# Stacked Model with Autoencoder for Financial Time Series Prediction

Haiying Zhang
*School of Informatics*
*Xiamen University*
Xiamen,China
zhang2002@xmu.edu.cn

Qiaomei Liang
*School of Informatics*
*Xiamen University*
Xiamen,China
qmliang0604@qq.com

Rongqi Wang
*Xiamen Rural Commercial Bank*
Xiamen,China
wangrongqi@fjnx.com.cn

Qingqiang Wu*
*School of Informatics*
*Xiamen University*
Xiamen,China
**Corresponding Author*
wuqq@xmu.edu.cn

***Abstract*—In this paper, we propose a stacked model with autoencoder for financial time series prediction. A stacked autoencoder model is used for feature extraction of high-dimensional stock factors. The factors after dimensionality reduction serve as input to the stacked model to predict the next-day returns of the stocks. In this paper, the stacked autoencoder not only has the effect of reducing the dimension, but also eliminates the redundant information in the data to a certain extent, which can effectively improve the predictive capacity of the model. The constituent stocks of CSI300 are used as backtest samples, and the experiment shows that the stacked model with autoencoder can obtain more than 50% of excess return in 2019.**

***Keywords—autoencoder; feature extraction; stacked model***

## I. INTRODUCTION

Artificial intelligence is a cutting-edge computer discipline that has emerged in recent years. Its essence is to try to use computer technology to study and imitate human thinking and patterns to achieve machine intelligence and enable machines to complete complex tasks that only humans can accomplish. There are many research fields of artificial intelligence, including machine language translation, speech recognition, image recognition, driverless, etc. Researchers have achieved good research results in the above fields. Combined with the current state of artificial intelligence development, intelligent investment consulting and intelligent investment research are likely to become the best match point for artificial intelligence technology and the financial system.

Current researches use machine learning or deep learning to build effective prediction models. The model is mainly divided into two modules: feature extraction and time series prediction.

In terms of factor processing, most researchers use raw stock information as factors, such as the opening price, the closing price, the highest price, the lowest price, and the volume.[1][2] In addition, some technical indicators are also used as stock factors, such as MA, TR, BIAS, etc.[3] With the development of technology, researchers have tried to construct factors through machines. AHP[4], ICA[5], PCA[6] and other methods have been widely used. Neural networks are gradually being used to extract stock factors, due to theirs excellent nonlinear problem processing capabilities. In order to capture the relationship between the technical indicators and the stock market for the period under investigation, Göçken M et al. hybrid Artificial Neural Network (ANN) models, which consist in exploiting capabilities of Harmony Search (HS) and Genetic Algorithm (GA), are used for selecting the most relevant technical indicators[7].

In terms of time series prediction, many stock prediction models have been built based on SVM[8-10] and neural networks[11-13]. Ensemble learning builds a strong learner by combining weak learners. Boosting and bagging are also very popular in the field of stock prediction[14-17]. Weng B et al. improve the model prediction performance by using machine learning ensemble methods[18]. However, some studies[19][20] have shown that, in terms of stock time series prediction, ensemble learning algorithms have stronger prediction capabilities than neural networks.

However, in the existing models, the problems of low factor prediction ability, single model type, and complex non-linear modeling ability problems are common. Therefore, this paper proposes a stacked model that incorporates an autoencoder. The stacked autoencoder is used to improve the predictive ability of the factors and reduce the redundant information of the data. By combining multiple ensemble learning models, the stacked model can improve the accuracy of prediction. The most important point of this paper is embedding an autoencoder in the stacked model, and the experiments have proved that it can obtain higher returns.

This paper is mainly divided into four parts, the second part will describe the construction of the model, the third part will describe a series of experimental results of the model, and the fourth part will make a summary of this paper

## II. METHODOLOGY

### A. Stacked autoencoder model

An autoencoder is a neural network model that implements data compression. It belongs to unsupervised learning. It tries to reproduce the input of its model to build a network, that is, its final target output is the model's input. However, there are some differences between the output value of the model and the input data itself. The essence of autoencoder model training is to reconstruct its input data by encoding and decoding. The simple autoencoder model is a three-layer neural network, as shown in

Fig.1, including an input layer, a hidden layer, and an output reconstruction layer. From the data input layer to the hidden layer is called the encoding process, and from the hidden layer to the output layer is called the decoding process. The autoencoder model is equivalent to being able to generate labels autonomously, and the labels are the input sample data itself.

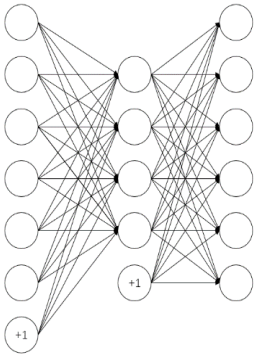

Fig. 1. A simple autoencoder model

Assuming that the given training sample data is x, first the input sample data is encoded by the autoencoder according to Eq.1 to obtain the intrinsic characteristics of the input sample data, and then the hidden layer is decoded according to Eq.2. Finally, it reconstructs the training sample data as output.

$$y(x) = f(W_1 + b_1) \tag{1}$$

$$z(x) = g(W_2 y(x) + b_2) \tag{2}$$

Where $W_1$ represents the weight matrix in the encoding process, $b_1$ represents the offset vector in the encoding process. $W_2$ represents the weight matrix in the decoding process, $b_2$ represents the offset vector in the decoding process. *f(.)* and *g(.)* represent the activation function in the encoding process and the decoding process, respectively. The activation functions used in this paper are all *sigmoid* activation functions, as shown in Eq.3.

$$F = \frac{1}{1+\exp(-x)} \tag{3}$$

The autoencoder is a lossy compression algorithm. By minimizing the loss function, the input data is similar to the output reconstructed data. The loss function is shown in Eq.4:

$$L(W,b) = \frac{1}{m}\sum_{r=1}^{m}\frac{1}{2}\left\|x^{(r)} - z(x^{(r)})\right\|^2 \tag{4}$$

The stacked autoencoder is constructed by stacking autoencoders, and feature extraction is performed on high-dimensional data by using the stacked autoencoder. The dimension of each autoencoder feature layer is less than the dimension of the autoencoder data input layer, so as to achieve the purpose of dimensionality reduction or feature extraction. The structure of this neural network is shown in Fig.2.

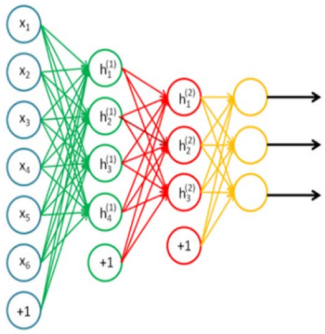

Fig. 2. Feature extraction structure diagram of stacked autoencoder

## B. *Ensemble learning method*

*1) Boosting:* Boosting is an algorithm that can promote weak learners to strong learners. Its working mechanism is: first train a weak learner from the initial training set, and then adjust the training sample distribution according to the performance of the weak learner, so that the weight of the misclassified samples which are trained by previous weak learner becomes higher, that is, the misclassified samples will receive more attention in the weak learners, and then the next weak learner is trained based on the adjusted sample distribution. The above process is repeated continuously until the number of weak learners reaches a predetermined number T, and finally the T weak learners are integrated through a set strategy to obtain the final strong learner.

*2) Bagging:* The training set of the weak learners in bagging is obtained by random sampling. Through T random samplings, we can get T sample sets, and then train T weak learners independently for these T sample sets. Finally, we construct these T weak learners into a strong Learner.

*3) Stacking:* The concept of stacking is to learn several different weak learners and combine them by training a metamodel, and then output the final prediction result based on multiple prediction results returned by these weak models. To build a stacking model, two things need to be defined: the m learners that you want to fit and the metamodel that combines them.

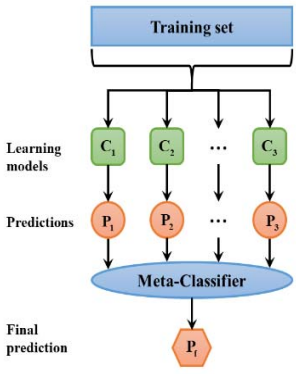


Fig. 3. The structure diagram of stacking

### C. Stacked model with autoencoder

The focus of bagging is to obtain an integrated model with smaller variance than its components, while boosting and stacking are mainly generate strong models with lower deviations than their components (even the variance can be reduced). Therefore, this paper aims to combine the advantages of bagging, boosting and stacking to build a model with high prediction accuracy and small error. The model structure is shown in Fig.4.

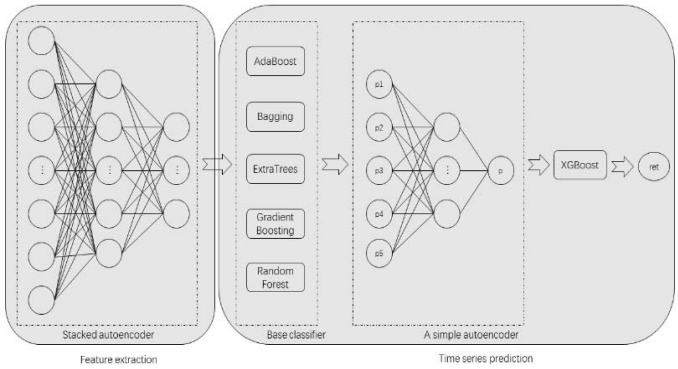


Fig. 4. Overall model structure diagram

The stock price prediction model in this paper is mainly divided into two parts: feature extraction module and time series prediction module. The feature extraction module uses a stacked autoencoder to extract effective factors. The time series prediction module is mainly a stacked model integrating regression algorithms. AdaBoost, Bagging, Extra Trees, Gradient Boosting, and Random Forest are used as elementary learners, and their prediction results are used as the input data of the XGBoost model to obtain the final prediction results.

Because financial time series are constantly changing, the generalization ability of the base classifiers will become weaker. Since the feature space of the metamodel is determined by the base classifiers, the effectiveness of the feature space becomes weaker. In order to improve the validity of the feature space, this paper chooses to add an autoencoder to the stacked model to improve the prediction performance of the metamodel.

## III. Experiment and Result

This paper uses the daily market data of A shares from January 1, 2012 to December 1, 2019, including the opening price, closing price, highest price, lowest price and volume of each stock on each trading day. Because the stacked model uses the predicted values of the junior learner as training samples, this paper divides the data set into three parts: 2012 to 2016, 2017 to 2018, and 2019 as the test set.

### A. Feature extraction

In this paper, we use 16 common technical indicators, as shown in Table 1. This type of indicator has clear inherent meaning from different perspectives to reflect the future trend of the stock.

TABLE I. THE TECHNICAL INDICATORS THAT ARE SELECTED AS INPUT VARIABLES

| Technical indicators | Technical indicators |
|---|---|
| volume_relative_ratio5 | ma5_ma10_dis |
| price_center5_5 | atr14 |
| ma10_close_dis | rsi14 |
| bias | cci14 |
| mfi | pvt |
| ar13 | logreturn5 |
| br13 | amplitude |
| vma6 | macd |

IC (Information Coefficient) represents the correlation coefficient between the factor value of the current cycle of the stock and the return rate of the next cycle. It is generally believed that if the absolute value of a factor's IC value is greater than 5%, then the factor is considered to have better predictive power; if the absolute value of the IC value is greater than 10%, the factor can be considered to have strong predictive power. The IC value of the stacked autoencoder factors proposed in this paper on the test set is shown in Table 2. ( 'Technical' represents the original 16 factors)

TABLE II. IC VALUE OF STACKED AUTOENCODER FACTORS ON TEST SET

| Factor | IC | | | |
|---|---|---|---|---|
| | Avg. | Max. | Min. | Med. |
| Technical | 0.0486 | 0.0664 | 0.0222 | 0.0477 |
| 64_l2_d5 | 0.0430 | **0.0690** | 0.0114 | **0.0482** |
| 64_l2_d10 | **0.0539** | **0.0710** | 0.0125 | **0.0583** |
| 64_l3_d5 | **0.0487** | **0.0735** | 0.0085 | **0.0616** |
| 64_l3_d10 | 0.0421 | **0.0681** | 0.0117 | 0.0426 |
| 128_l2_d5 | 0.0291 | **0.0685** | 0.0043 | 0.0160 |
| 128_l2_d10 | 0.0354 | **0.0674** | 0.0020 | 0.0402 |
| 128_l3_d5 | 0.0485 | **0.0725** | **0.0316** | 0.0462 |
| 128_l3_d10 | 0.0349 | 0.0621 | 0.0009 | 0.0382 |
| 256_l2_d5 | **0.0603** | **0.0706** | **0.0427** | **0.0645** |
| 256_l2_d10 | **0.0535** | **0.0715** | **0.0231** | **0.0552** |
| 256_l3_d5 | 0.0485 | 0.0527 | **0.0419** | **0.0512** |
| 256_l3_d10 | 0.0386 | **0.0671** | 0.0041 | 0.0421 |

In Table 2, different autoencoders generate different factors. The factors are distinguished in the form of 'N_lX_dY'. 'N' represents the number of neurons in the first feature extraction layer, 'X' represents the number of layers in the feature extraction layer, and 'Y' represents the dimension of the last data. The bold part indicates that the IC value of the autoencoder factor exceeds the original factor.

It can be seen from the experimental results that the stacked autoencoder can improve the predictive ability of factors, and it is related to the setting of the number of neurons. The IC value

of the '256_lX_dY' factor exceeds the benchmark at the minimum value, indicating that the model improves the lower limit of the factor performance and improves the predictive ability of the factor as a whole. Factors with a relatively large number of neurons have stronger predictive ability.

*B. Time series prediction*

In this section, from the perspective of machine learning and quantitative investment, the regression prediction results of the stacked model with autoencoder are evaluated, and the backtest results on the constituent stocks of CSI300 are given. The stacked model with autoencoder is called Autoencoder-Stacking, and the stacked model without autoencoder is called NoAutoencoder-Stacking. Model prediction error results are shown in Table 3.

TABLE III. MODEL PREDICTION ERROR(RMSE)

| Factor | NoAutoencoder-Stacking | Autoencoder-Stacking |
|---|---|---|
| Technical | 0.0271 | 0.0275 |
| 64_l2_d5 | 0.0280 | **0.0271** |
| 64_l2_d10 | 0.0280 | **0.0271** |
| 64_l3_d5 | 0.0277 | **0.0270** |
| 64_l3_d10 | 0.0280 | **0.0271** |
| 128_l2_d5 | 0.0280 | **0.0271** |
| 128_l2_d10 | 0.0280 | **0.0265** |
| 128_l3_d5 | 0.0280 | **0.0265** |
| 128_l3_d10 | 0.0282 | **0.0266** |
| 256_l2_d5 | 0.0283 | **0.0266** |
| 256_l2_d10 | **0.0268** | **0.0266** |
| 256_l3_d5 | 0.0281 | **0.0265** |
| 256_l3_d10 | 0.0279 | **0.0264** |

Compared with the original factors, the Autoencoder-Stacking proposed in this paper can reduce the prediction error, indicating that the metamodel has improved the generalized prediction ability of the primary model to a certain extent. The ability to express information on indicators has improved. And with the increase of the number of neurons in the feature extraction layer of the stacked autoencoder, the prediction error of the model also decreases accordingly, indicating the correlation between the number of neurons and model performance.

With the change of financial time series, the generalization ability of the base classifiers become weaker, which leads to a decrease in the validity of the input features of the metamodel, and a decrease in the predictive performance of the metamodel. The existence of the autoencoder improves the effectiveness of input features and improves the predictive performance of the metamodel.

Because the model in this paper predicts the next-day rate of return of each stock, we use the form of daily adjustment of positions, operate on the top five stocks each trading day and can only do long operations. The backtest period is from January 01, 2019 to November 30, 2019. During this period, the rate of return of the benchmark (CSI300) was 30.06%.

TABLE IV. THE BACKTEST RESULTS OF NOAUTOENCODER-STACKING

| Factor | Rate of return | Winning rate | Max drawdown | Sharpe |
|---|---|---|---|---|
| Technical | 41.45% | 50.17% | 19.10% | 2.72 |
| 64_l2_d5 | -11.54% | 46.42% | 28.86% | 0.19 |
| 64_l2_d10 | -5.67% | 47.62% | 27.40% | 0.75 |
| 64_l3_d5 | 2.52% | 47.62% | 30.40% | 1.35 |
| 64_l3_d10 | -4.65% | 45.64% | 34.67% | 0.74 |
| 128_l2_d5 | -1.97% | 46.84% | 83.26% | 0.69 |
| 128_l2_d10 | -11.80% | 46.96% | 34.00% | 0.28 |
| 128_l3_d5 | -1.03% | 45.21% | 34.27% | 0.66 |
| 128_l3_d10 | 3.36% | 48.93% | 29.64% | 1.27 |
| 256_l2_d5 | 0.93% | 47.30% | 30.99% | 1.31 |
| 256_l2_d10 | -0.05% | 49.54% | 82.69% | 0.73 |
| 256_l3_d5 | 5.69% | 45.19% | 27.33% | 1.14 |
| 256_l3_d10 | 12.04% | 48.43% | 25.86% | 2.07 |

TABLE V. THE BACKTEST RESULTS OF AUTOENCODER-STACKING

| Factor | Rate of return | Winning rate | Max drawdown | Sharpe |
|---|---|---|---|---|
| Technical | 2.99% | 47.85% | 23.97% | 1.77 |
| 64_l2_d5 | -9.82% | 47.14% | 33.54% | 0.62 |
| 64_l2_d10 | -9.78% | 46.65% | 26.33% | 0.2 |
| 64_l3_d5 | -0.73% | 49.52% | 25.78% | 1.15 |
| 64_l3_d10 | 3.13% | 48.17% | 25.43% | 1.29 |
| **128_l2_d5** | **56.75%** | **52.31%** | **14.90%** | **2.71** |
| 128_l2_d10 | 7.89% | 47.47% | 20.16% | 1.46 |
| 128_l3_d5 | 6.35% | 47.74% | 28.23% | 1 |
| 128_l3_d10 | -16.97% | 45.39% | 40.37% | 0.42 |
| 256_l2_d5 | 3.11% | 48.17% | 30.27% | 1.17 |
| 256_l2_d10 | 4.94% | 46.94% | 33.01% | 0.87 |
| 256_l3_d5 | 18.15% | 48.44% | 22.65% | 2.1 |
| 256_l3_d10 | 15.60% | 48.66% | 22.45% | 2.31 |

It can be seen from Table 4 and Table 5 that models based on different factors present significantly different backtest results and the stacked model can obtain an excess return of 56.75% after adding the autoencoder, which improves the winning rate and reduces the investment risk.

In general, the model base on '128_l2_d5' factor has the best prediction performance, has the advantages of high returns and low risks, and can make profits while bearing the risks.

The NoAutoencoder-Stacking model based on original factors gets better backtest results than that based on 'N_lX_dY'

factors, which proves that the model has a good modeling ability for the original factors

It is worth noting that the RMSE on the test set of the Autoencoder-Stacking model based on the '128_l2_d5' factor does not get the best performance, second only to the '256_lX_dY' series of factors, the difference is about 0.0005, but the backtest performance of them is very different. It may be because that '256_lX_dY' series of factors show the lowest average prediction error performance on the test set, but '128_l2_d5' has a good prediction ability for stocks with high future returns, and a poor prediction ability for stocks with low future returns, which resulting in the overall average prediction error is greater than the '256_lX_dY' series of factors.

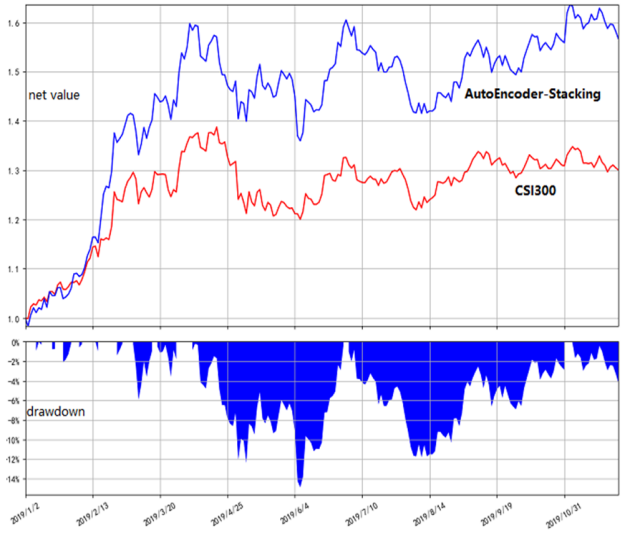


Fig. 5. Net value curve of factor '128_l2_d5'

As can be seen from Fig.5, the net value of the Autoencoder-Stacking model based on '128_l2_d5' is larger than the benchmark for a long time, and generally shows a steady growth trend. The large retracement in the medium term may be due to the problem of updating of the constituent stocks of CSI300. But generally it shows acceptable small fluctuations, indicating that the model has good stability and low investment risk.

## IV. Conclusion

In the field of quantitative investment, it has always been the goal to build a model with low risks and high returns. The stacked model with autoencoder proposed in this paper can obtain more stable benefits while improving the accuracy of prediction. But the model has many parameters, and there may be more effective models, which needs further research.

## References

[1] Jia H. Investigation into the effectiveness of long short term memory networks for stock price prediction. arXiv preprint arXiv:1603.07893, 2016.

[2] Chen K, Zhou Y, Dai F. A LSTM-based method for stock returns prediction: A case study of China stock market//2015 IEEE international conference on big data (big data). IEEE, 2015: 2823-2824.

[3] Singh R, Srivastava S. Stock prediction using deep learning. Multimedia Tools and Applications, 2017, 76(18): 18569-18584.

[4] Marković I, Stojanović M, Stanković J, et al. Stock market trend prediction using AHP and weighted kernel LS-SVM. Soft Computing, 2017, 21(18): 5387-5398.

[5] Grigoryan H. A Stock Market Prediction Method Based on Support Vector Machines (SVM) and Independent Component Analysis (ICA). Database Systems Journal, 2016, 7(1): 12-21.

[6] Chang P C, Wu J L. A critical feature extraction by kernel PCA in stock trading model. Soft Computing, 2015, 19(5): 1393-1408.

[7] Göçken M, Özçalıcı M, Boru A, et al. Integrating metaheuristics and artificial neural networks for improved stock price prediction. Expert Systems with Applications, 2016, 44: 320-331.

[8] Gong X L, Liu X H, Xiong X, et al. Forecasting stock volatility process using improved least square support vector machine approach. Soft Computing, 2019, 23(22): 11867-11881.

[9] Huang C F. A hybrid stock selection model using genetic algorithms and support vector regression. Applied Soft Computing, 2012, 12(2): 807-818.

[10] Nayak R K, Mishra D, Rath A K. A Naïve SVM-KNN based stock market trend reversal analysis for Indian benchmark indices. Applied Soft Computing, 2015, 35: 670-680.

[11] Sun H, Rong W, Zhang J, et al. Stacked Denoising Autoencoder Based Stock Market Trend Prediction via K-Nearest Neighbour Data Selection//International Conference on Neural Information Processing. Springer, Cham, 2017: 882-892.

[12] Liu H, Song B. Stock trends forecasting by multi-layer stochastic ann bagging//2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2017: 322-329.

[13] Li J, Liu G, Yeung H W F, et al. A novel stacked denoising autoencoder with swarm intelligence optimization for stock index prediction//2017 International Joint Conference on Neural Networks (IJCNN). IEEE, 2017: 1956-1961.

[14] Krauss C, Do X A, Huck N. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. European Journal of Operational Research, 2017, 259(2): 689-702.

[15] Jidong L, Ran Z. Dynamic Weighting Multi Factor Stock Selection Strategy Based on XGboost Machine Learning Algorithm//2018 IEEE International Conference of Safety Produce Informatization (IICSPI). IEEE, 2018: 868-872.

[16] Wang Q, Xu W, Huang X, et al. Enhancing intraday stock price manipulation detection by leveraging recurrent neural networks with ensemble learning. Neurocomputing, 2019, 347: 46-58.

[17] Guoying Z, Ping C. Forecast of Yearly Stock Returns Based on Adaboost Integration Algorithm//2017 IEEE International Conference on Smart Cloud (SmartCloud). IEEE, 2017: 263-267.

[18] Weng B, Lu L, Wang X, et al. Predicting short-term stock prices using ensemble methods and online data sources. Expert Systems with Applications, 2018, 112: 258-273.

[19] Park M, Lee M L, Lee J. Predicting stock market indices using classification tools. Asian Economic and Financial Review, 2019, 9(2): 243.

[20] Hah D W, Kim Y M, Ahn J J. A study on KOSPI 200 direction forecasting using XGBoost model. The Korean Data & Information Science Society, 2019, 30(3): 655-669.