CPT_S 515: FinalReport #1

Instructor: Zhe Dang

Sheng Guan

Problem 1

Answer:
1. The main methodology to tackle this problem is that I would like to apply machine learning approach to the time series stock market data and to achieve the best performance on prediction accuracy and profitability. The outline of my solution is as below: firstly, I will review some existing machine learning approaches, mainly focus on support vector machine (SVM) analysis(this will serve as solution 1), and other types of machine learning methods. The shortcomings of these methods will be analyzed and the challenges for the stock price forecasting problem will be talked. Then I introduce the state-of-the-art machine learning model– Long- short term memory model that is one of the many variations of recurrent neural network (RNN) architecture. The details of the model analysis will be given, and the corresponding implementation will be mentioned. At the same time, the measurement metrics of predictive accuracy and profitability will be introduced. In the end, several preprocessing thoughts will be discussed based on this Long-short term memory model, such as data preprocessing using the wavelet transform, and so on, to further enhance the performance of the proposed model(this will serve as solution 2).

(1). Related work
Stock market prediction is one of the most challenging problems among time series predictions due to its noise and volatile features. In the past decades, machine learning approaches such as Artificial Neural Networks(ANNs)[1], Back Propagation (BP) neural network[2], Wavelet neural networks[2], and Support Vector Machines(SVM) [3]have been widely studied and applied on the financial time series prediction and they are proved to be able to gain relatively good predictive accuracy.
BP neural network is a multi-layer feedforward neural network based on error backpropagation algorithm. It consists of the input layer, the middle layer, and the output layer. The basic principle relies on the gradient descent method, which can adjust the weights and the thresholds of the network continuously through error back propagation to achieve the total error of the network to be the minimum. Wavelet neural network is based on BP neural network topology and applies the wavelet basis function as a hidden layer node transfer function. The idea of a Wavelet neural network is to adopt the wavelet basis to the training data. Hence, the wavelet estimator is expected to be more efficient than a sigmoid neural network.[4] Wavelet neural network combines the advantages of both wavelet analysis and neural network which leads it to overcome the shortcomings of the two when they are separately applied as forecasting system. Therefore, applying Wavelet neural network to stock price prediction is of important theoretical significance and practical value. However, the empirical study on the stock price data, has shown that the results of BP neural network and wavelet neural network have great volatility with strong randomness. The results of SVM is relatively stable.[2]

(2). SVM solution
When dealing with nonlinear problems, the SVM first transforms the nonlinear problem into a linear problem in a high-dimensional space, and then uses a kernel function to replace the inner product operation in the high-dimensional space, thus cleverly reducing the computational complexity. The SVM effectively overcomes the dimensionality curse and local minimum issue that will happen in the neural network.[5] The introduction of SVM at first is to solve classification problem. Before we talk about our prediction problem, we need to talk about some concepts occurred in the classification problem first.
Suppose in an n-dimensional classification problem, we have l training points.

$$T = (x_1, y_1), ..., (x_l, y_l), x_i \in R^n, y_i \in \{1, -1\} \tag{1}$$

Given a linearly separable training set $(x_i, y_i) : i = 1, 2, ...l$, we would like to find a linear classifier with maximum margin. If we use w and b to represent classifier parameters, this can be represented as an optimization problem.

$$\begin{aligned} \underset{w,b}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y^i(w \cdot x^i + b) \geq 1, i = 1, ..., l \end{aligned}$$

This results in a quadratic optimization problem with linear inequality constraints. Now we formulae an equivalent dual optimization problem and solve that dual problem instead. Based on Lagrangian constrained optimization, the new objective function can be transferred to a function of α's only.

$$\begin{aligned} \text{maximize} \quad & w(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{l} \alpha_i \alpha_j y_i y_j Ker(x_i, x_j) \\ \text{subject to} \quad & \sum_{i=1}^{l} \alpha_i y_i = 0, c \geq \alpha \geq 0 \end{aligned}$$

c here is a regularization parameter that achieves the tradeoff between the maximum separation margin and the training error. Now we will talk about how to use SVM model. In the SVM we have two parameters to optimize, c and kernel function. From the empirical study, it is well known that the Gaussian kernel usually will give us the acceptable performance. Here we can directly choose Gaussian kernel and optimize its parameter σ.
In the classification problem, in the end, if we use l training data points, we can get the final decision function as below:

$$f(x) = sign \sum_{i=1}^{l} \alpha_i y_i Ker(x_i, x) + b \tag{2}$$

If we remove sign in the Equation (2), the g(x) will contain more information since g(x) in fact reflects the distance between the prediction point and the trained hyperplane. Then, we can use $\bar{x}$ to predict the stock price relative change of the following M^{th} day to the current day.

$$z = g(\bar{x}) = \sum_{i=1}^{l} \alpha_i y_i Ker(x_i, \bar{x}) + b \tag{3}$$

Now we assume in the following M^{th} day, the stock price actual change as y and $\hat{y}$ is the predicted change value. Suppose there exists a mapping such that z and $\hat{y}$ is 1-to-1 mapping relationship, that is:

$$\hat{y} = \varphi(z) \tag{4}$$

In our problem, we have a chart of $t = \langle openprice, closeprice, highestprice, lowestprice, volume \rangle$. We can construct a set of indexes based on this dataset. For example, MeanIndex: the average stock price over the past M days; RateChangeIndex: the maximum difference among past M days; HighIndex: the highest price in the past M days; LowIndex: the lowest price in the past M days, .etc.

The SVM algorithm can be applied in the following way:
1) Select the original stock data, construct forecasting index, and normalize these original data and indexes to form vector $x_i (i = 1, 2, ...)$.
2) Define the N as the length of the training data, M is the length of the prediction data.
3) Let $y_i = thecloseprice of day(i + M) - thecloseprice of day(i)$.
4) Select $N(x_1, x_2, ..., x_N)$ as the training data and select Gaussian kernel function to train these N data to obtain $\alpha_1, \alpha_2, ..., \alpha_N$ and b.
5) Calculate $z_i = g(x_i) = \sum_{j=1}^{N} \alpha_i \bar{y_i} Ker(x_j, x_i) + b, i = 1, 2, ..., N$
where $\bar{y_i} = sign(y_i)$
6) Select the M data following the above N data– $M(x_{N+1}, x_{N+2}, ..., x_{N+M})$ as testing data, compute

$z_i = \sum_{j=1}^{N} \alpha_i \bar{y_i} Ker(x_j, x_i) + b, where \quad i = N+1, N+2, ..., N+M)$.
7) Let

$$\begin{aligned}\delta_i &= (theclosepriceofday(i) + y_i) - (theclosepriceofday(i+M-1)), \quad i = N+1 \\ &= (theclosepriceofday(i) + y_i) - (theclosepriceofday(i-1) + y_{i-1}), \quad i = N+2, ..., N+M\end{aligned} \tag{5}$$

Then δ_i will represent the actual relative change on (i+M) day respect to i+(M-1) day (i=N+1,N+2,...,N+M).
8) Let

$$\begin{aligned}\hat{\delta_i} &= (theclosepriceofday(i) + \hat{y_i}) - (theclosepriceofday(i+M-1)), \quad i = N+1 \\ &= (theclosepriceofday(i) + \hat{y_i}) - (theclosepriceofday(i-1) + \hat{y_{i-1}}), \quad i = N+2, ..., N+M\end{aligned} \tag{6}$$

Then $\hat{\delta_i}$ will represent the predictive relative change on (i+M) day respect to i+(M-1) day (i=N+1,N+2,...,N+M).

We can use a validation dataset to make a comparison of real and forecast trend of the stock market and tune the SVM model to achieve the best performance regarding on prediction accuracy. When the prediction accuracy is high enough, we can according to the $\hat{\delta_i}$ to draw the stock price changing curve and make the purchase-sell decision accordingly, that is purchasing in the lowest point and sell in the highest point within the given time window.

(3). Long- short term memory model (LSTM) solution
Despite the SVM model can predict financial time series well and gain high predictive accuracy, a recent trend in the machine learning and pattern recognition communities considers that a deep nonlinear topology should be applied to time series prediction. An improvement over traditional machine learning models, the new one can successfully model complex real-world data by extracting robust features that capture the relevant information [6] and achieve even better performance than before [7].
At the same time, [5] reports that for the SVM model, some predictions can be really bad since the SVM model relies on the choice of the training set. If the training set is improperly selected, for short-term prediction, the larger training set will hide the short-term trends. Whereas, if the training set is too small, we cannot grasp the trend at all.
LSTM is a type of recurrent neural network (RNN), with feedback links attached to some layers of the network. Unlike conventional RNN, it is well-suited to learn from experience to predict time series when there are time steps with arbitrary size. In addition, it can solve the problem of a vanishing gradient by having the memory unit retain the time related information for an arbitrary amount of time[8]. Evidence has proved that it is more effective than the conventional RNN[9][10]. Thus, we decide to use LSTM model to predict the stock trends. We will introduce the model of RNN and its LSTM architecture for forecasting the closing price first.
(3)-1 RNN model:
The RNN is a type of deep neural network architecture that works on temporal dimension and has been widely adopted in time series modelling[9]. The RNN will make use of the sequential information in contrast the traditional neural network cannot make use of. The RNN will take as their input not just the current input example they see, but also what they have perceived previously in time. The decision in a recurrent net reached at time step t-1 affects the decision it will reach later at time step t. So RNN has two sources of input, the present and the recent past. The process of carrying memory forward can be defined mathematically[11]:

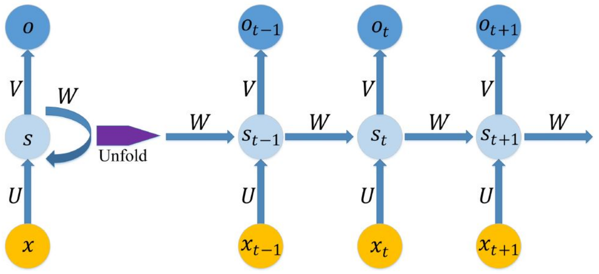
$$h_t = \phi(W x_t + U h_{t-1}) \tag{7}$$

The hidden state at time step t is h_t. It is a function of the input at the same time step x_t, modified by a weight matrix W (feedforward nets) added to the hidden state of the previous time step h_{t-1} multiplied by its own hidden-state-to-hidden-state matrix U, otherwise known as a transition matrix and similar to a Markov chain. The weight matrices are filters that determine how much importance to accord to both the

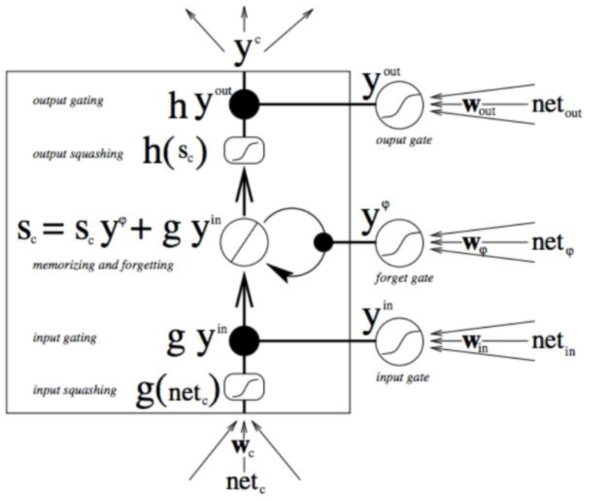
present input and the past hidden state. The error they generate will return via backpropagation and be used to adjust their weights until error cant go any lower.
o_t is the output at time t, which can be formulated as:

$$o_t = \phi(Vh_t) \tag{8}$$

The sum of the weight input and hidden state is squashed by the function ϕ either a logistic sigmoid function or tanh.
The figure below shows that how the output will be dependent on the hidden state.

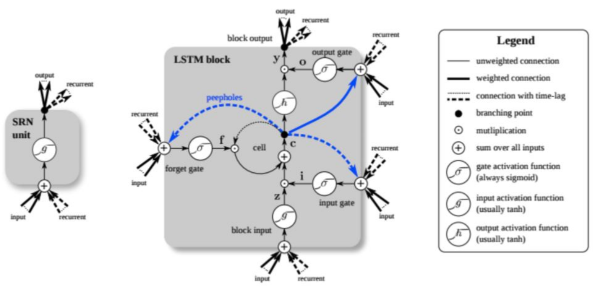


(3)-2 LSTM:
Although RNN models the time series well, it is hard to learn long-term dependencies because of the vanishing gradient problem [10]. LSTM is an effective solution for combating vanishing gradients by using memory cells[12]. A memory cell is composed of four units: an input gate, an output gate, a forget gate and a self-recurrent neuron. Those gates act on the signals they receive, and similar to the neural networks nodes, they block or pass on information based on its strength and import, which they filter with their own sets of weights. Those weights, like the weights that modulate input and hidden states, are adjusted via the recurrent networks learning process. That is, the cells learn when to allow data to enter, leave or be deleted through the iterative process of making guesses, backpropagating error, and adjusting weights via gradient descent[11].
The diagram below illustrates how data flows through a memory cell and is controlled by its gates.



Starting from the bottom, the triple arrows show where information flows into the cell at multiple points. That combination of present input and past cell state is fed not only to the cell itself, but also to each of its three gates, which will decide how the input will be handled.

The black dots are the gates themselves, which determine respectively whether to let new input in, erase the present cell state, and/or let that state impact the networks output at the present time step. S_c is the current state of the memory cell, and gy^{in} is the current input to it. Remember that each gate can be open or shut, and they will recombine their open and shut states at each step. Those flows are represented here. The large bold letters give us the result of each operation.
The difference between a simple recurrent network (SRN) (Left) to an LSTM cell (right) is shown in the below figure[11].



We now use another figure to show the value of each gate is updated and how the LSTM model will be unrolled into a full network.
The mathematical symbols are defined as below[13]:
1). x_t is the input vector to the memory cell at time t.
2). W_i, W_f,W_c,W_o,U_i,U_f,U_c,U_o and V_o are weight matrices.
3). b_i,b_f,b_c and b_o are bias vectors.
4). h_t is the value of the memory cell at time t.
5). i_t and $\tilde{C_t}$are values of the input gate and the candidate state of the memory cell at time t, respectively, which can be formulated as:

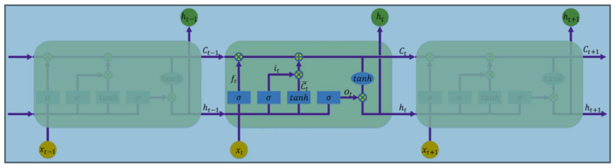
$$\begin{aligned} i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ \tilde{C_t} &= tanh(W_c x_t + U_c h_{t-1} + b_c) \end{aligned} \tag{9}$$

6).f_t and C_t are values of the forget gate and the state of the memory cell at time t, respectively, which can be calculated by:

$$\begin{aligned} f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ C_t &= i_t \tilde{C_t} + f_t C_{t-1} \end{aligned} \tag{10}$$

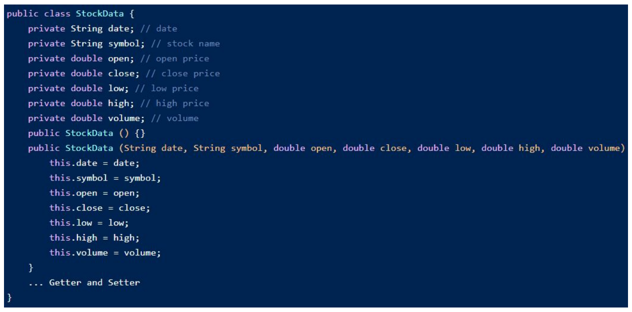
7). o_t and h_t are values of the output gate and the value of the memory cell at time t, respectively, which can be formulated as:

$$\begin{aligned} o_t &= \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o) \\ h_t &= o_t \cdot tanh(C_t) \end{aligned} \tag{11}$$

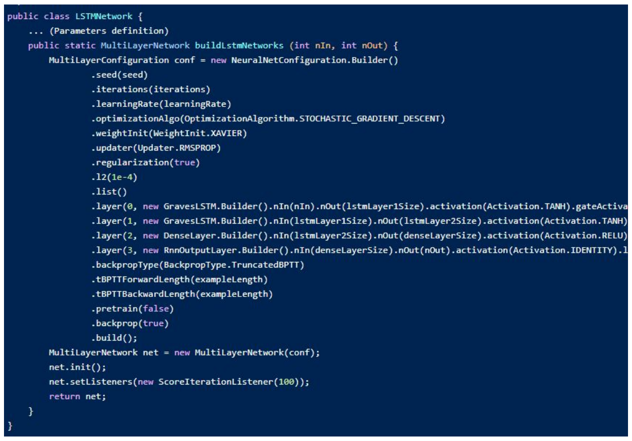


Here, x_t and h_t are the input vector and output result to the memory cell at time t, respectively. h_t is the value of the memory cell. i_t, f_t and o_t are values of the input gate, the forget gate and the output gate at time t, respectively. $\tilde{C_t}$ are values of the candidate state of the memory cell at time t.

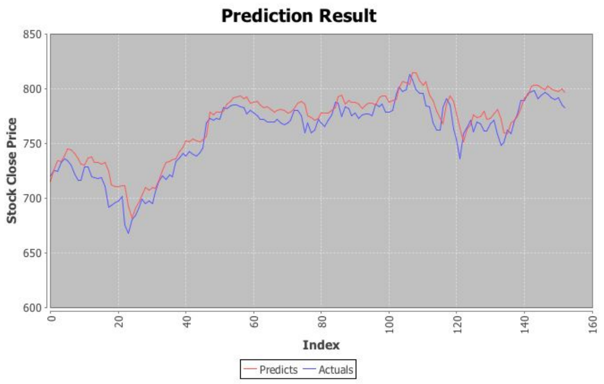
The architecture of a LSTM network includes the number of hidden layers and the number of delays, which is the number of past data that account for training and testing. Currently, there is no rule of thumb to select the number of delays and hidden layers[9]. In the experiment, we can use a validation dataset to tune these parameters or even use Bayesian optimization to tune these parameters.
There are several open-source projects using LSTM model to train and predict the stock open/close price, and these projects have reported the state-of-the-art results compared with other machine learning methods.[14][15].
One implementation example use Deeplearning4J module to implement LSTM along with its sample result is shown as below:



```
public class StockData {
    private String date; // date
    private String symbol; // stock name
    private double open; // open price
    private double close; // close price
    private double low; // low price
    private double high; // high price
    private double volume; // volume
    public StockData () {}
    public StockData (String date, String symbol, double open, double close, double low, double high, double volume)
        this.date = date;
        this.symbol = symbol;
        this.open = open;
        this.close = close;
        this.low = low;
        this.high = high;
        this.volume = volume;
    }
    ... Getter and Setter
}
```



```
public class LSTMNetwork {
    ... (Parameters definition)
    public static MultiLayerNetwork buildLstmNetworks (int nIn, int nOut) {
        MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
                .seed(seed)
                .iterations(iterations)
                .learningRate(learningRate)
                .optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
                .weightInit(WeightInit.XAVIER)
                .updater(Updater.RMSPROP)
                .regularization(true)
                .l2(1e-4)
                .list()
                .layer(0, new GravesLSTM.Builder().nIn(nIn).nOut(lstmLayer1Size).activation(Activation.TANH).gateActiva
                .layer(1, new GravesLSTM.Builder().nIn(lstmLayer1Size).nOut(lstmLayer2Size).activation(Activation.TANH)
                .layer(2, new DenseLayer.Builder().nIn(lstmLayer2Size).nOut(denseLayerSize).activation(Activation.RELU)
                .layer(3, new RnnOutputLayer.Builder().nIn(denseLayerSize).nOut(nOut).activation(Activation.IDENTITY).l
                .backpropType(BackpropType.TruncatedBPTT)
                .tBPTTForwardLength(exampleLength)
                .tBPTTBackwardLength(exampleLength)
                .pretrain(false)
                .backprop(true)
                .build();
        MultiLayerNetwork net = new MultiLayerNetwork(conf);
        net.init();
        net.setListeners(new ScoreIterationListener(100));
        return net;
    }
}
```



As we can see that the input can only have 5 features that are $\langle openprice, closeprice, highestprice, lowestprice, volume\rangle$ or input can include more features, such as 3-days MA, 5-days MA [15], and so on, from default feature set similarly as SVM solution.
We can use classical indicators (i.e., MAPE, R and Theil U) to measure the predictive accuracy of our model[1], here for simplicity we only use MAPE since predictive accuracy measurement is not our major

concern. The definitions of the MAPE indicator is as follows:

$$MAPE = \frac{\sum_{i=1}^{N} |\frac{y_i - \tilde{y_i}}{y_i}|}{N} \tag{12}$$

In the equation, y_i is the actual value and $\tilde{y}_i$ is the predicted value.
Worth to mention that, back to our solution 1-SVM approach, we also can apply this metrics to solution 1-SVM approach to measure the relative average of the error.
A buy-and-sell trading strategy is created based on the predicted results of the current LSTM model.The strategy recommends that we should buy when the predicted value of the next period is higher than the current actual value. On the contrary, it recommends that we should sell when the predicted value is smaller than the current actual value.
The strategy can be described by the following equations:

$$\begin{aligned} &Buy_{signal} : \tilde{y_{i+1}} > y_i \\ &Sell_{signal} : \tilde{y_{i+1}} < y_i \\ &R = 100 * (\textstyle\sum_{i=1}^{b} \frac{y_{i+1} - y_i}{y_i} + \sum_{i=1}^{s} \frac{y_i - y_{i+1}}{y_i}) \end{aligned} \tag{13}$$

Where R is the strategy returns, b and s denote the total number of days for buying and selling, respectively. Based on the above trading rule, we sell stocks when the predicted price is below the current price and buy the stocks when the predicted price is higher than the current one. The strategy for profit is as the same as solution SVM, in the given time window, we try to maximize R in the above equation.
(3)-3 Wavelet Transform:
Many papers talked about the idea to use wavelet transform to decompose the stock price time series to eliminate noise [16][17]. Here I would like to apply the Haar function as the wavelet basis function because it can not only decompose the financial time series into time and frequency domain but also reduce the processing time significantly[18].
Financial time series can be reconstructed by a series of projections on the mother and father wavelets with multilevel analysis indexed by $k \in \{0, 1, 2, ...\}$and by $j \in \{0, 1, 2, ...J\}$, where J denotes the number of multi-resolution scales. The orthogonal wavelet series approximation to a time series x(t) is formulated by:

$$x(t) = \sum_k s_{J,k}\varphi_{J,k}(t) + \sum_k d_{J,k}\psi_{J,k}(t) + \sum_k d_{J-1,k}\psi_{J-1,k}(t) + ... + \sum_k d_{1,k}\psi_{1,k}(t) \tag{14}$$

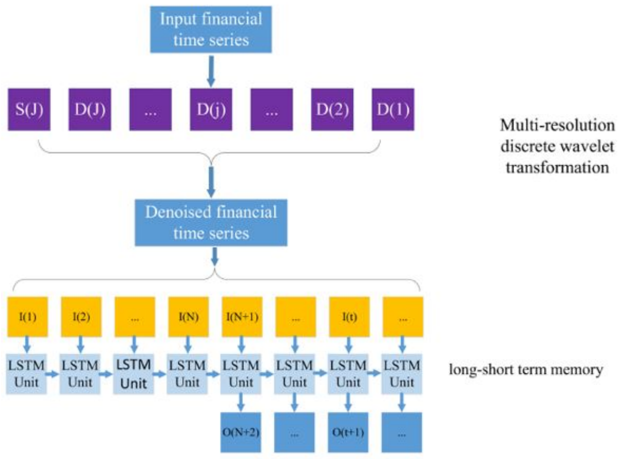
where the multi-scale approximation of time series x(t) is given as:

$$\begin{aligned} S_J(t) &= \textstyle\sum_k s_{J,k}\varphi_{J,k}(t) \\ D_j(t) &= \textstyle\sum_k d_{j,k}\varphi_{j,k}(t) \end{aligned} \tag{15}$$

Then, the brief form of orthogonal wavelet series approximation can be denoted by:

$$x(t) = S_J(t) + D_J(t) + D_{J-1}(t) + ... + D_1(t) \tag{16}$$

In the end, the flowchart of the solution-2 is shown as below:



References

[1] Guo Z, Wang H, Liu Q, Yang J. A Feature Fusion Based Forecasting Model for Financial Time Series. Plos One. 2014; 9(6):172200.

[2] Huang QP, Zhou X, Gan YJ, Wei Y. Application of SVM and Neural Network Models in Stock Prediction. Microcomputers and Applications 34, no. 5 (2015): 88-90.;

[3] Cherkassky V. The nature of statistical learning theory: Springer; 1997.

[4] Alexandridis, Antonios K., and Achilleas D. Zapranis. Wavelet neural networks: A practical guide. Neural Networks 42 (2013): 1-27.

[5] Zhang YC, Zhang ZC. Application of Support Vector Machines in Stock Price Prediction. Journal of Beijing Jiaotong University 31, no. 6 (2007): 73-76.

[6] Hinton GE, Salakhutdinov RR. Reducing the Dimensionality of Data with Neural Networks. Science. 2006; 313(5786):5047. https://doi.org/10.1126/science.1127647 PMID: 16873662

[7] Bengio Y, Courville A, Vincent P. Representation Learning: A Review and New Perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2013; 35(8):1798828.

[8] Chu KL, Sahari KSM. Behavior recognition for humanoid robots using long short-term memory. 2016; 13(6):172988141666336.

[9] Palangi H, Deng L, Shen YL, Gao JF, He XD, Chen JS, et al. Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval. IEEE-ACM Trans Audio Speech Lang. 2016; 24(4):694707. https://doi.org/10.1109/taslp.2016.2520371

[10] Palangi H, Ward R, Deng L. Distributed Compressive Sensing: A Deep Learning Approach. IEEE Transactions on Signal Processing. 2016; 64(17):450418.

[11] A Beginners Guide to Recurrent Networks and LSTMs. https://deeplearning4j.org/lstm.html

[12] Hochreiter S, Schmidhuber J. Long Short-Term Memory. Neural Computation. 1997; 9(8):173580. PMID: 9377276

[13] Bao, Wei, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS one 12, no. 7 (2017): e0180944.

[14] Plain Stock Close-Price Prediction via LSTM. https://isaacchanghau.github.io/2017/07/26/Plain-Stock-Close-Price-Prediction-via-LSTM-Initial-Exploration/

[15] US-Stock-Market-Prediction-by-LSTM. https://github.com/christsaizyt/US-Stock-Market-Prediction-by-LSTM

[16] Ramsey JB. The contribution of wavelets to the analysis of economic and financial data. Philosophical Transactions of the Royal Society B Biological Sciences. 1999; 357(357):2593606.

[17] Popoola A, Ahmad K, editors. Testing the Suitability of Wavelet Preprocessing for TSK Fuzzy Models. IEEE International Conference on Fuzzy Systems; 2006.

[18] Hsieh TJ, Hsiao HF, Yeh WC. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. Applied Soft Computing. 2011; 11(2):251025.