMH)[1], the market is not always fully efficient and there are stocks whose prices are undervalued. Thus we can choose these stocks and hold them until next period to get a higher excess. Capital Asset Pricing Model (CAPM)[2] shows that in the market equilibrium state, there is a simple linear relationship between securities returns and market returns. Fama and French in 1993 proposed a three-factor model[3] proving that stock returns are related to the company size and book market value ratio in addition to market β. Finding more accurate pricing models is one of the major development directions of asset pricing. In recent years, many scholars are no longer limited to linear models, using more advanced algorithms such as machine learning to estimate the stocks return. Fan. A.[4] used the Support Vector Machine (SVM)[5] to select stocks in Australia market and obtained a gain of 77% over the benchmark. Zhi Su[6] used the kernel principal component genetic algorithm and SVR finding that the prediction results are accurate. There are other scholars who have used fuzzy theory[7], neural networks[8] to select stocks and have achieved positive results.

However, from the existing research results, the current stock selection model still has some aspects needing to be improved. On the one hand, the dimension of the feature space is not high enough to comprehensively analyze the characteristic changes of the future market, on the other hand, the classification algorithm for stock selection has problems such as missing values processing, slow running speed and classification accuracy. Therefore, it is of great significance to apply excellent methods to Chinese financial market. Extreme Gradient Boosting (XGBoost)[9], proposed by Chen (2016), is an improvement algorithm based on the Gradient Boosting Decision Tree (GBDT)[10] in the loss function regularization, feature point segmentation search, sparse recognition and parallel computing, and has been welcomed by many data scientists in the Kaggle competition. XGBoost is widely used in industrial fields such as rocks classification [11], epilepsy patient identification [12] etc., but is rarely used for financial analysis. In this paper, we have creatively built a layered stock selection model based on the XGBoost with comprehensive indicators to identify excellent stocks that outperform the market in the next period. We also compare the results of XGBoost with two other machine learning algorithms, Logistic Regression (LR) and SVM, to verify the effectiveness of XGBoost on stock selection issues.

This paper is organized as follows: Section 2 introduces the XGBoost classification algorithm. Section 3 presents the construction process of the stock selection model, including the construction of feature space, the design of the layered training back-test model and the parameter optimization and evaluation. In Section 4, we apply the actual data to the XGBoost stock selection model for model verification, and compare the results of two other machine learning algorithms. Section 5 is the summary of this article and the improvement of the future work.

2 Classification Algorithm

2.1 Decision Tree

Decision Tree is the basic classifier of the Tree Boosting. The output function of XGBoost is a linear superposition of a series of regression trees (also known as CART [13]).

Given training data set:

$$D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\} \tag{1}$$

a regression tree corresponds to a partition of the input space (i.e. feature space) and an output value on the partitioning unit. Suppose that the input space has been divided into M units $R_1,R_2,\cdots,R_m$, there is a fixed output value c_m on each unit R_m, so the regression tree model can be expressed as:

$$f(x)=\sum_{m=1}^{M} c_m I(x\in R_m) \tag{2}$$

*This work is supported by National Natural Science Foundation (NNSF) of China under Grant U1613210.

When the division of the input space is determined, the squared error $\sum_{x_i \in R_m} (y_i - f(x_i))^2$ can be used to represent the prediction error of the regression tree for the training data, and the minimum error criterion is used to solve the optimal output value of each unit. It is easy to know that the optimal value of c_m on the unit R_m, $\hat{c}_m$,is the mean value of the output y_i corresponding to all input instances x_i on R_m:

$$\hat{c}_m = ave(y_i \mid x_i \in R_m) \tag{3}$$

CART uses a heuristic approach to partition the input space. Select the j-th variable $x^{(j)}$ and its value s as the splitting variable and the splitting point, and define two areas:

$$R_1(j,s) = \{x \mid x^{(j)} \le s\} \; and \; R_2(j,s) = \{x \mid x^{(j)} \ge s\} \tag{4}$$

Then find the optimal splitting variable j and the splitting point s:

$$\min_{j,s}[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2] \tag{5}$$

For the fixed input variable j, the optimal segmentation point s can be found.

$$\hat{c}_1 = ave(y_i \mid x_i \in R_1(j,s)) \; and \; \hat{c}_2 = ave(y_i \mid x_i \in R_2(j,s)) \tag{6}$$

Iterate through all the input variables and find the optimal splitting variable j to form a pair (j,s). The input space is divided into two regions in turn, and then the above division process is repeated for each region until the stop condition is satisfied, thus generating a regression tree.

2.2 Tree Boosting

XGBoost is an improved tree boosting algorithm based on GBDT. For a sample space D ,with n samples and m features, a tree boosting model is obtained by summing scores of every decision tree:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F \tag{7}$$

and the objective function of XGBoost can be defined as follows:

$$Obj(\theta) = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{8}$$

where $D = \{(\boldsymbol{x_i}, y_i)\}(|D| = n, \boldsymbol{x_i} \in R^m, y_i \in R)$, F is the space of regression trees (such as CART above). K is the number of CART, which can be seen as a function map from sample point to fraction . $L(y_i, \hat{y}_i)$ is a differentiable convex loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i . $\Omega(f_k)$ is a regularization term helping to avoid over-fitting.

When we are training the *t-th* tree, it is equivalent to minimizing the objective function:

$$\begin{aligned} Obj^{(t)} &= \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k) \\ &= \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + const \end{aligned} \tag{9}$$

The special feature of XGBoost is to approximate the original loss function with the second-order Taylor expansion of the loss function. The above objective function can be approximated as:

$$\begin{aligned} Obj^{(t)} \approx & \sum_{i=1}^{n} [L(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] \\ & + \Omega(f_t) + const \end{aligned} \tag{10}$$

here $g_i = \partial_{\hat{y}^{(t-1)}} L(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} L(y_i, \hat{y}^{(t-1)})$ are the first and second order partial derivatives of the loss function L with respect to the second variable at the *i-th* sample point respectively, and f_t is the *t-th* tree.

The part of the regularization in the objective function is (used to measure the complexity of the tree):

$$\Omega(f_t) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} \omega_j^2 \tag{11}$$

here T indicates the number of leaf nodes and ω_j indicates the score in *j-th* leaf nodes. γ and λ are penalty factors.

And the specific expression of f_t is as follows:

$$f_t(x) = \omega_{q(x)}, \omega \in R^T, q : R^m \to \{1, 2, \cdots, T\} \tag{12}$$

where ω is a T dimensional vector corresponding to the score on T leaf nodes and q is a map that maps sample points $x \in R^m$ to a leaf. As long as the structure q of the tree and the score ω of the leaves are determined, we can obtain the tree f_t .

XGBoost differs from GBDT in the way it constructs new trees. Define $I_j = \{i \mid q(x_i) = j\}$ as the instance set of leaf j .Substituting the expressions of Eq(11) and Eq(12) into the approximate objective function Eq(10), ignoring the constant part unrelated to f_t, we can obtain:

$$\begin{aligned} Obj^{(t)} &\approx \sum_{i=1}^{n} [L(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \\ &= \sum_{i=1}^{n} [g_i \omega_{q(x_i)} + \frac{1}{2} h_i \omega_{q(x_i)}^2] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} \omega_j^2 \\ &= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) \omega_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) \omega_j^2] + \gamma T \\ &\triangleq \sum_{j=1}^{T} [G_j \omega_j + \frac{1}{2} (H_j + \lambda) \omega_j^2] + \gamma T \\ & where \; G_j = \sum_{i \in I_j} g_i, H_j = \sum_{i \in I_j} h_i \end{aligned} \tag{13}$$

It can be seen from the last equation that the approximation of the objective function is a quadratic function of T independent variables ω_j, which can directly solve the minimum point:

$$\omega_j^* = -\frac{G_j}{H_j + \lambda} \tag{14}$$

and the minimum value of the objective function:

$$Obj^* = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T \tag{15}$$

Obj^* indicates how much more can be reduced on the objective function when we specify a tree. Therefore, it can be called a structural score and the smaller the Obj^* is, the better the structure of the tree is. Then the greedy algorithm is used to enumerate different tree structures and the tree with the smallest structural score is selected.

Assume that I_L and I_R are the instance sets of left and right nodes after the split. Let $I = I_L \cup I_R$. Each time an attempt is made to add a segment to an existing leaf, the gain of Obj^* is calculated by the following equation to determine whether or not to introduce the segmentation:

$$Gain = \frac{1}{2}[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}] - \gamma \tag{16}$$

Introducing the segmentation does not necessarily reduce the objective function, because there is a penalty term for introducing a new leaf in the objective function. The optimization target corresponds to the tree pruning. The segmentation can be cut off when the gain from the segmentation is less than the threshold. So far, the structure of the tree is determined.

3 Stock Selection model

3.1 Feature Space Construction and Preprocessing

If the number of indicators is m, the predicted variable for the *i-th* company at some time t is expressed as $\mathbf{x}_i = \{x_1, x_2, \cdots, x_m\}$.Calculate m indicators on the 5*th* trading day of each month as the original feature data of the sample. Calculate the excess returns of individual stocks in the next month (based on CSI 300 Index), and sort the returns from high to low, and take the top 30% marked Class 1, the last 30% marked as Class -1, while the middle ranked stock is marked as Class 0. Then the original label for the sample is expressed as $y = \{1, 0, -1\}$.If there are n sample points, the original dataset can be represented as the following set:

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_n, y_n)\} \subset R^m \times R \tag{17}$$

In order to make the model more effective in identifying stock feature data, we need to do some simple processing on the initial data set. Features preprocessing generally includes missing value processing, de-extremum, standardization and neutralization. If the number of missing value does not exceed 1%, we replace it with the average for the same stock. In this paper, we use $M \pm 3*\sigma$ to remove the extreme value to exclude errors on the model, where M and σ are the median and standard deviation of feature data, respectively.

Next we use the max-min method to standardize factor data to eliminate the influence of different indicators dimensions:

$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{18}$$

where $x_{\max}$ and $x_{\min}$ represents the maximum and minimum values of the feature data, respectively. Max-min normalization, also known as dispersion normalization, is normalized by linear variation so that the value of the data ranges between [0, 1].

In the quantitative stock selection, the factors used for stock selection may be affected by other factors, resulting in certain biases in the selected stocks. For example, the P/B ratio will have a high correlation with the market value. If we use the P/B ratio without market capitalization, the results of stock selection will be more concentrated. At the same time, the price-earnings ratio of the sunrise industry and the sunset industry also has certain characteristics in general, that is to say, the industry also has an impact on the valuation factors, then the result we get is that there are some redundant preferences. In order to let us use a certain factor eliminated the influence of other factors, we need to neutralize the factors, generally including market value neutralization and industry neutralization, so that the selected stocks are more dispersed.

For other style risk factors, the impact of market value is the most obvious and extensive. We use market value and industry neutralization:

$$Factor_i = \beta_M * \ln(MktVal_i) + \sum_{j=1}^{n} \beta_j * Industry_{j,i} + \varepsilon_i \tag{19}$$

where $Factor_i$ is the alpha factor of *i-th* stock, $MktVal_i$ is the total market value of *i-th* stock and $Industry_{j,i}$ is the industry dummy variable, that is, if *i*-th stock belongs to $Industry_j$, the exposure is 1, otherwise it is 0. Each stock belongs only to one industry.

3.2 Layered Training Back-Test Model

Assume that the training data set has a time span of T months, then the training data set in back test phase is：

$$\{(X_1, y_1), (X_2, y_2), \cdots, (X_T, y_T)\} \subset R^{m\times n} \times R \tag{20}$$

where X_t is an $n \times m$ dimensional matrix containing m features n stocks at time t, $t = 1, 2, \cdots, T$. Let's do a layered back test simulation. We shorted all the stocks bought last month at the beginning of each month, and merged all the stock data from the previous month into the training data set, and removed the oldest first month data set. New data set, as a data set for the next month's training model, is:

$$\{(X_2, y_2), (X_3, y_3), \cdots, (X_T, y_T)\} \subset R^{m\times n} \times R \tag{21}$$

and iterated over the training data every month. The data preprocessing process is required for all data before each training (as introduced in the previous section) because the newly incorporated data set has changed the distribution after merging. This approach takes longer to run, but each time new data was used to train the model, which made the training results relatively more accurate.

3.3 Parameter Optimization and Evaluation

The index for evaluating the performance of classifiers is generally the classification accuracy rate, but we consider the uneven distribution of the sample data of the stock market in different categories so that the traditional accuracy cannot properly reflect the performance of the classifier. In this paper the classification model was evaluated mainly using receiver operating characteristic (ROC) curve and area under roc curve (AUC).

The horizontal and vertical coordinates of the ROC curve are false positive rate (FPR) and true positive rate (TPR) :

$$FPR = \frac{FP}{FP + TN} \tag{22}$$

$$TPR = \frac{TP}{TP + FN} \tag{23}$$

where TP is the number of positive classes predicted as positive classes and FP is the number of negative classes predicted as positive classes and so on. Here TPR is also known as recall rate.

We use the grid search with cross-validation to optimize the parameters of the XGBoost model, as shown in Fig 1. The data set is divided into training set and test set, and the optimal set of grid search with cross-validation is adjusted. The model parameters corresponding to the highest AUC of the test set are taken as the optimal parameters.

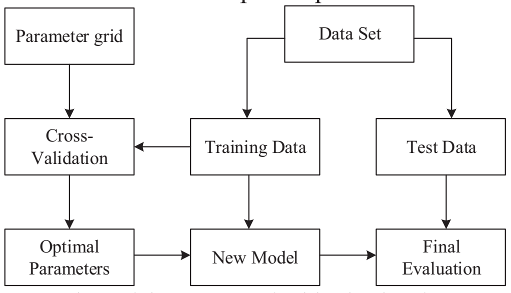

Fig. 1 Overview of the process of grid selection for parameters selection and model evaluation.

4 Experiment

We selected stocks from the CSI 300 Index in Chinese stock market as the stock pool from January 2009 to September 2018. The features data from JoinQuant website involved valuation factors, financial factors, momentum factors, leverage factors, emotional factors, and some technical factors, for a total of 62. We sorted the monthly returns of individual stocks from high to low and marked the top 30% of the stocks as Class 1, the last 30% of the stocks as Class -1, and the middle part of the stocks as Class 0. In order to avoid the high noise impact of stock data, we actually only used the top and the last 30% of the stock data to improve the generalization ability of the model.

Before running the stock picking model, we need to go through a series of data processing and model tuning. The grid search algorithm is used to adjust the parameters for XGBoost, and the evaluation index is the AUC of the test set. The based classifier of XGBoost used in this paper is regression tree, and we found the main influence parameter is $\max\_depth$,which optimal value is 3. We used LR and SVM with a Gaussian kernel to conduct the experiment based on the same data set as comparison to prove the effectiveness of XGBoost. The figure below is the optimal ROC curve obtained after grid search with cross-validation.

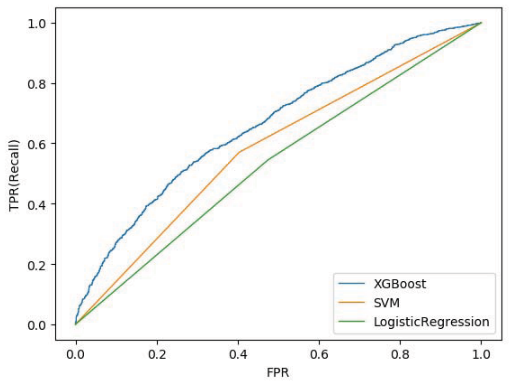

Fig. 2 Comparison of roc curves of three algorithms

All three classifiers ROC curves can exceed the random line $y = x$, indicating that the three methods are effective at a certain level of confidence. The AUC of the three algorithms XGBoost, SVM and LR are 0.6641, 0.5834, and 0.5353 respectively. It is found that XGBoost is superior to the other two algorithms in the prediction accuracy of the model, indicating that it is reasonable to apply XGBoost to the field of stock classification.

In the XGBoost stock selection model back-testing process, we can observe the relative importance of the feature factors, which facilitates the investor's further analysis of the factor's intrinsic market logic. Fig. 3 below is the most important 25 feature factors. We can see that the 20-day rolling Shape Ratio is the most important factor.

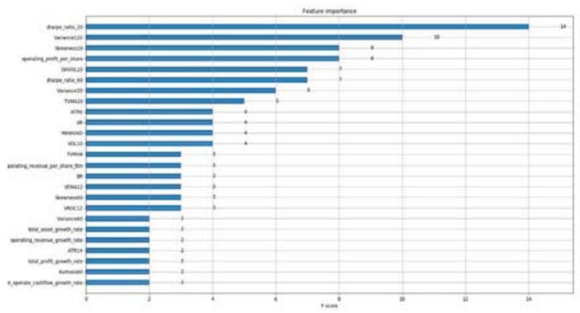

Fig. 3 Top 25 most contributing feature scores

Next, we conduct the stock selection back-testing using XGBoost. We use data from January 2009 to January 2016 as in-sample data for model training and parameters optimization. The actual verification of the model was carried out using data from February 2016 to September 2018 as out-sample data. Set position change by per month and we sold short the stocks selected by the classifier last month and bought the new 15 stocks undervalued by the market. Here, before the stock picking, the up limited and down limited stocks were removed from the stock pool in advance and the stocks bought are given equal weighted funds.

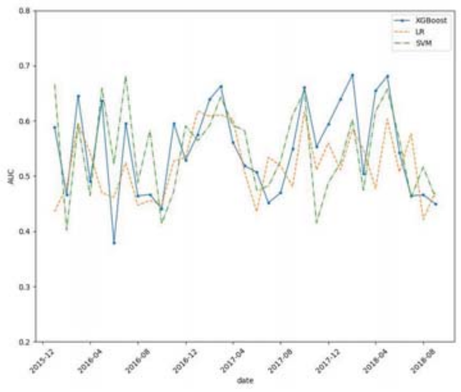

Fig. 4 Out-of-sample AUC for different classifiers during back test

Fig. 4 is the change of AUC in the layered training back test model. It can be seen that the AUC of the three classifier algorithms are not much different, most of them are between 0.45 and 0.65. In this paper, the rolling layered training model (as mentioned in 3.2) is used for training data, so that the model is updated as the market information changes. Here is the stock selection situation of XGBoost every month, compared with the benchmark:

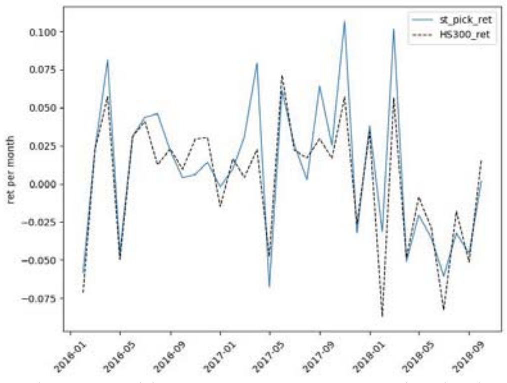

Fig. 5 Monthly return on XGBoost stock selection versus HS300 benchmark

From Fig. 5 we can see that the returns of stocks picked by XGBoost model higher than that of CSI 300 benchmark basically, but because the number of selected stocks is relatively small, the volatility is relatively large. This problem can be avoided by setting the corresponding stop loss conditions in actual trading.

We conducted a continuous rolling back-testing for 32 months from 2016 to 2018 to explore the effectiveness of the XGBoost algorithm. We also compared two other machine learning classifiers (LR and SVM) in addition to the benchmark's return. The following figure shows the cumulative returns over 32 months for three stock picking strategies:

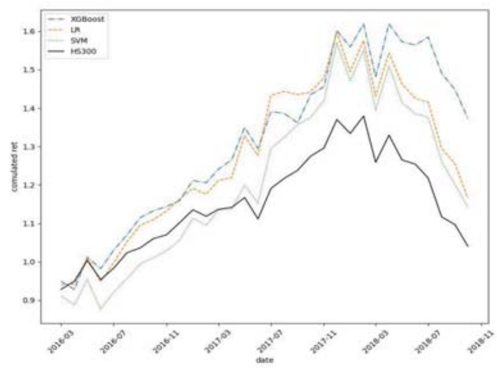

Fig. 6 Comparison of cumulative return of different stock selection strategies.

As seen from Fig 6, the cumulative return of XGBoost classifier is superior to the other two algorithms. Surprisingly, LR also has a good performance in stock selection problem. Because our stocks are selected from the CSI 300 benchmark stock pool, the underlying trend of the strategies is similar to the benchmark trend. From 2016 to 2017, the market's relative bull market stage, the stock selection strategies return has been higher than the benchmark and after entering the bear market in 2018, the return of strategies is also reduced, but the reduction is lower than the benchmark. Using the rolling layered training model, the cumulative return of XGBoost stock selection strategies for the 32 months was 134% which outperformed the benchmark of 28%.

In addition, we additionally explore whether the number of stocks selected has an impact on the final cumulative income. Considering the problem of investment funds, we tried to choose 5, 10, 15, 20 respectively. It is found that the volatility of choosing less than 10 stocks is very large, and the cumulative income of 15 and 20 stocks is not much different.

5 Conclusion

Whether Tree Boosting algorithm is applicable to the financial market, we have carried out preliminary exploration. The results showed that the XGBoost classifier is effective for the stock selection model. We selected CSI 300 index stocks as stock pool. Our rolling layered back test model produced a cumulative return of 134% over a 32-month period, exceeding the benchmark of 28%. We also compared two other machine learning strategies, SVM and LR, and found that XGBoost has two obvious advantages: Firstly, it is superior to the other two algorithms in the prediction accuracy AUC of the model, and has the highest cumulative return in back-testing period. Secondly, XGBoost can conveniently count the contribution of each factor to the model, which helps to reduce the workload of investors in indicators selection.

There are still many problems in the future in terms of quantitative stock selection. We found that AUC of the three methods during the training process is not high enough. The reason may be that the characteristics of the financial market are not particularly obvious, which requires us to further discover more effective factors to join the model. As future works, we can also try to integrate different algorithms into a stacking stock selection model, and also consider the weight optimization of the selected stocks.

References

[1] Malkiel, B. G., & Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, *25*(2), 383-417.

[2] Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The journal of finance*, *19*(3), 425-442.

[3] Fama, E. F., & French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of financial economics*, *33*(1), 3-56.

[4] Fan, A., & Palaniswami, M. (2001). Stock selection using support vector machines. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on* (Vol. 3, pp. 1793-1798). IEEE.

[5] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, *2*(2), 121-167.

[6] Zhi, S., & Xiaoyuan, F. (2013). Kernel principal component genetic algorithm and improved SVR stock selection model. *Statistical Research*, *30*(5).

[7] Huang, C. F., Tsai, M. Y., Hsieh, T. N., Kuo, L. M., & Chang, B. R. (2012, November). A study of hybrid genetic-fuzzy models for IPO stock selection. In *Fuzzy Theory and it's Applications (iFUZZY), 2012 International Conference on* (pp. 357-362). IEEE.

[8] Levin, A. E. (1996). Stock selection via nonlinear multi-factor models. In *Advances in Neural Information Processing Systems* (pp. 966-972).

[9] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd*

international conference on knowledge discovery and data mining (pp. 785-794). ACM.

[10] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189-1232.

[11] Zhang, L., & Zhan, C. (2017, May). Machine learning in rock facies classification: an application of XGBoost. In *International Geophysical Conference, Qingdao, China, 17-20 April 2017* (pp. 1371-1374). Society of Exploration Geophysicists and Chinese Petroleum Society.

[12] Torlay, L., Perrone-Bertolotti, M., Thomas, E., & Baciu, M. (2017). Machine learning–XGBoost analysis of language networks to classify patients with epilepsy. *Brain informatics*, *4*(3), 159.

[13] Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and Regression Trees, 1984: Belmont. *CA: Wadsworth International Group*.

[14] Gumus, M., & Kiran, M. S. (2017, October). Crude oil price forecasting using XGBoost. In *Computer Science and Engineering (UBMK), 2017 International Conference on* (pp. 1100-1103). IEEE.

[15] Harrell, F. E. (2015). Ordinal logistic regression. In *Regression modeling strategies* (pp. 311-325). Springer, Cham.

[16] Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, *28*(2), 337-407.