Prediction of Stock Market Closing Prices and Time Series Analysis using Recurrent Neural Networks with Long-Short Term Memory

Sirisha Satish
Erik Jonsson School of Engineering and Computer Science
University of Texas at Dallas Richardson, Texas
sxs210095@utdallas.edu

Anish Joshi
Erik Jonsson School of Engineering and Computer Science
University of Texas at Dallas Richardson, Texas
axj200101@utdallas.edu

Sandra Jayakumar
Erik Jonsson School of Engineering and Computer Science
University of Texas at Dallas Richardson, Texas
sxj210016@utdallas.edu

Sherin Jayakumar
Erik Jonsson School of Engineering and Computer Science
University of Texas at Dallas Richardson, Texas
sxj210018@utdallas.edu

***Abstract*—Forecasting of future trends is essential in risk management. Investors perform time series forecasting to predict future trends. Governments and other institutions heavily rely on stock market forecasting to make investment decisions. The analysis of stock prices possesses challenges due to the dimensionality and randomness of data. The aim of this research is to employ Long Short-Term Memory (LSTM) which is a variant of Recurrent Neural Network (RNN) for time series forecasting of the stock market. LSTMs are designed to handle long-term dependencies in data by employing gating mechanisms. LSTM also addresses the vanishing gradient problem that arises during neural network training making it an effective architecture for stock market forecasting. The results can aid organizations in aligning strategies depending upon the prevailing market conditions. R2 score and Root Mean Square Error (RMSE) metric are employed to support the results obtained. The results are visualized to analyze the change in stock trends.**

***Index Terms*—Recurrent Neural Network, Long Short-Term Memory, Time series analysis, Stock market.**

I. Introduction

The term stock market refers to buying and selling of shares of publicly listed companies. A country's economy is directly influenced by its stock market. According to Statista the global market capitalization is around 105 trillion USD. Almost all sectors that are involved in a nation's growth and development are influenced by the trading of stocks, like healthcare, information technology, energy and materials and real estate. This knowledge about the stock market and its worldly influence evokes an important question, what if the decisions made in the stock exchange are not only based on experience of traders but also influenced by data. Building models that help make data driven decisions regarding stock exchanges which in turn boosts the growth of the economy.

Our research is to develop a model to forecast stock markets by performing time series analysis. A LSTM model is a recurrent neural network and is perfect for our problem statement because the algorithm is designed to work with sequential data.The Long short-term memory (LSTM) consists of a memory cell, an input gate, an output gate and a forget gate. The job of the memory cell is to remember the previous state, which helps the RNN make decisions using data that is carried on through layers. Another major advantage of LSTM is that it retains input data for a long time in its memory. An informed decision can be made about the future performance of a particular stock by analyzing the past financial stock data.

The research paper is composed of a detailed explanation of the LSTM model and strategies used to implement the model, detailed explanation of the dataset and its preprocessing methods, and the findings of our analysis. The research also describes Recurrent Neural Networks, and the error functions employed in the construction of the LSTM model. The possibility of future work has also been discussed in the paper.

II. Background work

Time-series analysis involves examining data collected over time to identify patterns, trends, and dependencies. The main objective of time series forecasting is to build models that can make accurate predictions about future values based on the historical patterns identified in the data. In our research we use historical stock price data to predict future movements in stock prices. These predictions will help the traders make profitable decisions.

III. Theoretical and Conceptual Study

A. Time Series Analysis

Time series typically refers to data that is collected over time. Analyzing this data is known as time series analysis. It can be performed to understand patterns and underlying

dynamics. The analysis reveals how the variable changes with time and provides additional information about the dependencies between data.

Time Series Analysis is employed to investigate how the change that is associated with an individual data point compares to other variable shifts over a given time period. In stock market forecasting, time series analysis aids in factors such as determining correlation between seasons and stock prices. Financial or stock market data can be effectively analyzed as time series.

B. Recurrent Neural Networks

Output from the previous step is passed to the current layer which is unlike traditional networks where input and output is not dependent on each other. An important feature of RNN is the hidden state which is also referred to as memory state. The hidden state includes information about the previous inputs in the sequence. RNNs use the same parameters through every layer in the network. This is due to the backpropagation property of RNN. In the network, the error is backpropagated and the weights are updated accordingly.

RNN is employed in time series forecasting due to the network's ability to remember previous dependencies. Due to the presence of memory cells, RNNs are well suited for time series forecasting. RNNs can also handle time series data containing varying lengths of historical data.

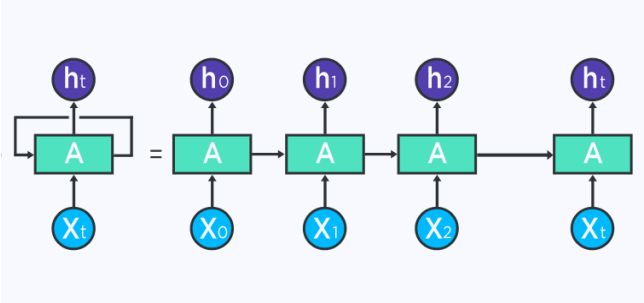


Fig. 1. Recurrent Neural Network Architecture

However RNNs result in vanishing gradients. During training, the network's weights are updated based on the gradients computed. These gradients become very small due to backpropagation resulting in vanishing gradients as RNNs have cyclic connections. LSTM which is RNN variant addresses this issue.

C. Long Short-Term Memory

This architecture is a type of RNN which mitigates the vanishing gradient which can arise in traditional RNN architecture. The memory cell in LSTM enables the network to store information over long sequences. Hence the memory cell resolves the vanishing gradient problem by enabling flow of information through the network for a long duration which is unlike traditional RNN systmes where each element in the sequence is processed one at a time.

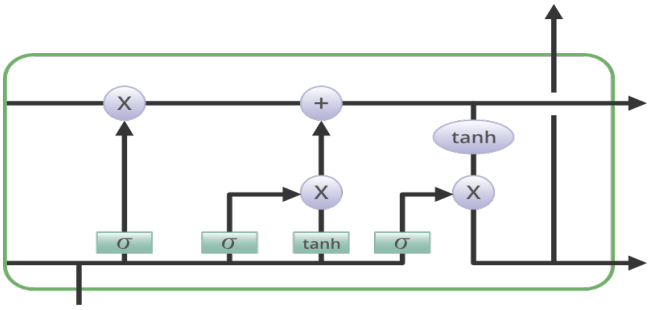


Fig. 2. Architecture of Long Short-Term Memory

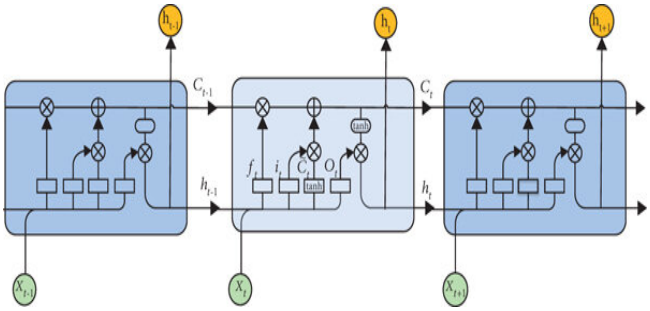


Fig. 3. Reapeating modules in LSTM

Since time series data includes patterns and dependencies that are spread across varying time, LSTM architecture can be effectively employed for time series forecasting. LSTM architecture consists of gating mechanisms which allows the model to direct information flow in the network.

- **Input gate** - It determines new information that would be stored in the memory cell. This ensures proper flow of information from the current step into the memory cell.

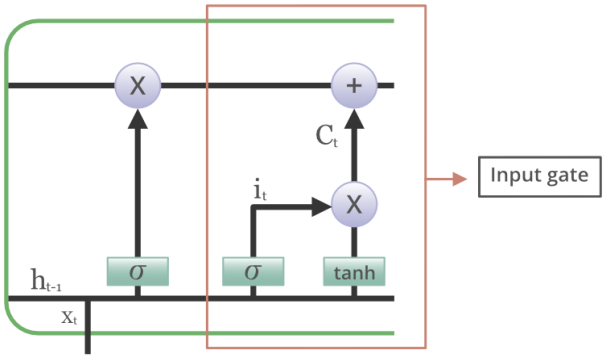


Fig. 4. Input gate in LSTM

- **Forget gate** - It determines whether to discard the information returned from the previous state. It is also known as remember gate.

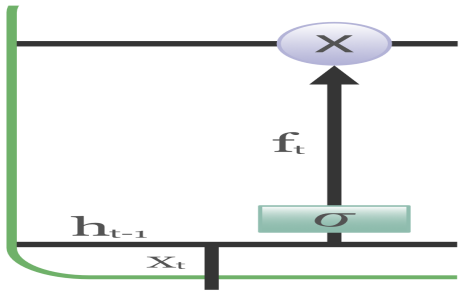


Fig. 5. Forget gate in LSTM

- **Output gate** - It determines what information in the memory cell should be included in the next time step.

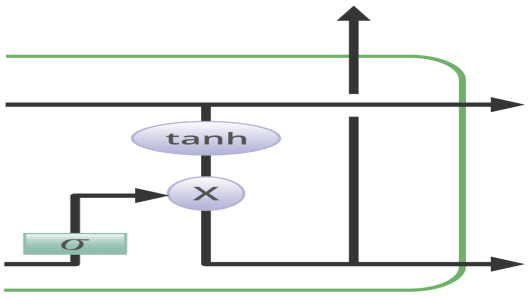


Fig. 6. Output gate in LSTM

- **Cell State** - It represents long-term memory nature that is found in LSTM. This state is updated at every new time step depending on the input, forget and the output gate. The cell state allows the network to capture dependencies spanning multiple time steps.

- **Hidden State** - The hidden state could be considered as a filtered cell state version that captures patterns and dependencies in data which extends across long sequences. While cell state retains long-term memory, hidden state works with information that is important for immediate decisions of the model.

  Incorporating these gating mechanisms enables LSTM model to efficiently study and record long-term dependencies. This makes the LSTM model a popular choice for time series forecasting.

- **Short-Term Memory:** - This memory refers to memory cell and its contents that are manipulated during the processing of the current input. The gating mechanisms control the flow of information in this memory.

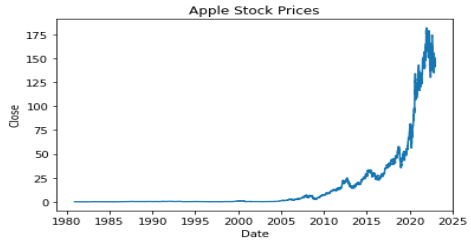
- **Long-Term Memory:** - It refers to the cumulative information that is stored across multiple time steps. The gating mechanisms control the information flow in long-term memory. The forget gate enables the LSTM network to selectively retain or discard information from long-term memory which captures long-term dependencies.

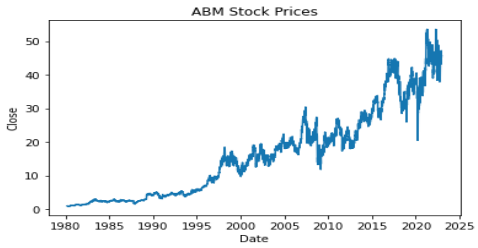
LSTM's short-term memory is responsible for processing of information at the current step whereas the long-term memory refers to information that is accumulated over multiple time steps. This design on LSTM model makes it an effective approach for tasks that involve sequential analysis such as time series forecasting.

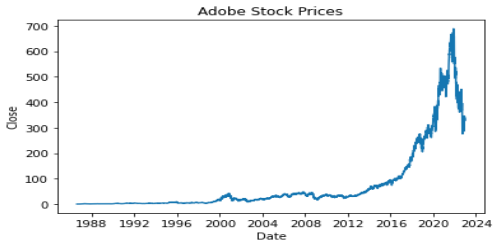
IV. Implementation Details

A. Dataset Details and Preprocessing

For this project, we have utilized three datasets of three different companies from the Stock Market Data dataset(NASDAQ, NYSE, S&P500 available on Kaggle. These datasets consist of six features (Date, High, Low, Open, Volume, and Adjusted Close) and one target variable (Close). To keep things uniform and prevent overfitting, we started dealing with null values. Upon close inspection, it was found that date isn't required for our use case. Hence, we will be removing the date column during the preprocessing and feature engineering stage. To analyze correlation between features and target variable, a correlation matrix using seaborn has been defined. In order to keep all the features in the same range, we used MinMaxScaler() from sklearn's preprocessing library. This step ensured that our LSTM model would be able to comprehend the datasets in the way we require. The three plots below show the trend of Apple, ABM, and Adobe Stock close prices.







B. Approach

In our approach, our main goal is to observe the trends and predict the closing price of a stock. To achieve this, we have implemented RNN with Long-Short Term Memory from scratch using PySpark in Databricks. Another objective of this project is to minimize the error in our predictions. To check the model performance we used Root Mean Squared Error and R2 score metrics.

C. Technical Details

We implement the LSTM class with the help of two main approaches, i.e., the Forward Propagation and the Backward Propagation.

- **Initialization** - The LSTM class is initialized in the init method with the input dimension, the output dimension, the no of neurons, and learning rate parameters. The Xavier initialization (also known as the Gorot initialization) is a technique used for initializing weights in neural networks. It uses Gaussian Distribution to set the weights with a mean 0 and a variance which is computed depending on the count of input and output neurons. It is beneficial to deal with the vanishing gradient problems in Neural Networks. In order to initialize the initial weights of the gates(input, forget and output gate), the Xavier Initialization was used. Further, cell and hidden state were initialized with zeros using numPy.

- **Activation Functions** - The main aim of an activation function is to introduce non-linearities in LSTM and allow it to capture complex patterns and dependencies in sequential data during both forward and backward propagation. We further defined 2 activation functions namely Sigmoid activation and Hyperbolic Tangent (tanh) activation function.

$$\text{Sigmoid Activation Function: } \sigma(x) = \frac{1}{1+e^{-x}} \quad (1)$$

$$\text{Tanh Activation Function: } \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad (2)$$

- **Forward Propagation** - In the forward pass operation we have calculated cell state update, forget, input and output gates using the above-defined sigmoid and tanh activation functions. This method returns the computed final hidden state.

- **Backward Propagation** - In the backward pass operation we have calculated the gradients with respect to the weights by backpropagating the error through the the sigmoid and tanh activation functions of the forget, input, and candidate value gates.

- **Update Weights** - In the Update Weights method of our LSTM class, the weights are updated using the gradients that were computed during the backward propagation. This step is an essential part of the training process because it involves the adjustment of weights in the direction that reduces error/loss. The update operation follows the basic gradient descent principle, where the weights are tweaked by subtracting the product of the learning rate and the respective gradient.

- **Root Mean Squared Error** - This method calculates the root mean squared error between actual and predicted values. A low RMSE value results in good model performance.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{\text{true},i} - y_{\text{pred},i})^2} \quad (3)$$

- **Fit LSTM method** - We have divided the data in 80 is to 20 ratio which means that the training process uses 80% of the data and the rest of the data is used for the testing process. For splitting the datasets, scikit Learn's train_test_split method has been used. The fit LSTM method trains our model by iterating through the training data for a certain number of epochs, performs forward and backward propagations, updates weights and lastly

calculates the root mean squared error for every epoch.

- **Predict method** - In this method, the model employs the learned LSTM model to predict the result for a given input sequence.

V. Results and Analysis

Our model was subjected to different hyperparameter settings. Three key hyperparameters are learning rate, neuron and epoch count. The tables below gives a detailed overview of the effects of different hyperparameter settings to each of the dataset. Upon further analysis, it is noted that as the epoch count increases, our model performs decently well with all the datasets (RMSE decreases and the R2 score increases). However, when the model runs with 50 epochs, it can be observed that the model performs poorly (RMSE increases and the R2 score decreases from the previous attempt) than the previous attempt of 20 epochs on each of the datasets. Coming to the learning rate, the model performs decently with changes in learning rate from 0.01 to 0.05. Major changes can be observed in the metrics of the model with the ABM dataset when there is a change in both the learning rate and the number of epochs. The final hyperparameter i.e., the number of neurons affects the model drastically as can be observed in the working of the model with the ABM dataset. We have also plotted the change in the RMSE values with each epoch for each dataset with the best hyperparameter settings.

LSTM Performance Analysis using RMSE and R2 score			
SR NO	**DATASET**	**PARAMETERS**	**EVALUATION METRICS**
1	Apple	No of neurons: 10 No of epochs: 10 Learning rate: 0.01	RMSE (Train Set) = 0.2702 RMSE (Test Set) = 0.25485 R2 score = – 0.94026
2	ABM	No of neurons: 10 No of epochs: 10 Learning rate: 0.01	RMSE (Train Set) = 0.3897 RMSE (Test Set) = 0.3783 R2 score = – 1.40001
3	Adobe	No of neurons: 10 No of epochs: 10 Learning rate: 0.01	RMSE (Train Set) = 0.2170 RMSE (Test Set) = 0.2177 R2 Score = – 0.3793

Fig. 7. Results Table - I

SR NO	DATASET	PARAMETERS	EVALUATION METRICS
1	Apple	No of neurons: 10 No of epochs: 20 Learning rate: 0.05	RMSE (Train Set) = 0.2144 RMSE (Test Set) = 0.2043 R2 Score = – 0.208
2	ABM	No of neurons: 10 No of epochs: 20 Learning rate: 0.05	RMSE (Train Set) = 0.3437 RMSE (Test Set) = 0.3227 R2 Score = – 0.7193
3	Adobe	No of neurons: 10 No of epochs: 20 Learning rate: 0.05	RMSE (Train Set) = 0.2354 RMSE (Test Set) = 0.2279 R2 Score = – 0.4057

Fig. 8. Results Table - II

LSTM Performance Analysis using RMSE and R2 score			
SR NO	**DATASET**	**PARAMETERS**	**EVALUATION METRICS**
1	Apple	No of neurons: 15 No of epochs: 20 Learning rate: 0.05	RMSE (Train Set) = 0.2457 RMSE (Test Set) = 0.2168 R2 Score = – 0.3638
2	ABM	No of neurons: 15 No of epochs: 20 Learning rate: 0.05	RMSE (Train Set) = 0.3872 RMSE (Test Set) = 0.4049 R2 Score = – 1.6196
3	Adobe	No of neurons: 15 No of epochs: 20 Learning rate: 0.05	RMSE (Train Set) = 0.2353 RMSE (Test Set) = 0.2216 R2 Score = – 0.4190

Fig. 9. Results Table - III

LSTM Performance Analysis using RMSE and R2 score			
SR NO	**DATASET**	**PARAMETERS**	**EVALUATION METRICS**
1	Apple	No of neurons: 15 No of epochs: 50 Learning rate: 0.05	RMSE (Train Set) = 0.2193 RMSE (Test Set) = 0.2229 R2 Score = – 0.4427
2	ABM	No of neurons: 15 No of epochs: 50 Learning rate: 0.05	RMSE (Train Set) = 0.4556 RMSE (Test Set) = 0.4484 R2 Score = – 2.2073
3	Adobe	No of neurons: 15 No of epochs: 50 Learning rate: 0.05	RMSE (Train Set) = 0.2560 RMSE (Test Set) = 0.2310 R2 Score = – 0.4627

Fig. 10. Results Table - IV

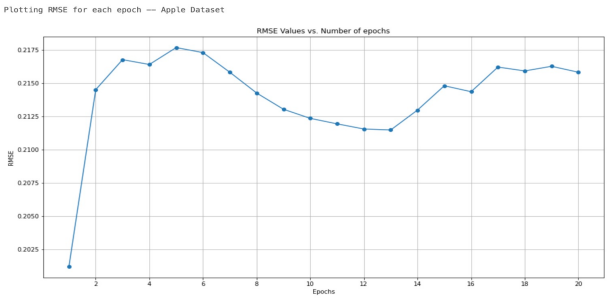


Fig. 11. Graph - I

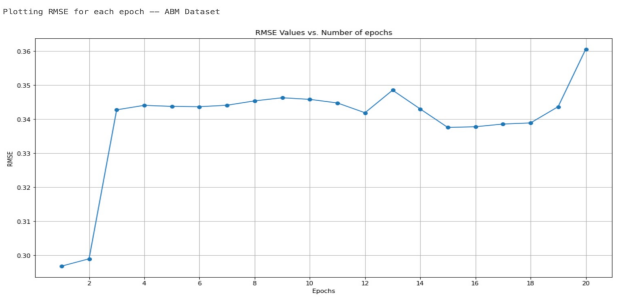


Fig. 12. Graph - II

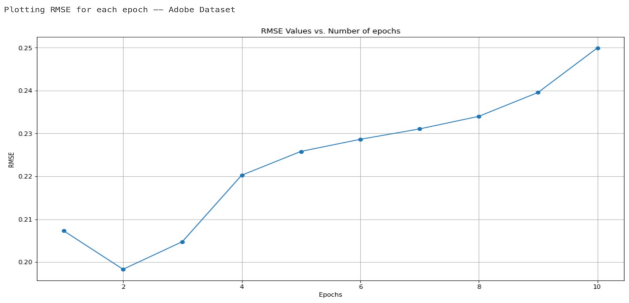


Fig. 13. Graph - III

VI. Conclusion and Future Scope

As a result of this LSTM implementation, we can observe that our model was able to perform decently on our datasets. The ABM dataset was not able to perform well (the R2 score was in high negatives and the error was around 0.40) with our model in comparison to the other two datasets. Hence, further analysis of the ABM dataset separately may help bring down the error. Even though the stock market is entirely uncertain, our model was able to perform decently well and analyze the pattern of ups and downs of the stock market dataset correctly. We can further enhance and extend our project by experimenting with other forecasting methods such as Auto Regressive Integrated Moving Average (ARIMA). Performing further feature engineering by incorporating the date-time (using sine or cosine transformation) into the prediction task might prove useful in developing a future deep-learning model.

References

[1] Hammer, Barbara. Learning with Recurrent Neural Networks. Springer, 2000, https://link.springer.com/book/10.1007/BFb0110016.

[2] Rahuljha. “LSTM Gradients.” Medium, Towards Data Science, 29 June 2020, towardsdatascience.com/lstm-gradients-b3996e6a0296.

[3] Mooney, Paul. “Stock Market Data (NASDAQ, NYSE, S&P500).” Kaggle, 16 Jan. 2023, www.kaggle.com/datasets/paultimothymooney/stock-market-data/data.

[4] Budiharto, Widodo. “Data Science Approach to Stock Prices Forecasting in Indonesia during COVID-19 Using Long Short-Term Memory (LSTM) - Journal of Big Data.” SpringerOpen, Springer International Publishing, 11 Mar. 2021, journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00430-0.

[5] Seth, Neha. “How Does Backward Propagation Work in Neural Networks?” Analytics Vidhya, 8 June 2021, www.analyticsvidhya.com/blog/2021/06/how-does-backward-propagation-work-in-neural-networks/.

[6] “Understanding LSTM Networks.” Understanding LSTM Networks – Colah’s Blog, colah.github.io/posts/2015-08-Understanding-LSTMs/.

[7] Zaccone, Giancarlo, et al. Deep Learning with Tensorflow: Take Your Machine Learning Knowledge to the next Level with the Power of Tensorflow 1.X. Packt Publishing Ltd., 2017.

[8] Author links open overlay panelAlex Sherstinsky, et al. “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network.” Physica D: Nonlinear Phenomena, North-Holland, 21 Jan. 2020, www.sciencedirect.com/science/article/abs/pii/S0167278919305974.

[9] Sherstinsky, Alex. “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network.” arXiv.Org, 31 July 2023, arxiv.org/abs/1808.03314.

[10] SALEM, FATHI M. Recurrent Neural Networks: From Simple to Gated Architectures. SPRINGER NATURE, 2023.