Predicting NYSE Stock Exchange prices using Recurrent Neural Network with Long Short-Term Memory

Fatema Hussain
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Richardson, Texas
fxh190006@utdallas.edu

Mariam Aafreen Muhammed Moinuddin
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Richardson, Texas
mxm210042@utdallas.edu

Venkata Kowsik Temididapathi
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Richardson, Texas
vxt200001@utdallas.edu

Veda Nandan Gandi
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Richardson, Texas
vxg200001@utdallas.edu

***Abstract*— New York Stock Exchange (NYSE), one of the world's largest marketplaces for securities and other exchange-traded investments. The prediction of the stock market has reached a technologically enhanced era because of global digitization. The analysis is still a major issue due to the dynamics of the data. This research demonstrates how to use the Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) approach to estimate future stock prices using data from the last ten years. Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) eliminates long-term dependency due to its unique storage structure, which aids in stock time series prediction. Furthermore, we train the model with various parameters. Mean Absolute Error (MAE), R2 Score and Root Mean Squared Error (RMSE) are used for accurate calculation. The results are visualized using graphs to bet a better idea using the training set and test set.**

***Keywords*— Long Short-Term Memory, Time series, NYSE, Recurrent Neural Network, Stocks**

I. INTRODUCTION

The course of business has changed dramatically in recent decades because of enormous technological breakthroughs. Financial markets are critical to a country's economy. The global stock market capitalization reached 68.654 trillion US dollars in 2018, according to the World Bank. The stock market is a marketplace where financial backers can buy and sell stocks or investable resources. Shareholders or investors profit either by selling their shares or by investing in companies that pay dividends on a regular basis. Making a model that learns from previous data and predictions stock values using that data is extremely profitable and beneficial.

The goal of this research is to develop a model for price prediction. Short-Term Memory (Long) because the data is a time-series dataset, one of the best fit is a RNN. Long short-term memory architecture allows the Recurrent Neural Network to remember input for a long time and is best suited for processing sequential data. After you've completed the learning process, you'll need to change the weights. To learn data and forecast future patterns, recurrent neural networks make use of prior stages. The data is stored in Long Short-Term Memory, which is then utilized by the Recurrent Neural Network. It is now possible to operate and anticipate in a dynamic manner.

We implemented our model using python, to predict stock prices efficiently. Our study will aid the stock market in the efficient prediction of stock prices.

In other parts, we've summarized our findings and approaches. This study will assist the stock market in better accurately anticipating stock values.

The following sections comprise the proposed paper: Section II discusses the literature study, which provides a summary of the stock market and prior strategies for predicting stock values. Our research is focused on machine learning techniques, such as deep learning and neural networks, time series analysis, and graph-based approaches[2].

The following sections comprise the proposed paper: The second section covers the basics of the stock market and outlines the numerous stock price prediction strategies. Our research in machine learning algorithms, RNNs and LSTMs, neural networks, Time Series Analysis, and graph-based methodologies is summarized in Section III. In Section IV, we look at our dataset and preprocessing processes, while in Section RESULTS AND ANALYSIS, we talk about our model and the results of our experiment. Finally, Section CONCLUSION AND FUTURE WORK brings the paper to a close and describes the scope of future work.

II. BACKGROUND WORK

We can evaluate longitudinal data collected on single units using time series analysis and associated research approaches. The goal of time-series forecasting is to utilize these models to anticipate future values based on historical data. It assists in discovering patterns in various observations and drawing conclusions from them. These findings are then used to better predict future stock prices data.

III. THEORETICAL AND CONCEPTUAL STUDY

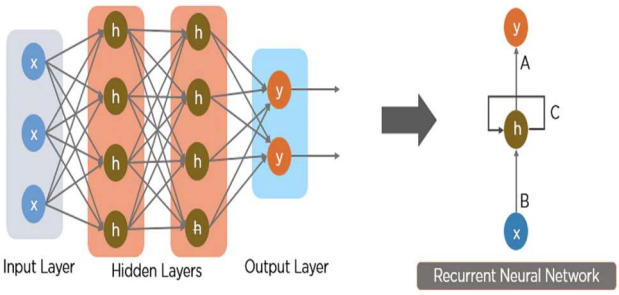
A. Time Series Analysis:

A time series is a set of data that follows the progression of a sample over time. It is a sequence of data points that occur in a specific order. Time series analysis is useful for determining how an asset, security, or economic variable change over time. This methodology uses a single-dimensional input consisting of historical data to forecast the stock price. The dataset will be used to train the model, which will then forecast future values.

The stock price was represented as a time series in the publication "Share Price Prediction Using Machine Learning Technique" [1]. For creating predictions that were very near to the actual number, the paper used a Recurrent Neural Network using normalized data.

B. Recurrent Neural Networks:

A recurrent neural network (RNN) is an artificial neural network that generally works with sequential time-series data. Recurrent neural networks' output is dependent on the sequence's preceding components, but traditional deep neural networks' inputs and outputs are assumed to be independent of one another. As a result, it is based on the concept of retaining the output of a layer and feeding it back into the input to forecast the layer's output [3].



RNN is the best suit for our scenario because of its memory persistence feature. Standard RNNs, on the other hand, have a minor flaw. To train an RNN back algorithm over time, we must determine the gradient at each step. This gradient is then used to update the network's weights. The gradient value will be low if the prior layer's effect on the current layer is minimal, and vice versa. If the gradient of the previous layer is modest, the gradient of the next layer will be even smaller. Because of the vanishing gradient problem, the gradients become progressively smaller as we backpropagate, making it difficult to learn some long-term dependencies. It is difficult to train an RNN. As a result, we'll have to include all past contributions up to the present time.

C. Long – Short Term memory architecture:

LSTMs are distinguished from more traditional feedforward neural networks by the existence of feedback connections. LSTMs process entire data sequences rather than dealing with each point separately. As a result, LSTMs are particularly well-suited to time series analysis [4]. Fundamental ideas in LSTMs are the cell state and its gates, which control how information in a sequence of data enters, is stored in, and leaves the network. The cell state functions as a highway that transports relative information down the sequence chain.

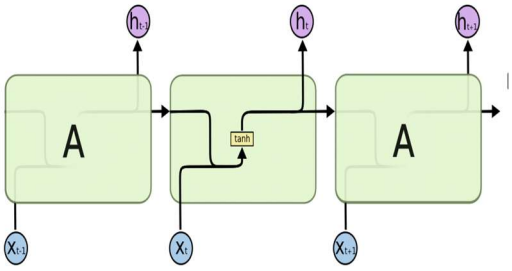
Long-Term Memory (LTM) and Short-Term Memory (STM) are dealt with by LSTMs, which make calculations simple and effective. The concept of gates is used.

1. Forget Gate: LTM discards information that is no longer useful as it enters the forget gate.

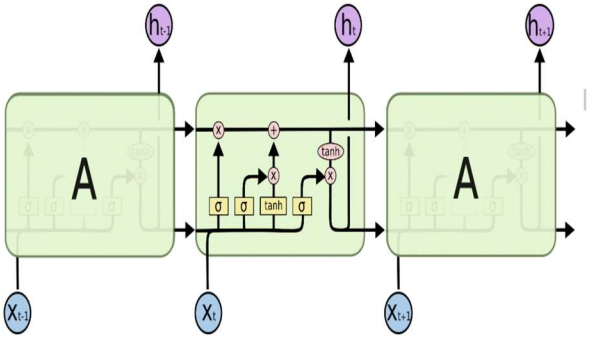
2. Learn Gate: The event and STM are combined so that STM's relevant data can be applied to the present input.

3. Remember Gate: Not-forgotten LTM data, as well as STM and Event data, are integrated into Remember Gate, which serves as an updated LTM.

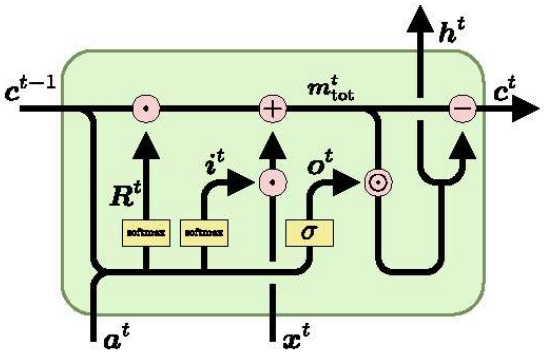
4. Use Gate: This gate, like the LTM, STM, and Event, use LTM, STM, and Event to forecast the output of the current event and acts as an updated STM. The image below shows the distinction between RNN and LSTM.



The repeating module in a standard RNN contains a single layer.



The repeating module in an LSTM contains four interacting layers.



D. RMSE (Root Mean Square Error):

The Root Mean Square Error (RMSE) is a standard method of calculating a model's error in predicting quantitative data. Based on the best available pattern, it illustrates how densely packed the data is.

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

E. MAE (Mean Absolute Error):

MAE may be used to measure the model's accuracy, and we can check how well the predictions match the dataset's original values. It's derived by dividing the difference between true and predicted/forecasted values by the mean of the absolute differences between true and predicted/forecasted values.

$$\mathrm{M} = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

F. R^2 (R-SQUARED):

The goodness of fit of a regression model is represented by R-squared, a statistical measure. The optimal r-square value is 1. The closer the r-square value is to 1, the better the model fits.
The R-square is a calculation that compares the residual sum of squares to the total sum of squares.

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

IV. DATASET AND PREPROCESSING

The dataset is a CSV file that was obtained from Kaggle Stock Market Data (NYSE). It includes a variety of stock rates, including date, high, volume low, and weighted price, but our aim would be on the closing rates. Every minute, data is collected. As the dataset contained a huge number of IsNull() values in the closed rates and date time-stamps, we have eliminated all of the IsNull values in order to avoid any errors in the calculations. The CSV file(dataset) was used to obtain the time-stamp and closed-rate column values. Before getting rid of the time-stamp column, the time-stamp was transformed into a date format and the last closed-rate value recorded in a single day was noted down.

Dataset consists of a wide range of information. In order to overcome the problem caused by overfitting or extreme values(high and low) of the gradient while implementing gradient descent in the back-propagation, we can update the weights of the various gates of LSTM architecture.

$$X_{normalized} = \frac{(X - X_{minimum})}{(X_{minimum} - X_{minimum})}$$

V. RESULTS AND ANALYSIS

A. Error/Loss functions:

We can evaluate the performance of the trained model using the three metrics: Squared Error (RMSE), R Squared(R2), and Mean Absolute Percentage Error (MAPE). Since stock market prices keep oscillating, MAPE is a preferable method to estimate the accuracy of our model. Moreover, R2 is a recommended method to estimate the efficiency of a regression-based model.

B. Activation functions:

The sigmoid activation function is used in LSTMs to obtain the results for the forget gate, input gate, and output gate. For calculating cell candidates and hidden states, the tank activation function is used.

C. Number of hidden neurons:

We experimented with various parameter combinations, as shown in the table. We came across the fact that as the number of neurons increases, model training takes a longer time even though the results do not improvise significantly. Thus, we can conclude that our model performs better once the count of hidden neurons is 15 (preferable) or 20.

D. Window size:

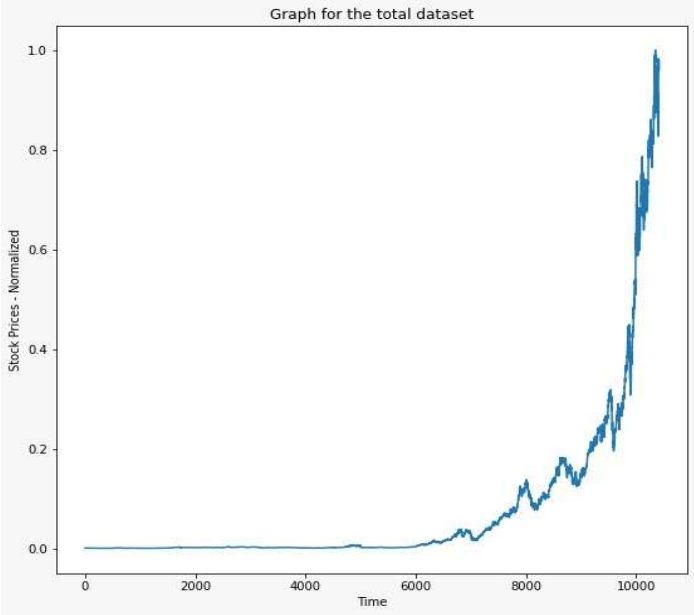
We evaluated our model with a couple of different window sizes (10, 15, and 20) and determined that the networks display better results when the size is 15.

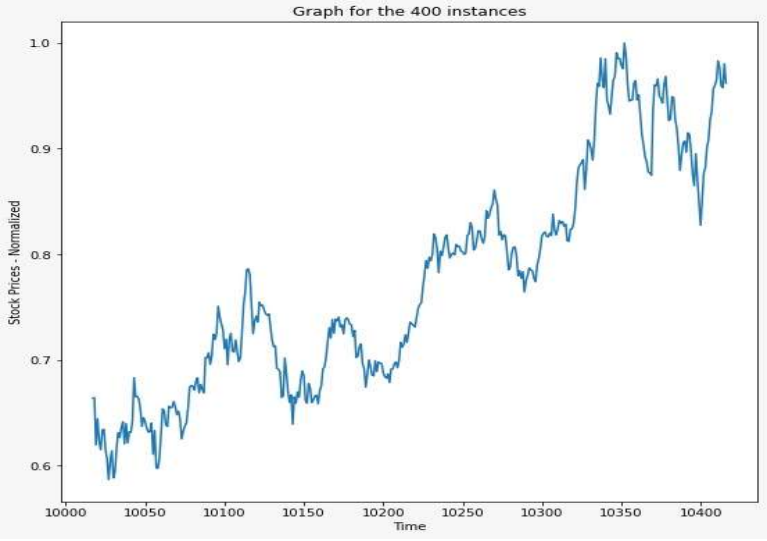
E. Number of epochs:

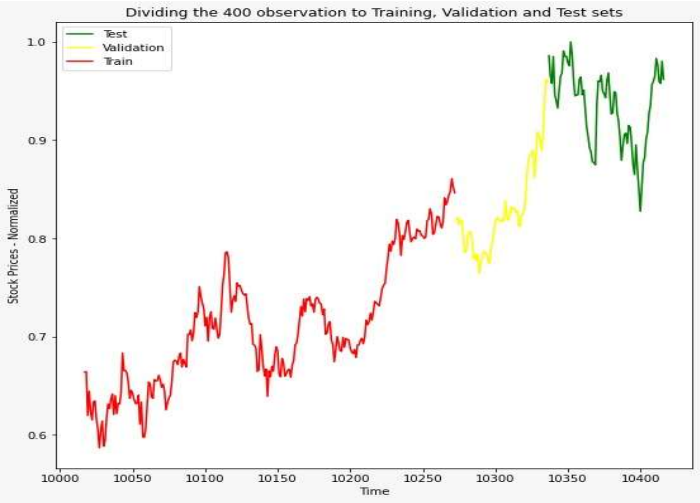
When the count of epochs grows, the time which is required to complete the fitting operation climbs up as well. It could not be too big since that would lead to overfitting of the training set and result in poor test results. We came to the conclusion that raising the number of epochs to big values does not improve the cost-time trade-off. We identified that the ideal number of epochs for our model to generate the best forecasts is around 400 epochs.

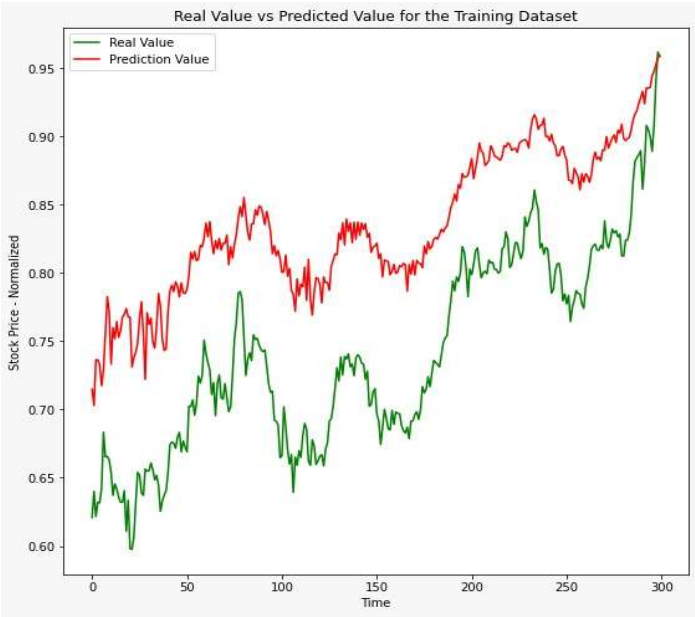
Since the stock market is highly volatile, it is a difficult task to forecast the exact pattern that the stock prices are likely to follow.

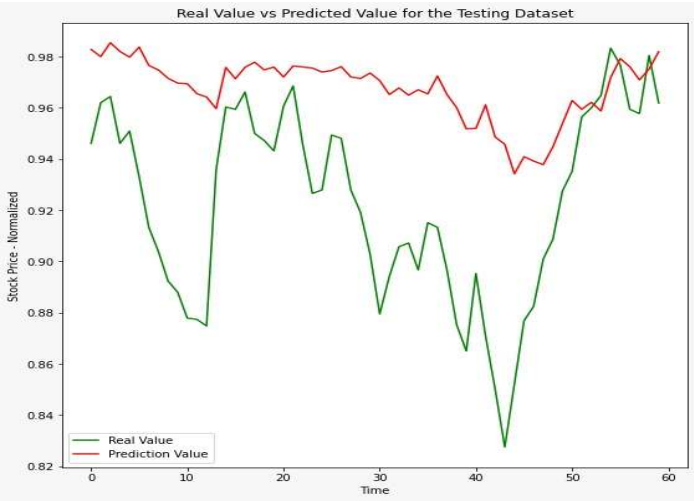
The charts below demonstrate the results of numerous trials on our model. All the charts on this page are for data from the NYSE stock price train and test sets. They display plots of real-world stock market values alongside the model's forecasted prices.











We experimented our model with different combinations of parameters and the model reflected the below results.

Index	Parameters	Results
1	Number of Input Layers: 15 Number of Output Layers: 1 Number of Hidden Neurons: 15 Number of Epochs: 600 Learning Rate: 0.1 Number of Observations: 400	RMSE for training set: 0.05896249236095784 RMSE for testing set: 0.02724568749004955 MAE for training set: 0.05588236680394179 MAE for testing set: 0.02218284472680114 R2 Score for training set: 0.36811965980627637 R2 Score for testing set: 0.4900496168074383 Time: 11 mins
2	Number of Input Layers: 15 Number of Output Layers: 1 Number of Hidden Neurons: 15 Number of Epochs: 200 Learning Rate: 0.1 Number of Observations: 400	RMSE for training set: 0.07656549716862078 RMSE for testing set: 0.024563224452605222 MAE for training set: 0.072655114919153 MAE for testing set: 0.019830390890924206 R2 Score for training set: -0.06549003204915849 R2 Score for testing set: 0.5855204249085233 Time: 2 mins
3	Number of Input Layers: 15 Number of Output Layers: 1 Number of Hidden Neurons: 15 Number of Epochs: 200 Learning Rate: 0.2 Number of Observations: 400	RMSE for training set: 0.08983140559822309 RMSE for testing set: 0.03462570742887589 MAE for training set: 0.08440930209350311 MAE for testing set: 0.028827109413039523 R2 Score for training set: -0.466694176613478 R2 Score for testing set: 0.17637467703146759 Time: 2 mins
4	Number of Input Layers: 15 Number of Output Layers: 1 Number of Hidden Neurons: 15 Number of Epochs: 200 Learning Rate: 0.05 Number of Observations: 400	RMSE for training set: 0.0486175448401107 RMSE for testing set: 0.02992526897563801 MAE for training set: 0.04593689516229911 MAE for testing set: 0.024968526658148527 R2 Score for training set: 0.5703951020230695 R2 Score for testing set: 0.38481107859667085 Time: 2 mins
5	Number of Input Layers: 10 Number of Output Layers: 1 Number of Hidden Neurons: 15 Number of Epochs: 200 Learning Rate: 0.05 Number of Observations: 400	RMSE for training set: 0.05055269329905594 RMSE for testing set: 0.02848040065890659 MAE for training set: 0.04831720870032131 MAE for testing set: 0.023136769821276645 R2 Score for training set: 0.5542181759392135 R2 Score for testing set: 0.4839186913347492 Time: 1 min
6	Number of Input Layers: 10 Number of Output Layers: 1 Number of Hidden Neurons: 15 Number of Epochs: 200 Learning Rate: 0.1 Number of Observations: 400	RMSE for training set: 0.06970745611341882 RMSE for testing set: 0.029993888691425677 MAE for training set: 0.0668372841611139 MAE for testing set: 0.024156491470473492 R2 Score for training set: 0.1523975019979208 R2 Score for testing set: 0.4276107195099449 Time: 1 min
7	Number of Input Layers: 10 Number of Output Layers: 1 Number of Hidden Neurons: 15 Number of Epochs: 200 Learning Rate: 0.2 Number of Observations: 400	RMSE for training set: 0.09481782114871092 RMSE for testing set: 0.025499290091826506 MAE for training set: 0.08909027678019307 MAE for testing set: 0.020280622338933674 R2 Score for training set: -0.5682442453544245 R2 Score for testing set: 0.5863032392725114 Time: 1 min
8	Number of Input Layers: 10 Number of Output Layers: 1 Number of Hidden Neurons: 15 Number of Epochs: 400 Learning Rate: 0.2 Number of Observations: 400	RMSE for training set: 0.06289634600463123 RMSE for testing set: 0.03424807776942217 MAE for training set: 0.05777786848781402 MAE for testing set: 0.029776024842866056 R2 Score for training set: 0.3099435951033688 R2 Score for testing set: 0.2537259453850119 Time: 3 mins
9	Number of Input Layers: 20 Number of Output Layers: 1 Number of Hidden Neurons: 20 Number of Epochs: 400 Learning Rate: 0.1 Number of Observations: 400	RMSE for training set: 0.09771884935852135 RMSE for testing set: 0.054530800540196885 MAE for training set: 0.09403826390663478 MAE for testing set: 0.045734083772192395 R2 Score for training set: -0.7708804432914778 R2 Score for testing set: -1.159634514812188 Time: 6 mins

5.1 Results Table

VI. CONCLUSION AND FUTURE WORK

As a result of this experiment, we were able to uncover a consistent pattern in the stock market, even though the stock price is clearly erratic. Since the amount of data to examine has been limited to 2 - 3 years due to the unexpected popularity of stocks and digital currency, we would be able to detect much better patterns in the next years as the amount of data increases. It is also obvious from past data that prices can unexpectedly jump or fall, thus we anticipate that by adding as many impacting elements as possible along with the spatial data, we would be capable of achieving better results in the near future.

Since there are several other models, including auto - regressive, ARIMA and moving average, have indeed been demonstrated to be effective at diagnosing time series difficulties accompanying RNNs and LSTMs. We would try to work on these models' versions as well. Furthermore, since each of such algorithms will have its own set of benefits and costs, an ensemble classifier can be designed to mitigate drawbacks simultaneously stacking the benefits and obtaining significant outcomes.

REFERENCES

[1] Jeevan, B., Naresh, E. and Kambli, P., 2018, October. Share Price Prediction using Machine Learning Technique. In 2018 3rd International Conference on Circuits, Control, Communication, and Computing (I4C) (pp. 1-4). IEEE.

[2] Payal Soni, Yogya Tewari, Deepa Krishnan. "Machine Learning Approaches in Stock Price Prediction: A Systematic Review", Journal of Physics: Conference Series, 2022.

[3] Sepp Hochreiter, Jürgen Schmidhuber. "Long Short-Term Memory", Neural Computation, 1997

[4] Y. Wang, S. Zhu and C. Li, "Research on Multi Step Time Series Prediction Based on LSTM," 2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE), 2019, pp. 1155-1159, DOI: 10.1109/EITCE47263.2019.9095044.

[5] Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras (https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/)

[6] Multivariate Time Series Forecasting with LSTMs in Keras (https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/)

[7] LSTMs Explained: A Complete, Technically Accurate, Conceptual Guide with Keras (https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2#:~:text=%E2%80%9CCell%20State%E2%80%9D%20vs%20%E2%80%9CHidden%20State%E2%80%9D&text=The%20cell%20state%20is%20meant,the%20previous%20time%2Dstep's%20data.)

[8] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[9] Predicting weather using LSTM (https://www.rs-online.com/designspark/predicting-weather-using-lstm)

[10] Brandon Rohrer – YouTube (https://www.youtube.com/watch?v=WCUNPb-5EYI&ab_channel=BrandonRohrer)