# VE445 Project Report

Group 5: Zhang Sijun, Shen Zhe, Zhao Hantao

*Abstract*— **This report is the project report for VE445. In this project, our goal is to make predictions on the stock market based on previous indicators. We first preprocess the dataset to extract the indicators and the labels, then we applied Principal Component Analysis(PCA) on the dataset to reduce the dimension of the features. After that, we tried different machine learning methods, like Logistic Regression, Support vector machine(SVM), and deep learning methods on the binary classification task respectively. In the first part of this report, we'll briefly introduce the basic theories of our models. Then we'll explain the details of our experiments. At the end of the report, the experiment results and corresponding discussion part will be given, and we will display our experiment result and do comparison and analysis on the results.**

## Keywords
**Stock market prediction; Classification; Machine learning Deep learning**

## Authors
**Shen Zhe 516370910148**
**Zhang Sijun 516370910155**
**Zhao Hantao 516370910046**

## I. INTRODUCTION

The stock market is full of opportunities. Unlike bonds or deposits, the values of stocks doesn't increase by itself. For stocks, the profits and losses are only caused by price changes. Therefore, those who can predict the rises and falls in the future are sure to make money in the stock market. However, given the tremendous number of indicators in the stock market, the process of prediction is nothing easy for human beings. Often, we have to rely on computers.

With the development of machine learning, nowadays we have a wide range of tools to facilitate the process of prediction. Traditional supervised-learning algorithms, including logistic regression and SVM (Support Vector Machine) are used and stated in our project. In addition, we implemented deep learning algorithms, like BP neural networks and Recurrent Neural Network(RNN).

In this project, the data for the stock market, including the indicators and the corresponding midprices at different timestamps are given. Our main task is to carry out binary-classification on given data set. First, we try to reduce the dimension of data features by PCA. Then we do the classification task using both traditional classical methods, Logistic regression, SVM, Random Forest contained, and deep Learning-based neural network frameworks.

Our code is written in Python 3.7. We referred to many machine-learning libraries for python, including tensorflow, keras and sklearn. For the numerical computation, we mainly utilized numpy. Experiments were carried out under environment of Win10 Operating System. Our thoughts and discussion are also presented in the experiment result parts. The following part in this report will introduce the basic theories of our models. Then, at the third part, we will introduce details of our experiments and show the experiment results. The result comparison, analysis and discussion are also in the third part.

## II. FEATURE ENGINEERING

In this part, we will mainly give a description of machine learning methods used in our experiment and how they are realized. In general, it could be divided into three sections: data preparation, resampling and dimension reduction.

### A. Data Separation

Due to the huge size of original data sets, we first separated the original data sets into 20 batches by linear separation to maintain the sequality of data in time series. Each batch of data sets consists of about 86,000 samples. Considering the characters of stock market is changing over the time, the relative small train and predict data sets is of more practical meanings.

### B. Resampling

We noticed that the stock data is sampled in about the time stamps of $\frac{1}{4}$ seconds and there is a malignant behavior of the midPrice that a mutation may exists in one seconds and quickly go back to the mean value, which is harmful for analyzing the rise and fall between two seconds.

Secondly, the original number of rise and fall between two time stamps is unbalanced, the resampling process also rebalance the training and validation data sets into a reasonable comparison.

Hence we applied a moving window to resample the stock data and obtained the mean of each 4 time stamps' data as the analyzing and validation input. The general idea is to realize a moving window feature selection with the window size of 4 and shift value also equals to 4.

As we maintain the shift value equal to the width of window, so each window will not be overlapped.

### C. Dimension Reduction

Since the given dataset has 108 indicators and 1721578 timestamps, it is a problem with large n and small p. Therefore, dimension reduction process before applying machine learning methods is necessary. To achieve the aim of dimension reduction, we chose the mainstream method PCA.

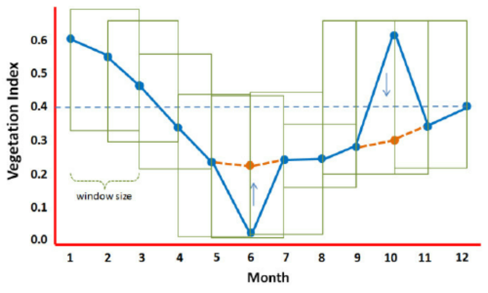


Fig. 1. Moving Window Feature Extraction

Here we will give a brief introduction of the basic theory of PCA. The details will be discussed in the experiment and result part.

*1) Correlation of Features:* We first analyze the correlation of each features and find the following two arguments:

- The indicator 1 to indicator 108 has some inner relationships with each other as each of them has some members share the nearly the biggest correlation value with itself.
- The midPrice, which is highly correlated to the label value, doesn't seems to be associated with the indicators tightly.

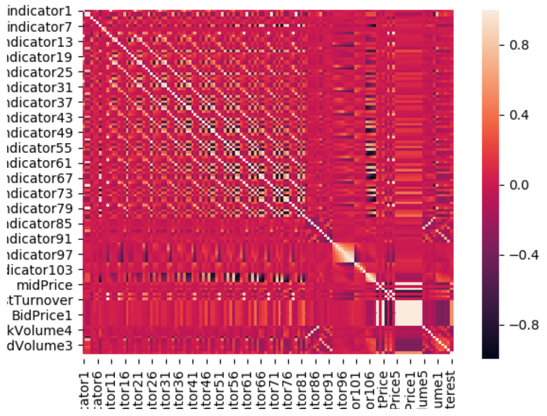


Fig. 2. Moving Window Feature Extraction

*2) PCA Method:* Principal Components analysis(PCA) aims to perform a linear mapping of the data to a lower-dimensional space. Suppose there is a given data matrix X with size n*p, where n is the quantity of sampling and p is the dimension of feature. In our project, our aim is to reduce p without lossing much information. Some key steps of PCA algorithms are listed as follows.

(i) Normalize the data with the empirical mean vector and let it be centered. Approximate the empirical mean vector $\mu$ from known samples by using the averaging expression

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

then center the data by

$$x = x - \mu \tag{2}$$

(ii) Calculate the covariance matrix C from the samples with

$$C = \frac{X^T X}{n-1} \tag{3}$$

(iii) Compute the eigenvectors and eigenvalues of the covariance matrix C, since C is a symmetric matrix and thus can be diagonalized

$$V^T C V = D \tag{4}$$

where the matrix V consists of eigenvectors which diagonalized the covariance matrix C, and D is the diagonal matrix of eigenvalues of C.

(iv) Rearrange the eigenvectors and eigenvalues in a way that the columns of the eigenvector matrix V and eigenvalue matrix D are sorted in decreasing order. The eigenvalues represent the distribution of the source datas energy among each of the eigenvectors, where the eigenvectors form a basis for the data.

(v) Select the first k columns of V as the p k matrix W, and the criterion should be followed

$$\frac{\sum_{i=1}^{k} D_{ii}}{\sum_{i=1}^{p} D_{ii}} \geq Threshold \tag{5}$$

where $Threshold$ is the desired proportion of cumulative energy. In this project, we tried different threshold, $Threshold$ = 99%, 95%, 90%, to get a proper final p.

(vi) Project the z-scores of the data onto the new basis

$$t_i = W^T(x_i - \mu) \tag{6}$$

In such a set of steps, PCA makes the feature has large variance while the noise has low variance. After PCA, we successfully reduce the correlation of features, which considerably accelerates training in following classification stages. Moreover, centered feature vector helps a lot for training without extra preprocessing.

In order to visualize the data set, we performed PCA to reduce the feaure dimension to 3 and plot them in Euclidean space.

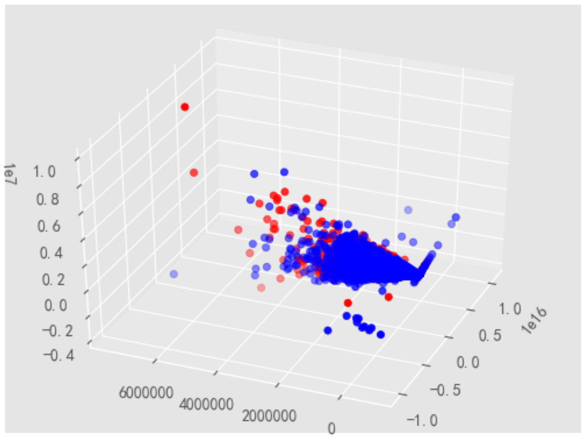


Fig. 3. Visualization of data with 3 dimensions

## III. Classification Methods

### A. Logistic Regression

Logistic regression model is a classical supervised linear classifier. The linear classification approach has two major components: a hypothesis function that maps the original data to class scores, and a loss function that quantities the gap between the results of the hypothesis function and the true labels. Logistic regression first calculates the boundary among different classes, and then it will predict the possibility of test data class based on the calculated data boundary. Let x denotes the feature vector of given training dataset, and Y denotes corresponding label of training dataset, then the calculated class possibility is defined as

$$P(Y=y_i|x_i)=\frac{e^{\omega x_i+b}}{1+\sum_i e^{\omega x_i+b}}$$

where $\omega$ denotes the regression weights and b denotes the regression bias. Based on the labels training data, using Maximum Likelihood Estimation(MLE) method we can determine the value of $\omega$. Let $h_\omega(x_i)$ denoting the possibilities, the log likelihood function is defined as

$$L(\omega)=\sum_i^N[y_i\log h_\omega(x_i)+(1-y_i)\log 1-h_\omega(x_i)]$$

There are several gradient-based optimizer can be applied to determine the value of $\omega$ and $b$. In our project, we used the logistic function in sklearn module to predict and compare the results.

### B. support vector machine

Support Vector Machine (SVM) algorithm has proves to be one of the most effective Machine Learning algorithms. When the data is linear or non-linear distributed, SVM could reach a high accuracy in classifying unlabelled data. When the data is linearly separable, the solution from SVM training is a hyperplane in $n-1$ dimensions. This hyperplane should

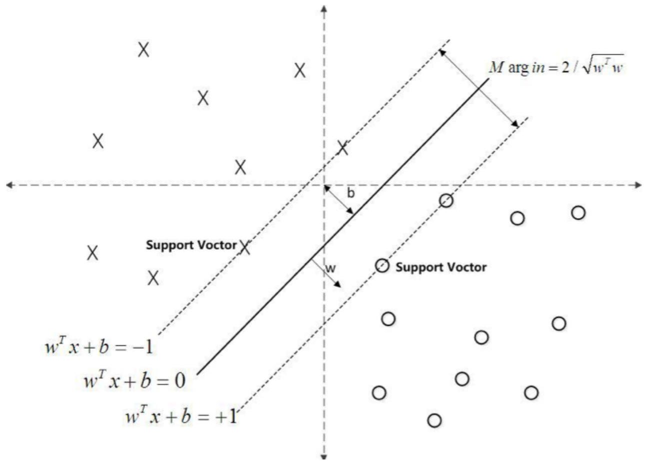


Fig. 4. Hyperplane and geometric distance in SVM

maximize the geometric distance between the plane and support vector, which is the measurement of the perpendicular lines to the hyperplane. When dealing with the non-linear separable problems, we can map the low-dimension problem to high-dimension space.

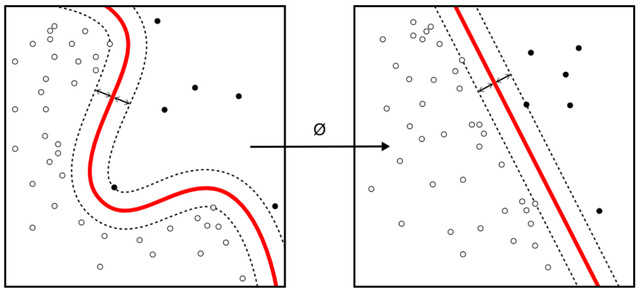


Fig. 5. mapping in SVM

However, this mapping relationship results in a serious problem. The time cost to multiply high dimension matrix significantly increases when the dimension is increasing, which is $O(D^2)$. When the linear regression could only be down in infinite dimension space, the multiplication can not even be conducted. The mathematicians provide us a tool named as **kernel trick** and defined as

$$K(x,z)=\phi(x_i)\cdot\phi(x_j)$$

which enable us to obtain the product of high dimension in the low dimension space. In other words, kernel function give us an access to the profit from high dimension mapping without charging us with the massive calculating cost.

### C. Random Forest

Random Forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because its simplicity and the fact that it can be used for both classification and regression tasks.

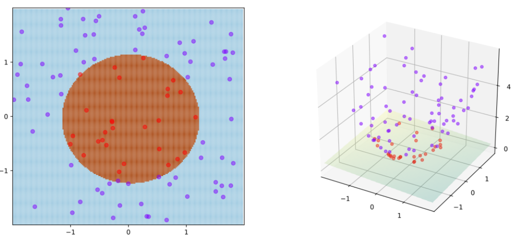


Fig. 6. the mapping process of rbf kernel function

Random Forest is a supervised learning algorithm. Like we can already drawn from its name, it creates a forest and makes it somehow random. The ”forest” it builds, is an ensemble of Decision Trees, most of the time trained with the bagging method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Random Forest is of great importance in ensemble learning, which is conferred with similar attention as GBDT, especially for the simplicity in parallel computing, which is attractive i the environment of huge data sets.

*1) Ensemble Learning:* In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. Unlike a statistical ensemble in statistical mechanics, which is usually infinite, a machine learning ensemble consists of only a concrete finite set of alternative models, but typically allows for much more flexible structure to exist among those alternatives.

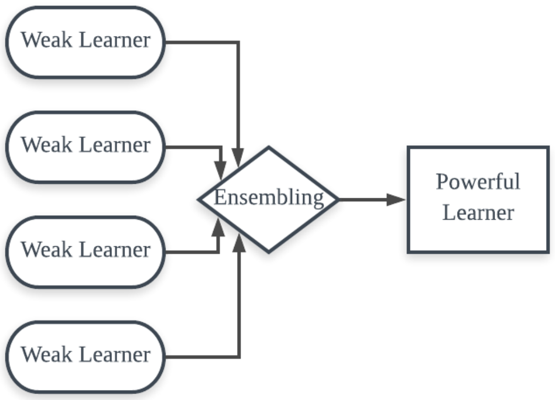


Fig. 7. Structure for Ensemble Learning

Hence we come to two problems in Ensemble Learning, how to get each single learner and how to select an algorithm to ensemble the learners. The single learner selection could be concluded in two types, homogeneous and heterogeneous learner, which is different in whether the learners are in the same type. Nowadays, homogeneous learner selection occupies the mainstream, which is represented by CART decision tree and neural cells. Distinguished by the existence of dependency among the single learner, Ensemble Learning can also be categorized as boosting and bagging.

*2) Boosting:* Boosting is selection of sample is made more intelligently. We subsequently give more and more weight to hard and classify observations. There always exists communication via each single learner.

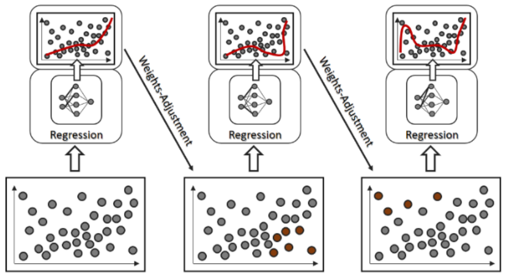


Fig. 8. Structure for Boosting

You can see that we have assigned equal weights to each data point and applied a decision stump to classify them as + (plus) or  (minus). The decision stump (D1) has generated vertical line at left side to classify the data points. We see that, this vertical line has incorrectly predicted three + (plus) as  (minus). In such case, well assign higher weights to these three + (plus) and apply another decision stump.

The most famous boosting algorithm is Adaboosting and Gradient Boosting Tree.

*3) Bagging:* Bagging could be concluded in the following chart

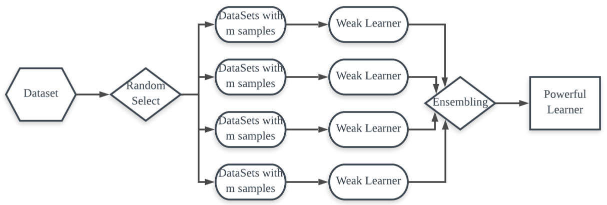


Fig. 9. Structure for Bagging

It should be mentioned that the sampling process in Bagging is put-back sampling and in Boosting, it is non-put-back sampling. In Bagging, for each sample, the probability of being sampled in each sampling process containing m samples is $\frac{1}{m}$, so the possibility of not being sampled is $1 - \frac{1}{m}$, so in the $m_{th}$ sampling process, the probability is $(1 - \frac{1}{m})^m$. When $m \Rightarrow \infty$, the possibility will be $\frac{1}{e}$, so it means about 36.8% data will not be sampled, which is called the data out-of-bag. The out-of-bag data is used to improve the generalization performance of the model, which is not included in the training process.

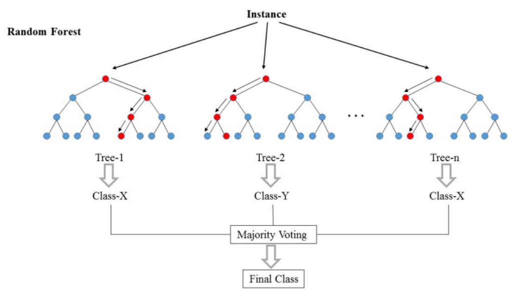


Fig. 10. Structure for Random Forest

*4) Random Forest:* Random Forest is basically a specific type of Bagging Algorithm, which makes improvements in the following two aspects. Firstly, it Random Forest applies CART decision tree in weak learner. Secondly, within each steps, Random Forest arbitrarily selects part of the features as the decision estimators and select the best one as the discriminator of right and left trees, which further improves the generalization performance of Random Forest.

### *D. XGBoost*

XGBoost is used widely by data scientists to achieve state-of-the-art results on many machine learning challenges. The most important factor behind the success of XGBoost is its scalability in all scenarios. The scalability of XGBoost is due to several important systems and algorithmic optimizations. These innovations include: a novel tree learning algorithm is for handling sparse data; a theoretically justified weighted quantile sketch procedure enables handling instance weights in approximate tree learning.

The theory of XGBoost is quite complex, however, we selected this model because of its advantage in handling skewed data, which is a big problem in our project.

### *E. Neural Network*

The concept of Neural Networks (NN) is originally inspired by the goal of modeling biological neural systems, but has since diverged and become a matter of engineering and achieving good results in Machine Learning tasks.

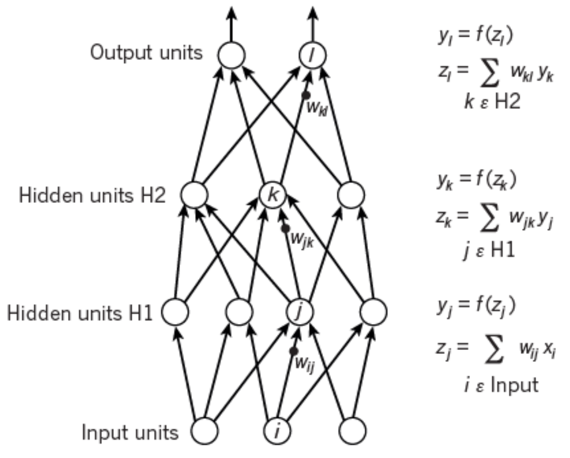


Fig. 11. Neural Network Architecture: feedforward

Many applications of deep learning use feedforward neural network architectures (fig.11), which learn to map a fixed-size input (for example, an image) to a fixed-size output (for example, a probability for each of several categories). To go from one layer to the next, a set of units compute a weighted sum of their inputs from the previous layer and pass the result through a non-linear function. At present, the most popular non-linear function is the rectified linear unit (ReLU), which is simply the half-wave rectifier f(z) = max(z, 0). Back-propagation is used to train the neural network. The gradients of the parameters are computed by chain rule, in order to minimize the cost function.

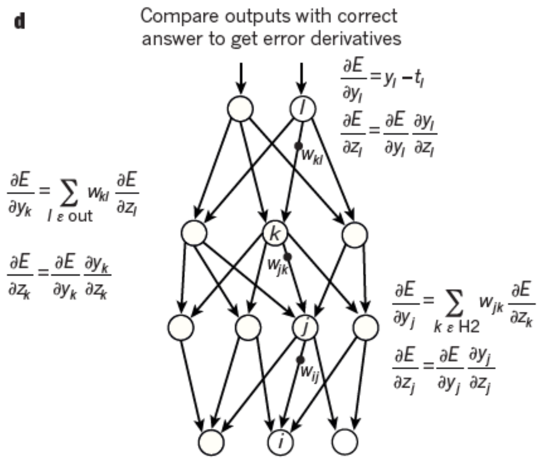

d
Compare outputs with correct answer to get error derivatives
$\frac{\partial E}{\partial y_l} = y_l - t_l$
$\frac{\partial E}{\partial z_l} = \frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial z_l}$
$\frac{\partial E}{\partial y_k} = \sum_{l \in out} w_{kl} \frac{\partial E}{\partial z_l}$
$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z_k}$
$\frac{\partial E}{\partial y_j} = \sum_{k \in H2} w_{jk} \frac{\partial E}{\partial z_k}$
$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_j}$
$w_{kl}$
$w_{jk}$
$w_{ij}$

Fig. 12. Neural Network Architecture: back-propagation

In this project, since we focused on binary classification, we used a simple neural network with one hidden layer. Using a sigmoid function, The output is a real number $y$ in the range (0,1). If $y \geq 0.5$, we mapped it to 1. If $y < 0.5$, we mapped it to 0. Then we could compare our predicted results with the true values.

### *F. Recurrent Neural Network (RNN)*

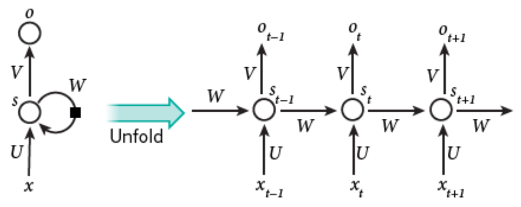


Fig. 13. Recurrent Neural Network

For tasks that involve sequential inputs, such as speech and language, it is often better to use RNNs. RNNs process an input sequence one element at a time, maintaining in their hidden units a state vector that implicitly contains information about the history of all the past elements of the sequence. When we consider the outputs of the hidden units at different discrete time steps as if they were the outputs of different neurons in a deep multi-layer network, it becomes clear how we can apply back-propagation to train RNNs.

## IV. Results

### A. Logistic Regression

The test accuracy, precision, recall and F1-score of logistic regression with different resampling are shown in the table below.

| | accuracy | precision | recall | F1-score |
|---|---|---|---|---|
| 0 | 0.5363 | 0.1014 | 0.5125 | 0.1693 |
| 4 | 0.5258 | 0.5346 | 0.5867 | 0.5595 |
| 8 | 0.4632 | 0.4511 | 0.4222 | 0.4362 |
| 12 | 0.6313 | 0.4935 | 0.4698 | 0.4814 |
| 16 | 0.7429 | 0.472 | 0.4856 | 0.4787 |

TABLE I
Results of Logistic Regression

It can be found that the resampling improves the performance of logistic regression.

### B. Support Vector Machine

Support vector machine performs poorly on the data set. We used different kernels to make the data set separable, but the accuracy and F1-score were low in all cases. The sigmoid and rbf kernel predicted negative for all inputs, while polynomial kernel predicted all positive. Moreover, when the parameter $C = \infty$ for polynomial kernel and $C = 1$ for linear kernel, the training process failed to converge in 20 minutes, which we regarded as an ill-posed problem.

It can be seen from figure 3 that the data forms an inseparable cloud, accounting for the failure of SVM method.

### C. XGBoost

The test accuracy, precision, recall and F1-score of XGBoost with different resampling are shown in the table below.

| | accuracy | precision | recall | F1-score |
|---|---|---|---|---|
| 0 | 0.7459 | 0.2389 | 0.7832 | 0.3661 |
| 4 | 0.7679 | 0.7618 | 0.8042 | 0.7824 |
| 8 | 0.7104 | 0.7053 | 0.7089 | 0.7071 |
| 12 | 0.7964 | 0.6725 | 0.8488 | 0.7505 |
| 16 | 0.7974 | 0.5521 | 0.8827 | 0.6793 |

TABLE II
Results of XGBoost

XGBoost achieved high performace compared with other methods, due to its advantage at dealing with skewed data. We set the parameter 'scale_pos_weight' to the ratio of negative samples/positive samples, so that the training data was balanced.

Another application of XGBoost is analyzing the importance of features. In order to make the picture more legible, we performed PCA to reduce the dimension of features to 30, and obtained the following figure.

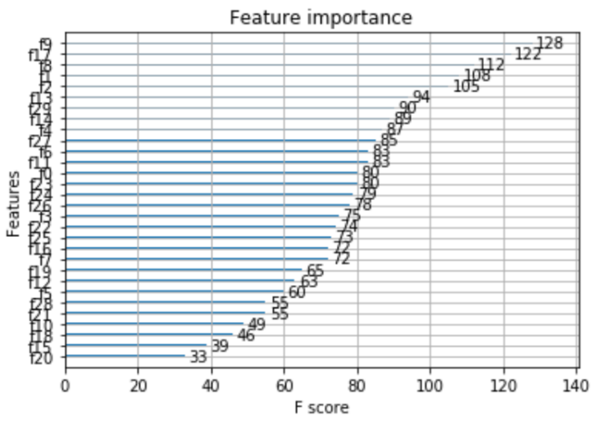


Fig. 14. Importance of features after PCA

It can be seen that feature 9 has the highest F score, thus contributing most to the prediction of price.

### D. Random Forest

The test accuracy, precision, recall and F1-score of Random Forest with different resampling are shown in the table below, which has eminent training and generalization performance.

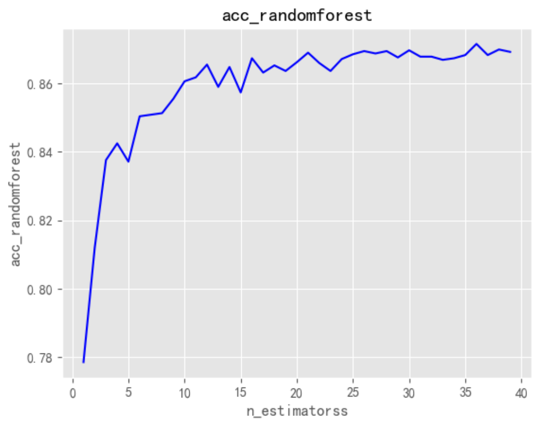


Fig. 15. Accuracy Trend of Random Forest

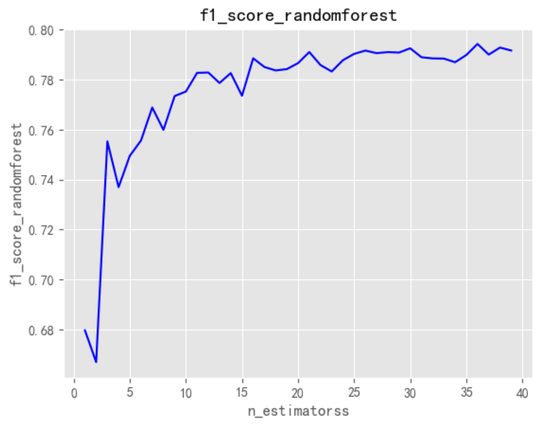


Fig. 16. F1-score Trend of Random Forest

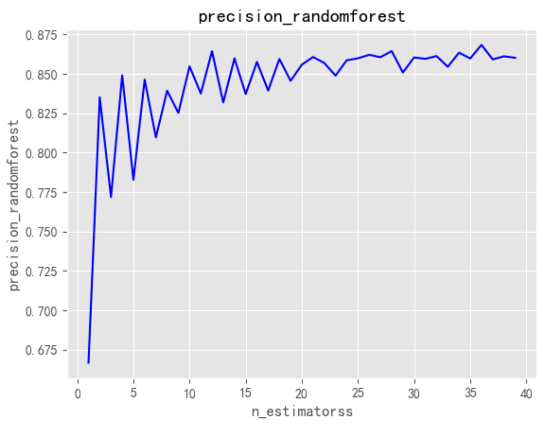


Fig. 17. Precision Trend of Random Forest

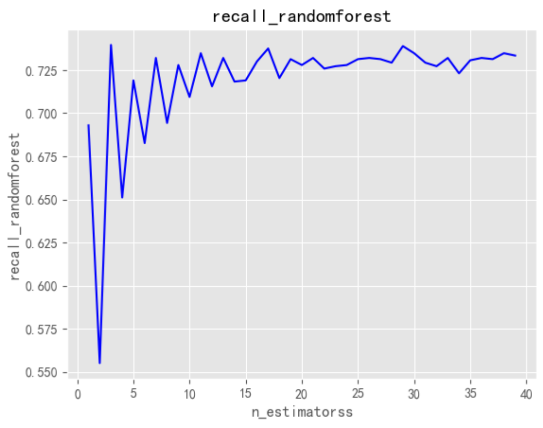


Fig. 18. Recall Trend of Random Forest

With the increasement of n_estimators, the classifier gains greater free degree and achieved higher accuracy.

### *E. Neural Network*

Neural network has a relatively bad performance in our task, due to the fragile structure and initialization problem.

| | accuracy | precision | recall | F1-score |
|---|---|---|---|---|
| 0 | 0.4037 | 0.0622 | 0.6179 | 0.1131 |
| 4 | 0.4962 | 0.4949 | 0.8787 | 0.6332 |
| 8 | 0.5008 | 0.5 | 0.008 | 0.0158 |
| 12 | 0.5048 | 0.5046 | 0.4359 | 0.4677 |
| 16 | 0.2861 | 0.1298 | 0.2905 | 0.1794 |

TABLE III
RESULTS OF NEURAL NETWORKS

Actually the loss function failed to converge given the training data. We tried one hidden layer and two hidden layers with different parameter dimensions, but that didn't improve much.

### *F. Recurrent Neural Network (RNN)*

It seems that the stock market don't really need memory. In other words, the previous patterns do not necessarily have much influence on latter times, at least not convincingly, according to the result of our experiment.

Another problem for RNN is that it tends to overfit. In order to solve the problem of overfitting, the method of random dropout is adopted, however, this makes the experiment not repeatable. Sometimes the results are good, but there is no guarantee that it'll be good next time.

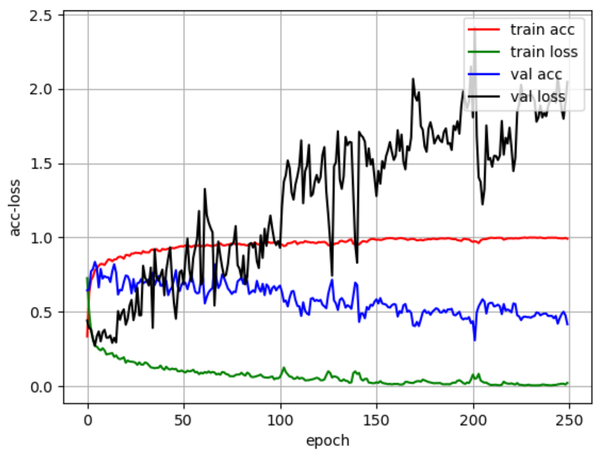


Fig. 19. RNN Overfitting

From this graph, we can see that as the training proceeds, the loss on the training set decreases fast, but the loss on the testing set very soon boosted.

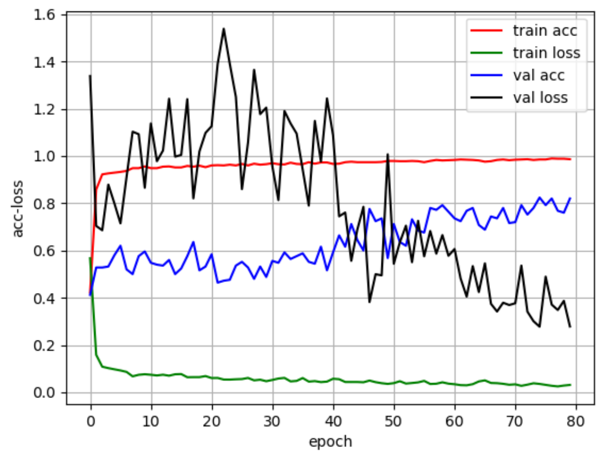


Fig. 20. A well-trained RNN

By adjusting the drop-out rate and epochs of training, we can have a RNN that avoids overfitting, as we can see from the loss curves. Here, in each of the four layers, I set the drop-out rate to be 0.2, which is very aggressive. This example turns out to be satisfactory.

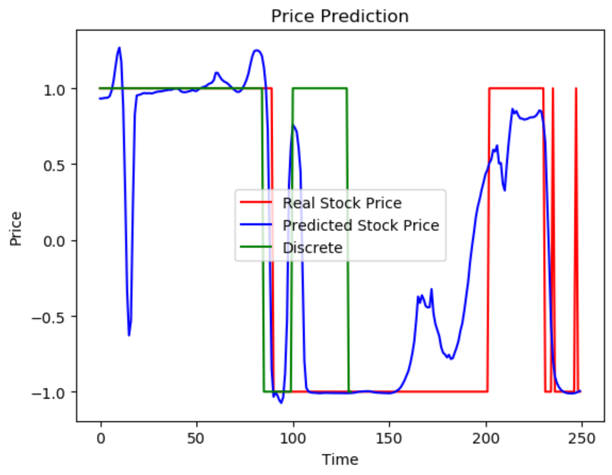
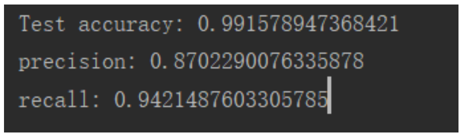


Fig. 21. Prediction results

In this image, the red curve is the prediction result, where 1 indicates a rise in the price, and -1 otherwise. The blue curve is the predicted value. It's trained in a continuous form. And the green line is the discrete form of the prediction, namely, if the prediction result is larger than 0, then the green line take 1, and -1 otherwise. Sometimes we can see only the green line or only the red line, it's because these two lines overlap.

We can see that the blue line can sometimes predict the rises and falls in the stock price.



```
Test accuracy: 0.991578947368421
precision: 0.8702290076335878
recall: 0.9421487603305785
```

Fig. 22. A well-trained RNN

We can see that in this example, the testing accuracy is about 99%, which is very high. The precision is 87%, and the recall is 94%. The corresponding f1 score is 90%. However, such good results are not always repeatable.

## V. Conclusions and Thoughts

The accuracy and f1 score of our models are not really satisfactory. The reasons, we guess, are as follows.

1) In the preprocessing and PCA part, we had to discard some features, which surely lead to decrease in accuracy.
2) The indicators might not be homogeneous. There might be some outliers in the dataset that are not detected.
3) For each timestamp, there are 37 indicators. If we use, say, 100 timestamps as the training set, we only have 37 labels, but we have 3700 features. The fact that features far outnumber labels could easily lead to overfitting.
4) The data is skewed. Most of the prices rise between two consecutive timestamps. Thus some models like SVM predicted all outputs to be negative and have very low F1-score. While the models that can handle this situation, like random forest and XGBoost, achieve both high accuracy and F1-score.

## VI. Acknowledgement

Sincerely thanks to Prof. Bo Yuan and Teaching Assistant Geng Zichen for their patient guiding and assistance during our learning process. This project report is successfully produced and we benefited a lot from this course project.

## VII. Distribution of work

Shen Zhe: Logistic Regression, Neural Network, XGBoost, Project Report
Zhang Sijun: SVM, Random Forest, Preprocessing, PCA, Project Report
Zhao Hantao: DataSet Preprocessing, PCA, RNN, Project Report

## References

[1] Bengio, Yoshua, Ian J. Goodfellow, and Aaron Courville. "Deep learning." An MIT Press book. (2015).

[2] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature 521.7553 (2015): 436-444.