Hindawi
Complexity
Volume 2022, Article ID 7447229, 17 pages
https://doi.org/10.1155/2022/7447229

Research Article

Multifactor Stock Selection Strategy Based on Machine Learning: Evidence from China

Jieying Gao, Huan Guo, and Xin Xu

School of Finance, Capital University of Economics and Business, Beijing 100070, China

Correspondence should be addressed to Xin Xu; xuxin@cueb.edu.cn

Received 20 August 2021; Accepted 30 December 2021; Published 3 February 2022

Academic Editor: Sameh S. Askar

Copyright © 2022 Jieying Gao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Machine learning methods have been used in multifactor stock strategy for years. This paper uses three machine learning methods and linear regression method to find the most appropriate approach. First, a framework is established and 10 style factors and 30 industry factors are chosen. Second, four methods are used to forecast portfolio returns and compared by predicting returns, successful rate, and Sharpe ratio. Finally, this paper draws conclusion. The main findings are as follows: the support vector regression has the most stable successful rate for predicting, while ridge regression and linear regression have the most unstable successful rate with more extreme cases; algorithm of support vector regression fitting higher-degree polynomials in Chinese A-share market is optimized, compared with the traditional linear regression both in terms of stock return and retracement control; the results of support vector regression significantly outperforming the CSI 500 index prove further.

1. Introduction

Quantitative trading in securities market usually adopts CTA (commodities trading adviser) strategy, intraday high-frequency strategy, and multifactor quantitative strategy. The multifactor models are widely used in the stock market, including Fama–French three-factor asset pricing model [1], Carhart four-factor model [2], and the further improved five-factor and six-factor models. Scholars have found hundreds of market anomalies which might provide excess returns and created a "factor zoo". Bridgewater Associates, Renaissance Technologies, and AQR Capital Management, the top hedge funds by assets in the world, trade in global financial markets achieving exceptional returns for their investors by strictly adhering to quantitative strategies. The vast majority of nonquantitative stock funds also introduce the multifactor model to analyze and allocate their securities positions to a certain extent.

The traditional factor strategies are usually used to forecast stock returns by scoring factor exposures, and linear regression methods commonly used are time series regression, cross-sectional regression, Fama and MacBeth [3] regression, and Hansen GMM regression [4]. However, the relationship between the factor value and the return of individual stocks in the actual stock market is often nonlinear which leads to linear regression that cannot well fit in many cases. In addition, Green et al [5]and Hou et al [6] studies have shown that out-of-sample testing finds that most factors cannot consistently provide excess returns. One of the reasons for the disappearance of excess returns is the increasing convergence of prediction and trading models using traditional methods in the security market, which leads to failure. With the development of artificial intelligence technology, Mullainathan and Spiess [7], Kleinberg et al [8] show data mining, machine learning, and other technical methods are applied to the field of economics and management research. Major financial institutions have also adopted new technologies and methods to improve quantitative trading strategies in security market transactions.

The finance analytical method is improved by introducing machine learning methods, which make the empirical research paradigm expand from linear to nonlinear, from focusing on parameter significance to the model structure and dynamic feature. Appropriate and robust models are built to capture the effective characteristics of financial data and to interpret economic meaning, making great efforts to improve prediction accuracy.

Liu et al. [9] use a support vector machine (SVM) to classify forecasting stock price index and find that support vector function can accurately reflect the variation trend and improve the prediction accuracy. On the basis of the multifactor stock selection model, Wang et al. [10] verify the predictive performance of the random forest algorithm in China's stock market by using it to predict the rise and fall of stocks and analyze the returns of selected stocks. Xie et al. [11] use LASSO regression and elastic net in the process of factor screening to select factors and determine the weight and find that the factors screened by this method could obtain excess returns. Gu et al. [12] test the performance of machine learning algorithms in the US market and find that machine learning models can effectively outperform traditional linear regression models. Wang and Li [13] use the gcForest algorithm to classify individual stocks and predict the probability of rise and fall of stocks. They build an investment portfolio and a back test shows that the portfolio could achieve significant excess returns. They compare the back-test results of SVM and random forest algorithm and find that the gcForest algorithm has obvious advantages over other algorithms in both stable and rising period in the stock market from a comprehensive analysis of various technical indicators.

Although machine learning methods have been used in return forecasts in the security market in recent years, there are still questions about which method is the best or most appropriate for the emerging stock market? Security markets are more volatile in developing countries and have their own features. Based on the Chinese stock market, this study aims to establish a forecasting framework to predict the relationship between abnormal factors and excess returns with different methods, conducting a systematic test and evaluating which method is best. Therefore, this study puts forward three research questions:

(1) Is the machine learning model superior to the traditional predictive model? To verify the first observation, a traditional linear regression model and three machine learning algorithm models are selected in this study. Rapach et al. [14] show traditional linear regression has been used in financial forecasting and achieved good results.

(2) If the prediction model $f(\cdot)$ adopts linear function form, whether the performance of the nonlinear model is better than that of the linear regression model. To verify the second observation, traditional regression and linear ridge machine learning models are used to compare with random forest and support vector machine models. Ridge regression is chosen because it can solve the problem of the sparse model as Hastie et al. [15] research, and random forest and support vector machine algorithms are chosen because both of them are the core algorithms according to machine learning theory and have achieved good results in many tasks as Fernández-Delgado et al. research [16].

(3) If the predictive model $f(\cdot)$ adopts the machine learning methods, which performance is best among the three machine learning models and why?

2. Factor Variable Selection and Model Selection

The task of multifactor model forecasting is a standard supervised learning and regression task, that is, to explore the following functional form:

$$R_{t+1,j} = f\left(x_{t,j};\theta\right) + \varepsilon_{t,j}, \tag{1}$$

where X_{tj} is the factor, the explanatory variable selected by the researcher in advance that has an influence on the return rate of the stock j at time t. The function $f(\cdot)$ can take any form and represents all the possible ways in which x can act on y that the researcher can imagine. The residual term ε represents other possible influence factors beyond control. Compared with the traditional factor regression model, formula (1) does not require that the number of variables in X is smaller than the number of samples and allows x to take effect on y of almost any form. Under specific sample conditions with the traditional econometric analysis framework, researchers can estimate $f(\cdot)$ by the reduced phenomenon regression model and nonparametric methods. But in the reduced linear regression model, the explanatory variable x_j is easily correlated with the residual term ε_j. In addition, when the sample size is limited and x contains many variables, traditional nonparametric estimation is difficult to overcome technical obstacles, and how to select variables has not been solved as showed by Henderson et al. [17].

2.1. Factors Selection.

The essence of the multifactor model is to build an optimal asset portfolio through factor selection. Therefore, factors should be selected as many as possible to explain the return of stocks, so as to minimize the residual of the regression model which represents the return of stocks that cannot be explained by factors. The selection of individual stocks is based on the results of portfolio earnings of the forecasting model. The characteristics of samples, that is, the independent variables in the model, are determined by researchers on account of their market experience. At present, the common practice in the investment industry is to divide factors into industry factors and style factors. The industry factor is a dummy variable, if the individual stock belongs to a certain industry, the corresponding factor value is 1, and the factor value of other industries is 0. Style factors are selected by investors' study and comprehension of the market. The number of style factors excavated by quantitative institutions and the ability to interpret alpha of the stock manifest academic competence of financial institutions, and different institutions may choose different style factors.

Based on the situation of China's A-share stock market, twelve primary factors from four categories are selected: the valuation factor which includes price-to-earnings ratio, price-to-book, total market capitalization, the financial factor which includes price-to-cash-flow ratio and price-to-sales ratio, the momentum factor which includes turnover rate, turnover, yield, the length of the cylinder, and the closing price, and technical factor which includes the length of upper wick and length of lower wick. Technical factors are improved because few researches have paid attention to the length of the wick, but it shows the trading mood which has a great effect on stock price, especially in emerging security markets.

The primary factors are back-tested with the stratified method, and the results are sorted in descending order listed in Table 1. Group a buys the top 20% stocks with the largest factor value ranking in each cross-sectional period each week, while group e buys the stocks with the last 20% factor value ranking in each cross-sectional period each week. The frequency of position adjustment is weekly. At the same time, stocks with an absolute weekly return of more than 15% were removed, which account for less than 2% of the number of stocks, in order to eliminate the impact of stocks with a consecutive daily limit up and daily limit down.

The results of long-short portfolios of the stratified back test show that the net value of four factors, namely, length of wick, length of the lower shadow, price-cash flow ratio, and price-sales ratio, is low, indicating these factors are not correlated with the stock return rate strongly. Considering the net value curve of the long-short portfolio and the stratified back test, two financial factors, price-cash-flow ratio and price-sales ratio, are filtered out, and ten style factors are selected. Combining with 30 industry factors, now 40 factors are selected for the return forecasting model as shown in Table 2.

2.2. Model Selection. Machine learning is a collection of many forms of predictive functions $f(\cdot)$ and all kinds of algorithms. As stock return prediction is a supervised learning regression task, theoretically, all machine learning algorithms adapted to regression task can be used to build stock return prediction models. In this study, three machine learning regression algorithms (ridge regression, random forest regression, and support vector regression) are used to predict the returns of individual stocks. Based on the predicted returns of individual stocks, the investment portfolio is constructed for back test, and the efficiency of the machine learning algorithm is analyzed.

2.2.1. Ridge Regression Model. In the traditional linear regression model, the parameter estimation is generally obtained by minimizing the loss function. The formula of the loss function is as follows:

$$\text{Loss}_{\text{OLS}}(\beta) = \|Y - X\beta\|^2, \tag{2}$$

where *LOSS* is the loss function, X and Y are data matrix and outcome variable, respectively, and β is regression coefficient vector. In contrast to *OLS* estimates, Hoerl and Kennard [18] propose to add a constant λ to the principal diagonal of the $X\prime X$ matrix to ensure the matrix $(X\prime X + \lambda I)$ is invertible and alleviate the multicollinearity problem. In order to obtain the unique solution of the parameter vector β, the paper regularizes it to limit its data range. The penalty term is introduced into the loss function for penalized regression.

$$\text{Loss}(\beta) = (y - X\beta)'(y - X\beta) + \lambda\|\beta\|_2^2, \tag{3}$$

where the first term of (3) is the sum of squares of residuals. The second term is the penalty term, and λ is the adjustment parameter to control the penalty intensity. The optimal solution of parameter β in the ridge regression model $\widehat{\beta}_{\text{ridge}}(\lambda) = (X'X + \lambda I)^{-1}X'y$ can be obtained when the loss function is minimum. The choice of parameter λ determines the degree to which the regression coefficient is compressed. Different values of λ will generate different results. A common method in machine learning proposed by McNeish and Daniel [19] is K-fold cross-validation. The cross-test error is as follows:

$$\begin{aligned} CV(\lambda) &= \overline{\text{MSE}}\,\lambda \\ &= \frac{1}{K}\sum_{k=1}^{K}\frac{1}{n_k}\sum_{i\in\text{fold}_k}\left[y_i - \widehat{y}_i(\lambda)\right]^2. \end{aligned} \tag{4}$$

The cross-validation error $CV(\lambda)$ is a function of λ, and λ is optimal when $CV(\lambda)$ is the smallest.

2.2.2. Random Forest. Random forest is also a combined prediction model, belonging to a Bagging algorithm variation in the family of integrated algorithms. It is a tree-based integrated learning model proposed by Breiman [20] and widely used to solve classification and regression.

The paper uses random forest algorithm of Bagging, using bootstrapping to generate random training samples n from the initial dataset. The probability of each sample being selected is *1/n*, and the probability of each sample not being collected k times is $\lim_{k\longrightarrow\infty}(1-(1/n))^k \longrightarrow (1/e) \approx 0.368$. The 36.8% dataset that did not participate in the training model composes the out-of-bag sample, which can be used to evaluate the out-of-bag error. D_t is used to represent the training sample set actually used by h_t, and $H^{\text{oob}}(\text{x})$ represents the out-of-bag sample prediction of sample x, whose formula is

$$H^{\text{oob}}(x) = \arg\max_{y\in Y}\sum_{t=1}^{T} II\left(h_t(x) = y\right) * II\left(x \notin D_t\right), \tag{5}$$

and the out-of-bag estimation of the generalization error of the Bagging algorithm is

$$\epsilon^{\text{oob}} = \frac{1}{|D|}\sum_{(x,y)\in D} II\left(H^{\text{oob}}(x) \neq y\right). \tag{6}$$

Cawley et al. [21] use the above results as the criteria for model pruning and overfitting to reduce the risk of overfitting.

The random forest method is similar to the bagging method both of which rely on initial data and use the bootstrap method to build the training set. Random forest also introduces a random attribute selection in the training process of the decision tree. In other words, in random forest generation, a subset containing j, attributes are randomly selected from the attribute set of each node of each decision

Table 1: Net value of factor stratified back-test and long-short portfolio.

		a	b	c	d	e	Long-short portfolio
Valuation factors	PE (price/earnings)	1.15	1.75	2.30	2.58	2.32	1.43
	PB(price/book value)	1.07	1.78	2.29	2.36	2.81	2.01
	Market value	1.05	1.06	1.27	2.05	10.05	10.64
Financial factors	Price/cash flow	1.40	2.01	2.61	2.57	1.60	1.10
	Price/sales	1.38	1.93	2.11	2.21	2.47	1.43
	Turnover	0.29	0.77	2.03	4.43	14.89	47.62
	Turnover rate	0.34	1.91	2.94	3.75	4.08	7.58
Momentum factors	Yield	0.35	1.72	3.45	4.16	3.51	10.08
	Length of wick	1.25	2.18	2.61	2.33	1.88	1.35
	Closing price	0.53	1.29	2.24	3.15	6.39	10.78
Technical factors	Length of lower shadow	1.43	2.83	2.65	2.10	1.39	1.06
	Length of upper shadow	0.59	2.38	2.88	2.76	2.77	3.74

Table 2: 40 selected factors.

Style factors	Industry factors		
PE	Petroleum and petrochemical	Electrical equipment and new energy	Banking
PB	Coal industry	National defense and military industry	Nonbank finance
Market value	Nonferrous industry	Auto industry	Real estate
Turnover	Electricity and utilities	Trade and retail	Comprehensive finance
Turnover rate	Steel	Consumer service	Transportation
Yield	Basic chemical engineering	Home appliance	Electron
Length of wick	Architectural industry	Textile and garment	Communication
Closing price	Architectural material industry	Pharmaceutical industry	Computer
Length of lower shadow	Light manufacturing	Food and beverage industry	Media industry
Length of upper shadow	Machinery industry	Agriculture, forestry, animal husbandry, and fishery	Comprehensive industry

tree, and then, an optimal attribute is selected from this j subset for partitioning after random selection. The more machine learners there are, the better the random forest learns. In the study, the method of the weighted mean for regression is adopted in the integrated strategy of random forest, and its formula can be expressed as

$$H(x) = \frac{1}{T}\sum_{i=1}^{T}\omega_i h_i(x). \quad (7)$$

2.2.3. Support Vector Machines. The support vector machine algorithm, first proposed by Vapnik and Vladimir [22], is to maximize the interval among training samples of different categories in the sample space so as to achieve optimal classification. For the nonlinear samples applied in the study, the feature space can be mapped into a higher-dimensional space, and all samples can be correctly classified by a mapping function. The sample space partition in the hyperplane can be expressed by the following linear equation.

$$\omega^T x + b = 0, \quad (8)$$

where ω is the normal vector that determines the direction of the hyperplane, and b is the displacement term that determines the distance between the hyperplane and the origin point. Thus, the distance from any point x in the sample space to the hyperplane (ω, b) can be obtained as follows:

$$r = \frac{\left|\omega^T + b\right|}{\omega}. \quad (9)$$

Assuming that any point $(x_i, y_i) \in D$, the sample space points are classified as

$$\begin{cases} \omega^T x_i + b \geq +1, & y_i = +1, \\ \omega^T x_i + b \leq -1, & y_i = -1. \end{cases} \quad (10)$$

It can be seen from the above formula that in order to find the partition hyperplane with the maximum interval, it is necessary to find the parameters ω and b which satisfy the constraints in (10) so that the sum of the distances from the two heterogeneous support vectors to the hyperplane can be maximized. The constraint conditions can be obtained as

$$\begin{cases} \min\limits_{\omega,b} & \frac{1}{2}\omega^2, \\ \text{s.t.} & y_i\left(\omega^T x_i + b\right) \geq 1, \quad i = 1, 2, \ldots, m. \end{cases} \quad (11)$$

For nonlinear classifiers, the support vector machine has several kernel functions to realize hyperplane partition, including polynomial kernel, Gaussian radial basis kernel, Laplacian kernel, and Sigmoid kernel. These nonlinear kernel functions mainly transform the original feature space into a higher-dimensional feature space and are separated by a hyperplane. In this paper, the Gaussian radial basis is chosen as the kernel function to establish the support vector machine model.

3. Data Preprocessing and Training Model

Predicting the returns of individual stocks is the most important part of the multifactor stock selection strategy, and the alpha of the strategy usually comes from stocks selected. Value of factor of individual stocks is taken as the characteristics of the data (independent variable) and the return rate of individual stocks in the next period as the label of the data (dependent variable). After using the data from t-24 to t-1 period as the model training set, the factor data of individual stocks in the t period are used to predict the return rate of t+1 period. The period of stock portfolio transfer selected in the paper is weekly, so the corresponding training set is the data of 24 weeks from the forecast day to the week 24 weeks before.

3.1. Data Preprocessing. Because the dimensions of each factor are not consistent, it is necessary to standardize the factors so as to compare and regress. Before data standardization, in order to avoid interference caused by the estimation of the correlation between a few extreme value data factors and the rate of return, the extreme data are excluded first.

Figure 1 shows the probability density comparison of factor data of stock market value before and after the de-extreme operation. It can be seen that the de-extreme method effectively reduces the impact of extreme values on the prediction results.

After the market value factor data of stock is de-extreme, the distribution before and after standardization is compared in Figure 2. Figure 2(a) shows the data distribution before standardization, and Figure 2(b) shows the data distribution after standardization. It can be seen that the dimensions of the normalized data are adjusted.

3.2. Model Training. After deleting extremes and standardization of all factor loading data, four algorithms including linear regression, ridge regression, random forest regression, and support vector regression are used, respectively, to predict the returns of individual stocks. In the ridge regression algorithm, the penalty parameter alpha is set to 90. In the random forest regression, 500 trees are selected to test with regression tree as the base learner. In the support vector machine algorithm, radial basis function is used, the radial kernel gamma parameter is set to 0.5, and penalty parameter is set to 100.

There are 40 sample features including the 10 style factors and 30 industry factors in the model. The label of the sample is the return rate of individual stocks in the next cross-sectional period. The sample characteristics and labels of 24 weeks before the prediction are selected as the training set, and rolling prediction is carried out. Finally, the forecasting value of the weekly return of all stocks in the security market from July 9, 2010, to November 15, 2019, is obtained for the construction of the investment portfolio. The data

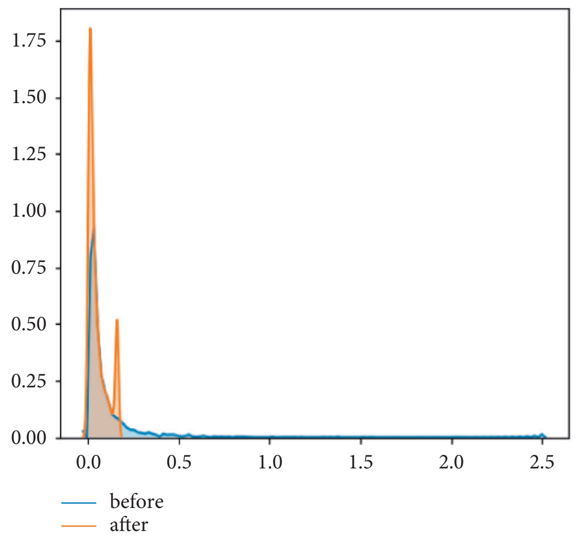

Figure 1: Distribution of samples de-extreme.

come from the Chinese A-stock market, obtained from CSMAR (China Stock Market & Accounting Research Database) and WIND Information Financial Terminal Market Sequence.

4. Prediction Results and Analysis

4.1. Mean Square Error Comparison. Mean square error (MSE) is the sum of squares of the difference between the true value and the predicted value of the test set divided by the number of samples in the test set. In a linear regression model, it refers to the loss function. Generally, the MSE index can intuitively reflect the deviation between the model prediction results and the real results and indicates the generalization ability of the model to the new data in the test set.

$$\frac{1}{m}\sum_{i=1}^{m}\left(y_i-\hat{y}_i\right)^2. \tag{12}$$

Figure 3 shows the MSE statistical results of the four algorithms in each back-test section period.

It can be seen that the MSE indexes of the four algorithms are very close, indicating that for all stocks in the security market, the generalization ability of the four algorithms is close to each other. Compared with the trend of turnover of Shanghai and Shenzhen stock exchanges in Figure 4, the deviation of the model forecast result is greater as the turnover of the market magnifies. The characteristic is in line with the actual situation of the Chinese A-share market, for market sentiment often being hot and retail investors entering the market in a concentrated way when the transaction volume is enlarged, which corresponds to the two highest transaction volumes in Figure 4 of 2015 and early 2019. At these times, irrational investors increase in the market, and market efficiency decreases, which are reflected in the price deviation of individual stocks. In this case, historical data usually cannot accurately predict the future, so the prediction deviation of the corresponding model increases.

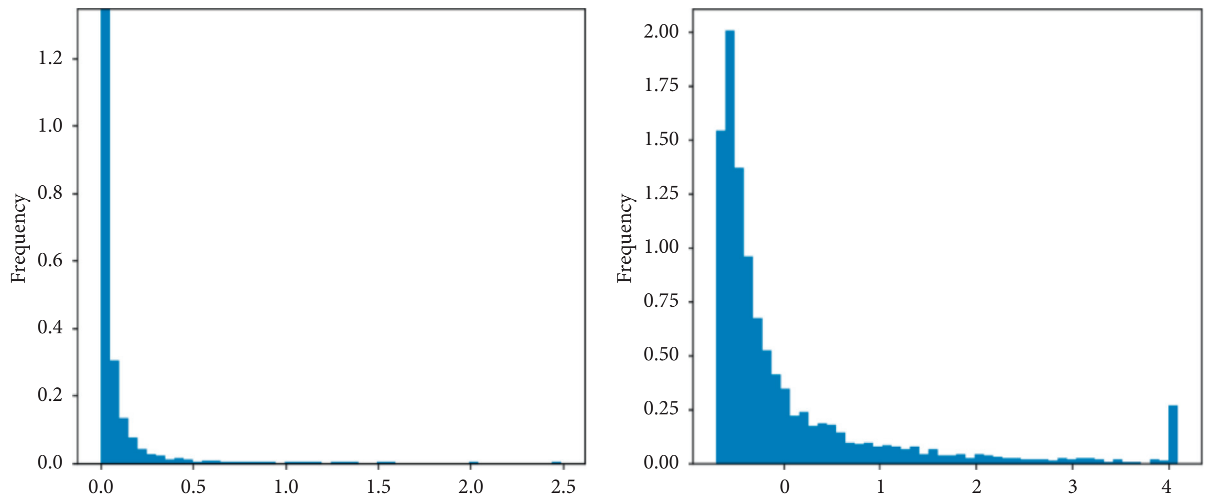

Figure 2: Distribution of samples before (a) and after (b) standardization. Note: the unit of the X-axis in (a) is 1 billion, and the unit of the X-axis in (b) is 1.

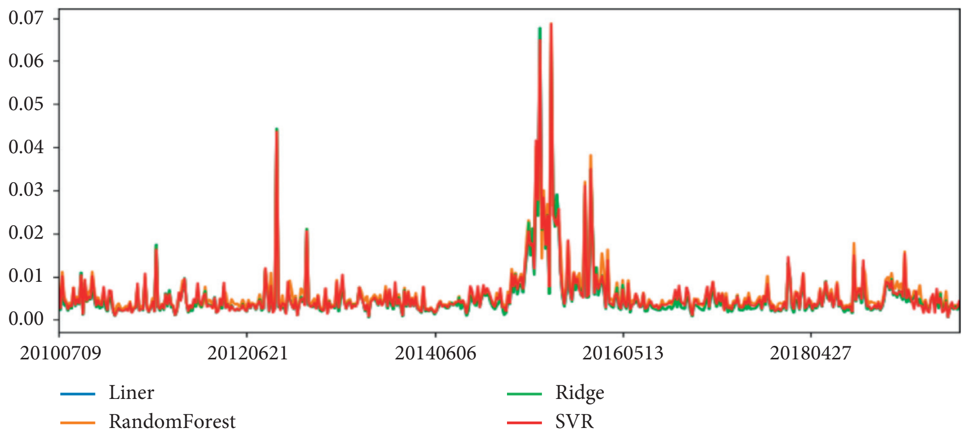

Figure 3: MSE indexes of four algorithms.

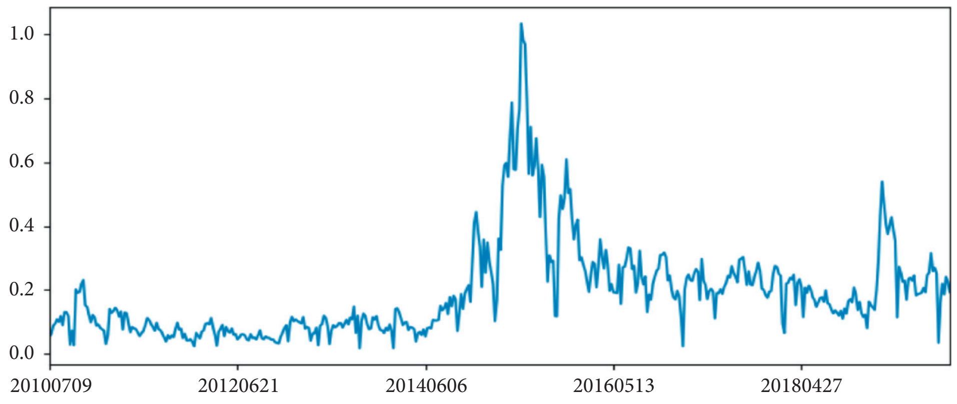

Figure 4: Turnover trend of the Chinese A-share market in the same period.

4.2. Success Rate of Forecast. In the multifactor stock selection model, the deviation between the forecast return and the actual return often cannot completely determine the merits of the strategy. The deviation can be divided into two kinds: one is the actual return of the selected stock being higher than the forecast return, and the other is the actual return of the selected stock being lower than the forecast return. Obviously, the first bias is favorable for investors, while the second bias is an adverse result that should be avoided as far as possible. Therefore, some other indicators are used to help evaluate the model, such as predicting success rate.

The success rate of forecast refers to the probability that the actual return of the stock which the model predicts is positive, that is, the accuracy of the model to predict the rise of the stock. In many cases, the absolute value of the prediction results of the model is not high. For example, although the model predicts a 3% return on individual stocks, the actual stock rise of 2% or 4% is acceptable. Because if the actual return falls in the end, the forecast will cause a loss on the investment. Therefore, the success rate is also an important index of model evaluation.

Figure 5 shows the probability density distributions of the four algorithms for predicting the success rate during the back test.

Except for SVR, the distributions of the other three algorithms all have two peaks, among which linear regression and ridge have the most unstable success rate, with the two peaks close to 0 and 1, respectively, and the peak near 0 is higher. Random forest is slightly better than the two algorithms, but many extreme values of the prediction of the success rate still exist. Therefore, from the perspective of prediction success rate, SVR is the most stable, followed by random forest regression, while ridge regression and traditional linear regression are very unstable and have many extreme values as shown in Table 3.

4.3. Model Comparison and Analysis. Figure 6 shows the net value curves of the investment portfolio constructed using the corresponding earnings forecast results of the four algorithms. Liner regression corresponds to linear regression, random forest corresponds to random forest regression, ridge corresponds to ridge regression, SVR corresponds to support vector regression, and benchmark corresponds to CSI 500 index trend.

The back-test results show the following.

4.3.1. SVR Is Superior to the Traditional Linear Regression Algorithm. Compared with the traditional linear regression, the return of the portfolio constructed by the SVR is significantly improved from the perspective of return rate and retracement control. The traditional linear regression is not suitable for high-dimensional data, the number of independent variables of high-dimensional data is greater than the sample size, and the rank of matrix X is less than the number of rows, which will lead to the matrix X is not full rank, and the unique solution cannot be obtained.

In addition, even if there is no problem of high-dimensional data, approximate (incomplete) multicollinearity which means the high correlation between characteristic variables often appears in the traditional linear regression model. The matrix becomes almost irreversible under multicollinearity, magnifying the variance and underestimating the significance of OLS estimation.

The back-test results show that SVR has a better prediction effect on the return of stocks than linear regression, and the algorithm fits the characteristics of higher-degree polynomial and is more suitable for the stock market. For example, it can be seen from the results of the stratified back test in the primary factor chosen part that the style factors with better effects in the model have fluctuated to a certain degree since 2018, such as factors of market value and turnover rate. In 2017, China's stock market saw a record number of IPOs, and the regulator cracked down on high increasing the number of common shares and other subject speculation, and there was a big shift in market style such as the market's small-cap effect changed significantly. For this kind of nonlinear behavior, the machine learning algorithm is relatively well adapted. Compared with the CSI small-cap 500 index of the Chinese stock market, Sharpe [23] ratio calculated is 0.27, which means at the same risk, the portfolio gains more than CSI 500.

However, the result of SVR has got a large retracement since 2018, which may be caused by the increasing use of machine learning algorithms by quantitative institutions in China's A-share market. That is, the increase of funds in the market for return prediction using the SVR algorithm reduces the alpha of the algorithm itself.

4.3.2. The Results of Linear Regression Are Similar. The back-test results show that the trend of ridge regression is very close to that of linear regression, which is determined by the two algorithms themselves. Ridge regression only adds a penalty term to the linear regression; in this study, the penalty term is rather big which leads to similar results. Although ridge regression and linear regression algorithms had relatively high returns before 2018, they began to plunge after 2018, which reflecting the distribution of the success rate of linear regression prediction was unstable. Analysis of the forecast results shows the successful rate distributions of ridge regression and linear regression are extreme, and Xu et al. [24] find that stocks with high return asymmetry exhibit low expected returns. In the market, if the forecast successful rate is unstable, it will have a negative impact on the net value.

The advantage of ridge regression over classical linear regression models lies in its tradeoff between prediction error and variance. With the increase of λ, the smoothness of the fitting of ridge regression decreases, although the variance decreases, but the deviation increases. In general, when the relationship between the response variable and the prediction variable is approximately linear, the least squares estimate will have a low bias but a large variance, which means that small changes in the training data may lead to large changes in the least squares regression coefficient. When the number of variables and the number of observations are close, the variance of the least squares estimation will be larger, and when the number of variables is greater than the number of observations, the least squares have no unique solution. Ridge regression method can still get a large decrease of variance by a small increase of deviation, and a better fitting effect can be obtained by using this tradeoff. The back-test results show that the ridge regression fitting trend effect is very close to that of the ordinary linear regression,

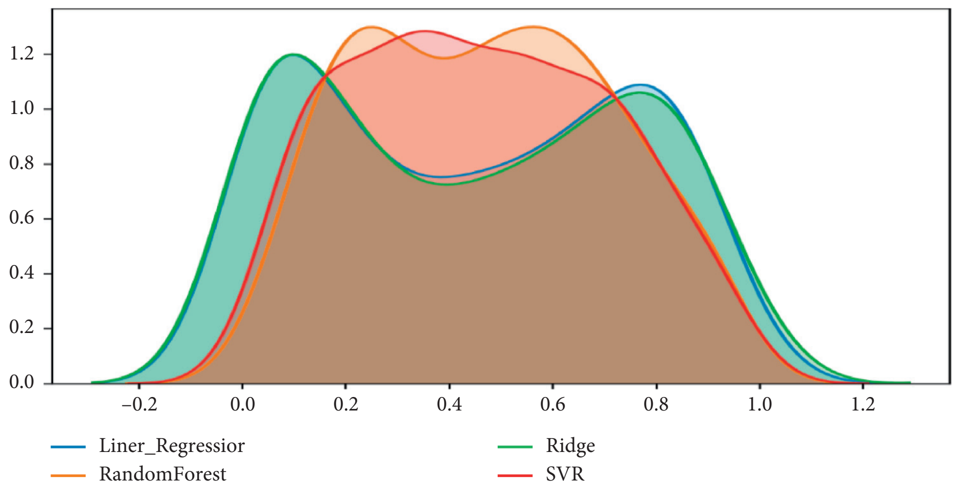

Figure 5: Distributions of the success rate of forecast earnings.

Table 3: Back-test results of different algorithms.

Algorithm	Annualized return	Maximum retracement rate (%)	Sharpe ratio weekly
Linear regression	−5.38%	94.73	——
Ridge regression	−6.61	95.56	——
Random forest regression	−4.74	83.43	——
Support vector regression (SVR)	70.12%	58.63	0.27

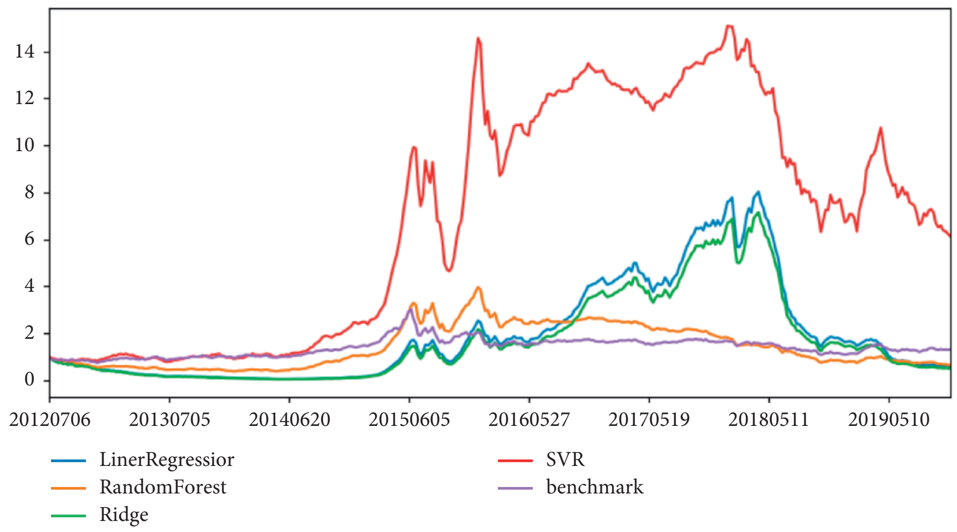

Figure 6: Net value curves of back test.

which also reflects from another side that the variance of the least square estimation is not large.

4.3.3. Stochastic Forest Regression Is Insignificant in the Sample. Random forest only uses some node variables in the decision tree. Because different nodes are forced to split with different variables, the correlation between different decision trees can be reduced, thus reducing the variance. Therefore, in the tradeoff between variance and bias, random forest sacrifices a small amount of bias for a smaller variance, so as to reduce the mean square error. Since all characteristic variables are used for splitting in this study, even though the deviation is small, the correlation between different decision trees is strong, resulting in a large variance. So, the portfolio constructed by the stochastic forest regression algorithm does not generate significant excess returns, and this algorithm has no obvious advantage over the traditional linear regression algorithm in the construction of the multifactor stock selection model.

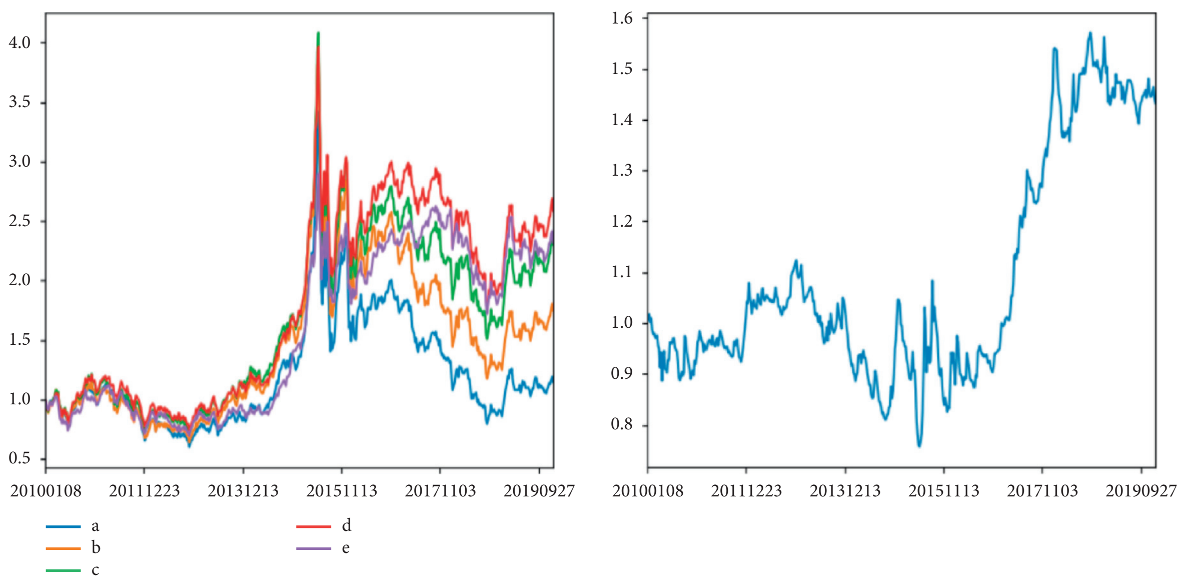

Figure 7: Stratified back test of price/earnings factor (a) and net value curve of long-short portfolio (b).

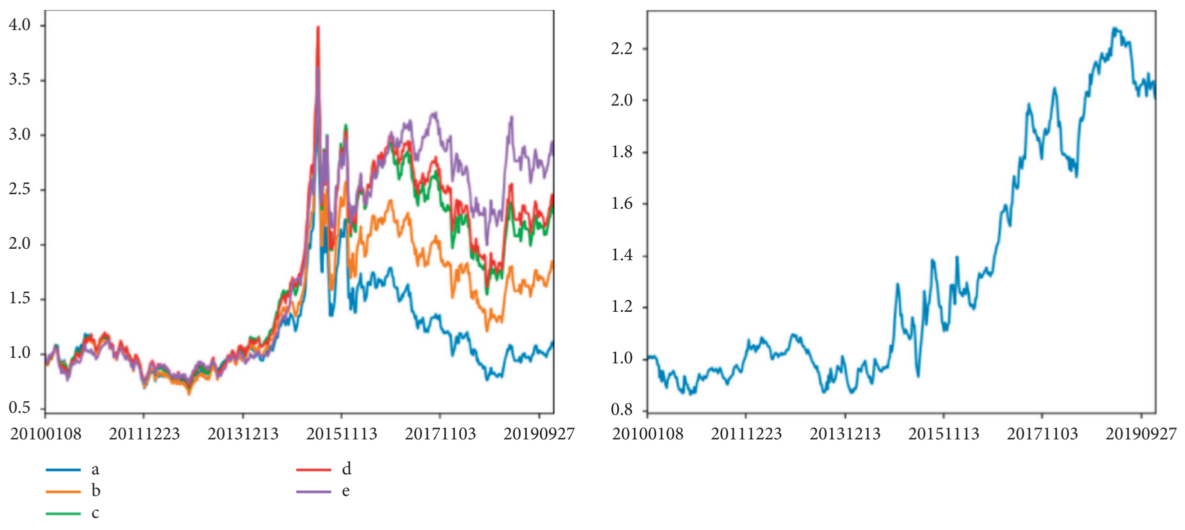

Figure 8: Stratified back test of price/book value factor (a) and net value curve of long-short portfolio (b).

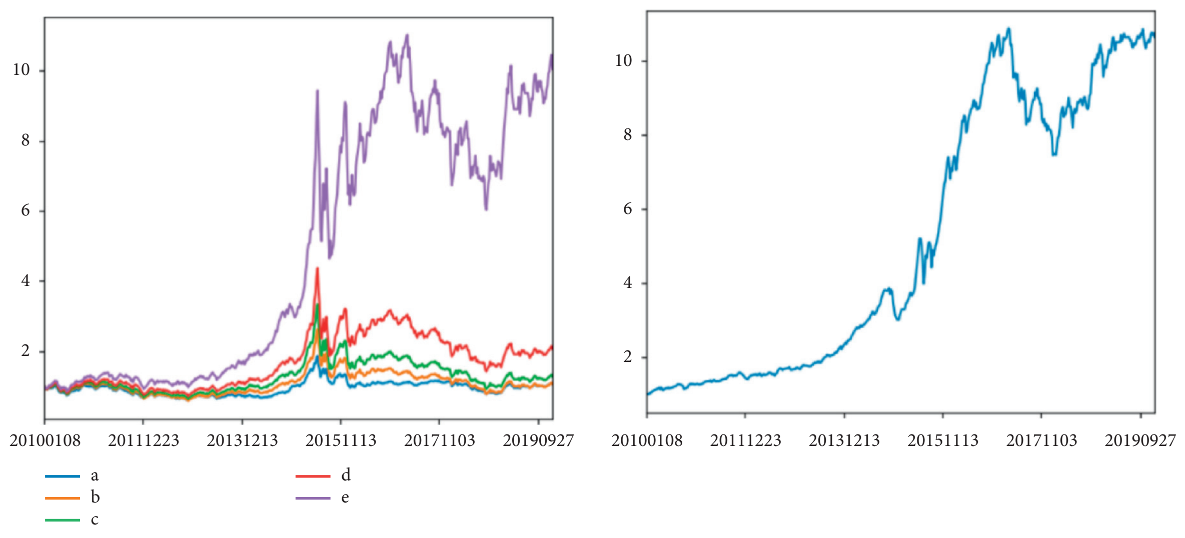

Figure 9: Stratified back test of market value factor (a) and net value curve of long-short portfolio (b).

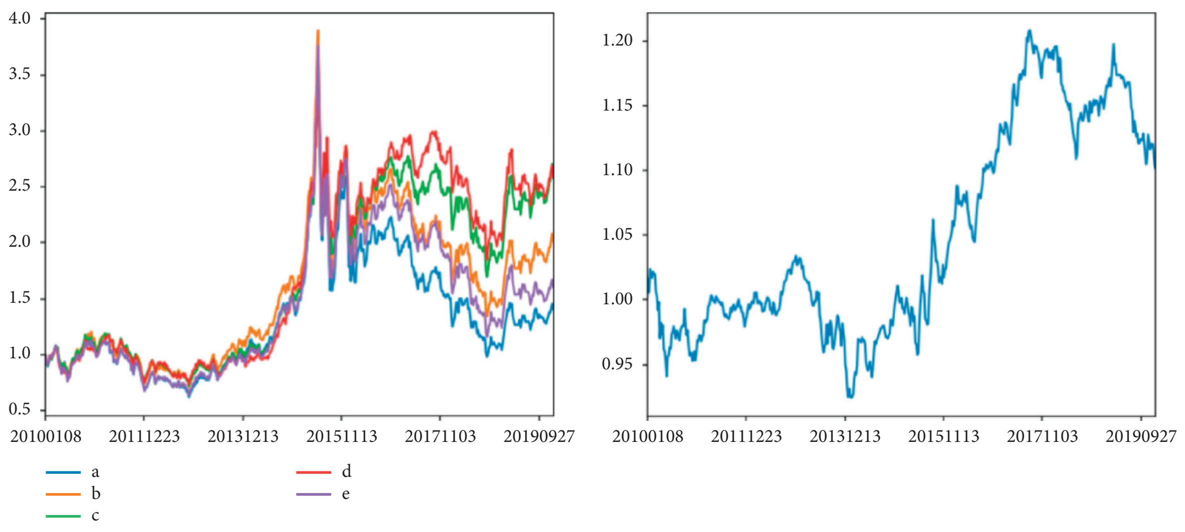

Figure 10: Stratified back test of price/cash flow factor (a) and net value curve of long-short portfolio (b).

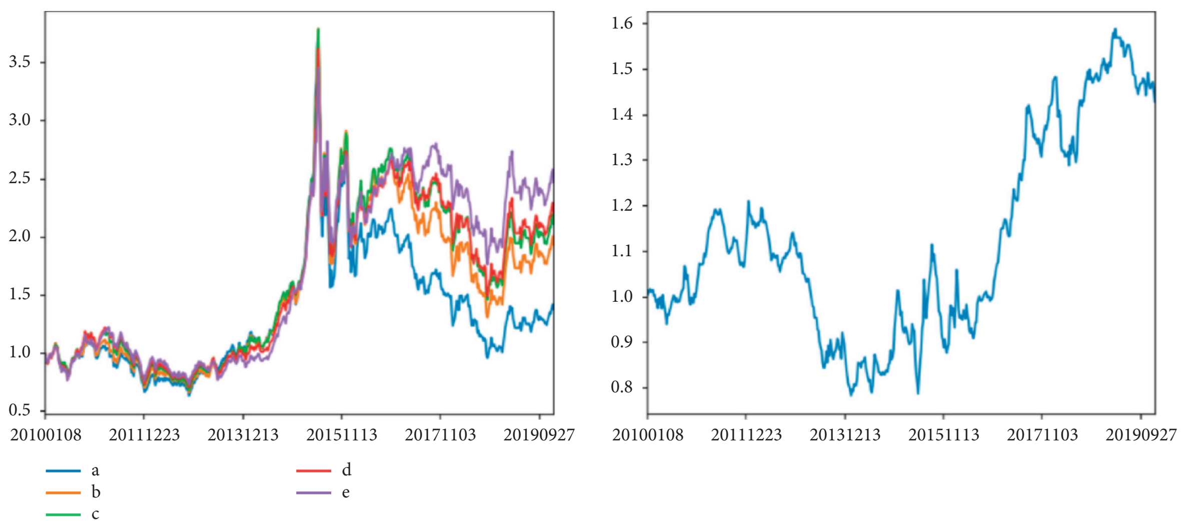

Figure 11: Stratified back test of price/sales factor (a) and net value curve of long-short portfolio (b).

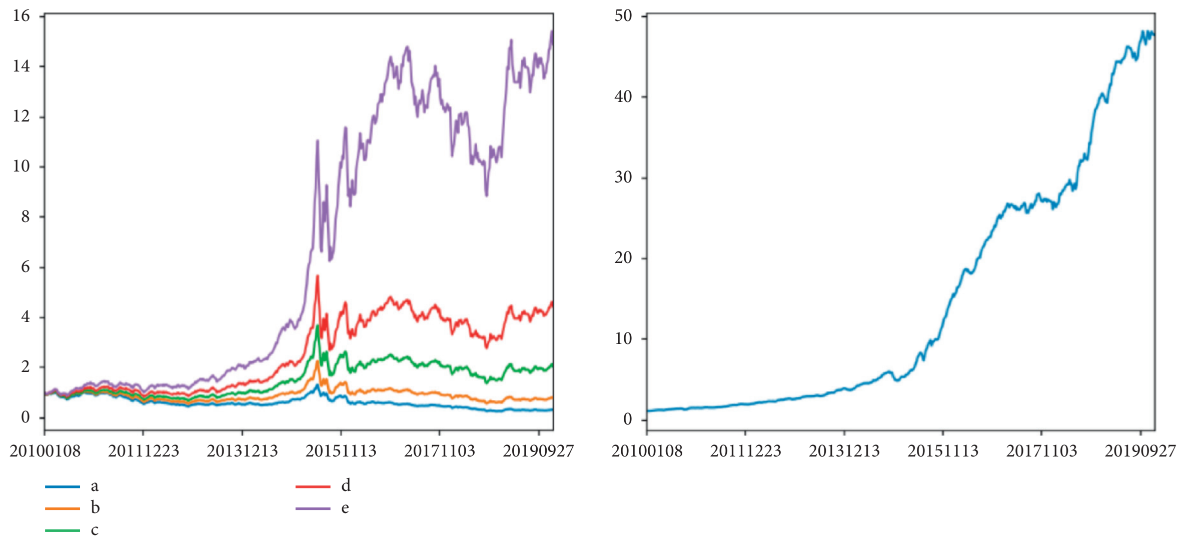

Figure 12: Stratified back test of turnover factor (a) and net value curve of long-short portfolio (b).

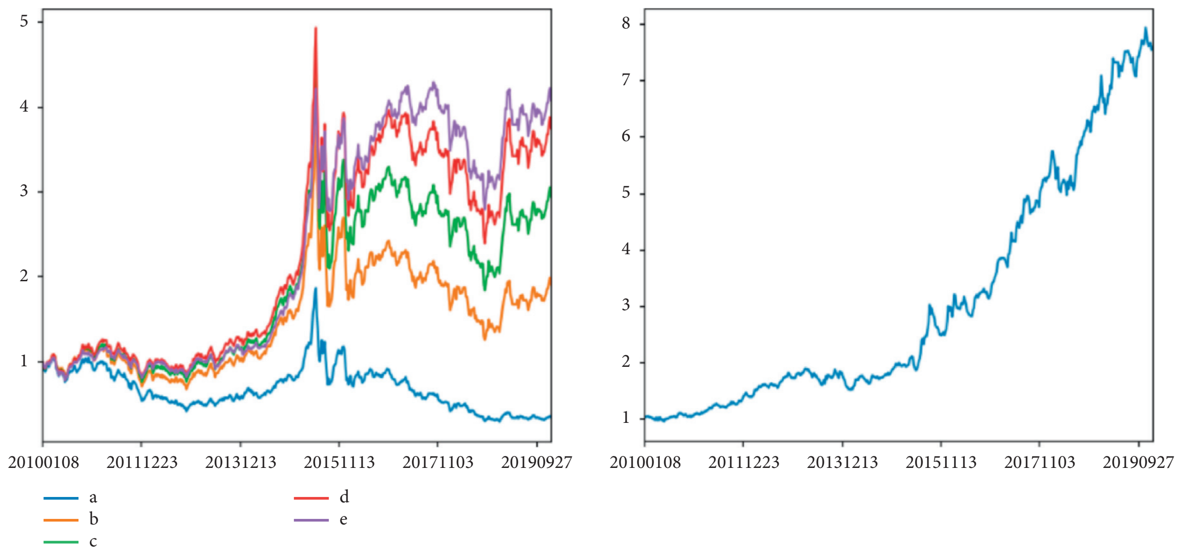

Figure 13: Stratified back test of turnover rate factor (a) and net value curve of long-short portfolio (b).

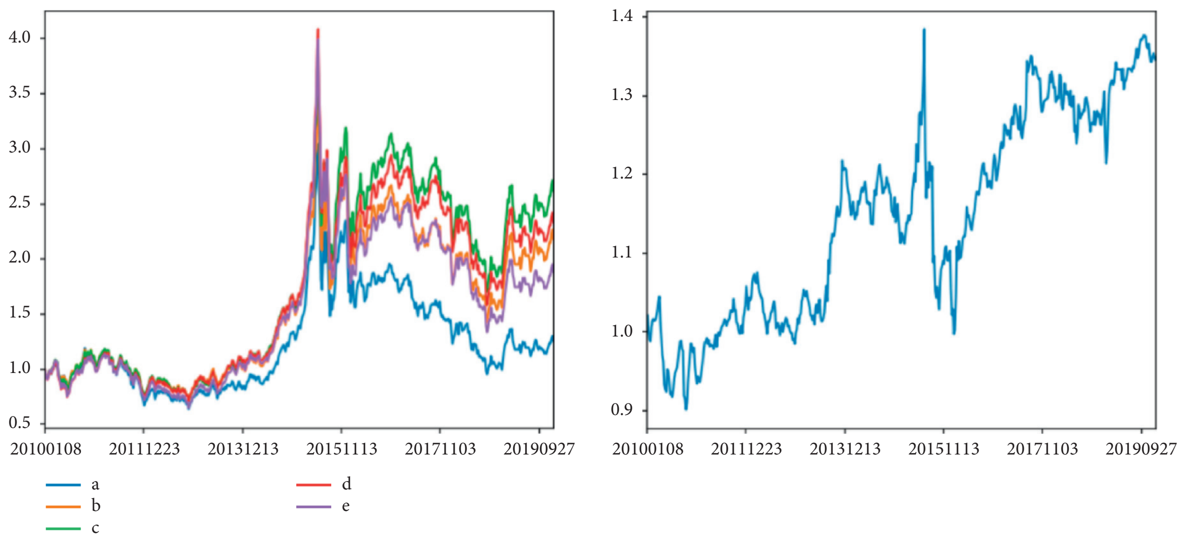

Figure 14: Stratified back test of length of wick factor (a) and net value curve of long-short portfolio (b).

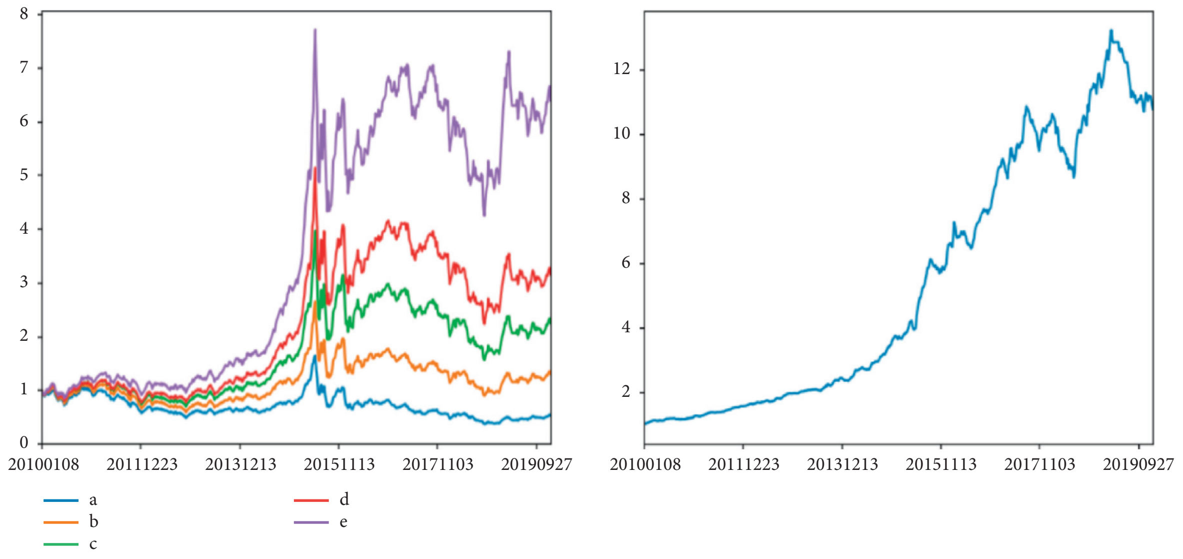

Figure 15: Stratified back test of closing price factor (a) and net value curve of long-short portfolio (b).

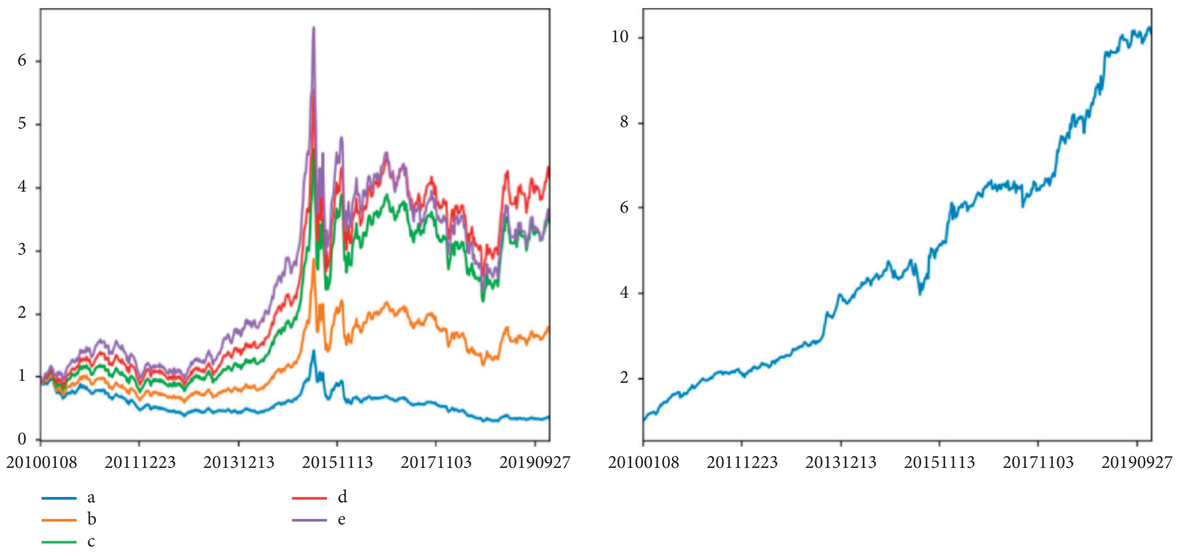

Figure 16: Stratified back test of yield factor (a) and net value curve of long-short portfolio (b).

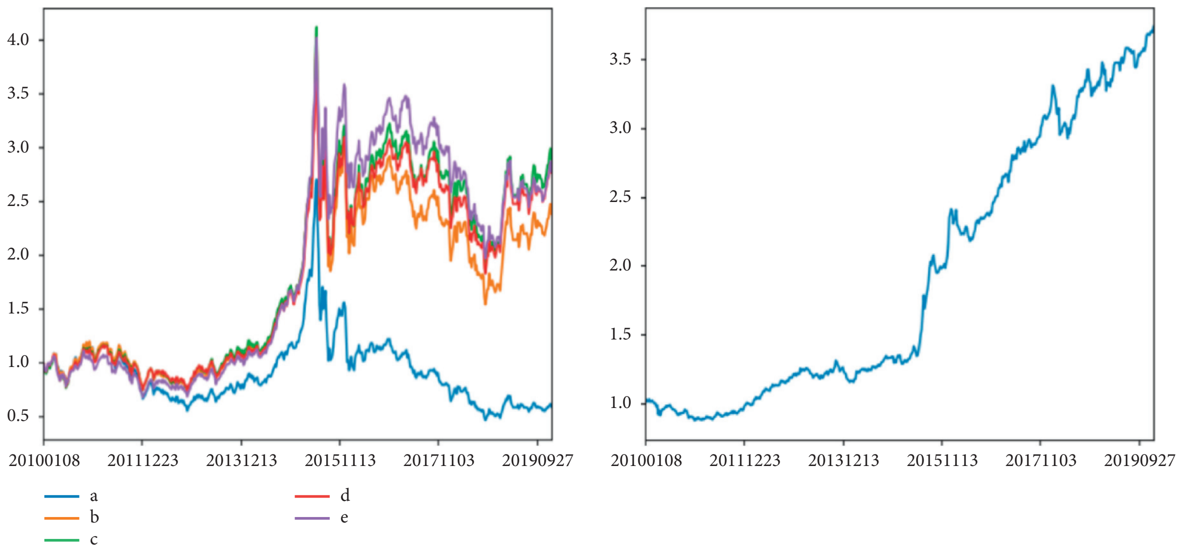

Figure 17: Stratified back test of length of upper shadow factor (a) and net value curve of long-short portfolio (b).

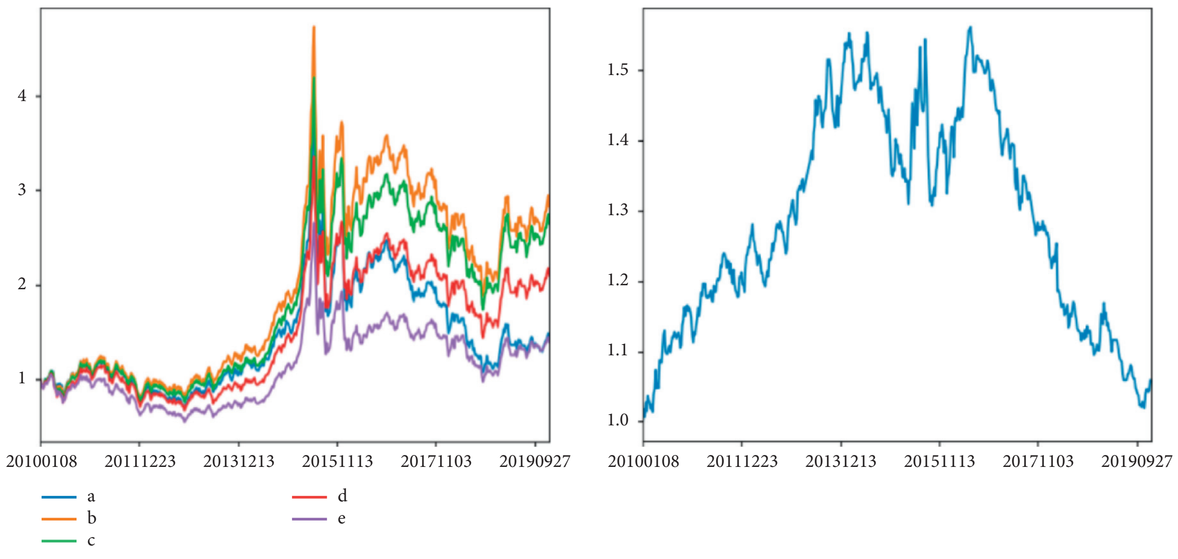

Figure 18: Stratified back test of length of lower shadow factor (a) and net value curve of long-short portfolio (b).

5. Conclusion and Prospect

5.1. Conclusion. This study aims to study the application of machine learning algorithm in multifactor selection strategy.

First, according to results of the stratified back-test and long-short portfolio, the price-to-sales ratio and price-to-cash-flow ratio factors which are not strongly correlated with the return rate of stocks are removed. The remaining 10 style factors and 30 industry factors constitute the independent variables in the return forecast model.

Next, four algorithms, including linear regression and three machine learning regression, are used, respectively, to predict the return of stocks. The deviation of the forecast results increases when the stock market turnover is enlarged, indicating that the prediction effect of the model will weaken when the market sentiment is high and the irrational pricing of investors increases. Among the four algorithms, the support vector regression has the most stable successful rate for predicting stocks return, while the ridge regression algorithm and linear regression algorithm have the most unstable successful rate for predicting with more extreme cases.

Finally, portfolios are built of which the position weight of individual stocks is determined by the weighted average of expected returns, and the forecast results of the four algorithms are back-tested. It is found that the support vector regression has a significant improvement compared with the traditional linear regression, both in terms of return and retracement control. The result can significantly outperform the CSI 500 index, indicating that the support vector regression algorithm in machine learning has a better effect in predicting returns in the multifactor stock selection strategy.

In conclusion, support vector regression can be used to fit higher-degree polynomials in the Chinese A-share market, and its applicability is strong.

5.2. Prospect. Multifactor stock selection model can help investors not only to make more efficient and accurate decisions in investment but also have a clearer understanding of the huge and intricate security market and price fluctuation to seize trade opportunity. With the advent of the Internet of things, the acceleration of the development of big data and cloud computing, and the continuous innovation of data mining algorithms, more and more reasonable algorithms will be explored and used in the Chinese equity market to gain an excess return.

From the perspective of the data relationship, the nonlinear relationship between the prediction excess return rate variable and the anomaly factor maybe not very strong, which leads to the prediction effect of some machine learning algorithms used in this study is not as good as that of the traditional linear regression model. On the other hand, it is also limited by the problems of the algorithm itself. Although the ridge regression model can solve the $X'X$ irreversible problem in the linear regression model, the cost paid is to "compress" the regression coefficients, thereby making the model more stable and reliable. Since the penalty term is the quadratic function of the regression coefficient β, when seeking the minimum value of the objective function, its partial derivative always retains the independent variable itself. So, sometimes ridge regression cannot realize the choice of variables in a true sense. Although the results of SVM model performance are sometimes excellent, its biggest disadvantage is that when the data scale is large, the operating cost is relatively high. Therefore, future work can be further studied from the following two aspects: if the real data do have sparse problems, LASSO regression can be considered to achieve better results; and if the correlation between decision trees is strong, the prediction accuracy can be further improved by using AdaBoost algorithm.

Appendix

Figure interpretation

Figures 7–18 show the results of stratified back test of primary factors and net value curve of long-short portfolio, which is used for factor screening.

Python code

The models are implemented in Python 3.7.3, and back-test part program code is listed below.

```
import pandas as pd
import matplotlib.pyplot as plt
import math
import numpy as np
import os
def get_hundred (arr):
  temp = []
  for i in arr:
    if i == 0:
      temp.append( 0)
    else:
      temp.append(math.floor(i / 100) * 100)
  return temp
def get_net_value(day):
  '''
  get net value
:   param day: back test date
:   return: net value
  '''
  global position
  nv = money
  for symbol, quantity in position.items():
    if pd.isnull(df_close[symbol][day]):
      print('stock without price : {} , use the nearest
price '.format(symbol))
      nv = nv + price[symbol] * quantity
      continue
    else :
        nv = nv + df_close[symbol][day] * quantity
        price[symbol] = df_close[symbol][day]
    return nv
def judge(day, next_day):
    “‘
: param day: back test date
: return: record word of position change : change_quantity
    “‘
    global change_quantity
    new_nv = get_net_value(day)
    df_judge = pd.DataFrame({‘new_weight’: df_weight [next_day]},
index=df_weight.index)
    df_judge[ ‘new_symbol_mv’] = df_judge[‘new_-weight’] * new_nv
    df_judge[‘price’] = df_close.loc[day, : ]
    df_judge = df_judge[df_judge[ ‘price’] > 0]
    df_judge[ ‘new_symbol_quantity’] = df_judge [‘new_symbol_mv’] / (df_judge[‘price’] * ((1000 + trade_fee) / 1000))
    df_judge = df_judge.fillna( 0)
    # df_judge = df_judge.sort_va-lues(by=‘new_symbol_mv’, ascending=False)
    df_judge[‘new_symbol_quantity’] = get_hun-dred(df_judge[‘new_symbol_quantity’])
    df_old_quantity = pd.DataFrame([position], index= [‘old_quantity’])
    df_old_quantity = df_old_quantity.T
    df_judge[ ‘old_quantity’] = df_old_quantity [‘old_quantity’]
    df_judge = df_judge.fillna( 0)
    df_judge[ ‘change_quantity’] = df_judge[‘new_s-ymbol_quantity’] -df_judge[‘old_quantity’]
    change_ = df_judge[ ‘change_quantity’][df_judge [‘change_quantity’] != 0]
    return change_
def buy(symbol, quantity, cost):
    global position, money
    if symbol in position:
        position[symbol] = position[symbol] + quantity
    else:
        position[symbol] = quantity
    money = money - cost
def sell(symbol, quantity, cost):
    global position, money
    position[symbol] = position[symbol] - quantity
    if position[symbol] == 0:
        del position[symbol]
    money = money + cost
def trade(day):
    “‘
    trade according to position change
    :param day: back test date
    :return:
    “‘
    stop_trade = []
    for symbol in change_quantity.index:
        if df_close[symbol][day] == 0:
            print(‘{}suspended , can not trade’.-format(symbol, day))
            stop_trade.append(symbol)
    for symbol in change_quantity.index: # sale
        if change_quantity[symbol] < 0:
            if symbol in stop_trade: # judge if suspended
                print(‘{}suspended , can not sell’.-format(symbol, day))
                continue
            if df_ret[symbol][day] > -0.09: # judge if limit down
                if position[symbol] >= abs(change_quantity [symbol]): # judge if amount of stocks arrive to the number to sell
                    trade_money = abs(change_quantity [symbol]) * df_close[symbol][day] * (1000 - trade_fee) / 1000
                    sell(symbol=symbol, quantity-=abs(change_quantity[symbol]), cost=trade_money)
                else: # see all
                    trade_money = position[symbol] * df_close[symbol][day] * (1000 - trade_fee) / 1000
                    print(‘{}number not enough{} , sell all {}’.format(day, symbol, abs(change_quantity[symbol]), position[symbol]))
                    sell( symbol=symbol, quantity=position [symbol], cost=trade_money)
            else:
                print(‘{}日,{}litmit down, can not sell’.-format(day, symbol))
                continue
    else :
        continue
for symbol in change_quantity.index: # buy
    if change_quantity[symbol] > 0:
        if symbol in stop_trade: # judge if suspended
            print(‘{}suspende , can not buy’.-format(symbol, day))
            continue
        if money < 10000: # cash below 10000 give up buying
            print('{} ash below 10000,give up buying {}'.format(day, symbol))
            continue
        if df_ret[symbol][day] < 0.09: # if Limit Up
            trade_money = abs(change_quantity[symbol]) ∗ df_close[symbol][day] ∗ (1000 + trade_fee) / 1000
            if money > trade_money: # cash enough
                trade_money = change_quantity[symbol] ∗ df_close[symbol][day] ∗ (1000 + trade_fee) / 1000
                buy(symbol, change_quantity[symbol], trade_money)
            else: # cash not enough, buy as can
                trade_quantity = 100 ∗ math.floor(money / (df_close[symbol][day] ∗ 100 ∗ ((1000 + trade_fee) / 1000)))
                if trade_quantity > 0:
                    trade_money = trade_quantity ∗ df_close [symbol][day] ∗ ( 1000 + trade_fee) / 1000
                    buy(symbol, trade_quantity, trade_money)
                    print('cash is not enough，{}日{}buy as it can{}'.format(day, symbol, trade_quantity))
                if trade_quantity < 0:
                    print('error，{}日{}buy number is negative'.format(day, symbol))
        else: # skip if limit down
            print('{}日,{}limit up,can not buy'.format(day, symbol))
            continue
        else :
            continue
def get_account(day):
    '''
    get the account information for program debugging
:   param day:
:   return: account information of today, back csv file
    '''
    symbol_list = []
    quantity_list = []
    price_list = []
    mv_list = []
    for symbol, quantity in position.items():
        symbol_list.append(symbol)
        quantity_list.append(quantity)
        price_list.append(price[symbol])
        mv_list.append(quantity ∗ price[symbol])
    df_account = pd.DataFrame({ 'symbol': symbol_list, 'quantity': quantity_list, 'price': price_list, 'mv': mv_list})
    df_account[ 'money'] = money df_close = pd.read_csv # matrix of closing price
df_close[df_close.columns[0]] = list(map(lambda x: str(x),
df_close[df_close.columns[0]]))
df_close = df_close.set_index(df_close.columns[ 0])
df_ret = pd.read_csv
df_ret[df_ret.columns[ 0]] = list(map(lambda x: str(x), df_ret[df_ret.columns[0]]))
df_ret = df_ret.set_index(df_ret.columns[ 0])
df_sum = pd.DataFrame()
df_change_rate = pd.DataFrame()
model_name = os.listdir
for m_name in model_name:
        bug_record = []
        position = {}
        price = {}
        money = 500_0000
        origin_money = 500_0000
        net_value = []
        change_rate = []
        long_short = []
        money_record = []
        trade_fee = 9
        df_weight = pd.read_csv.format(m_name))
        df_weight = df_weight.set_index(df_weight.columns[0])
        start_day = '20120706'
        all_time_interval = list(df_weight.columns)
back_test_interval = all_time_interval[all_time_interval.index(start_day): -1]
        for date in back_test_interval:
            next_date = list(df_weight.columns)[1 + list(df_weight.columns).index(date)]
            change_quantity = judge(date, next_date)
            trade(date)
            # get_account(date)
            net_value.append(get_net_value(date) / origin_money)
            if date == start_day:
                print('{}net value change{}%'.format(date, 0))
                change_rate.append( 0)
                money_record.append(money)
            else:
```

```
            # print((get_net_value(date) / 1_000_000_000 - net_value[-2]) * 100 / net_value[-1])
            print('{}{}net value{}'.format(m_name.split('_')[1], date, round(net_value[-1], 6)))
            print('{}{}net value{}'.format(m_name.split('_')[1], date, round(net_value[-1] / net_value[-2] - 1, 6)))
            money_record.append(money)
            change_rate.append( round(net_value[-1] / net_value[-2] - 1, 6))
        df_result = pd.DataFrame({ 'net_value': net_value, 'change_rate': change_rate},
        index=back_test_interval)
        df_sum[ '{}'.format(m_name[:-4])] = df_result ['net_value']
        df_change_rate[ '{}'.format(m_name[:-4])] = df_result['change_rate']
        df_result.to_csv.format(m_name))
    df_sum[ 'index'] = df_result.index
    df_sum = df_sum.set_index( 'index')
    df_sum.plot.line()
    plt.show()
    df_change_rate[ 'index'] = df_change_rate.index
    df_change_rate = df_change_rate.set_index( 'index')
    df_change_rate.plot.line()
    plt.show().
```

Data Availability

The data generated and analyzed in this manuscript are available from the CSMAR (China Stock Market & Accounting Research Database) and WIND Information Financial Terminal Market Sequence.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the "Great Wall Scholar Training Program" for the Construction of High-level Teachers in Universities of Beijing and the Planning Project of Beijing Social Science Fund (grant no. 20SRB010) and Cultivation of Major Research Projects of CUEB: Study on the mechanism, measurement, early warning and prevention of systemic financial risk contagion in China.

References

[1] F. Eugene and K. French, "The cross-section of expected stock returns," *The Journal of Finance*, vol. 47, no. 2, pp. 427–465, 1992.

[2] M. M. Carhart, "On persistence in mutual fund performance," *The Journal of Finance*, vol. 52, no. 1, pp. 57–82, 1997.

[3] E. F. Fama and J. D. MacBeth, "Risk, return, and equilibrium: empirical tests," *Journal of Political Economy*, vol. 81, no. 3, pp. 607–636, 1973.

[4] L. P. Hansen, "Large sample properties of generalized method of moments estimators," *Econometrica*, vol. 50, no. 4, pp. 1029–1054, 1982.

[5] J. Green, J. R. M. Hand, and X. F. Zhang, "The characteristics that provide independent information about average U.S. Monthly stock returns," *Review of Financial Studies*, vol. 30, no. 12, pp. 4389–4436, 2017.

[6] K. Hou, C. Xue, and L. Zhang, "Replicating anomalies," *Review of Financial Studies*, vol. 33, no. 5, pp. 2019–2133, 2020.

[7] S. Mullainathan and J. Spiess, "Machine learning: an applied econometric approach," *The Journal of Economic Perspectives*, vol. 31, no. 2, pp. 87–106, 2017.

[8] J. Kleinberg, H. Lakkaraju, J. Leskovec, J. Ludwig, and S. Mullainathan, "Human decisions and machine predictions," *Quarterly Journal of Economics*, vol. 133, no. 1, pp. 237–293, 2018.

[9] D. W. Liu, "Modeling and forecasting stock price Index based on support vector machine," *Statistics and Decision making*, vol. 2, p. 3, 2013.

[10] S. Y. Wang, Z. F. Cao, and M. Z. Chen, "Research on application of random forests in the quantitative stock selection model," *Operations Research and Management Science*, vol. 3, pp. 163–168, 2016.

[11] H. L. Xie and H. U. Di, "The application of multi-factor quantization model in portfolio: the comparative research on LASSO and elastic net," *Statistics & Information Forum*, vol. 32, no. 10, pp. 36–42, 2017.

[12] S. Gu, B. T. Kelly, and D. Xiu, *Empirical Asset Pricing via Machine Learning (No. w25398)*, National Bureau of Economic Research, Cambridge, MA, USA, 2018.

[13] L. Wang and L. Li, "Multi-factor quantitative stock selection strategy based on gcForest," *Computer Engineering and Applications*, vol. 56, no. 15, pp. 86–91, 2020.

[14] D. E. Rapach, J. K. Strauss, and G. Zhou, "Out-of-sample equity premium prediction: combination forecasts and links to the real economy," *Review of Financial Studies*, vol. 23, no. 2, pp. 821–862, 2010.

[15] J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer open, New York, NY, USA, 2017.

[16] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[17] D. J. Henderson, C. Papageorgiou, and C. F. Parmeter, "Growth empirics without parameters," *The Economic Journal*, vol. 122, no. 559, pp. 125–154, 2011.

[18] A. E. Hoerl and R. W. Kennard, "Ridge regression: biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[19] D. M. McNeish and M. Daniel, "Using lasso for predictor selection and to assuage overfitting: a method long overlooked in behavioral sciences," *Multivariate Behavioral Research*, vol. 50, no. 5, pp. 471–484, 2015.

[20] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[21] G. C. Cawley and N. L. C. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," *Journal of Machine Learning Research*, vol. 11, no. 1, pp. 2079–2107, 2010.

[22] V. N. Vapnik, *The Nature of Statistical Learning Theory*-Springer, New York, NY, USA, 1995.

[23] W. F. Sharpe, “Capital asset prices: a theory OF market equilibrium under conditions OF risk∗,” *The Journal of Finance*, vol. 19, no. 3, pp. 425–442, 1964.

[24] Z. Xu, T. Chevapatrakul, and X. Li, “Return asymmetry and the cross section of stock returns,” *Journal of International Money and Finance*, vol. 97, no. OCT, pp. 93–110, 2019.