Stock Futures Market Movement Prediction

Yansen HAN, Yichen ZHOU, Yuntao ZHANG, PANDA Soonam Kalyam, Yi LUO
The Hong Kong University of Science and Technology

Abstract

At present, stock prediction is still a tough problem. However, plenty of researchers still research on it. In our report, we have applied statistical models (VAR and ARIMA) and classical machine learning methods (Random Forest, Support Vector Machine, and Neural Networks) to forecast stock movement. In statistical models, we predict the price of next time. In machine learning models, we defined some stock labels and then trained several classification models. We empirically find that predicting the exact price of next time is still very hard by using time series models, while the classification results is very good, and we can achieve an accuracy of 70% or so with the unsupervised labels. Therefore, our report points out that after fine process the labels and features, we can make good predictions even in low information-noise ratio problems, the prediction of stock movement.

1 Data Preprocessing

1.1 Data Smoothing

Normally, the financial time series is embedded with high level of noise (random trading behaviors). If the raw noisy time series is less denoised, the prediction model performs poorly due to the high level noise; if the noisy time series is over denoised, the filtered time series loses some genuine information of raw time series. Traditionally, There are three kinds of denoising methods, Kalman filters, Wiener filers and traditional filters. All of these filters are based on an assumption that signal information is smooth while noise is not. Therefore, we can filter the noise from our data. However, Financial time series data break this assumption and make filters not work or scrape some real information. Fortunately, the appearance of wavelet decomposition overcome this drawback, because it has good properties, like self-adopted and focus, which enable wavelet decomposition project different signals into different frequency channel. After wavelet smoothing, we can obtain highly informative-noise-ratio data.

In this part, we use Discrete Wavelet Transform (DWT) to denoise our data. Within a Discrete Wavelet Transform (DWT), we map $f \to w$ from the signal domain to the wavelet coefficient domain, or in other words we apply the transformation $w = Wf$, such that one obtains the coefficients for fine scales, which capture high frequency information, and those for coarse scales, which capture low frequency information.

Therefore, a sequence of smoothed signals and of details giving information at finer resolution levels, may be used to represent a signal expansion:

$$f(x) = \sum_k c_{j0,k}\phi_{j0,k}(x) + \sum_{j>j_0}\sum_k d_{j,k}\psi_{j,k}(x)$$

where $\phi_{j0,k}$ is a scaling function with the corresponding coarse scale coefficients $c_{j0,k}$ and $d_{j,k}$ are the detail (fine scale) coefficients, i.e., $c_{j,k} = \int f(x)\phi_{jk}(x)dx$ and $d_{jk} = \int f(x)\psi_{jk}(x)dx$.

In our processing, we reconstruct our data by only using the coefficients of level 5. After implementing wavelet decomposition, we obtain smoother data (Figure 1):

Figure 1: Denoised Stock Price

2 Financial Feature Engineering

In this section, we will introduce parts of our features. There are a lot of features you can find online, but all of those features are technical features and have high correlations. Therefore, we just choose some classical features. In the following, We will list several of our features' formula (Note: the bars below means the frequency we choose):

- Stock gains $i-th$ on bar:

$$rise_i = \frac{PRICE^i_{close} - PRICE^{i-1}_{close}}{PRICE^{i-1}_{close}} \times 100\% \quad (1)$$

- Stock gains on 10 bars:

$$rise_i = \frac{PRICE^i_{close} - PRICE^{i-10}_{close}}{PRICE^{i-10}_{close}} \times 100\% \quad (2)$$

- K-line values on i_th day:

$$K-line_i = \frac{PRICE^i_{close} - PRICE^i_{open}}{PRICE^i_{high} - PRICE^i_{low}} \quad (3)$$

3 Methodology

3.1 Regression: Time Series Models

3.1.1 Vector Autoregression (VAR)

In this section, I will introduce the background of the VAR models at the beginning, then why I chose VAR model, how I built VAR model, the forecast result from the model, and lastly my evaluation on the models results.

VAR, vector autoregression, models was publicly supported by Christopher Sims, an American econometrician and macroeconomist; indeed, he also criticized the claims and performances of earlier modeling in macroeconomic econometrics. VAR models had previously used in time series statistic and in statistical control theory; moreover, VAR models provide a theory-free method to estimate economic relationship, which let VAR models be an alternative to the incredible identification restrictions in structural models. For example, health research for dairy data or sensor datas automatic analysis heavily use VAR models to conduct the analysis.

In the analysis of multivariate time series, VAR model is one of the most flexible, useful, successful, and easy to models; in fact, VAR model is useful and helpful in describing the dynamic behavior of economic and forecasting financial and economic markets. Since VAR models can be made conditional on the potential future paths of specified variables, VAR models results (forecasts) are very flexible. This made me choose VAR model as one of models in our project.

Before building the model, the feature engineering was conducted by calling Seaborn.heatmap function. I deployed all the features in our datasets to complete the feature engineering. The heatmap of features correlation is shown in Figure 2:

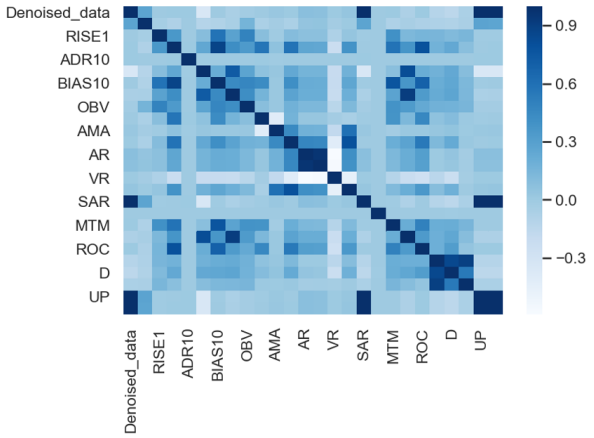

Figure 2: Heatmap of Features Correlation: The heatmap shows the relationship between how a pair of features relate to each other. The darker colour means the stronger relationship, and vice versa.

For building the VAR model in this project, I employed an vector autoregressive model of order 5, denoted as VAR(5). The two time series variables I modelled are price and volume, denoted by $price_t$ and $volume_t$. The VAR(5) model is as follow:

$$\begin{aligned} price_t &= \alpha_1 + \phi_{1,1} \cdot price_{t-1} + \phi_{1,2} \cdot volume_{t-1} \\ &\quad + \cdots + \phi_{1,9} \cdot price_{t-5} + \phi_{1,10} \cdot volume_{t-5} + w_{t,1} \\ volume_t &= \alpha_2 + \phi_{2,1} \cdot price_{t-1} + \phi_{2,2} \cdot volume_{t-1} \\ &\quad + \cdots + \phi_{2,9} \cdot price_{t-5} + \phi_{2,10} \cdot volume_{t-5} + w_{t,2} \end{aligned}$$

Each variable is a linear function of values from lag 1 to lag 5 for all variables. $\alpha_1, \alpha 2$ and $\phi_{i,j} (i \in \{1,2\}, j \in \{1,\cdots,10\})$ are regression coefficient. $w_{t,1}$ and $w_{t,2}$ are white noise which are independently distributed with a normal distribution. we divided the chronological data into two parts, where data at last 100 time point forms the testing set, and data at previous time points forms the training set. After we fitted the VAR model to the training set and use the trained model to make prediction for the testing data, we obtained the result (see 3):

The above result shows different specific price at the specific timepoint; moreover, the blue line means the ground truth and yellow line means the forecast result from VAR model.

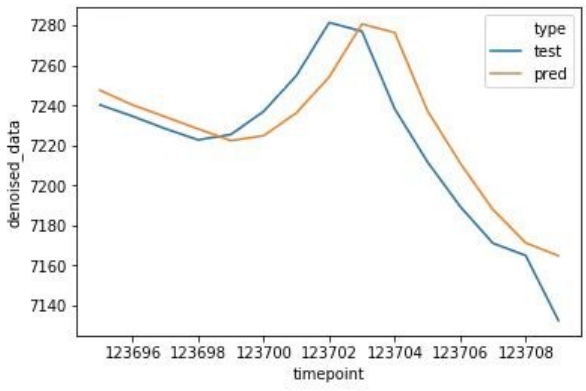

Figure 3: Prediction of VAR model

After completing the forecast through VAR model, I found out that the model did not work well on our datasets. As shown on the result, the prediction line just parallelly moves to right compared to the ground truth. This is because $\phi_{1,1} = 1.0002, \phi_{1,2} = -6.004e-02, \phi_{1,3} = -2.1421e-04, \phi_{1,4} = 5.3688e-02, \phi_{1,5} = -1.1009e-04, \phi_{1,6} = 1.2974e-02, \phi_{1,7} = -2.5114e-05, \phi_{1,8} = 3.4680e-02, \phi_{1,9} = -1.6254e-05, \phi_{1,10} = 2.8372e-02$ when I ran the model. In fact, after running the model, only ϕ_1,1has significant influence from $price_{t-1}$, and others have no significant influence. Thus, the forecast result approximately just generated T timepoint from T-1. In other word, from the learning result, given the T-1 timepoint data, other timepoints' data (price) have no use for the prediction outcome.

3.1.2 Autoregressive Integrated Moving Average (ARIMA)

Before applying ARIMA model on our data we did data analysis using different visualisation and statistic techniques and found it contains an upward trend and seasonality. That means the data does not have constant mean and variance. The ARIMA model cant applied to our data and we need to remove those trend and seasonality. We used following techniques to verify if our data contains any trend or seasonality.

1. Plotting Rolling Statistics : Visual representation
2. Augmented DickeyFuller Test(ADF) : Statistical test that give information regarding critical values and test statistics.

If the Test Statistics is less than the Critical Value, we can reject the null hypothesis and say that the series is stationary.As our data contains some trend we need to apply some Aggregation , Smoothing or Polynomial fitting.We calculated the rolling mean and standard deviation and plotted the data and still found an upward trend in the data.

The test statistics is higher than critical values which shows that it contains trend.

Then we applied different techniques to remove the trend and every time we apply one transformation to our data we need follow the above two methods to check our data if it contains some trend or seasonality.

1. Apply logarithm and check rolling mean and standard deviation (see Figure 4).
2. Apply exponential weighted average and verify data(see Figure 5).
3. Apply Sessional differencing (see Figure 6)
4. Apply one left shift of series.

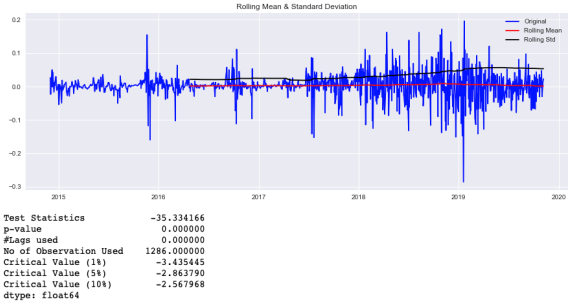

Figure 4: Apply logarithm and check rolling mean and standard deviation

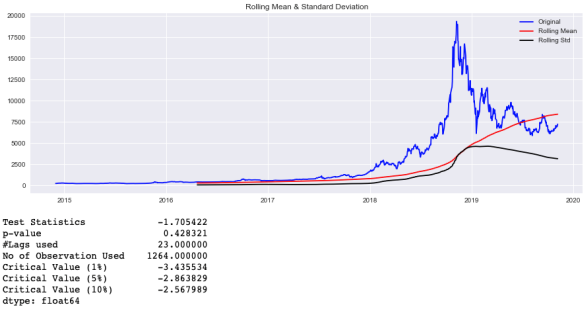

Figure 5: Exponentiation

We found after applying so many techniques the left shift of the series gives a better statistic.The result looks good, test statistics is less even p-value.

Now we can apply the ARIMA model. ARIMA model(p,d,q) is combination of p (Auto Regression part) + d (Difference after shift part) + q(Moving average). p and q values can be found from the ACF and PACF graphs of ARIMA model.and it shows that p and q are nearly 1 and 1 respectively. But we need to take these

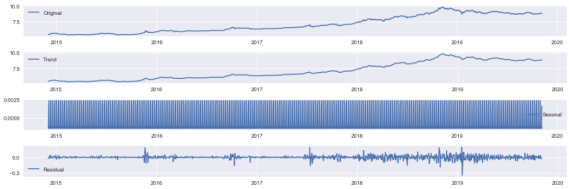

Figure 6: Seasonal Decomposition

hyperparameters and try different cases. Finally, the prediction is shown in Figure 7.

Figure 7: Predictions

3.2 Classification: Machine Learning Models

3.2.1 Unsupervised pattern recognition

In the machine learning model, stock prices of the early trade days are used to predict whether the predefined shape will appear in the whole fixed duration. Here the fixed duration is defined as pattern duration denoted as PD, the duration of stock prices used for constructing train samples is defined as training duration denoted as TD. The correlation between PD and TD is illustrated as the figure 8. After PD and TD are set, we slide a window of length PD on the original dataframe which contains time series of stock prices. Then we get a large number of frames containing prices information in a fixed length duration PD. Next we can label each frame by applying pattern recognition algorithm. Finally training samples are extracted from the first few rows of these frames where the number of rows is TD. And the label of each training sample is the label of corresponding frame in the pattern duration.

- **Introduction of classes:** Since our project aims to predict the tendency of stock prices, six classes are defined in our model based on the shapes of close prices of a stock, i.e., the price will (1) rise up steadily (Continuous Up), (2) rise up more and more rapidly (Slides Up), (3) drop down steadily (Continuous Down), (4) drop down more and more rapidly (Slides Down), (5) stay approximately the same (Flat), (6) fluctuate without predictable tendency (Unknown).

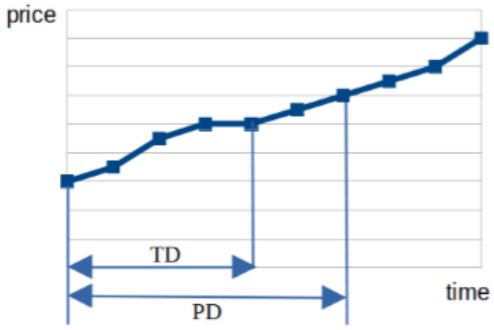

Figure 8: Pattern Duration and Training Duration

- **Unsupervised labeling algorithm:** To classify different shapes of close prices automatically, some labeling algorithms were applied on the close prices in the pattern duration to label each frame of stock prices. In this report we will not introduce algorithms of identifying continuous down and slides down labels since the continuous up and slides up labels are mirrors of the previous two. Algorithm for recognizing continuous up is relatively simple. First, we calculate the growth of stock price in the pattern duration by comparing the close price on the first day and last day. Then we can get four points A, B, C, D shown in the Figure 9, where AC, BD denote the price vibrating between the range $[-\alpha G, \alpha G]$ (e.g., α=0.5) of first days price and last days price respectively. Finally if a proportion larger than δ (e.g., δ=0.95) of points of the period are located in the area ABCD, the duration of stock data will be labeled as continuous up. Flat pattern recognition is similar with the algorithm above, if the growth is less than a threshold and the proportion of points in the pattern period located in the area ABCD is larger than δ (e.g., δ=0.95), then it can be classified as flat pattern.

3.2.2 Random Forest

We use a database containing stock prices with 15 minutes time interval between neighbor records as well as the technical indices of stock market. The time span of our database is within the range from February 22, 2015 to September 3, 2018.

We first measure accuracy of test samples with different combinations of PD-TD. We generated training samples and labels by applying slide window method on the dataframe, which only contain open, close, high and low

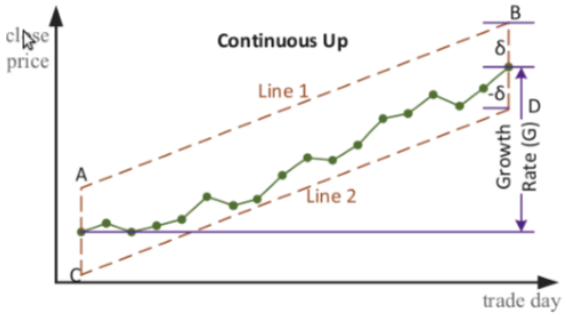

Figure 9: Continuous Up

PD-TD	10-6	15-10	20-13	30-20
Accuracy	0.520	0.590	0.629	0.653
PD-TD	40-26	50-33	60-40	avg.
Accuracy	0.693	0.714	0.738	0.648

Table 1: Results of Random Forest

prices of the duration of PD*15 minutes.then use 70% of the training samples and labels to train the random forest model. And the remaining 30% are used as test data. Prediction accuracy is calculated by validating the test data. The results are listed in the table 1.

Feature selection: Initially only four features of the stock market are used to train the model, which are the close price, open price, high price and low price. To achieve better performance of the model, We extend the features with technical indices to train and evaluate the model. Since there are totally over 30 features in our dataframe, it is time-consuming to evaluate every combination of features. We use *Forward Sequential Search method*, which evaluate the model by adding one feature into the original feature set in one iteration and cannot go back. It may not find the optimal feature combination but has high search speed. The table below shows part of experiment result with different combinations of features when the PD-TD is fixed as 30-10. From the experiments, we find that the recommended feature combination is ['$denoised_data$', 'BIAS10', 'AMA', 'AR', 'WAWVAD', 'up', 'close', 'high', 'low', 'open', 'sar', 'dn', 'ADR10', 'TRMA', 'RSI10']. In fact the prediction accuracy of the combinations are so close that there may not exists a global best feature combination which can be applied in all cases based on different data sets. We can only find the relatively better feature combination for our model.

3.2.3 Support Vector Machine

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high-or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

Whereas the original problem may be stated in a finite dimensional space, it often happens that the sets to discriminate are not linearly seperable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot product of pairs input data vectors may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $k(x,y)$ selected to suit the problem. The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant, where such a set of vector is an orthogonal (and thus minimal) set of vectors that defines a hyperplane.

With this choice of a hyperplane, the points in the feature space that are mapped into the hyperplane are defined by the relation: $\sum_i \alpha_i k(x_i,x) =$ constant Note that if $k(x,y)$ becomes small as y grows further away from x, each term in the sum measures the degree of closeness of the test point to the corresponding data base point . In this way, the sum of kernels above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated. Note the fact that the set of points mapped into any hyperplane can be quite convoluted as a result, allowing much more complex discrimination between sets which are not convex at all in the original space. The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

Neural Network Architecture **Accuracy on Dev** Set	**Accuracy on Training Set**
Fully Connected Neural Network(1 Layer)	0.6112
Fully Connected Neural Network(3 Layers)	0.5978
LSTM(1 Layer)	0.6354
LSTM(3 Layer)	0.6903

Table 2: Results of Deep Neural Network

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

Since our data set is relatively larger than normal problem weve handled before, which contains more than 120,000 samples and 30 features, SVM can efficiently help us save the memory.

The support vector machines in scikit-learn support both dense (numpy.ndarray and convertible to that by numpy.asarray) and sparse (any scipy.sparse) sample vectors as input. However, to use an SVM to make predictions for sparse data, it must have been fit on such data. For optimal performance, use C-ordered numpy.ndarray (dense) or *scipy.sparse.csr$_m$atrix* (sparse) with dtype=float64.

Results of SVM: Based on the SVM classifier, we use cross validation to evaluate our model. According to the result of 5-fold cross validation, the accuracy of the classifier is around 0.5325, which is slightly higher than the expectation of the accuracy of a random prediction (0.5). However, for a financial trading strategy, a prediction with expected accuracy higher than 0.5 means a positive expected profit for every minutes. Therefore, considerable profit can be anticipated in the long run.

Using Pattern Recognition to Improve the Model: Although the result of SVC on 0-1 categorical label is expected to bring stationary profit to us, we still seek for methods that can achive higher accuracy. To reach the target, we classify the movement of the stock price into 6 different state instead of 2. The new label help us get an accuracy of 0.67 when tested with cross validation, which is a dramatic rise from that brought by the old model. Compared with the result this result is quite beyond our expectation. It seems like SVM classifier is quite useful when handling with this kind of problem.

3.2.4 Long short-term memory

Long short-term memory (LSTM) units are units of a recurrent neural network (RNN). An RNN composed of LSTM units is often called an LSTM network. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

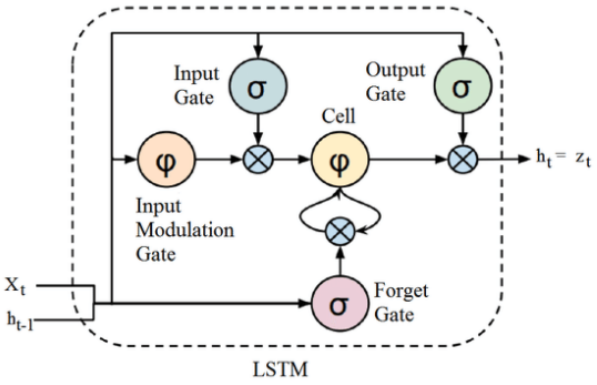

Figure 10: LSTM Unit

Results of Neural Networks: In this experiment, we apply Fully-connected neural network and LSTM to our denoised data and use the unsupervised labels as the labels in our model. Finally, we obtain the results in table 2. As we can see from this table, we can find the performance of LSTM is much better than the one of Fully-connected neural network by using our unsupervised labels.

4 Acknowledgement

All of us are very grateful to the course MSBD5001 and the efforts of professors and TA.