UM–SJTU Joint Institute

Introduction to Machine Learning
(VE445)

# Stock Market Prediction

## VE445 Course Project Report

Name: Wu Guangzheng ID: 515370910014
Name: Wang Hanqin ID: 5140809063
Date: 1 May. 2019

# 1 Introduction

This project is meant to practice your ability of handling actual business problems about financial transaction. There is only one task to predict the rise and fall of the price, and one should apply as many as models and algorithms as you can including supervised learning and unsupervised learning.

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit. The efficient-market hypothesis suggests that stock prices reflect all currently available information and any price changes that are not based on newly revealed information thus are inherently unpredictable. Others disagree and those with this viewpoint possess myriad methods and technologies which purportedly allow them to gain future price information.

# 2 Data Pre-processing

In this section, we would like to talk about how we pre-process the data for our training. We will talk about our methodology in the section, and provide a general view for the whole dataset, and show our understanding of it.

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | indicator1 | indicator2 | indicator3 | indicator4 | indicator5 | indicator6 | indicator7 | indicator8 |
| 2 | 1.00009 | -0.08651 | -1374.48 | -2.07612 | 0.000178 | 0.999911 | 0.998495 | 0.75 |
| 3 | 1.00009 | -0.08651 | -1374.48 | 0 | 0.000178 | 0.999911 | 0.998506 | 0.5 |
| 4 | 1.00009 | -0.08651 | -1374.48 | 0 | 0.000178 | 1.00009 | 0.998516 | 0.5 |
| 5 | 1.00009 | 0 | 490.311 | 1.03806 | 0.000178 | 1.00009 | 0.998703 | 0.666667 |
| 6 | 1.00009 | 0 | 571.371 | 0 | 0.000178 | 1.00009 | 0.998712 | 0.5 |
| 7 | 1.00009 | 1.41129 | 8503.23 | 0 | 0.000178 | 1.00009 | 0.998721 | 0.5 |
| 8 | 0.999911 | 1.00806 | 6239.11 | 1.00806 | 0.000178 | 1 | 0.998907 | 0.5 |
| 9 | 1.00009 | -0.40323 | -1690.32 | -2.01613 | 0.000178 | 1 | 0.998739 | 0.95 |
| 10 | 1.00009 | -0.30242 | -1123.79 | 0 | 0.000178 | 1 | 0.998749 | 0.5 |
| 11 | 1.00009 | -0.50403 | -2256.45 | -0.40323 | 0.000178 | 1 | 0.998757 | 1 |
| 12 | 1.00009 | -0.50403 | -2256.45 | 0 | 0.000178 | 1 | 0.998766 | 0.5 |
| 13 | 1.00009 | -0.70565 | -3388.91 | 0 | 0.000178 | 1 | 0.998773 | 0.5 |
| 14 | 1.00009 | -0.20161 | -557.258 | 0 | 0.000178 | 1 | 0.99878 | 0.5 |
| 15 | 1.00009 | -0.40323 | -2769.35 | -0.20161 | 0.000178 | 1 | 0.998787 | 0.5 |
| 16 | 1.00009 | -0.59289 | -3824.9 | -0.59289 | 0.000178 | 1.00004 | 0.998794 | 0.8 |
| 17 | 1.00009 | -0.79051 | -4935.18 | -0.59289 | 0.000178 | 1 | 0.9988 | 1 |
| 18 | 1.00009 | -0.70281 | -4450.4 | 0 | 0.000178 | 1 | 0.998807 | 0.5 |
| 19 | 1.00009 | -0.60484 | -3902.02 | 0 | 0.000178 | 1 | 0.998814 | 0.5 |
| 20 | 0.999911 | -0.49801 | -3296.31 | 0.199203 | 0.000178 | 1 | 0.998998 | 0.5 |

| DE | DF | DG | DH | DI | DJ | DK | DL |
|---|---|---|---|---|---|---|---|
| midPrice | UpdateTim | UpdateMil | LastPrice | Volume | LastVolum | Turnover | LastTurnov |
| 5617.5 | 9:01:06 | 395 | 5617 | 3240 | 24 | 1.82E+08 | 1.35E+06 |
| 5617.5 | 9:01:06 | 637 | 5617 | 3240 | 0 | 1.82E+08 | 0 |
| 5617.5 | 9:01:06 | 943 | 5617 | 3244 | 4 | 1.82E+08 | 224700 |
| 5618.5 | 9:01:07 | 185 | 5618 | 3256 | 12 | 1.83E+08 | 674220 |
| 5618.5 | 9:01:07 | 340 | 5618 | 3256 | 0 | 1.83E+08 | 0 |
| 5618.5 | 9:01:07 | 684 | 5618 | 3256 | 0 | 1.83E+08 | 0 |
| 5618.5 | 9:01:07 | 747 | 5619 | 3266 | 10 | 1.84E+08 | 561900 |
| 5618.5 | 9:01:08 | 156 | 5618 | 3286 | 20 | 1.85E+08 | 1.12E+06 |
| 5618.5 | 9:01:08 | 234 | 5618 | 3286 | 0 | 1.85E+08 | 0 |
| 5618.5 | 9:01:08 | 614 | 5618 | 3290 | 4 | 1.85E+08 | 224720 |
| 5618.5 | 9:01:08 | 911 | 5618 | 3290 | 0 | 1.85E+08 | 0 |
| 5618.5 | 9:01:09 | 368 | 5618 | 3290 | 0 | 1.85E+08 | 0 |
| 5618.5 | 9:01:09 | 629 | 5618 | 3290 | 0 | 1.85E+08 | 0 |
| 5618.5 | 9:01:09 | 902 | 5618 | 3292 | 2 | 1.85E+08 | 112360 |
| 5618.5 | 9:01:10 | 189 | 5618 | 3302 | 10 | 1.86E+08 | 561820 |
| 5618.5 | 9:01:10 | 613 | 5618 | 3308 | 6 | 1.86E+08 | 337080 |
| 5618.5 | 9:01:10 | 913 | 5618 | 3308 | 0 | 1.86E+08 | 0 |
| 5618.5 | 9:01:11 | 191 | 5618 | 3308 | 0 | 1.86E+08 | 0 |
| 5618.5 | 9:01:11 | 406 | 5619 | 3310 | 2 | 1.86E+08 | 112380 |

Figure 1: Indicators and other important things of the data

The data contains a large amount of features named indicators, which are data that has already been processed, and we can use them directly. Besides, we also have other features with names, including mid prices and sell time, and so on. Since with these names, we can have a better understanding of these parts of data, and trying to find the data leakage within them, to help us establish a better model for machine learning.

## 2.1 Data Reprocessing

We reprocess the data by deducting the data at time step $t$ by the data at time step $t-10$ and the data at time $t-60$, and concat them together as the input data for most of the methods. The reason for this is that the absolute value of some data is somehow meaningless, but we need to know how the data changes. On the other hand, in general, time step 10 and 60 is generally chosen by the field, so we just simply applied this same strategy. Besides, we drop the feature "time", which will remain the same after doing this reprocessing. The result of reprocessed data is a data with 276 features.

## 2.2 Data Sampling

Because of the limited time, we need to sample the data as well, in order to reduce the time needed for training. In this project, we sampled 30000 samples out of the whole image for training. And then the data is split into a training set of size 21000 and a validation set of size 9000, for most of the methods mentioned afterwards.

## 2.3 Data Labelling

Since the data given is just raw data, and we need to label them in order to train the model with the data, or validate how good our data really is. We label the data "rise" if the data at time step $t$ has a lower mid price than the data at time step $t + 10$, and fall otherwise.

# 3 Methods

In this section, we would like to describe some algorithms we used to handle this dataset. We tried Support Vector Machine, K Nearest Neighbours, and Logistic Regression. Besides, we also tried a generally used way in the field of business, named moving average, and consider this way as the baseline of our project.

## 3.1 Moving Average

Let's start with the most general strategy in the field of stock market prediction: moving average. This strategy has long been considered as an efficient way of predicting the rise and fall of the stock, and it is very simple. The idea of this is to find out the average middle price of the recent 10 days and recent 60 days, and if the average of recent ten days is higher than the recent 60 days, then the stock will have a potential to rise, and otherwise fall.

Figure 2: Moving Average Sample Graph

In the sample graph, we can see that this stock has a higher average in the short term than in the long term, then we can simply predict that this stock has a higher potential to rise.

Let's move back to our problem. We wrote a program to run the baseline using the moving average algorithm. And we reached an accuracy of 54.06%. Since this algorithm predicts rise and fall without using the true "rise or fall" value, so we do not need to split the training set and the

validation set, so we test the accuracy via the whole system. The accuracy of 54.06% is higher than 50%, which proves that using this kind of strategy, one can do earn extra money, and this kind of method is efficient, and really do find some data leakage.

## 3.2 K Nearest Neighbours

K Nearest Neighbours is a kind of algorithm that first define a distance, and for each data point in the validation set, find the nearest k points in the training set, and let them vote for the group of this point.

```
Till Interation  3100 , the correct rate is  0.7800709448564979
Till Interation  3200 , the correct rate is  0.7822555451421431
Till Interation  3300 , the correct rate is  0.7861254165404423
Till Interation  3400 , the correct rate is  0.7924139958835636
Till Interation  3500 , the correct rate is  0.7914881462439303
Till Interation  3600 , the correct rate is  0.7917245209663982
Till Interation  3700 , the correct rate is  0.7895163469332612
```

Figure 3: KNN Result

In this project, we just measure the basic KNN implementation, and we can change the constraint for votes, so that improve the KNN model. It is very unlucky that we met some unknown problem when dealing with this model, so we only test some of the validation set. But since it is converging, we can easily predict that the final result will be close to 78%. The following graph shows the relation between KNN and iterations. The accuracy converges.

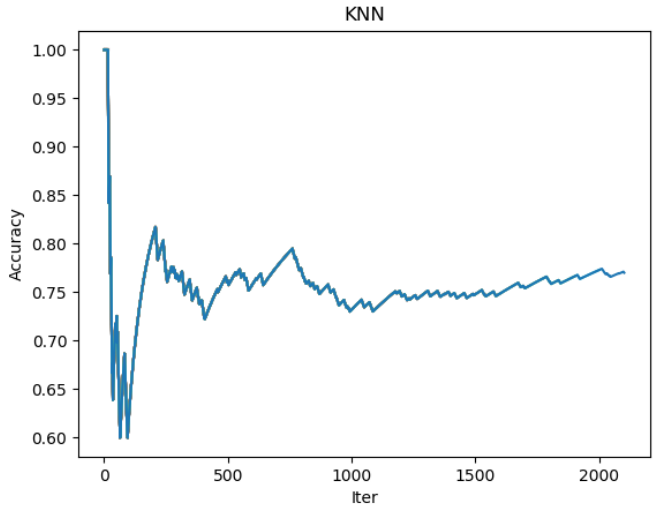


Figure 4: KNN Result v.s. Iterations

## 3.3 Logistic Regression

We also tried the logistic regression model for stock market prediction. We split the model into 7 : 3, and train the model with simple logistic regression. The following graph shows the result of the logistic regression. We get an accuracy of 75.05% after training the data with 67 epochs.

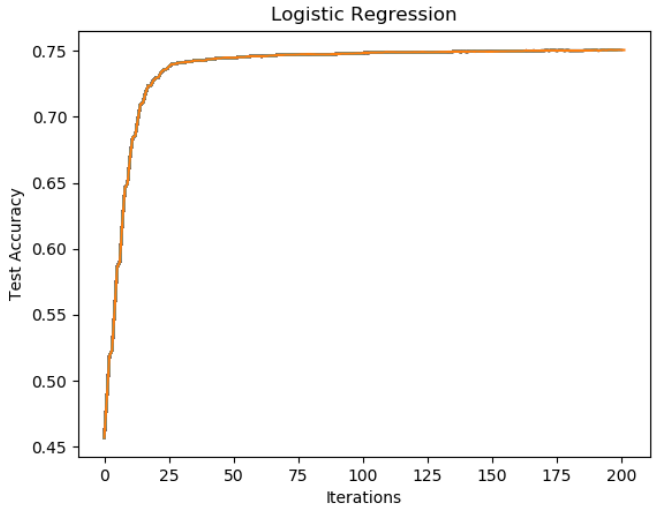


Figure 5: Logistic Regression Result v.s. Iterations

```
At epoch  63 , and iter at  0  the accuracy is  0.7503333333332467
At epoch  63 , and iter at  10000  the accuracy is  0.7505555555554688
At epoch  63 , and iter at  20000  the accuracy is  0.7501111111110246
At epoch  64 , and iter at  0  the accuracy is  0.7503333333332467
At epoch  64 , and iter at  10000  the accuracy is  0.7506666666665799
At epoch  64 , and iter at  20000  the accuracy is  0.7503333333332467
At epoch  65 , and iter at  0  the accuracy is  0.7506666666665799
At epoch  65 , and iter at  10000  the accuracy is  0.7506666666665799
At epoch  65 , and iter at  20000  the accuracy is  0.7505555555554688
At epoch  66 , and iter at  0  the accuracy is  0.7505555555554688
At epoch  66 , and iter at  10000  the accuracy is  0.7506666666665799
At epoch  66 , and iter at  20000  the accuracy is  0.7505555555554688
At epoch  67 , and iter at  0  the accuracy is  0.7505555555554688
At epoch  67 , and iter at  10000  the accuracy is  0.7506666666665799
```

Figure 6: Accuracy of Logistic Regression

## 3.4 Support Vector Machine

Since the data size is really large, so calculating a $O(N \times N)$ kernel matrix is a waste of memory. We apply the I-SVM algorithm in this method, and split the training set into small sets of size 500, and only remains the support vectors and train the following small sets with the data in that set and the support vectors remained. We also use a Gaussian Kernel with parameter 5.0, and a soft margin with parameter 1.0 in it. Generally, the SVM model works good with these two parameters. If given more time, we can try more parameters and try to improve the results of SVM.

Let's see the results here, shown in the graph above. The training accuracy is 75.3%. SVM here might not be a very good idea, because we use the L2 norm in SVM, but however, the L2 norm might not be a good representation for the data, since using L2 norm will miss the constraint of different features, and also if some of the features has no relation to rise and fall, then this feature will affect the result and reduce the accuracy. But finally, it does work better than a simple moving average strategy.

```
start iteration number:  20
     pcost       dcost       gap    pres   dres
 0: -1.3470e+03 -2.2497e+03  1e+04  2e+00  2e-16
 1: -8.6700e+02 -1.6131e+03  8e+02  2e-02  4e-16
 2: -1.0487e+03 -1.0917e+03  4e+01  1e-03  3e-16
 3: -1.0826e+03 -1.0832e+03  6e-01  1e-05  4e-17
 4: -1.0830e+03 -1.0830e+03  6e-03  1e-07  2e-16
 5: -1.0830e+03 -1.0830e+03  6e-05  1e-09  2e-16
Optimal solution found.
[ 9.99999873e-02 -1.99454760e-23 -1.99454760e-23 ...  1.28040969e+00
  1.28040969e+00  1.28040969e+00] 1.280409692590178
0.753
```

Figure 7: SVM result

## 3.5 Random Forest

We also tried the random forest algorithm to implement the stock market prediction system. Random forest contains a set of decision trees, and each decision tree is only given some parts of the input. The output is voted among all the decision tree. The structure of a random forest is shown below.

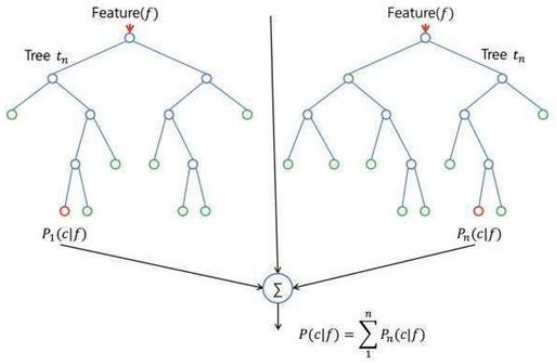


Figure 8: Structure of Random Forest

We trained the training set with random forest classifier with sklearn, and test the validation set. The separation of the data is shown in the former section. We finally get an accuracy of 71.94%, and since we didn't try extra parameters with random forest, with only the default values, so the accuracy is somehow very high.

We also tried to show the importance of different features with the help of the random forest. It does show that some indicators, not the later features, play a very important role. And 60-step is more important than 10-step.

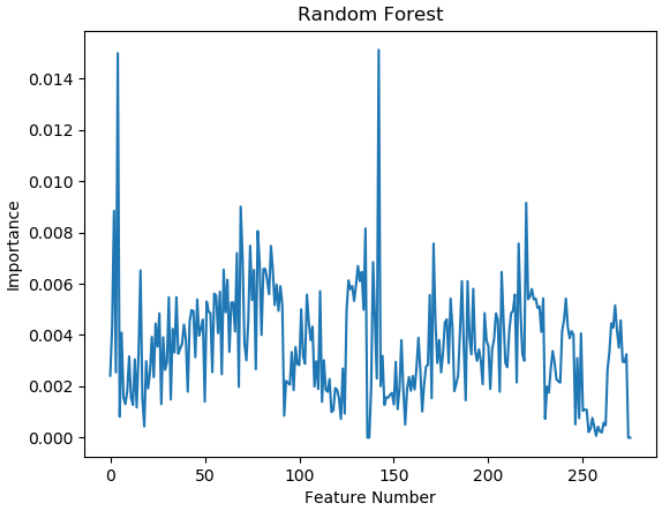


Figure 9: Importance Result from Random Forest

## 3.6 K-means

We can also try some unsupervised learning algorithms such as K-means. We can ignore the label at very beginning, and try the K-means to separate it into two clusters. And then we let every point in the same cluster vote for the label of the whole cluster, and then determine the accuracy of the whole system.

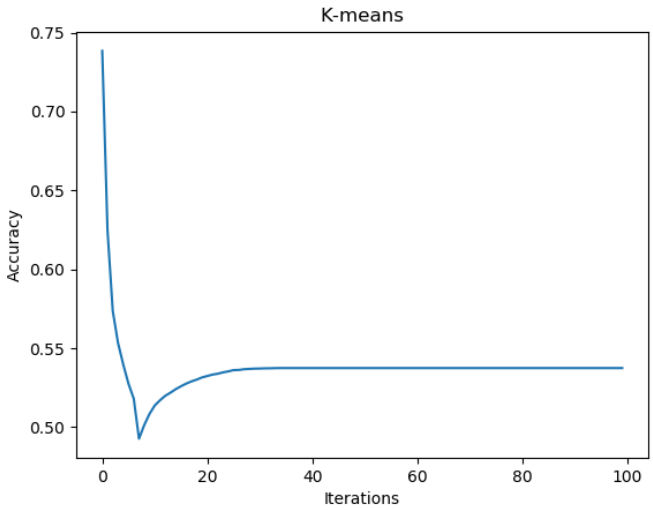


Figure 10: K-means result

The accuracy of the K-means algorithm is 53.743%.

The result of the whole K-means system is not very good, with a final result just a little bit better than our baseline. We guess that the reason for this is that we use the L2 norm to determine the distance between each point, so we cannot add some constraints for any of the feature, so features worth equally. But actually, some features should be more important than

others, and that makes the difference.

Another thing is that, K-means, as well as KNN, uses the L2 norm, and we do normalize the data for both the algorithms. But KNN's result is far better than K-means. We guess that it is the data that is formed in some style, where the main feature that separate it into two parts is not something that really affects rise and fall, but some second important features do affects the rise and fall. So if we only consider a small cluster with only five data points in it, the second important feature will play a far more important role, and then improve the accuracy.

# 4 Conclusion

We've tried various methods onto this stock market prediction, and got some insight from the results. In general, those machine learning methods mainly enjoys a same level of accuracy, around 75%, and the difference might be caused by the parameters, and if we tried to improve the parameters, the result might be even better.

The unsupervised learning is actually worse than other supervised algorithms, and the reason for this might be the difference of importance of different features. As shown in the random forest part, different features should have different constraints, which makes the L2 norm not so fit for the model.

The tradition baseline is somehow useful, which remains a positive expectation according to the result. The moving average do find some data leakage. But on the other hand, we found that for the feature importance, some indicators are the most important ones, but we only receive data with name "indicator", so we cannot know where this indicator comes from.

# 5 Discussion

We will first try to improve the parameters for each model, and try to use more data to train models. In addition, we may try more models such as RNN.

# 6 Resources


1. VE445 Course Project Introduction.
2. Wikipedia Page of Stock Market Prediction.