Stock Market's Closing Price Prediction with LSTM, CNN and ARIMA

Zixuan Wang
zixuan14
zixuan14@illinois.edu
Team 6 (ygw)

Yuhang Gong
yuhangg2
yuhangg2@illinois.edu
Team 6 (ygw)

Gan Yao
ganyao2
ganyao2@illinois.edu
Team 6 (ygw)

***Abstract*—The stock market is a volatile and dynamic system which makes the prediction of the stock market prices a very challenging problem. There are Many existing predicting methods that applied deep learning algorithm to address that problem and reached satisfactory results. This project explored the prediction of four stocks–Facebook (FB), Microsoft (MSFT), Procter & Gamble (PG) and Amazon (AMZN), from different stock market sectors. Three different models –LSTM, CNN and ARIMA, were developed to predict the daily wise closing stock prices based on different characteristics of stocks. The performance of different models was evaluated using RMSE. The result shows that ARIMA achieved the highest mean RMSE followed by LSTM and CNN.**

Keywords—Stock Market, Price Prediction, LSTM, CNN, ARIMA, Time Series

I. Introduction

The prediction of stock market price is a fruitful wide-open avenue to explore. The accuracy of the stock prices prediction can determine the profit investors can make and can also have a positive influence on the economy. However, the stock market is also a volatile and dynamic system which makes the prediction of the stock market prices a very challenging problem. From financial industry to academic field, various methods were applied to address that problem. The forecasting analysis involves two classes in general—linear models like ARIMA, and non-linear models like ARCH and deep learning algorithms. This project aims at applying both linear (ARIMA) and non-linear models (LSTM, CNN) to predict and study the trends in stock prices based on historical stock data. The performance of the three models were compared and evaluated using RMSE.

The motivation for pursuing this problem can be two folds. On the one hand, the accurate prediction of stock market's price can be highly lucrative. The study of the trend of the stock market's price can provide insight and help the investors in making proper decisions. On the other hand, the deep learning algorithm has demonstrated promising outcome in predicting stock market's price compared with many of the other algorithms. Therefore, this project can be served as a beginner approach that explores the performance of two deep learning algorithms in predicting stock market's price.

Three different models—LSTM, CNN, ARIMA were used in this project. The data set was split into training (first 80% of the data) and testing set (rest of the data). For LSTM, the input neurons contained a tensor of length n with 1 feature (close price) of the preceding n days, to predict the close price at the n+1 day, which is the expected output. As for CNN, the input neurons contained a tensor of length n with 6 features (open price, close price, high price, low price, adjusted close price, volume) of the preceding n days, to predict the close price at the n+1 day, which is the expected output. Finally, the ARIMA used the daily wise closing price from the training set as the input first and predicted the next day's closing price. The predicted next day's closing price was then updated into the training set and was used to predict the closing prices for the following days. This process was repeated until the prediction of all the data from testing set was done.

II. Related work

The prediction of the stock market's price using different algorithms has been widely studied by many scholars in the recent years. Selvin (2017) introduced both linear and non-linear algorithms used on price forecasting and used three different deep learning architectures—RNN, LSTM, CNN and one linear model—ARIMA to predict the stock price of NSE listed companies. They used error percentage as the metric to compare the performance of the four selected models. This paper provided the overall structural framework of our project. In terms of the evaluation metrics, we decided to use RMSE instead of error percentage.

Nelson (2017) demonstrated the standard process of developing a LSTM classification model on stock market price prediction. They focused on predicting if the price of a particular stock is going to go up or not in the near future. Liu (2018) explored more methods that can be used in data preprocessing stage. They calculated the moving average, exponential moving average based on the stock data and used them as features when training LSTM model. Roondiwala (2015) predicted the stock market's price using a more complicated LSTM architecture in their paper. They added 2 LSTM layers and dense layer with ReLU activation and used RMSE to evaluate the performance of the LSTM model. We integrated the LSTM architectures mentioned above and added more layers to achieve better predictions.

Mehtab (2020) proposed a simple CNN method using 1D convolutional layer to process single-feature data and multi-feature data, which provided an overview of the standard processing of predicting stock market's price using CNN model. Lu (2020) proposed a CNN-LSTM method to predict the market stock's price, where they used CNN to extract features and adopt LSTM to predict the price with the extracted features. Nascimento (2020) proposed a new model called Multichannel CNN Trading Classifier and discussed the process of using 2D convolutional layers to handle multi-feature data. Similar architectures were used to develop the CNN model for this project. On the other hand, rather adapting traditional architecture, Wang (2017) introduced a different Convolutional Neural Network, in which number of fully connected layer are reduced and global average pooling is used instead of max pooling. Such architecture turns out performing really well working with time series data.

Shi (2022) presented the extensive process of developing stock market price predictive model using ARIMA model. We used similar methods to develop our ARIMA model.

III. Dataset Description

As to the data set, we decided to explore the trends of stocks from four different stock market sectors. For each of the stock market sector, we selected one representative stock accordingly. The four stocks are Facebook (FB) from communication services sector, Microsoft (MSFT) from information technology sector, Procter & Gamble (PG) from consumer staples sector and Amazon (AMZN) from consumer discretionary sector.

The history stock data were collected from Yahoo finance for the period 2017-04-11 to 2022-04-08. The links to the datasets can be found in the references section. The data of each stock contains a "date" column and six other columns describing the daily wise stock price ("open price", "highest price", "lowest price", "close price", "adjusted close price" and "volume"), all of which are continuous variables. The data of each stock contains 1259 records with no missing value. Min max normalization was applied to columns other than "date". Value was transformed into [0,1].

From the original time series dataset, the data was split into two parts sequentially. The first part contains first 80% of the original data and was used to train/validate the model. The rest of the data was used to test the model. The sample of the first five observations of the Amazon stock data is shown in Table 1.

Date	Open	High	Low	Close	Adj Close	Volume
2017-04-11	907.04	911.24	897.50	902.36	902.36	3012700
2017-04-12	903.09	904.09	895.25	896.23	896.23	2456100
2017-04-13	891.45	894.97	884.49	884.67	884.67	3174600
2017-04-17	887.50	902.38	887.50	901.99	901.99	2854700
2017-04-18	900.99	909.61	900.78	903.78	903.78	2999200

Table 1 Observations of the Amazon stock data

IV. Long Short Term Memory

Long Short Term Memory (LSTM) is a special kind of Recurrent Neural Network. For sequences of reasonably short lengths, Long Short Term Memory do a wonderful job of decoding correlations and capturing them to build a robust model and it can solve the gradient vanishing and gradient explosion problems during long sequence training (Nelson 2017).

A. Model Architecture

In the standard architecture of Long Short Term Memory networks, there are an input layer, a recurrent LSTM layer and an output layer. We use LSTM cells to replace usual hidden layers. It contains input gate, cell state, forget gate and output gate (Roondiwala 2017). The following are the definitions and functions of the various gates.

- Input gate: Input gate consists of the input.
- Cell State: Runs through the entire network and has the ability to add or remove information.
- Forget gate layer: It is a gating device that controls how much after the historical state c (t-1) flows to time t in the RNN.
- Output gate: A gating device controlling controls the time t state value c (t) is visible.
- Sigmoid layer taking the value range of (0,1), it can map a real number to the interval of (0,1) to describing how much of each component should be let through.
- Tanh layer solved the sigmoid non-zero center problem.
- ReLU layer check if the input is greater than 0, the value provided as input directly returns a value of 0 if the input is 0 or less.

The following are the mathematical formula expressions of these definitions.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (3)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (4)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t * tanh(C_t) \quad (6)$$

where f_t: forget gate, h_t: output, i_t: input gate, c_t: cell state, o_t: output gate, h_t: output, x_t: input, W, b are the parameter matrix and vector (Selvin 2017). The structure diagram of the entire neural network is shown in Figure 1.

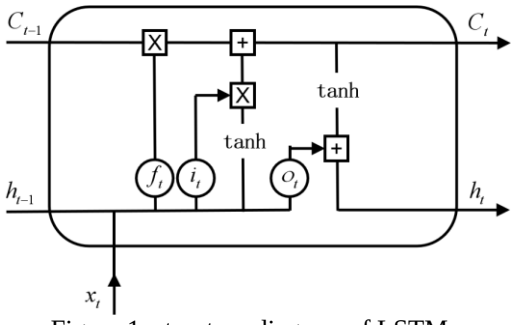

Figure 1: structure diagram of LSTM

In our project, we use Adaptive Moment Estimation (Adam) optimizer. It is the most widely optimization algorithm in deep learning, and it use the same learning rate for each parameter and adapt independently as learning progresses. In addition, Adam is a momentum-based algorithm that utilizes historical information on gradients. It gives great results fast, that is why we use it. Reference (Rana 2019) used different optimizer when they do the LSTM networks and the results showed that when they use Adam optimizer, the RMSE is really small than other optimizers.

And we use Mean square error as loss function. It is because the function curve of MSE is smooth, continuous, and derivable everywhere, which is convenient to use the gradient descent algorithm. It is a commonly used loss function. Moreover, as the error decreases, the gradient also decreases, which is beneficial to convergence, even if a fixed learning rate is used, it can converge to the minimum value faster.

LSTM improves the original coverage parameter update of RNN. It controls the information flow through the gate structure, and selectively adds the new state to the old state. So the derivative (gradient) in backpropagation is also an accumulation form, thus solving the vanishing gradient problem.

B. *Training Details*

According to our data and code, we have one hidden layer; in the layer, it has eight neurons. We also have two fully connected layers and three activation functions. They are Sigmoid function, tanh function and ReLU function. First, we input 1×64×10 data, via LSTM networks, the output is 1×64×8. Then we take this as input to the full connection layer, the output is 1×64×4. Next we use this result as input to generate results via the ReLU activation function, it dimension does not change. Finally, we use previous result as input, via the full connection layer and get the output result, it is 1×64×1.

For different stock data, we have different parameter setting. For AMZN data, sequence length is 10, batch size is 64, input size is 1, hidden size is 8, number of neural network layers is 1, learning rate is 0.001, epoch is 2000. For FB data, we set batch size to 128, others are same as AMAZ. For PG data, we set sequence length to 20, others are same as AMAZ. For MSFT data, we set sequence length to 20, batch size to 128, others are same as AMAZ.

V. Convolutional Nueral Network

One of the approaches we choose to address our problem, is convolutional neural network. Due to the use of convolution technique in it, this kind of neural network can successfully extract patterns hidden in the spatial structure of data. Our target data, stock prices, have strong temporal correlation between different timesteps, and thus is an ideal target for convolutional neural networks.

By our progress report, we had built a convolutional neural network with 1 convolutional layer and 1 dense layer, which gave us acceptable prediction accuracy, but not good enough. We have since tried to optimize the model by adding both convolutional layers and dense layers, but failed to make any notable improvement.

Then the work of Wang (2017) drew our attention to another path. The model they proposed, known as Fully Convolutional Network, introduced some uncommon features to CNN and is more suitable for sequence data. Based on their idea, we built a new network that significantly outperforms the precious one. More details will be discussed in the following content of this section.

A. *Model Architecture*

The Architecture of our Convolutional Neural Network is as presented in Figure 2.

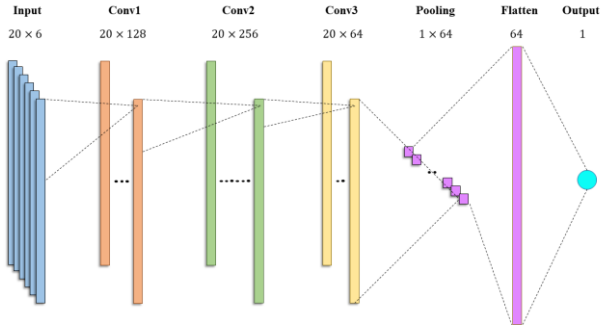

Figure 2: The architecture of CNN

- Input Layer: one training example consists of a tensor of length 20 with 6 channels, which means we use all 6 features(open price, close price, high price, low price, adjusted close price, volume) of the preceding 20 days, to predict the close price at 21st day.
- 1D Convolutional Layer: a 1D convolutional layer with 128 filters of size 8, as well as stride of 1, and padding of 4 to maintain original tensor size.
- ReLU activation function: an activation function defined as the positive part of its argument.

$$f(x) = \max(0, x)$$

- 1D Convolutional Layer: a 1D convolutional layer with 256 filters of size 5, as well as stride of 1, and padding of 2 to maintain original tensor size.
- ReLU activation function.
- 1D Convolutional Layer: a 1D convolutional layer with 64 filters of size 3, as well as stride of 1, and padding of 1 to maintain original tensor size.
- ReLU activation function.
- Global Average Pooling Layer: take the average of all neurons in each channel, and out put a tensor of length 1 with 64 channels.
- Flatten Layer: Flatten the output from global average-pooling layer, which is a tensor of length 1 with 64 channels, to a tensor of length 64 with 1 channel.
- Output Layer: Technically also a dense layer, mapping from 64 neurons to 1 output by weighted averaging.

A general working process of this network can be described as: A training example enters the network; when it go through

the convolutional layer, 64 filters each performs convolution on the input, convert it into a feature map of same size, 128 such feature maps are formed into a 2D tensor, a ReLU function check all the elements in this tensor and modify all negative values to zero and output it to the next layer; The following 2 convolutional layers performs similar job, and further map the features to higher levels, In the global average-pooling layer, all elements in one feature map are reduced to just 1 elements which is the average value; The flatten layer then tear down the 2D tensor, and connect all elements into a long 1D tensor; At last, 64 output from previous flatten layer are weighted averaged again to give the final prediction.

There are several novel features in this CNN architecture that set it apart from traditional ones. First, the number of fully connected layers is reduced. A network with such deep convolutional layers always has multiple fully connected layers. This architecture completely gives up fully connected layers, which reduce the size of model and brings great generalization ability. Second, a global average pooling layer is used. Rather than applying max-pooling of size smaller than the feature map, which is a much more technique in CNN, this architecture employs an average-pooling with size equal to the feature map, reducing the whole feature map to 1 element.

B. Training Details

Since our task falls into the category of regression, we choose Mean Squared Error loss function in our training phase. As the dominant optimization algorithm in the field of deep learning, Adam is widely used in training of convolutional neural networks, and therefore is also our choice of optimizer. We set learning rate as 0.0005 and train this model for 100 epochs, but with an early stop strategy to prevent overfitting. That is, if the training loss fail to decrease for 10 epochs, we stop the training. In other words, if $Loss(t-10) > Loss(i),\ i = t-9, t-8, \dots, t$, then we stop training at epoch t.

VI. Results/Discussion

A. Loss Graphs

The LSTM and CNN training and testing loss based on the loss function are shown in Figure 3 and Figure 4. As for CNN loss graphs, we can see that the early stop strategy makes the training process on PG end at around 60 epochs, while the remaining ones go through all 100 epochs.

B. Model Evaluation

We use Root Mean Squared Error (RMSE) to evaluate our model. LSTM, CNN, ARIMA performances on four stocks can be found in Table 2. Compare with the previous models, the new architectures have been proved to be much more suitable for our problems. On average, ARIMA achieved the highest RMSE followed by LSTM and CNN. However, LSTM outperforms ARIMA in predicting the stock price of Procter & Gamble. The prediction and real testing data plot of LSTM, CNN and ARIMA are demonstrated in Figure 5, Figure 6 and Figure 7.

	LSTM	**CNN**	**ARIMA**
Amazon	0.0223	0.0300	**0.0222**
Fachebook	0.0319	0.0366	**0.0317**
Procter & Gamble	**0.0161**	0.0194	0.0162
Microsoft	0.0175	0.0194	**0.0162**

Table 2: The RMSE of three models for four stocks

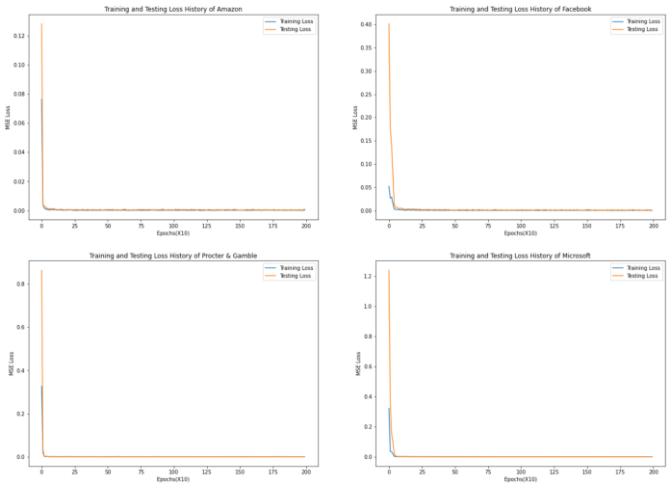

Figure 3: Training and testing loss plot of LSTM

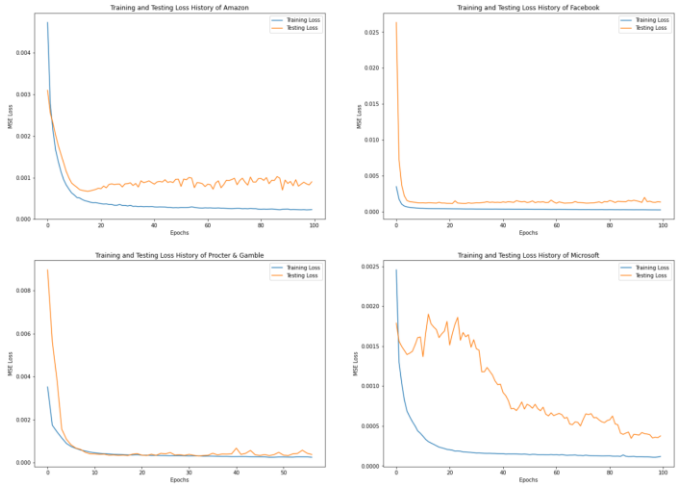

Figure 4: Training and testing loss plot of CNN

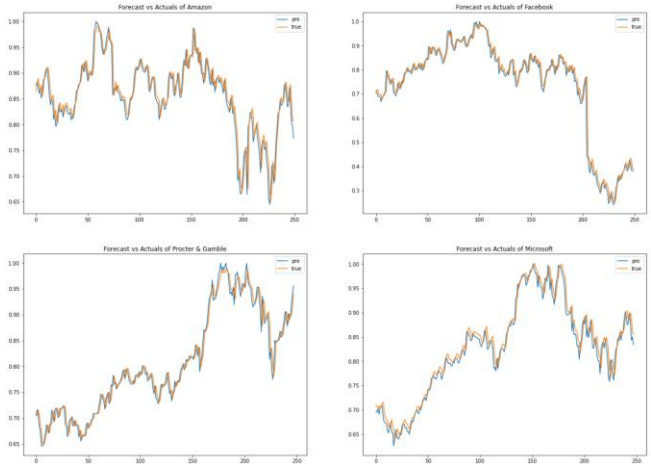

Figure 5 Prediction vs. real testing data of LSTM

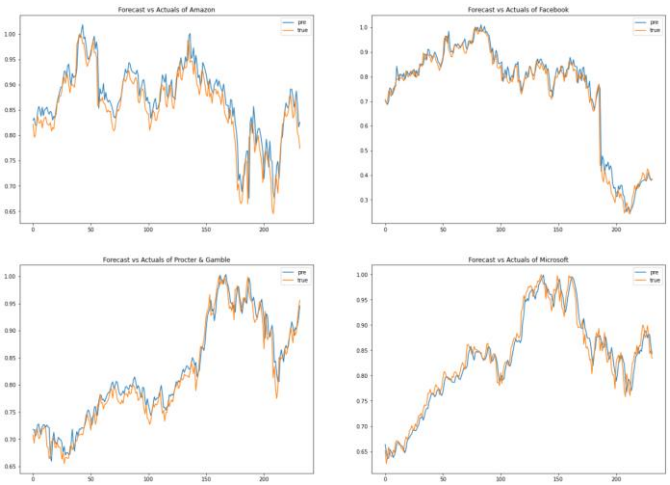

Figure 6 Prediction vs. real testing data of CNN

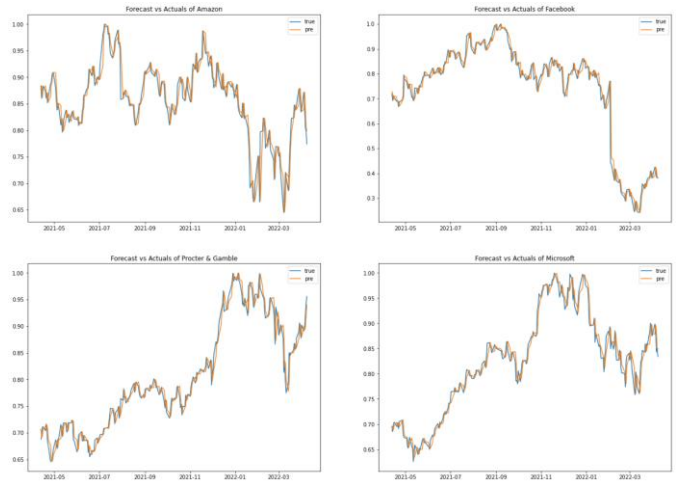

Figure 7 Prediction vs. real testing data of ARIMA

C. *Gradio Application*

We designed demos using Gradio. The input of the demo contains the testing dataset and the name of the stock. The output contains the predicted values, RMSE of the testing dataset and the forecast and actual plot. The screenshots of the outputs of Gradio application of LSTM and CNN algorithm are shown in Figure 8 and Figure 9.

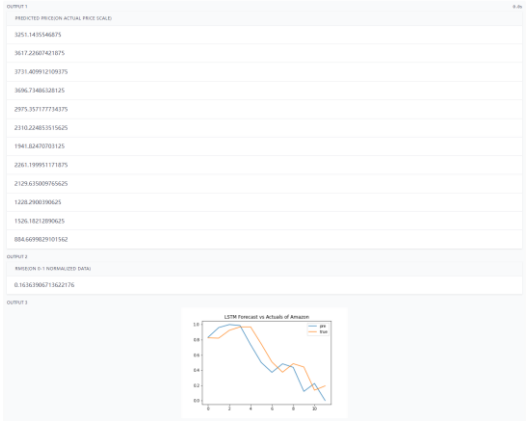

Figure 8 LSTM Gradio demo

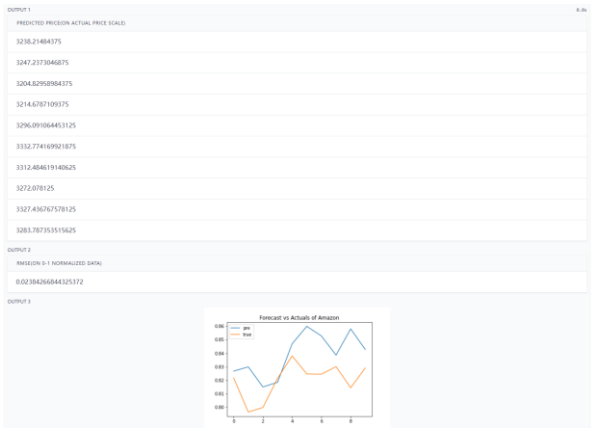

Figure 9 CNN Gradio demo

VII. Conclusion and Future Work

A. *Conclusion*

To conclude, this project explored the prediction of four stocks–Facebook (FB), Microsoft (MSFT), Procter & Gamble (PG) and Amazon (AMZN), from different stock market sectors. Three different models –LSTM, CNN and ARIMA, were developed to predict the daily wise closing stock prices based on different characteristics of stocks. The result shows that ARIMA achieved the highest RMSE followed by LSTM and CNN. Both LSTM and CNN can be easily trained and yield decent prediction accuracy, even though most of them do not outperform the ARIMA model.

B. *Future Work*

As to LSTM, the next thing we need to do is checking is there any other way to make the result better, like use Bi-directional LSTM way. We also can try another network, named Gated Recurrent Unit (GRU). GRU is a very effective variant of LSTM network, which is simpler in structure than LSTM network. In terms of computational complexity, GRU has fewer parameters and fewer tensor operations, so the training speed will be relatively faster than LSTM.

As to CNN, we can further improve our model by collecting more data, and deepen then convolutional layers. We have also noticed that there are plenty researches on combining CNN with other network architectures to achieve better performance. As we have built CNN and LSTM individually, we hope to mix them and see if we can create a better model.

References

Liu, S., Liao, G., & Ding, Y. (2018, May). Stock transaction prediction modeling and analysis based on LSTM. In *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)* (pp. 2787-2790). IEEE.

Lu, W., Li, J., Li, Y., Sun, A., & Wang, J. (2020). A CNN-LSTM-based model to forecast stock prices. *Complexity*, 2020.

Mehtab, S., & Sen, J. (2020). Stock price prediction using convolutional neural networks on a multivariate timeseries. *arXiv preprint arXiv*:2001.09769.

Nascimento, D., Costa, A., & Bianchi, R. (2020, October). Stock Trading Classifier with Multichannel Convolutional Neural Network. In *Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional* (pp. 282-293). SBC.

Nelson, D. M., Pereira, A. C., & De Oliveira, R. A. (2017, May). Stock market's price movement prediction with LSTM neural networks. In *2017 International joint conference on neural networks (IJCNN)* (pp. 1419-1426). IEEE.

Rana, M., Uddin, M. M., & Hoque, M. M. (2019, December). Effects of activation functions and optimizers on stock price prediction using LSTM recurrent networks. In *Proceedings of the 2019 3rd International Conference on Computer Science and Artificial Intelligence* (pp. 354-358).

Roondiwala, M., Patel, H., & Varma, S. (2017). Predicting stock prices using LSTM. *International Journal of Science and Research (IJSR)*, *6*(4), 1754-1756.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017, September). Stock price prediction using LSTM, RNN and CNN-slidingwindow model. In *2017 international conference on advances in computing, communications and informatics (icacci)* (pp. 1643-1647). IEEE.

Shi, Z., Hu, Y., Mo, G., & Wu, J. (2022). Attention-based CNN-LSTM and XGBoost hybrid model for stock prediction. *arXiv preprint arXiv*:2204.02623.

Wang, Z., Yan, W., & Oates, T. (2017, May). Time series classification from scratch with deep neural networks: A strong baseline. In 2017 International joint conference on neural networks (IJCNN) (pp. 1578-1585). IEEE.

Facebook (FB) Historical Stock Price Data Source: https://finance.yahoo.com/quote/FB/history?p=FB

Microsoft (MSFT) Historical Stock Price Data Source: https://finance.yahoo.com/quote/MSFT/history?p=MSFT

Procter & Gamble (PG) Historical Stock Price Data Source: https://finance.yahoo.com/quote/FB/history?p=FB

Amazon (AMZN) Historical Stock Price Data Source: https://finance.yahoo.com/quote/AMZN/history?p=AMZN

Contributions

Zixuan Wang was responsible for the literature review and implementation of ARIMA and part of the implementation of LSTM, as well as the writing of introduction, dataset description, results and conclusion. Totally contributed 34% to the whole project.

Yuhang Gong was responsible for the literature review and implementation of LSTM code and the LSTM section of the progress report. Totally contributed 33% to the whole project.

Gan Yao was responsible for the literature review and implementation of CNN, as well as the writing of CNN part in the progress report. Totally contributed 33% to the whole project.