Classic Deep Learning Methods in Stock Price Prediction

Tingfu Zhou[1], Yifei Sun[1*]

[1] University of Michigan, Michigan, Ann Arbor, USA
Email: tingfu@umich.edu, yifeisun@umich.edu

***Abstract*—Using deep learning model to predict stock price is an attractive field. In this paper, we implement three classic deep learning models, CNN, RNN, and LSTM in stock price prediction. And we compare these deep learning models with a non-deep learning stacking models. Several experiments show that RNN has the best performance. LSTM performance is a bit better RNN. And Non-deep learning stacking method is the worst. With further analysis, we find that the change rate of stock price is largely random, which makes complex models difficult to predict.**

I. Introduction

In the stock market, investors often expect continuous short-term returns. Also, due to the existence of market anomalies, it is plausible to use information about the past to predict the stock market [1]. Predicting stock prices using machine learning is one of the most attractive fields in applied machine learning [2].

In this paper, we predict stock prices using classic deep learning models like CNN, RNN, and LSTM. We use the price feature data of the first 50 days to predict the price data of the next 5 days. By harnessing the power of these advanced algorithms, we aim to provide investors, traders, and financial analysts with more accurate and actionable insights, aiding in better decision-making and risk management. The ability to forecast stock prices can lead to improved portfolio performance reduced financial risks, and a deeper understanding of market dynamics. This approach is not merely about financial gain, but also about gaining a deeper comprehension of the intricate world of stock markets, contributing to financial stability, and ultimately making informed choices for a more secure financial future.

The main contributions to this work are as follows:

- We use CNN, RNN, LSTM, adn non-deep learning stacking model to predict the stock price.
- By analyzing the predicted result of these model and character of stock price data, we found that short-term stock market is largely gambling, since holder are basically guessing a random number.

The remainder of this article is organized as follows: Section II briefly introduces the related work. Section III introduces the predicted models we use. Section IV represent the experiment set up. Section V shows experiments and analysis of the result. Section VI is the conclusion.

*Corresponding author.

II. Existing work

There are many papers focus on applying CNN to stock market predict. One of the popular method is combining price information and news[3]. Another popular method focus on the stock price itself, trying to predict the short-term value from past information [4]. The situation of LSTM is similar, with only price information provided, LSTM can also get excellent results[5].

Compare to LSTM, RNN is less popular in stock prediction[2]. However, since we only focus on short-term prediction, RNN can also receive good performance with past price information[6].

III. Method

The target of this report is predicting the price of a stock. We will implement 3 classic deep learning methods for predicting stock price. They are CNN, RNN and LSTM.

CNNs are primarily known for their use in image processing, but they can also be effective for time-series prediction like stock prices. This is because CNNs can identify patterns and trends in data, making them suitable for analyzing the sequences found in stock market data. By applying convolutional layers, CNNs can extract and learn features from historical price data, which can then be used to forecast future stock prices.

RNNs are designed to handle sequences of data, making them ideal for time-series prediction tasks such as stock price forecasting. Unlike traditional neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes RNNs particularly good at understanding the temporal dynamics and dependencies in stock market data, allowing for predictions based on historical price trends and patterns.

LSTMs are a special kind of RNN, specifically designed to avoid the long-term dependency problem. This makes them excellent for stock price prediction, as they can remember information for long periods, which is crucial in understanding the complex patterns in stock market data. LSTMs can capture long-term dependencies in time-series data, making them more

effective than standard RNNs in predicting stock prices, which often depend on long-term historical trends.

Each of these models has unique features that make them suitable for stock price prediction: CNNs for feature extraction, RNNs for understanding temporal dynamics, and LSTMs for capturing long-term dependencies in the data. By leveraging these characteristics, these models can be effectively used to predict stock market movements.

We would also compare them with non-deep learning method such as Support Vector Regression, Random Forest Regression, and Gradient Boost Tree. The input features would be adjusted to open, close, high and low prices of the previous m days. The output values would be adjusted to close price of later n day. We will also use methods in Multivariate statistical analysis, like variable selection method, to analyze and select statistical significant features.

A. problem set-up

We use the price feature data of the first 50 days to predict the stock price of the next 5 days.

B. Preprocess

Firstly, we drop some features that are not so useful for our prediction. The features that the dataset provide are 'Open', 'High', 'Low', 'Close', 'Volume', 'Ex-Dividend', 'Split Ratio', 'Adj.Open', 'Adj.High', 'Adj.Low', 'Adj.Close', 'Adj.Volume'. Since adjusted features are more accurate, thus we drop the original not adjusted value. Since we focus on the price itself, we also drop feature "Ex-Divided" and "Split Ratio". It is worth to note that there is no NAN in the stock price dataset.Thus, we do not need to drop-NA or interpolation.

Then, we split the data into train-test pair. For each pair, the training part is 50 days' price data (X) and 5 days' test data (Y).

C. Training method

Following tables are the detail of the CNN I, RNN II, and LSTM III model we use. Also, we use a non-deep learning stacking method IV to compare. The num of epochs is 25 for all three models.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 393, 4, 32)	544
max_pooling2d (MaxPooling2D)	(None, 196, 2, 32)	0
conv2d_1 (Conv2D)	(None, 191, 1, 64)	24,640
max_pooling2d_1 (MaxPooling2D)	(None, 95, 1, 64)	0
conv2d_2 (Conv2D)	(None, 92, 1, 16)	4,112
conv2d_3 (Conv2D)	(None, 90, 1, 8)	392

TABLE I: CNN Architecture

Firstly, we split the pre-processed dataset to training and testing set. Then we normalize all the features and train the model. After training, we use the model to predict the test dataset. Lastly, we reverse-normalized the result.

Layer (type)	Output Shape	Param #
simple_rnn (SimpleRNN)	(None, 50, 50)	2800
simple_rnn_1 (SimpleRNN)	(None, 50)	5050
dense (Dense)	(None, 1)	51
Total params		7901 (30.86 KB)
Trainable params		7901 (30.86 KB)
Non-trainable params		0 (0.00 Byte)

TABLE II: RNN Architecture

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 50, 50)	11200
lstm_1 (LSTM)	(None, 50)	20200
dense_1 (Dense)	(None, 1)	51
Total params		31451 (122.86 KB)
Trainable params		31451 (122.86 KB)
Non-trainable params		0 (0.00 Byte)

TABLE III: LSTM Architecture

IV. Experiment Setup

A. Hardware

A laptop with Intel I7-13700HX, NVIDIA RTX 4070 (8GB), 16 GB of memory.

B. Datasets

We will use stock price data from quandl API as the primary dataset. This data retrieved using this API contains close, open, high and low prices of every day since this stock existed, as well as the split ratio, trading volume and respective adjusted prices. The registration link can be found at https://data.nasdaq.com/.

C. Evaluation

We want to keep our evaluation method Simple and direct. Thus, the main evaluation metrics are Mean Square Error (MSE).

V. Experiment and Analyize

A. Models Comparison

Table VI shows the mean square error of different models. Table V shows the training loss of different models. The predicted outcome of CNN is shown in figure 7. The predicted outcome of RNN is shown in figure 1. And the predicted outcome of LSTM is shown in figure 2.

From table VI and table V, we can see that the performance of RNN is the best among the three deep learning method. While LSTM is slight behind RNN. The third one is CNN. And non-deep stacking method has the worst performance. The gap between deep learning and non-deep learning stacking model is several orders of magnitude.

It is obvious that the gap between deep learning models and non-deep learning model is too large. We believe the reason is the Flatten of the input matrix. Since the input feature matrix is flattened, It makes the model learn the wrong feature. Which means that stacking is not a good option for stock price prediction. Also, from Figure 1 2 7 we find that the

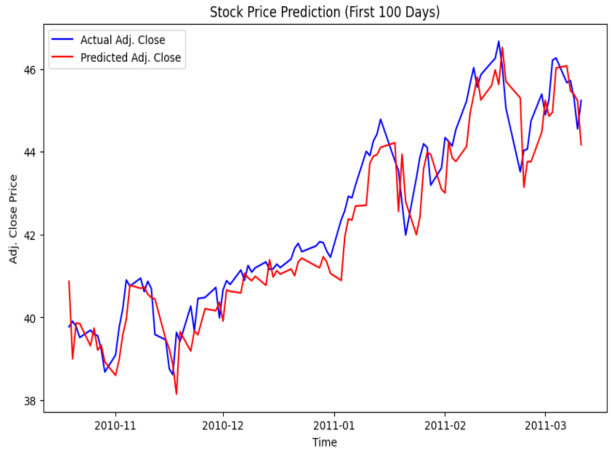

Fig. 1: RNN

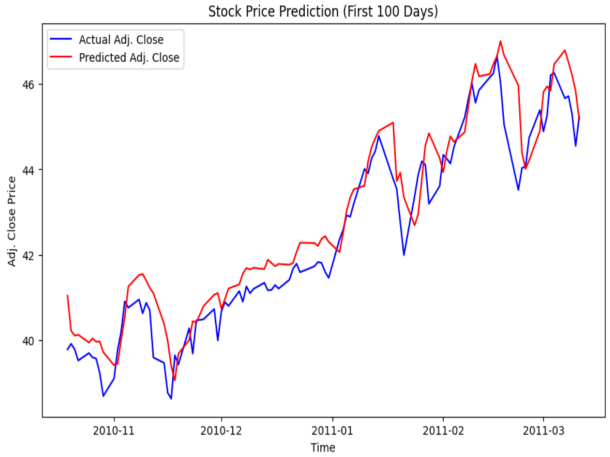

Fig. 2: LSTM

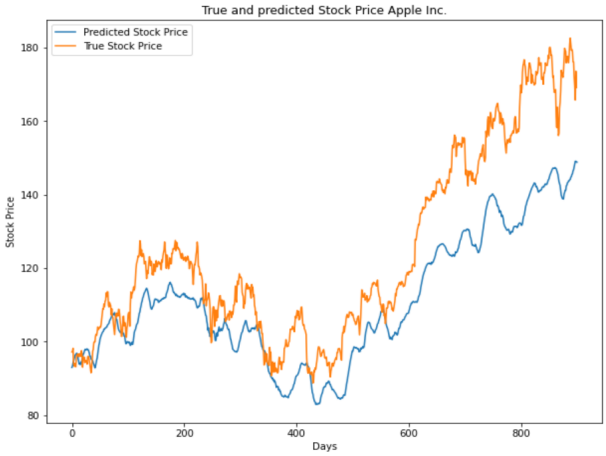

Fig. 3: CNN

Model Type	Model Name	Param #
Base Learner	SVR	$C = 1, gamma = auto$
Base Learner	GradientBoostingRegressor	$nestimators = 10$
Base Learner	RandomForestRegressor	$nestimators = 10$
Meta Model	Ridge	Default parameter

TABLE IV: Stacking Architecture

TABLE V: Training Loss

Algorithm	MSE
CNN	1.4417e-04
RNN	2.1835e-06
LSTM	2.2215e-06
Non-deep Stacking	52870545870477.15

TABLE VI: Mean Square Error

Algorithm	Training Loss
CNN	15.189550
RNN	13.179923
LSTM	3.9783600
Non-deep Stacking	100885019625606.98

predicted curves of these three deep learning models seem to be the translation of the original curve. Thus, we use a dummy predictor that only use the previous day's price as today's price. The MSE of the experiment is shown in table VII.

TABLE VII: Mean Square Error (Dummy predictor)

Algorithm	Training Loss
CNN	15.189550
RNN	13.179923
LSTM	3.9783600
Dummy predictor	3.120314

The result shows that the dummy predictor has the best performance. It seems that in this stock price predicted problem, simple models are more efficient than complex models. To analyze the reason, we need to check the stock price itself.

We also noticed that in the CNN implementation, intuitively, the performance of the model should be increasing as we increase the length of days of the training data, however, the opposite is observed. Our prediction looks like a lagging of the true values. Generally speaking, as the number of days decreases in each of the training data, the MSE decreases and performance in terms of shear price becomes better. (See fig 7) And when the number of days decreases to 20 and 10, the predicted curve looks like a shift of the true curve into the future. (See fig 4-6) A reasonable explanation is that the neural network actually does not learn anything particularly interesting in the days far ahead of the predicted day. The longer the days in the training data are, the more distraction and interference are for the model. Also notice that our baseline model, which is just using yesterday's value as the prediction for today's value, can be interpreted as only using one day as training data. We indeed observed the curve look more and more like the baseline curve as the length of days decreases.

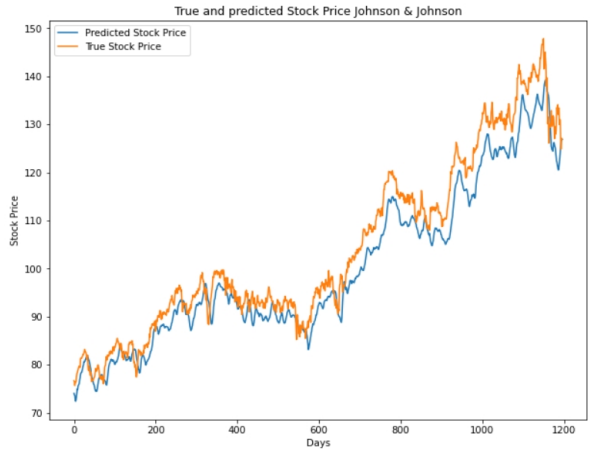

Fig. 4: CNN Prediction for J & J Stock using previous 200 Days

Fig. 5: CNN Prediction for J & J Stock using previous 20 Days

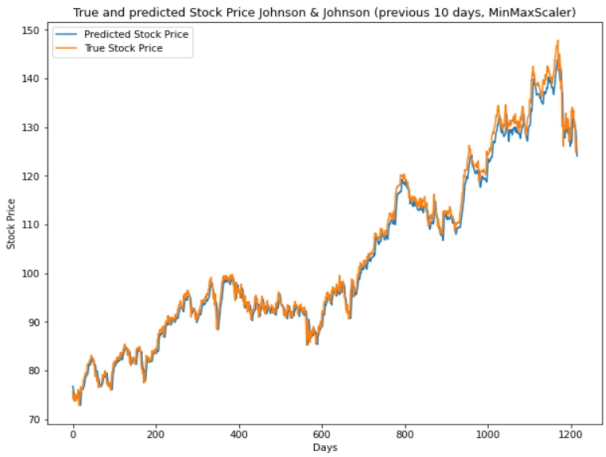

Fig. 6: CNN Prediction for J & J Stock using previous 10 Days

	MinMaxScaler Prediction (400 previous days)	Baseline (Use Last Day)	MinMaxScaler Prediction (200 previous days)	MinMaxScaler Prediction (100 previous days)	MinMaxScaler Prediction (50 previous days)	MinMaxScaler Prediction (20 previous days)	MinMaxScaler Prediction (10 previous days)
Apple Inc.	270.109784	3.120314	35.515093	18.426735	77.594101	15.189550	15.189550
Microsoft Corporation	54.455868	0.738092	7.651248	4.052051	16.542605	3.135019	3.135019
Alphabet Inc.	19924.141530	156.821904	2536.948921	1310.657837	4864.986719	1024.629327	1024.629327
Amazon.com Inc.	13391.496637	222.393537	2553.502891	1342.470078	6213.079913	1154.141854	1154.141854
Intel Corporation	16.855977	0.275912	3.049557	1.691925	5.795468	1.179489	1.179489
Cisco Systems, Inc.	2.933339	0.163733	1.368182	0.971677	2.558565	0.586067	0.586067
NVIDIA Corporation	715.541567	15.329314	159.636914	85.804693	246.640694	55.181193	55.181193

Fig. 7: MSE of CNN Prediction for Some Stocks

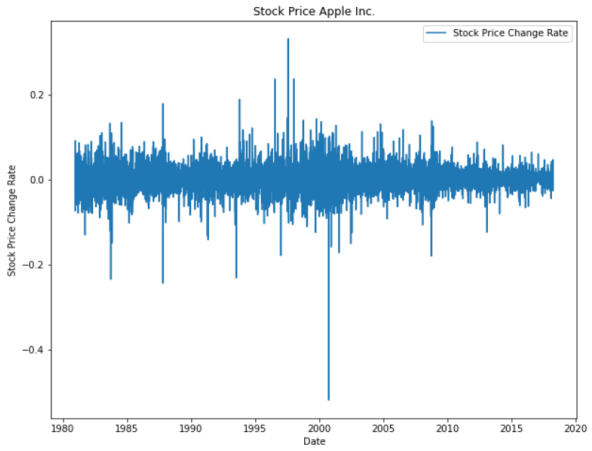

Fig. 8: Change Rate

The phenomenon above indicates two things. The first is in the stock prediction task, using the stock price itself as the target is not ideal. What we care about is how the price changes. Because stock price can be seen as a sampling from a continuous curve, the values yesterday and today can be very close most of the time, and the neural network will always try to as the previous value to predict the current value, which invariably achieves the minimum MSE in terms of shear price. Therefore, for such a random process, we could consider taking the difference of the sequence as the target, or use the change rate, which is the difference of the sequence normalized by the value of the sequence, as the target. The second is that our model may be too simplified as can not catch all the nuances in the data, and the data we have is not enough for prediction. We may need variables outside of the price value itself, such as news extraction, sentiment analysis, stock market index. We will try to explore the first point and leave the second point for future exploration.

B. Prediction of Change rate

Figure 8 shows the change rate of the stock price. And Figure 14 shows the result of stock price's autocorrelation function.

Figure 8 represents that the change rate is a variable with a positive mean close to zero. The rate of change of stock price fluctuates greatly at different time points. Some years, such as 1987, 2000, and 2008, saw spikes and sharp declines that could be related to the stock market crash or other major

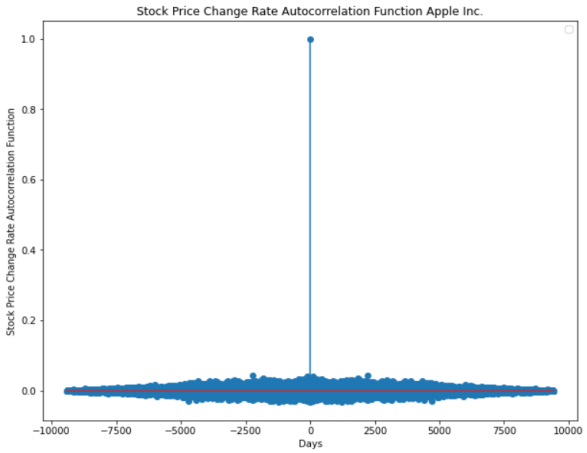

Fig. 9: Autocorrelation function

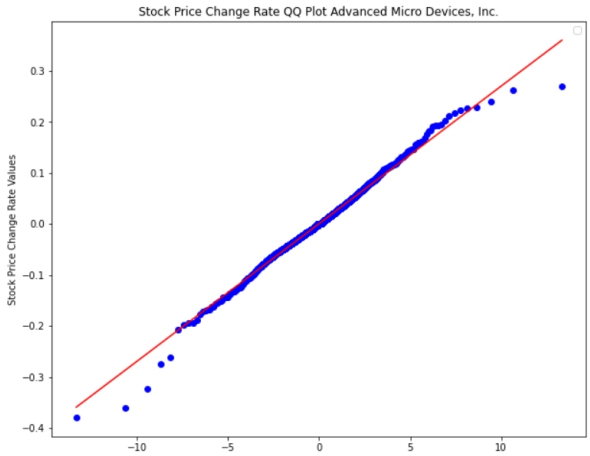

Fig. 11: QQ Plot of the Distribution of Change Rate against T Distribution of DOF of 4

	Mean	Variance	Skewness	Kurtosis
Apple Inc.	0.000736	0.000211	-0.006166	2.760613
Microsoft Corporation	0.001151	0.000206	0.930114	9.308568
Alphabet Inc.	0.000728	0.000147	-0.889537	3.587724
Amazon.com Inc.	0.001881	0.000212	1.726632	16.457004
Intel Corporation	0.000661	0.000217	0.244814	6.038590
Cisco Systems, Inc.	0.000818	0.000174	0.027081	7.242541
NVIDIA Corporation	0.003937	0.000813	2.703834	27.960642
Adobe Inc.	0.001477	0.000232	0.177842	8.639795
IBM	-0.000053	0.000142	-0.608616	7.569054
Oracle Corporation	0.000221	0.000172	-0.582776	8.692239
Salesforce.com Inc.	0.001524	0.000134	-0.753936	3.632293

Fig. 10: Statistic Summary of the Distributions of Change Rate

	RandomForestRegressor	Baseline
Apple Inc.	0.000096	0.000177
Microsoft Corporation	0.000082	0.000169
Alphabet Inc.	0.000105	0.000176
Amazon.com Inc.	0.000089	0.000162
Intel Corporation	0.000103	0.000206
Cisco Systems, Inc.	0.000036	0.000067
NVIDIA Corporation	0.000232	0.000497
Adobe Inc.	0.000078	0.000153

Fig. 12: MSE of Random Forest Regressor and Baseline

financial events. In general, the fluctuation is in the range of plus or minus 0.2, but occasionally there are large fluctuations beyond this range.

To further test whether the change rate of stock price is a random number. We use the autocorrelation function (ACF) to analyze it. ACF is a tool in time series analysis, It is widely used to measure the correlation of time series at different time intervals. From Figure 14, we find that near day 0, the autocorrelation coefficient is close to 1, which means that the sequence is highly correlated with itself. However, as the time interval increases, the autocorrelation coefficient decreases rapidly and approaches 0, indicating that there is little correlation between stock price changes at different points in time.

We also explore more into the distribution of change rate as seen in fig 10. We found that the change rate has a distribution with higher kurtosis than the normal distribution, meaning it has heavier tails and lower peaks. The distributions have very small variances, centered around a positive number near 0. Using QQ plot, we found it has a distribution close to the t distribution of the degree of freedom of 4 as seen in fig 11. T distribution decays polynomially, making it a heavy tail distribution and more likely to have extreme values.

Thus, due to the randomness of change rate, Stock prices change almost irregularly. This also shows that the short-term stock market is in fact gambling. After all, holders are essentially guessing a random number.

We continued to use CNN to predict the change rate multiple times. Even if we tried a lot of different configurations, such as using change rate as features, changing hyperparameters, and training every stock one by one, the performance is not satisfying. CNN will try to predict the change rate as nearly a constant close to zero every time, completely ignoring the variability every day. This also corresponds to the aforementioned CNN for predicting price It almost looks like there is nothing a neural network can learn from the past data that is helpful for predicting future change rate. We used Random Forest Regressors to predict the change rate, and transform the predicted values into stock prices. The result we have is performing slightly better than the baseline as seen in fig 12. Noted here our prediction is also extremely close to the baseline. Also, though the random forest regressor fits the training data pretty well, it still performs poorly on the test data. (see fig13 and fig14) This shows that predicting stock is extremely difficult and could be far beyond our current capability.

Fig. 13: Random Forest Regressor's Prediction of Change Rate on Training Data

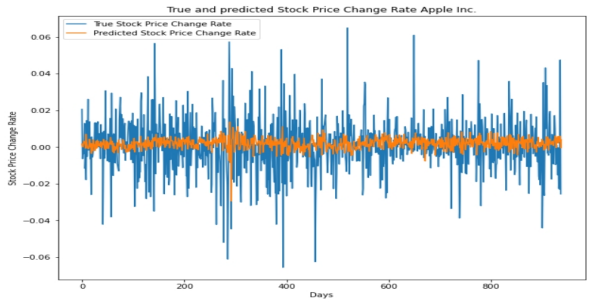

Fig. 14: Random Forest Regressor's Prediction of Change Rate on Testing Data

VI. Conclusion

In this paper, we use CNN, RNN, LSTM, and a non-deep learning stacking model to predict stock prices. Experiments show that RNN has the best performance. LSTM performance is a bit better than RNN. The non-deep learning stacking method is far worse than deep learning method. With further analysis, we find that the change rate of stock price is largely a random variable, with a little correlation between them. Hence, when the target of the model is set to be the price, the prediction is largely based on the previous day, as the neural network becomes lazy and finds a shortcut to the problem, which makes complex models less efficient than simple models. To really develop a method to predict stock price could be far beyond our current capability.

VII. Duty of each group member

Workload and tasks will be distributed evenly to the two team members. The idea is proposed by Yifei. And the methods are discussed and introduced by both Yifei and Tingfu. Coding is done by two teammates. Tingfu is in charge of RNN, LSTM, and Stacking. Yifei is in charge of CNN and the change rate. The writing is done by Tingfu and Yifei.

References

[1] M. M. Kumbure, C. Lohrmann, P. Luukka, and J. Porras, "Machine learning techniques and data for stock market forecasting: A literature review," *Expert Systems with Applications*, vol. 197, p. 116659, 2022.

[2] Z. Hu, Y. Zhao, and M. Khushi, "A survey of forex and stock price prediction using deep learning," *Applied System Innovation*, vol. 4, no. 1, p. 9, 2021.

[3] H. Maqsood, I. Mehmood, M. Maqsood, M. Yasir, S. Afzal, F. Aadil, M. M. Selim, and K. Muhammad, "A local and global event sentiment based efficient stock exchange forecasting using deep learning," *International Journal of Information Management*, vol. 50, pp. 432–451, 2020.

[4] P. Patil, C.-S. M. Wu, K. Potika, and M. Orang, "Stock market prediction using ensemble of graph theory, machine learning and deep learning models," in *Proceedings of the 3rd international conference on software engineering and information management*, 2020, pp. 85–92.

[5] Y. Baek and H. Y. Kim, "Modaugnet: A new forecasting framework for stock market index value with an overfitting prevention lstm module and a prediction lstm module," *Expert Systems with Applications*, vol. 113, pp. 457–480, 2018.

[6] L. Ni, Y. Li, X. Wang, J. Zhang, J. Yu, and C. Qi, "Forecasting of forex time series data based on deep learning," *Procedia computer science*, vol. 147, pp. 647–652, 2019.