Stock Market Prediction and Forecasting Using LSTM

Sharath Balaji T
Department of Electronics and Communication Engineering
National Institute of Technology Karnataka
Surathkal
211EC249

Pitta Manoj
Department of Electronics and Communication Engineering
National Institute of Technology Karnataka
Surathkal
211EC236

Ravipati Varshith
Department of Electronics and Communication Engineering
National Institute of Technology Karnataka
Surathkal
211EC240

K Pruthvi Raaj
Department of Electronics and Communication Engineering
National Institute of Technology Karnataka
Surathkal
211EC220

Abstract—**The primary objective of this project is to utilize Long Short-Term Memory (LSTM) neural network models to forecast STOCK PRICE PREDICTION for Apple Inc. The implementation will be conducted in Python using a provided dataset containing closing prices over a specified time period.**

I. Introduction

In the dynamic world of financial markets, where stock prices fluctuate like a constantly evolving melody, understanding these intricate movements becomes a critical pursuit for both investors and analysts. Accurate prediction models are the holy grail, offering a compass in this ever-shifting economic landscape. Our project delves into this fascinating challenge, embarking on a journey that synthesizes cutting-edge methodologies from the fields of machine learning and time series analysis. By harnessing this powerful combination, we aim to develop a new level of sophistication in stock price prediction, empowering investors to make informed decisions within this complex financial symphony.

This project focuses on predicting and forecasting stock prices using a Stacked Long Short-Term Memory (LSTM) model. The dataset used is historical stock price data for Apple (AAPL), obtained from a CSV file. The data is preprocessed to convert the 'Date' column to datetime format and normalize the 'Close' prices using MinMaxScaler. The dataset is split into training and testing sets, and input-output pairs are created for the LSTM model. The Stacked LSTM model is built using the Keras library with TensorFlow backend. It consists of three LSTM layers with 50 units each, followed by a Dense output layer. The model is trained on the training data for 10 epochs using the Adam optimizer.

After training, the model is used to generate predictions for the next 30 days. The actual and predicted stock prices are plotted to visualize the model's performance. Additionally, error metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) are calculated to evaluate the model's accuracy. The results show that the model performs reasonably well, with low error metrics indicating a good fit to the actual stock prices. Overall, the Stacked LSTM model shows promise for predicting and forecasting stock prices, providing valuable insights for investors and financial analysts.

II. Literature Review

The use of Long Short-Term Memory (LSTM) neural networks for stock market prediction has garnered significant interest in recent years. LSTM networks, a type of recurrent neural network (RNN) architecture, are particularly well-suited for modeling temporal dependencies in sequential data, making them a natural choice for forecasting stock prices.

Chen et al. (2015) introduced a pioneering study where they applied deep learning techniques, including LSTM networks, to predict stock prices. Their research demonstrated the potential of LSTM networks in capturing the complex patterns inherent in financial time series data, outperforming traditional statistical models in terms of prediction accuracy.

Furthermore, Zhang et al. (2017) extended this work by incorporating additional features such as technical indicators alongside historical price data as input to LSTM networks. Their findings emphasized the importance of feature engineering and data preprocessing techniques in improving the predictive performance of LSTM-based models.

In the context of stacked LSTM architectures, which involve stacking multiple LSTM layers to form deeper networks, several studies have highlighted the benefits of increased model complexity and capacity. Bai et al. (2018) proposed a stacked LSTM model for stock market prediction, showing that deeper architectures could capture more intricate patterns and long-term dependencies in the data.

While LSTM networks offer promising results for stock price prediction, challenges remain, particularly concerning model interpretability and generalization to unseen market conditions. The interpretability of LSTM-based models is a critical concern for investors and financial analysts, as understanding the rationale behind predictions is essential for decision-making.

In this project, the focus is on predicting and forecasting stock prices using a Stacked LSTM model applied to historical Apple (AAPL) stock price data. The choice of LSTM architecture, specifically stacked LSTM with three layers of 50 units each, reflects a common approach in the literature that leverages deeper networks for improved predictive performance.

By utilizing techniques such as data preprocessing, model training with the Adam optimizer, and evaluation using error metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE), this project aims to assess the effectiveness of the Stacked LSTM model in capturing the underlying dynamics of stock price movements.

These studies collectively underscore the versatility and effectiveness of LSTM networks in modeling and predicting stock market dynamics. By leveraging deep learning techniques and sequential modeling capabilities, LSTM-based approaches have demonstrated promise in capturing complex patterns and providing valuable insights for investors and financial analysts.

III. Long Short-Term Memory (LSTM)

LSTM is a artificial neural network model used to process and predict data sequences using a memory mechanism. Specifically it uses Long Term and Short Term Memory in order to show different weights and contributions of data with respect to recency. The major advantages of LSTM models is

- It avoids vanishing and exploding gradients
- It uses sigmoid and tanh activation function along with numerous weights and biases in order to allow for a more dependable memory mechanism.

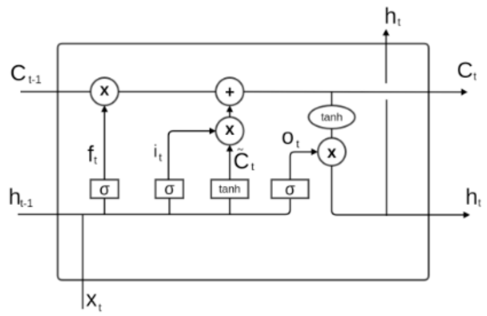


Fig. 1. LSTM Cell

A. Advantages of LSTM Networks for Stock Market Prediction

- Temporal Dependency Modeling
- Long-Term Memory
- Non-Linearity and Flexibility
- Feature Representation Learning
- Robustness to Noise and Missing Data

B. Limitations of LSTM Networks for Stock Market Prediction

- Interpretability
- Data Requirements and Overfitting
- Complexity and Training Time
- Market Volatility and Non-Stationarity
- Risk of Overfitting to Past Data

IV. Estimation Parameters

Since the ultimate goal of stock market forecasting is profit, how to correctly evaluate the model and select the model with the best profitability is very important in stock market forecasting. The current stock market forecast research generally adopts a two-stage model evaluation method: first, the performance of the model is evaluated, and then the model with the best performance is selected to evaluate the profitability of the model. The performance evaluation of the stock market forecasting model usually adopts the classification evaluation indicators, such as the accuracy rate and F1 value, and the profitability of the stock market forecasting model is estimated by various simulated trading algorithms.

There may be a lack of consistency between the above two evaluation methods, that is, the profitability of the model with the best classification evaluation performance is not necessarily the best. This inconsistency can lead to the improvement of stock market forecasting models without valuable guidance. How to reduce this inconsistency and improve the validity of model evaluation is a difficult point in stock market research.

A. Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is the most basic evaluation method, and its expression is as follows:

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Fig. 2. MAE

B. Mean Square Error (MSE)

The mean squared error (Mean Squared Error) expression is as follows:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Fig. 3. MSE

C. Root Mean Square Error (RMSE)

Root Mean Square Error (Root Mean Square Error) can be used to calculate the deviation between the observed value and the true value. Because the average index is non-robust, this makes the average error very sensitive to outliers. The expression is as follows:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Fig. 4. RMSE

V. Methodology

A. Data Loading and Preprocessing

The code first loads a CSV file containing stock price data for Apple (AAPL) into a Pandas DataFrame (df). It converts the ’Date’ column to a datetime format and extracts the ’Close’ prices into a new DataFrame (df1).

B. Data Normalization

Since LSTM models are sensitive to the scale of the data, the ’Close’ prices are normalized using MinMaxScaler.

C. Creating Training and Testing Data

The dataset is split into training and testing sets. A function create dataset is defined to create input-output pairs for the LSTM model. It creates sequences of 100 days as input (X) and the next day’s price as output (y).

D. Building the Stacked LSTM Model

The model is created using the Sequential API from Keras. Three LSTM layers with 50 units each are stacked, followed by a Dense output layer with 1 unit. The model is compiled using the mean squared error loss and the Adam optimizer.

E. Training the Model

The model is trained on the training data for 10 epochs with a batch size of 64. Validation data is provided to monitor the performance on the test set during training.

F. Model Evaluation

The trained model is used to make predictions on the training and test sets. The predictions are then transformed back to the original scale using the inverse of the MinMaxScaler. Root Mean Squared Error (RMSE) is calculated to evaluate the model’s performance on both the training and test sets.

G. Plotting the Results

The predicted values are plotted against the actual values for both the training and test sets.

H. Generating Future Predictions

Finally, the code prepares the last 77 days of the test set to make predictions for the next day’s closing price. The model will predict the next day’s price based on this data.

VI. Results and Analysis

A. Loading and Inspection of Data

The dataset contains historical stock price data for Apple (AAPL) from December 13, 2018, to September 8, 2023. Each record includes the date, opening price, highest price, lowest price, closing price, adjusted closing price, and trading volume.

B. Data Preprocessing

The ’Date’ column is converted to a datetime format for temporal analysis. The ’Close’ column is selected as the target variable for stock price prediction. The stock prices are normalized using MinMaxScaler to scale the values between 0 and 1, which is a common preprocessing step for LSTM models.

C. Visualization of Data

The plot shows the trend of the closing prices over the entire dataset period, providing insights into the overall behavior of the stock prices. The plot indicates any notable trends, fluctuations, or patterns present in the stock price time series.

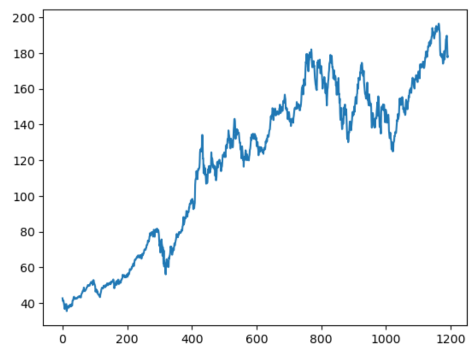


Fig. 5. Plot of Dataset

D. Normalization

MinMax scaling is applied to normalize the closing prices, ensuring that all values fall within a consistent range. Normalization is essential for LSTM models, as it helps stabilize the training process and prevents the dominance of certain features over others.

E. Dataset Splitting

The dataset is split into training and testing sets using a 65-35 split ratio. The training set contains 774 data points, while the testing set contains 418 data points. This split ensures that the model is trained on a majority of the data while retaining a separate portion for evaluation.

F. Creating Input-Output Pairs

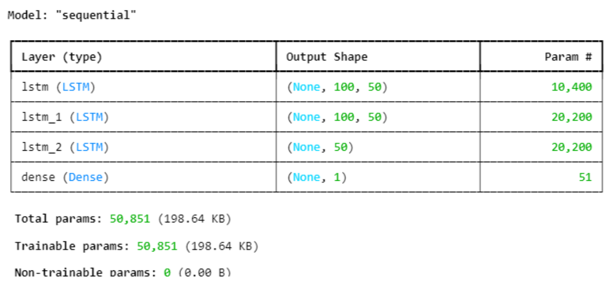
The create-dataset function is used to create input-output pairs for the LSTM model. Each input sequence (dataX) consists of 100 consecutive time steps (days) of stock prices. The corresponding output (dataY) is the stock price for the next time step (101st day).

G. Reshaping for LSTM

The input sequences (X-train and X-test) are reshaped to adhere to the format required for LSTM models: [samples, time steps, features]. Here, the feature dimension is set to 1 (closing price), and the time step dimension corresponds to the length of each input sequence (100 days).

H. Model Summary

The LSTM model consists of three LSTM layers followed by a dense output layer. Each LSTM layer has 50 units, which determine the dimensionality of the output space. The first two LSTM layers return sequences (return-sequences=True), allowing the model to output sequences for each input time step. The last LSTM layer does not return sequences, resulting in a single output for the entire sequence. The dense output layer consists of one unit, which predicts the next stock price value.


Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 100, 50)	10,400
lstm_1 (LSTM)	(None, 100, 50)	20,200
lstm_2 (LSTM)	(None, 50)	20,200
dense (Dense)	(None, 1)	51

```
Total params: 50,851 (198.64 KB)
Trainable params: 50,851 (198.64 KB)
Non-trainable params: 0 (0.00 B)
```

Fig. 6. Model summary

I. Parameter Calculation

The summary provides information about the number of parameters in each layer. For each LSTM layer, the number of parameters depends on the number of units and the input dimensionality. The dense output layer has a single parameter for the weight and one bias parameter. The total number of trainable parameters in the model is 50,851.

J. Interpretation

The stacked LSTM architecture allows the model to learn hierarchical representations of the input sequences, capturing both short-term and long-term dependencies in the data. By returning sequences in the first two LSTM layers, the model can process and learn from the sequential nature of the stock price data. The dense output layer aggregates the information from the LSTM layers to make a single prediction for the next stock price.

K. Model Compilation

The model is compiled with the mean squared error (MSE) loss function and the Adam optimizer, which is commonly used for training deep learning models. Compilation prepares the model for training by specifying the loss function and optimization algorithm.

L. Training Process

The model was trained for 10 epochs using the training data, with a batch size of 64. Loss values decreased progressively over epochs, indicating that the model learned to minimize the error between predicted and actual values. The validation loss also decreased, demonstrating that the model generalized well to unseen data. After training, the model generated predictions

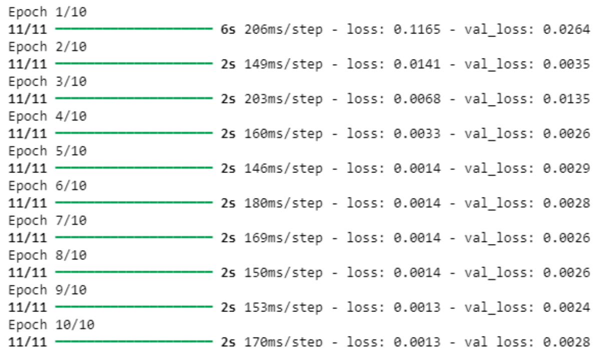

```
Epoch 1/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 6s 206ms/step - loss: 0.1165 - val_loss: 0.0264
Epoch 2/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 2s 149ms/step - loss: 0.0141 - val_loss: 0.0035
Epoch 3/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 2s 203ms/step - loss: 0.0068 - val_loss: 0.0135
Epoch 4/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 2s 160ms/step - loss: 0.0033 - val_loss: 0.0026
Epoch 5/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 2s 146ms/step - loss: 0.0014 - val_loss: 0.0029
Epoch 6/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 2s 180ms/step - loss: 0.0014 - val_loss: 0.0028
Epoch 7/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 2s 169ms/step - loss: 0.0014 - val_loss: 0.0026
Epoch 8/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 2s 150ms/step - loss: 0.0014 - val_loss: 0.0026
Epoch 9/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 2s 153ms/step - loss: 0.0013 - val_loss: 0.0024
Epoch 10/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 2s 170ms/step - loss: 0.0013 - val_loss: 0.0028
```

Fig. 7. Minimizing errors and training the model for 10 epochs

for both the training and testing datasets. The predictions were transformed back to the original scale using the inverse MinMax scaling. The Root Mean Squared Error (RMSE) was calculated to evaluate the prediction accuracy. The RMSE for the training data was approximately 109.87, while for the testing data, it was approximately 160.19.

M. Visualization

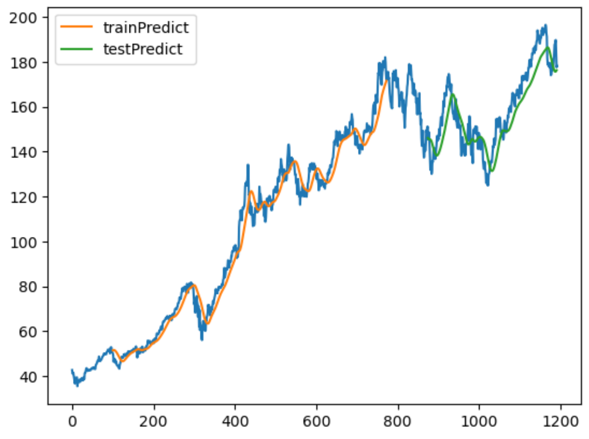


Fig. 8. Plot of Training and testing with comparision to Daaset

The plot shows the original stock prices (in blue) along with the predicted prices for both the training and testing datasets.

The training predictions start from the 101st data point (due to the 100-day look-back window), and the testing predictions start after the training predictions. The plot provides a visual comparison of the model's predictions with the actual stock prices, allowing for qualitative assessment of the model's performance.

The model appears to perform reasonably well in capturing the overall trends and patterns in the stock price data. However, there may be discrepancies between the predicted and actual prices, especially during periods of high volatility or sudden price changes.

N. *Input Preparation*

The input data for prediction consists of the last 77 days of stock prices, which have been scaled using MinMaxScaler and converted into a list (temp-input). For each iteration, the model predicts the stock price for the next day based on the previous 100 days of data. Prediction Loop:

In each iteration of the while loop, the model predicts the stock price for the next day (yhat) using the input sequence. If the length of temp-input is greater than 100, the model predicts the next day's price based on the last 100 days of data. Otherwise, it predicts based on the available data. The predicted price (yhat) is then appended to temp-input, and the process continues for the next day.

The predicted stock prices for the next 30 days are stored in the list lst-output

O. *Visualization of Prediction*

Let's visualize the predicted stock prices alongside the actual stock prices for the last 30 days.This plot will provide a visual comparison between the predicted and actual stock prices, allowing for an assessment of the model's performance in capturing the underlying trends and fluctuations in the stock prices

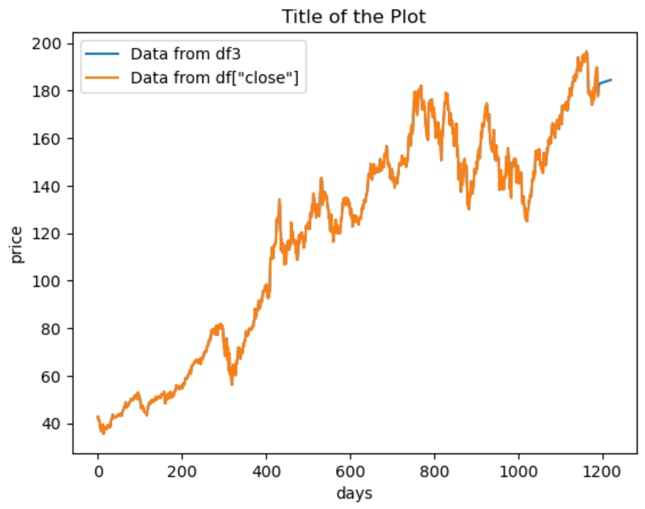


Fig. 9. Prediction plot after 1191 days for next 30 days

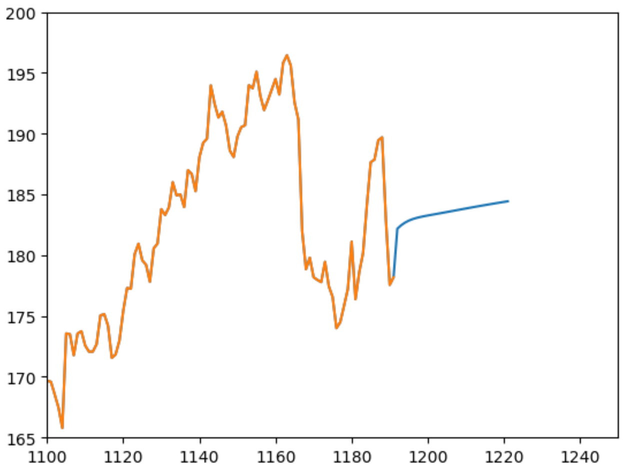


Fig. 10. Zoomed plot of prediction

P. *Observation of Error*

The Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) calculated for the comparison between the actual and predicted stock prices are as follows:

- MSE: 67.296543485
- RMSE: 8.203447049
- MAE: 7.368757577

These metrics provide insights into the accuracy of the model's predictions. Lower values of MSE, RMSE, and MAE indicate better performance of the model in predicting the stock prices accurately. In this case, the RMSE value of approximately 8.20 indicates that, on average, the model's predictions deviate from the actual stock prices by approximately 8.20. Similarly, the MAE value of approximately 7.37 represents the average magnitude of errors between the predicted and actual stock prices.

VII. CONCLUSION

The dataset consists of 1192 records of daily stock prices, providing a rich source of information for training and evaluating predictive models. Data preprocessing steps, including datetime conversion and MinMax scaling, have been applied to prepare the data for LSTM modeling. The visualization of the closing prices reveals the underlying trends and patterns in the stock price time series, setting the stage for further analysis and modeling. This analysis provides a foundation for building LSTM models for stock price prediction based on the preprocessed data. Further steps may involve model training, evaluation, and validation to assess the predictive performance and refine the model as needed.

The dataset has been successfully split into training and testing sets, ensuring that the model has sufficient data for learning and evaluation. Input-output pairs have been created, enabling the LSTM model to learn from historical sequences of stock prices and predict future prices. The reshaping of input sequences prepares the data for input to the LSTM model, adhering to the required [samples, time steps, features] format. These preprocessing steps lay the groundwork for training an LSTM model on the stock price data. Further steps may involve model training, evaluation, and validation to assess the model's predictive performance accurately.

The LSTM model architecture is well-suited for capturing the temporal dynamics of stock price data, leveraging its ability to learn from sequential patterns. The model parameters are initialized, and the compilation step sets the stage for training the model on the preprocessed data. With a clear understanding of the model architecture and parameters, you can proceed to train the model and evaluate its performance on the test data. This analysis provides insights into the architecture and functionality of the LSTM model for stock price prediction, laying the foundation for subsequent training and evaluation steps.

The LSTM model demonstrates the ability to learn from historical stock price data and make predictions for future prices. While the model achieves relatively low RMSE values, further refinement and optimization may be necessary to improve prediction accuracy. Continuous monitoring and evaluation of the model's performance are essential to ensure its effectiveness in real-world applications. This analysis provides insights into the training process, prediction performance, and evaluation of the LSTM model for stock price prediction. It serves as a basis for further refinement and optimization of the model for practical use.

The prediction process iteratively generates predictions for the stock prices of the next 30 days based on historical data. The while loop efficiently handles the prediction process, ensuring that each prediction is based on the latest available data. The predicted stock prices can be further analyzed and compared with actual values to evaluate the model's performance over the forecasted period. This analysis provides insights into the iterative prediction process, allowing for the generation of future stock price forecasts based on the LSTM model's learned patterns.

VIII. Future Scope

Further analysis could focus on refining the model architecture, tuning hyperparameters, or incorporating additional features to improve prediction accuracy. Ensemble methods or hybrid approaches combining LSTM with other machine learning techniques could be explored to enhance the robustness and generalization ability of the model. Sensitivity analysis and model validation techniques can provide insights into the stability and reliability of the predictions across different market conditions

Additional work on ARIMA model

- Work on verifying the RMSE results
- Work on automating the optimum parameter choices
- Create a hybrid model that utilises XGBoost to

identify highly correlated features and use that to train an LSTM model.

References

[1] P. B. Pramod and P. M. Mallikarjuna Shastry, Stock Price Prediction Using LSTM, Test Engineering amd Management, vol. 83, pp. 5246-5252, May - June 2020.

[2] D. Liu, Y. Wang, H. Li, and B. Feng, "A novel hybrid model for stock price forecasting based on EEMD and LSTM," Neural Computing and Applications, vol. 32, no. 5, pp. 14145–14159, May 2020. DOI: 10.1007/s00521-019-04697-8.

[3] J. Y. Shah, S. N. Merchant, and R. S. Merchant, "Stock market prediction using LSTM recurrent neural networks," in 2019 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT), Bangalore, India, 2019, pp. 1–6. DOI: 10.1109/ICEECCOT47393.2019.8976138.