Overview of Machine Learning for Stock Selection Based on Multi-Factor Models

Haoxuan Li [1st,1,a*,] Xueyan Zhang [2nd,2,b,] Ziyan Li [3rd,3,c,] Chunyuan Zheng [4th,4,d]

[1]College of Mathematics, Sichuan University, Chengdu, China
[2]College of Mathematics, Sichuan University, Chengdu, China
[3]College of Mathematics, Sichuan University, Chengdu, China
[4]College of Mathematics, Sichuan University, Chengdu, China

Abstract—In recent years, many scholars have used different methods to predict and select stocks. Empirical studies have shown that in multi-factor models, machine learning algorithms perform better on stock selection than traditional statistical methods. This article selects six classic machine learning algorithms, and takes the CSI 500 component stocks as an example, using 19 factors to select stocks. In this article, we introduce four of these algorithms in detail and apply them to select stocks. Finally, we back-test six machine learning algorithms, list the data, analyze the performance of each algorithm, and put forward some ideas on the direction of machine learning algorithm improvement.

1 INTRODUCTION

There are many mature algorithms for stock selection that have been widely used by quantitative investment companies. At present, strategies such as fundamental analysis [1], qualitative analysis [2], and value investing [3] have been used by many stock investors, and have obtained considerable abnormal return.

In this article, we first obtain relevant data of the CSI500 component stocks and the ETF500 index from January 1, 2015 to December 31, 2019 through WIND information. We select 19 candidate factors for machine learning algorithms and divide these factors into six categories. We expect that a quantitative strategy to adjust stock portfolios every three months can be given based on the multi-factor model. Note that the data may not be complete, partly because the data was extracted too early so that the stock was not listed at that time. As a result, we ignore this part of these stocks within the relevant data missing time, and do not consider using these stocks to invest in the next time period.

This article takes the first time adjustment as an example, using the yield and factor value on March 31, 2015, excluding stocks with non-existent yield and partially blank factor values, and then evaluates the performance of the remaining 382 stocks. We use different machine learning algorithms such as linear regression, adaboost, support vector machine, random forest, gradient boosting and XGBoost for stock selection, and then calculate the abnormal returns separately. Finally, we use python to simulate different strategies to obtain data, evaluating the performance of different strategies and propose a quantitative strategy based on machine learning multi-factor models for Chinese low and medium market capitalization stocks, together with some ideas on the direction of machine learning algorithm improvement.

2 LINEAR REGRESSION MODEL

2.1 Data Selection and Preprocessing

Above all, we list 19 factors which would be used in this linear regression model and machine learning algorithms mentioned later:

TABLE I. CANDIDATE FACTORS LIST AND CLASSIFICATION

Valuation factor	pe; pb; ps; pcf_ncf
Growth factor	yoyop; yoyprofit; yoy_or; growth_roe
Profit factor	roe; roa; wgsd_ocftosales; dividendyield2
Liquidity factor	val_lnfloatmv; tech_turnoverrate10
Momentum factor	tech_revs60; tech_revs120
Leverage factor	debttoassets; fa_equityassetradio; fa_fixedassettoasset

Due to the difference in the numerical range of different factors and the unit of measurement, we normalize the same factor for different stocks on March 31, 2015: $Z_i = \frac{X_i - \bar{X}}{S}$, where $\bar{X}$ denotes sample mean and S denotes sample standard deviation. For writing convenience, we still take the normalized factor value as X_i . At the same time, let $(\vec{x}_i, y_i)$ be the vector of factors' value and the corresponding return of the next period.

* Corresponding author: [a]lihaoxuan_2002@126.com
[b]zhxy_1998@163.com, [c]liziyan99@163.com, [d]chunyuanzheng1999@163.com

2.2 Linear Regression Model

Above all, perform a linear regression between each factor and next period's yield to obtain the correlation coefficient, then test the level of each coefficient. We use SPSS to get the t values and p values, and remove the irrelevant factors. As a result, there are ten factors remaining. Next, we test the correlation between the ten remaining factors and find that some of them are significantly correlated. In order to eliminate the problem that the mean square error (MSE) of the linear regression model caused by multicollinearity is too large which will cause a poor performance of this model, we choose to use *Stepwise Regression Method* to filter the remaining factors to obtain a better model.

The basic idea of stepwise regression [4] is to introduce variables one by one. The condition for introducing variables is that the sum of partial regression square is significant by the F test. Meanwhile, after introducing new variables, the selected variables must be tested again, and those underperforming factors will be removed.

Here, we set the significant level at 0.15 and results are as follows:

TABLE II. SUMMARY OF STEPWISE SELECTION

Step	Variable Entered	Partial R-Square	Model R-Square	C(p)	F Value	Pr > F
1	X7	0.7379	0.7379	62.7105	1066.88	<.0001
2	X10	0.0159	0.7538	38.0665	24.38	<.0001
3	X6	0.0143	0.7680	16.1036	23.22	<.0001
4	X9	0.0073	0.7754	5.7798	12.30	0.0005
5	X3	0.0016	0.7770	5.0516	2.74	0.0990
6	X8	0.0015	0.7785	4.4944	2.57	0.1095

As a result, six ultimate factors are selected by stepwise regression. It can be seen that the R-square value is 0.7785 > 0.75, which indicates the model performs well. Finally, through SAS the coefficients of the multiple regression model are determined:

$$Y = 0.00002301X_3 - 1.64751X_6 + 0.84189X_7 - 0.03471X_8 - 0.00137X_9 - 0.07059X_{10} - 0.57707$$

where Y is *the rate of returns*, X_3 is *yoyop*, X_6 is *tech_turnoverrate10*, X_7 is *tech_revs60*, X_8 is *tech_revs120*, X_9 is *fa_fixedassettoasset*, and X_{10} is *dividendyield2*.

2.3 Validation Testing

Model fitting effect and residual plot are shown below.

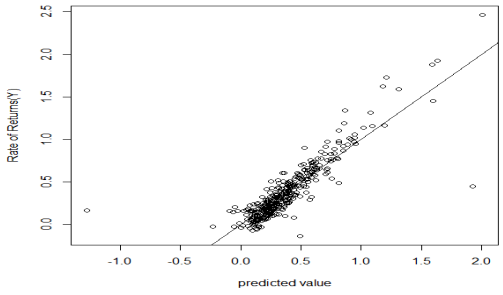

Figure 1. Multivariate linear regression model fitting effect

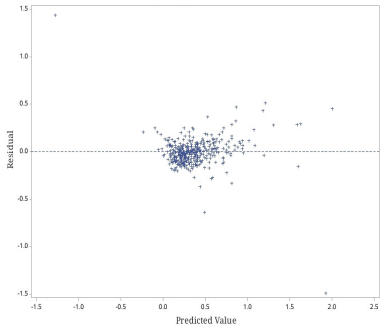

Figure 2. Multivariate linear regression model residual plot

From Fig. 1, it can be seen that the predicted value and the actual rate of returns have a clearly linear relationship with a slope of 1. Besides, from Fig. 2, except for some extreme values, the majority of values are distributed in a horizontal band-shaped region centered on the origin. We can consider that the model estimates the rate of returns well.

3 ADAPTIVE BOOSTING ALGORITHM

Adaboost algorithm was proposed by Freund and Schapire in 1995 [5]. As an improvement of the boosting algorithm, Adaboost's principle is to use iteration to train and filter the weak classifiers by adjusting the sample weight coefficients, and finally form a strong classifier.

For stock selection, we use the following adaboost algorithm:

3.1 Data Preprocessing

Due to the sensitivity of the machine learning algorithm to the data and to prevent overfitting, we first sort all the stocks in the training set for each factor according to the value of the corresponding factor from large to small, and divide the ranking by the total number of shares. This can control the corresponding factor value to $(0,1]$.

Then sort the stock returns for the next period, taking the top 40% of the stocks as strong buy stocks and recording them as $+1$. Correspondingly, taking the last 40% of stocks as strong sell stocks and recording as -1. The middle 20% of stocks are served as noise data and no longer considered in the next discussion.

3.2 Training Algorithm

In the initial state of first-round training, we give equal weights to all stocks marked as $+1$ or -1. Then, the factor values of N stocks corresponding to each factor are sorted by quantity. For all factors, N stocks are divided into K categories, which are recorded as 1, 2, 3,…, K. Then calculate the sum of the corresponding weights of the stocks with $+1$ or -1 in each category (denoted by k) of factor (denoted by i) in particular, and record it as ${}_iW_{\pm}^{k}$.

Next, we construct the statistic:

$$Z_i = \sum_{k=1}^{K} \sqrt{{}_iW_{+}^{k} \cdot {}_iW_{-}^{k}}.$$

We use the value of Z_i to distinguish the weak classifier. Intuitively, since the sum of weights in each classification is always 1, the larger difference between ${}_iW_{+}^{k}$ and ${}_iW_{-}^{k}$, the smaller the value of statistic Z_i, which implies the effect of the corresponding weak classifier is more significant.

For the statistic Z_i calculated by each factor, we choose the factor i_1 with the smallest Z_i to build the first weak classifier:

$$h_1(x) = \frac{1}{2}\ln\left(\frac{{}_{i_1}W_{+}^{k} + \varepsilon}{{}_{i_1}W_{-}^{k} + \varepsilon}\right)$$

where $$i_1 = \arg\min_{i\in\{1,2,\ldots,\}} Z_i, \varepsilon = \frac{1}{N}.$$

It is worth noting that every time we generate a new weak classifier, as a result we need to adjust the weight of the data.

For the data that has been classified correctly, we reduce its weight; on the contrary, for the data that has not been classified correctly, we increase its weight appropriately. We update the weights by the following recursive formula:

$$w_{m+1}^{(\vec{x}_i)} = w_m^{(\vec{x}_i)} \cdot e^{-y_i h_m(\vec{x}_i)}$$

where m represents the weak classifier of the m-th tier.

Then, we readjust the weights in a proportional manner, in order to make the sum of the weights be 1. We repeat the above process, and can get a weak classifier $h_2(x)$ relevant to the second factor denoted by i_2, which is different from i_1.

Finally, we add these weak classifiers to get a strong classifier:

$$H(x) = \sum_i h_i(x).$$

4 RANDOM FOREST ALGORITHM

In the 1980s, the decision tree algorithm was first proposed, using repeated bifurcation strategies for classification or regression. However, the time complexity of this algorithm is too high, instead, Breiman first proposed the random forest algorithm in 2001 [6] based on decision tree algorithm. We will briefly describe the principle of the random forest algorithm, and apply this algorithm to stock selection strategy.

4.1 Decision Tree Algorithm, CART

In the random forest algorithm, we use CART algorithm to classify attributes. First, we introduce the *Gini index* to express the information gain ratio. For sample set Ω, assuming that there are K categories, and the probability that a random sample point in Ω belongs to the k-th category is denoted as P_k, then the *Gini index* of sample set Ω is defined as

$$Gini(\Omega) = \sum_{k=1}^{K} P_k(1-P_k) = 1 - \sum_{k=1}^{K} P_k^2.$$

Obviously, the smaller the *Gini index*, the higher the purity of Ω. Then for each factor A, sorting from large to small, we divide stocks into two categories: Ω_1, Ω_2. Then after splitting according to this factor, the *Gini index* is defined as:

$$Gini(\Omega, A) = \frac{|\Omega_1|}{|\Omega|} Gini(\Omega_1) + \frac{|\Omega_2|}{|\Omega|} Gini(\Omega_2).$$

Starting from the root node, we try different factors to calculate the split *Gini index* and choose the factor that minimizes the *Gini index* to construct the decision tree. Not until the *Gini index* after a node split is less than the given threshold, we stop recursion to get a complete decision tree that can be used for classification prediction.

It is worth noting that if the importance of factors to the algorithm needs to be considered, we should use the following method:

$$D_i(A) = Gini(\Omega_i) - Gini(\Omega_i, A), S(A) = \sum_i D_i(A)$$

where $D_i(A)$ represents the decreasing value of the *Gini index* after the node i splits into two sub-nodes according to the factor A, and $S(A)$ represents the total decreasing value of the *Gini index* caused by the factor A at all nodes, reflecting the contribution of the factor to the construction of the classifier .

Finally, we use pruning to reduce the complexity of the decision tree, thereby reducing the risk of overfitting.

4.2 Random Forest Algorithm

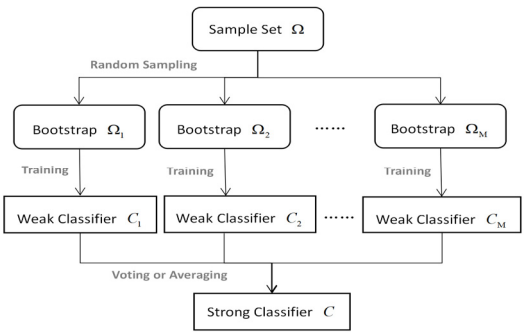

Figure 3. Flow Chart of Random Forest Algorithm

As shown in Fig. 3, we generate M Bootstrap datasets from the original data, train a weak classifier for each Booster dataset, and finally use voting and averaging methods for classification and regression.

4.3 Application in Stock Selection

1) Data Preprocessing: First, we find the median of all stocks corresponding to each factor, and remove extreme data (extremely large or extremely small) based on the median. Then normalize all data to obtain a uniformly distributed sequence on (0,1].

2) Algorithm Application: By section data at the end of each quarter, we select the top 10% stocks with the best performance in the next quarter as strong buy stocks, which are recorded as $+1$. Similarly, the worst performing 10% stocks are selected as the strong sell stock, which is recorded as -1.

Using the interactive verification method, 90% of the samples are randomly selected as the training set each time, and the other 10% of the samples are used as the validation set.

Then use the random forest algorithm to train the training set, and repeat the training for different training sets in each time interval. After the training is completed, the obtained model is used to predict the cross-validation set. In this stage, we use linear regression model for comparison.

3) Algorithm Evaluation: On the one hand, we test the validity of the model, that is, the accuracy rate of the test set, AUC and other indicators; on the other hand, we construct a long position portfolio strategy and compare it with the performance of ETF500, the specific results will be shown in Part VI.

5 Support Vector Machine Algorithm

As one of the most widely used machine learning algorithms, support vector machine was first proposed in the 1990s. Support vector machine have a good performance on linear and nonlinear classification problems [7], which can be divided into linear support vector machines and kernel support vector machines. Since multiple linear regression model has an outstanding performance in stock selection, we will only introduce linear support vector machine below.

Assuming that n candidate factors are considered for each stock, we want to use an $n-1$ dimensional hyperplane to separate outstanding stocks from poor stocks. As in the past, we select the top 10% of stocks with cross-sectional returns as $+1$, and the worst 10% of stocks as -1.

Based on the principle of maximum distance separation, we try to find the maximum margin hyperplane.

In fact, unfortunately, we cannot find a hyperplane that can completely separate the two parts mentioned above. Instead, we can construct a hyperplane that almost completely separates the two and is denoted as:

$$w^T x + b = 0$$

where $w \in R^{n-1}, b \in R$.

Next, we introduce the concept of slack variable to punish misclassified or fuzzy classified stocks.

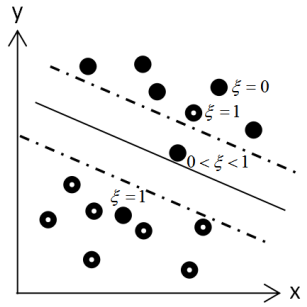

Figure 4. A simple example of linear support vector machine algorithm

As shown in Fig. 4, for those correctly classified points, we assign a value of 0 to the slack variable; for the sample points that are completely misclassified, we assign the value of the slack variable to 1. For the sample points of fuzzy classification, that is, between the solid line and dotted line in the figure, we assign a certain number between 0 and 1 to the slack variable based on the distance between the sample point and the hyperplane. According to the slack variable defined above, we can transform the problem of solving the linear support vector machine into the following extreme value problem:

$$\min_{w,b,\xi_i}(\|w\|^p + C\sum_{i=1}^{N}\xi_i)$$

$$s.t., y_i(w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

where $\|w\|^p$ is reciprocal of a measure of the distance between the sample point and the $n-1$ dimensional hyperplane in the sense of p-norm, and N is the total number of sample points, C is the punishment coefficient, and ξ_i is the value of the slack variable corresponding to each sample point.

It is worth noting that we should give the punishment coefficient C in advance. In general, we need to find the appropriate C through multiple numerical simulations to achieve the balance of bias and variance. Once the punishment coefficient C is given, we can use Lagrange multiplier method and duality principle to solve the optimization problem above.

6 Data Analysis

In addition to using the multivariable linear regression, adaboost, random forest and support vector machine algorithms described above, we also use XGBoost and gradient boosting algorithms to select stocks. We buy stocks with a predicted return of the top 10% with equal weight every three months, and calculate the abnormal return and cumulative function of each time. In this process, ETF500 index will be used to reflect market performance. Cumulative functions corresponding to the strategies obtained by different machine learning algorithms are shown in the following figures:

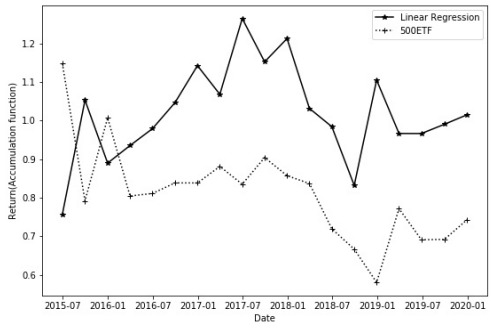

Figure 5. Back-testing result of Linear Regression Model

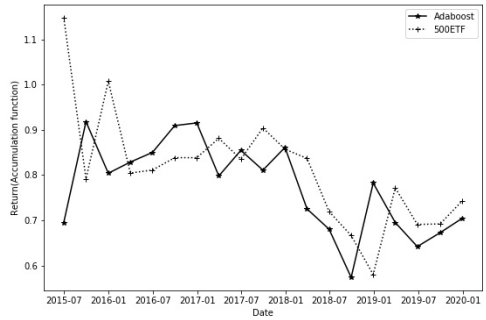

Figure6. Back-testing result of Adaptive Boosting Algorithm

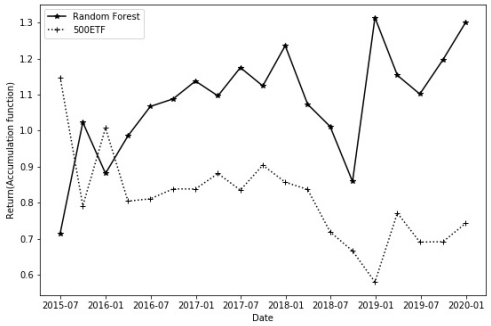

Figure7.Back-testing result of Random Forest Algorithm

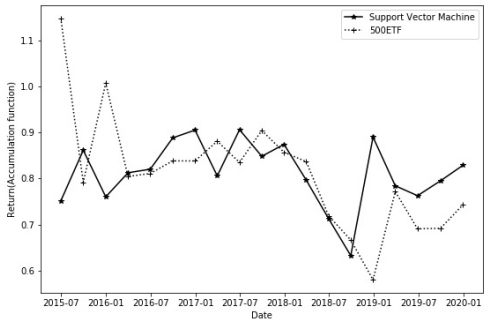

Figure8. Back-testing result of SVM Algorithm

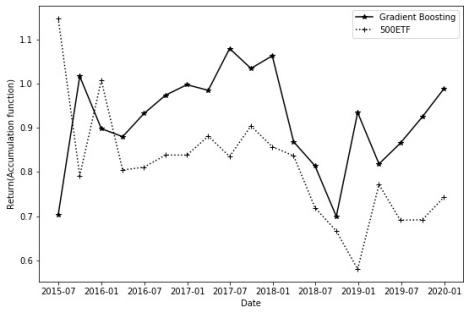

Figure9.Back-testing result of Gradient Boosting Algorithm

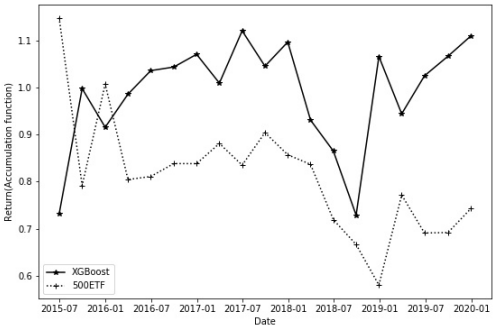

Figure10.Back-testing result of XGBoost Algorithm

TABLE III. ABNORMAL RETURNS OF THE FIRST SIX STOCK SELECTIONS WITH DIFFERENT ALGORITHMS

	LR	SVM	ABR	GBR	RFR	XGBR
2015/9/30	6.705%	6.163%	0.507%	1.268%	2.510%	4.132%
2015/12/31	11.966%	-12.547%	4.920%	17.550%	15.909%	9.294%
2016/3/31	4.619%	8.395%	7.712%	8.470%	6.307%	11.901%
2016/6/30	4.314%	6.061%	2.273%	-2.816%	10.986%	6.865%
2016/9/30	1.289%	-2.427%	-0.818%	2.567%	4.918%	1.696%
2016/12/30	6.954%	8.315%	6.999%	4.471%	1.956%	0.755%

We select the first six stock selections of different algorithms and calculate abnormal returns. Compared with the ETF500, from TABLE III, there are much more positive terms than negative terms, which implies machine learning algorithms do have a better performance in stock selection.

From Fig. 5, 7, 9 and 10, the linear regression model, random forest, gradient boosting and XGBoost algorithm has stable and significant returns compared to ETF500, while from Fig. 6, the back-testing result of the adaboost algorithm is generally the same as ETF500. From Fig. 8, SVM obtains a certain return with a slight advantage, but still lacks stability.

The reason for linear regression model performs well is that we only use a small number of factors. If we increase the number of factors, other machine learning algorithms will perform better. Adaboost performs mediocrely because each time we only use one single factor to develop a weak classifier. As the algorithm with the highest complexity, random forest performs best. The lack of stability of SVM is because, when we assign values to the slack variables, those misclassified sample points are all assigned as 1. The distance of the sample points from the hyperplane is not considered, that is, the degree of misclassification. This will increase the probability of extreme values, as shown in TABLE Ⅲ for the second selection, the return of selected stocks is 12% lower than that of ETF500. As an improved algorithm of ababoost, gradient boosting [8] and XGBoost [9] perform better and more stable.

7 CONCLUSIONS

This article uses six traditional machine learning algorithms for stock selection, taking CSI 500 as an example, and back-tests each selection strategy. We introduce linear regression, adaboost, random forest and SVM in detail. Most algorithms have good performances.

These algorithms could be further extended by using multiple factors to develop weak classifiers of adaboost, or giving a more realistic loss function to SVM to reduce uncertainty.

REFERENCES

1. Yuh-Jen Chen, 2013, "A Fundamental Analysis-based Method for Stock Market Forecasting", Fourth International Conference on Intelligent Control and Information Processing (ICICIP), 2013, Beijing, China.
2. Ghezzi Luca Luigi, Peccati Lorenzo. Qualitative analysis of a nonlinear stock price model [J]. Elsevier, 1994, 63 (2-3).
3. Hua Li, Shuanghong Qu, Hua Zhu. PROMETHEE: a Fuzzy Algorithm for Decision Making in Stock Market Value Investing. 2013, 1(1):75-80.
4. Cheng, Shou Hsiung. "A Hybrid Predicting Stock Re turn Model Based on Logistic Stepwise Regression a nd CART Algorithm." Asian Conference on Intellige nt Information & Database Systems Springer Interna tional Publishing, 2015.
5. Freund, Yoav, and R.E. Schapire. "A decision-theoretic genea generalization of on-line learning and an application to boosting." Proceedings of the Second European Conference on Computational Learning Theory Springer-Verlag, 1995.
6. Breiman, L. Random forests. Machine Learning 45 (1): 5-32, Web of Science, 2001.
7. Cortes C, Vapnik V N. Support vector networks [J]. Machine Learning, 1995, 20(3):273-297.
8. Friedman J H. Stochastic gradient boosting [J]. Computational Statistics & Data Analysis, 2002, 38.
9. Chen, Tianqi, Guestrin, Carlos. XGBoost: A Scalable Tree Boosting System [J]. 2016.