Project Report
Stock Market Prediction using LSTM Neural Networks

Lucía Mañas, Sara Toledo, Noel Otero, Laura Koltraka, Valento Bardhoshi
School of Innovation, Design and Engineering
Mälardalens University, Västerås, Sweden
Email: [lmn25007, sta25002, noa24001, lka24002, vbi24001] @student.mdu.se

***Abstract*—This report presents a comprehensive analysis of our project on stock market prediction using advanced neural network technology. Our work focuses on developing a supportive tool that can forecast stock market prices by analyzing historical pricing data. The system leverages the power of deep learning to identify patterns and trends in stock price movements that might not be immediately apparent through conventional analysis methods.**

***Index Terms*—LSTM Neural Network, Stock Market**

I. Introduction

Stock market prediction has always been an area of great interest for investors, financial analysts and economists. It is very difficult to make exact predictions of stock prices in financial markets because stock prices are influenced by many factors including company performance, economic indicators, global events and market sentiment among others. The results of using traditional statistical methods to model stock price dynamics have been unsatisfactory due to the non-linearity and time dependence of stock prices.

This project solves the problem through the use of a particular type of neural network that is particularly useful for working with sequential data and identifying relationships in time series data. The approach enables the system to capture relevant historical information while discarding noise, thus improving the prediction model. The solution we propose is able to handle large volumes of historical data and learn from time series data, unlike most conventional prediction methods.

The main objective of it is to design and implement a functional prototype that can work with historical stock data and make predictions about the future stock prices within a certain time period. Our system uses the stock data from Yahoo Finance and analyzes the complex patterns in the price history to make forecasts that can be used to inform investment decisions. It is not the intention of the authors to replace human judgment with this tool but rather to provide the user with additional data-driven insights into the possible market trends.

It is a multidisciplinary project that brings together the knowledge of neural network architecture, data processing, and financial market to come up with an interface that allows the user to choose a stock, load the historical data, and even view the past data as well as the forecasted future data.

Although it is impossible to guarantee absolute accuracy of any prediction system for such a complex object as the financial market, our approach shows encouraging results in the identification of general price dynamics and possible market changes. It is a stock market prediction work that uses the advantages of LSTM neural networks to provide predictions on the close price of stocks based on historical data from Yahoo Finance.

Traditional RNNs suffer from short-term memory due to a vanishing gradient problem, which makes it difficult to capture long-range dependencies in sequences. LSTM neural networks were designed to solve the vanishing gradient problem by using memory cells and gating systems that allow effective detection of extended relationships between data points in time series sequences.

II. Related work

For many years, Stock market prediction research has operated by utilizing statistical models together with machine learning techniques and deep learning architectures. During the early stages of research, scientists employed the Autoregressive Integrated Moving Average (ARIMA) and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) as econometric models to analyze time series data and forecast stock prices [1]. While these models showed acceptable accuracy for detecting linear patterns, they were unable to recognize the complex non-linear patterns that existed in financial data.

Because they allowed models to detect non-linear relationships, the implementation of support vector machines (SVMs) and random forests as machine learning techniques improved the predictions [2]. However, the models failed to maintain sequential data dependencies, which made them less suitable for stock market prediction tasks.

Stock market prediction accuracy improved dramatically through the development of deep learning models, especially Recurrent Neural Networks (RNNs). Traditional RNNs experience a vanishing gradient problem that prevents them from effectively handling long-term time series predictions [3]. The solution to this problem came from Hochreiter and Schmidhuber (1997) through the development of Long Short-Term Memory (LSTM) networks.

The memory cells and gating systems in LSTM networks allow the effective detection of extended relationships between data points in time series sequences, which makes them ideal for stock market prediction tasks. Numerous recent studies prove that LSTM-based models deliver outstanding results in predicting stock market performance.

Fischer and Krauss [4] have shown that LSTMs generate more accurate predictions than traditional machine learning methods for stock price movement forecasting. In a study comparing LSTMs to multiple deep learning models, Selvin et al [5] reached the conclusion that LSTMs produced better results because of their enhanced capacity at processing sequential data patterns.

The availability of the Yahoo Finance API for financial data retrieval allows real-time data access, which enhances the practicality of LSTM-based predictive models. According to the article [7], the implementation of hybrid models, which integrate LSTMS with CNNS and attention mechanisms, improved the accuracy of stock market prediction.

Our research assignment implements LSTM networks for stock market prediction by analyzing historical price data retrieved from Yahoo Finance. In addition to financial time series forecasting studies, the research will evaluate how well LSTMs perform in recognizing market patterns to predict upcoming stock prices.

III. Approach and Method

A. *Dataset*

For the project it was used a dataset from "Yahoo Finance" [17], a resource of financial market data, from which we used stock data of different companies to train and evaluate our model. Precisely, the acronyms or symbols of the company for which we want to make predictions are entered into the model and analyzed (for example "TSLA" for Tesla, Inc.) in order to predict its future stock values.

B. *Recurrent Neural Networks*

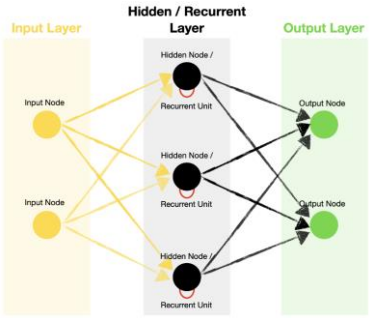
A Recurrent Neural Network (RNN) is a deep learning model designed to process sequential data, such as time series. Neural Networks are inspired by two fundamental principles observed in biological neurons:

- Distributed processing in units (neurons).
- Learning through modification of connections.

Specifically, RRNs are inspired by how the human brain processes sequential information. Unlike traditional feedforward neural networks, RNNs try to imitate how humans remember past information while processing new data using memory, feedback and time-dependent learning. They can carry out these tasks by maintaining a hidden state that carries past information forward, introducing loops in the network
architecture (allowing previous outputs to influence current decisions) and capturing time-dependent patterns by sharing weights across time steps respectively.

The architecture of these neural networks is similar to the feedforward's ones. They are composed of an input layer, hidden layers and an output layer. However, they contain recurrent units in their hidden layers which are responsible for providing the algorithm with the capacity of handling sequential data. It is achieved by recurrently passing a hidden state from the previous timestep and integrating it with the current input.

Fig. 1. Architecture of a Recurrent Neural Network



Recurrent units are represented in Figure 1 [11].

- Xt is the input at time step t, a vector of size N.
- A is the hidden state at time step t, it provides the memory to the network. It is calculated based on the previous hidden state and the input at the current step.

$$A_t = f(W X_t + U A_{t-1})$$

Where W are the weights for input and U the weights of previous state value input and f is the non-linearity applied to the sum to generate final cell state [12].

Nevertheless, traditional RNNs suffer from short-term memory due to a vanishing gradient problem that emerges when working with longer sequential data. This problem occurs during training of deep neural networks when gradients, which are used to update weights in backpropagation, become so small leading to very slow learning (or no learning at all) and difficulty in capturing long-range dependencies in sequences.

During backpropagation, updating the weights relies on the chain rule of differentiation. In deep networks, gradients are multiplied multiple times by small values from activation functions such as sigmoid or tanh, which leads to their exponential reduction.

$$\frac{\partial L}{\partial W} \approx (\text{small number})^n$$

- L is the loss function.
- W represents weights.
- n is the number of layers or time steps.

To solve the vanishing gradient problem, Long Short-Term Memory (LSTM) was designed.

C. *LSTM Neural Network*

LSTMs are a type of RNNs designed to handle sequential data and learn long-term dependencies meaning that it is capable of processing and predicting future data, such as, in our case, predicting future actions in the stock market. LSTM is specifically designed to perform this function thanks to its ability to process the flow of information with different gates present in

its basic unit, the LSTM cell.

Architecture

LSTM is composed of a set of basic units called LSTM cells. Each unit has the same architecture, which is composed of a forget gate, an update gate and an output gate. It also has a state cell which provides to the LSTM the long-term memory.

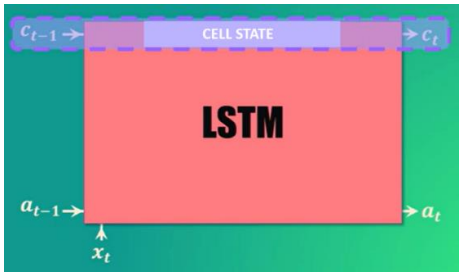


Fig. 2. LSTM Cell Structure

Figure 4 illustrates the structure of an LSTM cell, highlighting its key components. The Cell State (Ct−1→Ct) is represented at the top, showing how information flows through time. At the bottom left, the previous activation state (at−1) and input (xt) enter the LSTM block. The LSTM processes these inputs and produces the new activation state (at) as output.

The previous activation state (at−1a) in an LSTM represents the hidden state from the previous time step. It carries short-term memory and helps the LSTM decide what information to retain, update, or forget at the current step.

The cell state is a fundamental part of the LSTM. Its main function is to keep and transport relevant information across the time, letting the network remember information during long sequences.

Fig. 3. Functioning diagram of a Cell State

The forget gate decides what information from the previous cell state (Ct-1) should be discarded or kept. The gate takes the previous activation state (at-1), which contains short-term memory and current input (xt), which is the new information at times step t, as inputs.

These inputs pass through a sigmoidal activation function (σ), producing values between 0 and 1.

- A value close to 0 means “forget this information”.
- A value close to 1 means “keep this information”

Finally, the output of this gate is multiplied by Ct-1, modifying the cell state.

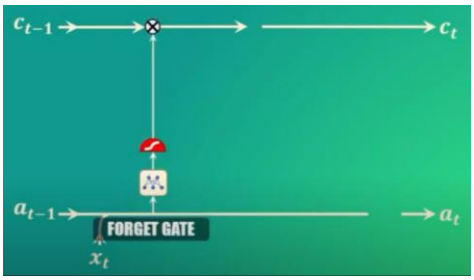


Fig. 4. Functioning diagram of the Forget Gate

The update gate determines what information should be added to the cell state. To carry out this action, the previous activation state and the current input pass through a backpropagation neural network with a final sigmoidal activation function that is going to decide what values are going to be updated, those that are close to 1.

The second part is a tanh function that scales the input values between -1 and 1, normalising them and creating in that form a candidate memory. Then, by multiplying both results elementwise and summing this result to the forget gate one, it is going to be obtained the new cell state.

The result is multiplied element-wise and then added to the cell state to update it.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

- ft is the forget gate output.
- it is the input gate output.
- C~t is the candidate memory.

The output gate decides which part of the updated cell state should be used as the new activation state (at). First a sigmoidal activation function processes at-1 and xt. Then, the updated cell state Ct is passed through a tanh function, scaling it between -1 and 1. Finally, the output gate´s result is multiplied element-wise with tanh(Ct) to produce at:

$$a_t = o_t \times \tanh(C_t)$$

This final output will be used in the next time step.

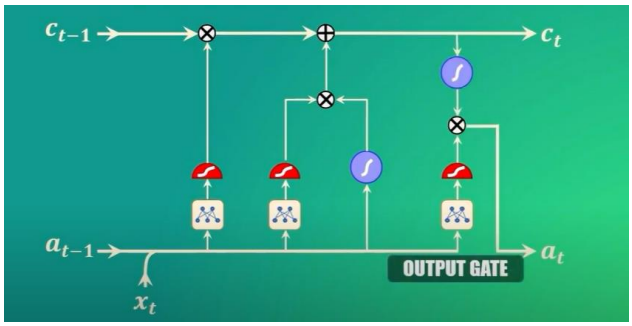


Fig. 5. Functioning diagram of the Output Gate

Finally, this is the full LSTM process by combining all three gates:

1. Forget gate removes unnecessary information from Ct-1.
2. Update gate adds new relevant information to Ct.
3. Output determines the final activation state at.

The final equations that summarize the entire mechanism are:

$$f_t = \sigma(W_f \cdot [a_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [a_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [a_{t-1}, x_t] + b_C)$$
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$
$$o_t = \sigma(W_o \cdot [a_{t-1}, x_t] + b_o)$$
$$a_t = o_t \times \tanh(C_t)$$

The weights (W) and biases (b) are learned parameters, meaning they are initialized randomly and adjusted during training using backpropagation through time. Each LSTM gate has its own weights and biases.

W_f → Forget gate

W_i → Input gate

W_C → Candidate memory (new cell state proposal)

W_o → Output gate

b_f → Forget gate bias

b_i → Input gate bias

b_C → Candidate memory bias

b_o → Output gate bias

D. Practical implementation of LSTM

In order to make use of this deep learning model to predict the future stock market, it has been used python to implement the practical approach.

The LSTM network follows a specific architecture:

- 3 LSTM layers with 50 units each.
- 3 Dropout layers.
- 1 Dense layer.

TABLE 1

Layer (type)	Output Shape	Units	Activation Function	Dropout rate
LSTM_1	(None, 60, 50)	50	activation="tanh" recurrent_activation="sigmoid"	
Dropout_1	(None, 60, 50)			0.2
LSTM_2	(None, 60, 50)	50	activation="tanh" recurrent_activation="sigmoid"	
Dropout_2	(None, 60, 50)			0.2
LSTM_3	(None, 50)	50	activation="tanh" recurrent_activation="sigmoid"	
Dropout_3	(None, 50)			0.2
Dense_1	(None, 1)		linear	

Architecture (Layers) of the LSTM Neural Network

LSTM Layers

LSTM layers are responsible for processing sequential data and capturing relations across the time. Each LSTM layer is composed of 50 units, that is to say, 50 neurons with their recurrent unit (LSTM cell). Keras assumes as activation functions of each neuron the sigmoid and the tanh ones, so it is not necessary to specify them in the code unless other activation functions want to be used.

Dropout Layers

Dropout layers are used as a regularization technique to prevent overfitting. They randomly set a percentage of input units to 0 at each step during training time. The percentage is known as the dropout rate. Then, the rest of the inputs that have not been set to 0 are scaled up by the factor 1/(1-rate) to keep the activation average constant.

In this LSTM, the first dropout layer has a rate of 0.2 which means that the 20% of the input neurons are set to 0 while the remaining 80% are scaled by 1.25, which ensures the total sum of activations to keep stable. Without this scaling, the model will receive less information in each iteration, and it could become unstable.

Dense Layer

Dense layers are completely connected in the neural network, where each neuron of the layer is connected to all the neurons in the previous layer (third LSTM layer), which provides the final hidden state. Their function is to combine

all the temporal features extracted by the LSTM layers to make the final prediction.

In this case, the first two LSTM layers return full sequences ensuring structured information flows through the next layer, while the last LSTM layer produces only its final hidden state. This hidden state passes to the Dense layer with a single neuron which transforms it into the predicted value.

Since this is a regression task, the Dense layer uses a linear activation function (keras assumes it and it is not necessary to specify it) to allow continuous outputs. To sum up, LSTM layers refine temporal features, and the Dense layer consolidates them into the final result.

Model Training

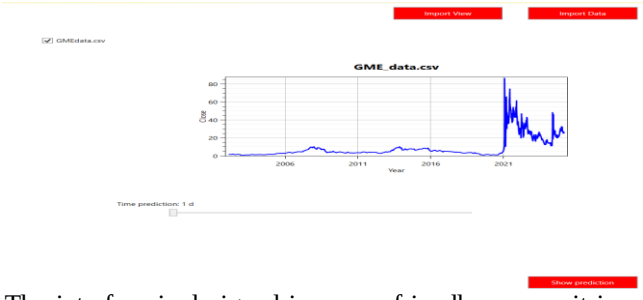
In terms of how the model is trained, two aspects should be mentioned:

- Utilization of "Adam" (Adaptive Moment Estimation), an optimization algorithm to adjust the weights of the neural network and minimize the loss function.
- Use of the $$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$ "mean_squared_error", which defines how the error of the model is measured in each iteration. It calculates the difference between the prediction of the model (ŷ) and the real value (y) and n is the total number of data. It´s formula can be defined as:

Regularization techniques

Some regularization techniques were used to avoid overfitting, such as dropout layers or EarlyStopping technique, which stops the training process if the loss in validation does not improve in a determined consecutive number of epochs, 15 in our case.

E. Interface design



The interface is designed in a user-friendly way so it is easily available and usable for anyone expert or novice in the stock market fields. The data to import is available using a python script from Yahoo Finance which could be added to the interface in a search bar program to improve the accessibility. After importing the data, a graph view of it is shown to the user with the bar to select time and the button to make the prediction.

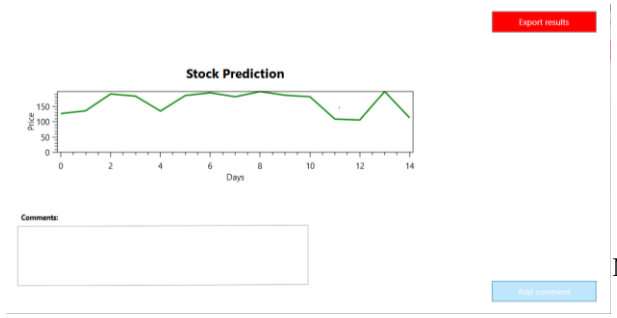


Fig. 6. Main window to make predictions and import data

After pressing the "Show prediction" button, another window is opened with the results of the prediction giving the option to put comments and export the data as a .csv file.

Fig. 7. Show prediction window with the prediction from the LSTM

Using this data we can import it again from the main window using the button "Import View" and selecting the file, after that another window is open showing the view and its comments.

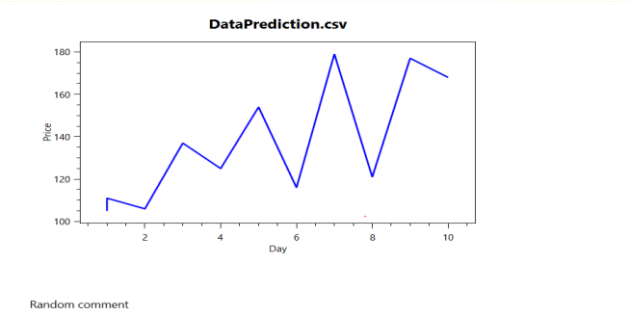


Fig. 8. ShowView window with the imported view and its comments

This design allows the user to import, export and see data so it can be analyzed and used in other programs or mixed with other inputs to make predictions.

IV. Results and Evaluation

The results obtained within this project after the implementation of the LSTM reveal the prediction in the evolution of stock prices in a determined time interval. These predictions allow us to observe the stock's price trend (rise, fall or remain unchanged), as well as estimate their price at a given time.

In order to evaluate the previous results, a validation graph was generated based on predicted stock prices for the training and testing sets over time for a selected companie, as shown in Figure

Fig. 9. Graph of the actual and the predicted stock prices for training and testing datasets.

The previous figure represents the actual stock prices for a selected company (symbol "PSLV") plotted in black, compared to the predicted prices for training and test sets, plotted in red and blue, respectively. This visual representation testifies a strong and successful learning trend as we can observe, due to the next observations:

- The train set prediction (red line) closely follows the actual price (black), suggesting the model has learned well on the training data.
- The test set prediction follows the actual price as well, but its accuracy has been further evaluated by using the Root Mean Squared Error (RMSE).

The Mean Squared error (MSE) is a metric used to evaluate the accuracy of a predictive model. It measures the average squared difference between the predicted values (predicted stock prices) and the actual values [16]. By taking the square root (RMSE), we obtain those values in the original units of the stock dataset. The formula for the root mean squared error is:

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

Where:

- n is the number of observations in the dataset.
- Yi are the actual values of the observations.
- $\hat{Y}$i are the predicted values.

A lower RMSE value indicates that the model's predictions are closer to the actual values, signifying better accuracy. On the contrary, a higher RMSE suggests poorer performance as the model's predictions deviate further from true values.
By applying the RMSE metric on both datasets (train and test) we obtained the values shown in Figure .

```
Root mean squared error:
Train set: 8.817686661079847
Test set: 8.577297402864092
```

As both training and testing RMSE values are similar and low (around 8), it is apparent that the model generalizes well and it is not overfitting.
Overall, taking into account previous observations from the graph and the RMSE values from evaluation, it can be said that the model is accurate and reliable. Therefore, it is useful for forecasting future prices.

V. Conclusion

After analyzing the results obtained by the test programs we could reach severe conclusions regarding the usefulness of the LSTM in Stock Market operations, compared with other tendency prediction methods. LSTM provides more capacity to handle big inputs of data such as historical stock data from different businesses, keeping better management of memory than other methods like regression methods, which can be more easy to use but with smaller inputs thus being less accurate.

Making a summary of the results, we can observe a pretty big percentage of accuracy in the prediction of several days after the last data input, which provides us a useful tool to study and predict the price tendency of any business in the stock market.

The program is also open to further development, adding filters to see and predict different data inputs such as change percentage, volume, market cap… It can also be used in real cases, not as the only and main, but a useful support tool to study the market and make investments mixed with other tools or techniques like studying the business news or new products.

There are real life examples of tools with the same functionality [8] or using other approaches like sentiment analysis [9] and also using social media data or the news [10]. This proves the flexibility of LSTM in Stock Market predictions and its general usefulness.

Overall, the program proved to be accurate so it could reach more utilities with further development in the future, providing a useful tool in stock market operations.

VI. References

1. Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control*. John Wiley & Sons.
2. Cao, L. J., & Tay, F. E. H. (2001). Financial forecasting using support vector machines. *Neural Computing & Applications*, *10*(2).
3. Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2).
4. Hochreiter, S., & Schmidhuber, J. (1997). Long

SHORT-TERM MEMORY. *NEURAL COMPUTATION*, *9*(8).

5. FISCHER, T., & KRAUSS, C. (2018). DEEP LEARNING WITH LONG SHORT-TERM MEMORY NETWORKS FOR FINANCIAL MARKET PREDICTIONS. *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH*, *270*(2).
6. SELVIN, S., VINAYAKUMAR, R., GOPALAKRISHNAN, E. A., MENON, V. K., & SOMAN, K. P. (2017). STOCK PRICE PREDICTION USING LSTM, RNN, AND CNN-SLIDING WINDOW MODEL. IN *PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTING, COMMUNICATIONS AND INFORMATICS (ICACCI)* .
7. KIM, Y., & KIM, H. (2019). STOCK MARKET PREDICTION USING HYBRID MODELS OF LSTM AND CNN. *IEEE ACCESS*, *7*.
8. PINYU CHEN, ZOIS BOUKOUVALAS & ROBERTO CORIZZO (2024). *A DEEP FUSION MODEL FOR STOCK MARKET PREDICTION WITH NEWS HEADLINES AND TIME SERIES DATA. SPRINGER NATURE LINK.*
9. WENJUN GU, YIHAO ZHONG, SHIZUN LI, CHANGSONG WEI, LITING DONG, ZHUOYUE WANG, CHAO YAN(2024). *PREDICTING STOCK PRICES WITH FINBERT-LSTM: INTEGRATING NEWS SENTIMENT ANALYSIS. CORNELL UNIVERSITY*
10. WASIAT KHAN, MUSTANSAR ALI GHAZANFAR, ALI JAVED, FAHEEM ULLAH KHAN(2024). *STOCK MARKET PREDICTION USING LSTM MODEL ON THE NEWS AND SOCIAL MEDIA DATA*
11. *SAUL DOBILAS(2022). LSTM RECURRENT NEURAL NETWORKS - HOW TO TEACH A NETWORK TO REMEMBER THE PAST*
12. *VIVEK SINGH BAWA(2017). BASIC ARCHITECTURE OF RNN AND LSTM..*
13. *SHIVANSU AGGARWAL(2024). THE ULTIMATE GUIDE TO BUILDING YOUR OWN LSTM MODELS. PROJECTPRO*
14. *SEPP HOCHREITER, JÜRGEN SCHMIDHUBER(1997). LONG SHORT-TERM MEMORY.*
15. *KAMILYA SMAGULOVA & ALEX PAPPACHEN JAMES(2019). A SURVEY ON LSTM MEMRISTIVE NEURAL NETWORK ARCHITECTURES AND APPLICATIONS. SPRINGER NATURE LINK.*
16. *(2025). MEAN SQUARED ERROR. GEEKS FOR GEEKS.*
17. *YAHOO FINANCE.*