Stock Market Prediction Using Long-ShortTerm Memory

Anand Shankar
Erik Jonsson School of Engineering and Computer Science
University of Texas at Dallas
Richardson,USA
axs210036@utdallas.edu

Arush Sharma
Erik Jonsson School of Engineering and Computer Science
University of Texas at Dallas
Richardson,USA
axs200241@utdallas.edu

Bhargava Guruvula
Erik Jonsson School of Engineering and Computer Science
University of Texas at Dallas
Richardson,USA
bdg200002@utdallas.edu

Harish Gontu
Erik Jonsson School of Engineering and Computer Science
University of Texas at Dallas
Richardson,USA
hxg200007@utdallas.edu

Abstract - Strategies related to the stock market are quite complicated and rely on a massive amount of data. One of the most difficult tasks for the investors and experts has been the prediction of stock prices. Many machine learning techniques have been developed as a result of extensive research to handle complicated computational issues and increase predictive capacities without being explicitly programmed. This project aims to investigate the capabilities of Long Short Term Memory, a form of Recurrent Neural Network, in predicting future stock values.

Keywords: Market Prediction, Root Mean Square Error, Recurrent Neural Networks, LSTM.

1. INTRODUCTION

The stock market's function is to offer a place for investors to purchase and sell individual business shares, funds, or other financial instruments. The stock market facilitates these exchanges through the technique of price discovery based on fundamental and technical analysis. A portfolio is a collection of investments that an individual holds. A portfolio spread out over a range of elements is one method to diversify against risk. Asset class, firm size, sector, industry, geographical location, yield, value, and many other characteristics may be considered.
The stock market is critical to a country's development and economic progress. This project attempts to improve stock price forecasting as then individuals can improve investing decisions.

Investors grew less competent at identifying market trends based on personal experience as market size and transaction execution speed increased. As technology advanced, investors and academics devised a plethora of strategies and models to address emerging issues.

2. BACKGROUND

In this section, let us understand the basic concepts behind the building of our model and few conceptual definitions.

2.1. Recurrent Neural Networks

Recurrent neural networks allow former results to be reused as inputs since they have hidden states. RNNs have the benefit of being able to handle sequential. In addition, the model's size does not grow in proportion to the size of the input. Third, the weights are identical throughout time. They also have an important representation for storing information about previous time steps. The parameter accessible at moment t affects the result generated at time t+1. RNNs store two types of input, such as the current one and the most recent output, to construct the output for fresh data in this way.
There are various types in RNN's, namely:

a) One-to-One
b) One-to-Many
c) Many-to-One
d) Many-to-Many

2.2. LongShort-Term Memory

Based on an artificial recurrent neural networks (RNNs), deep learning architecture is LSTM. Other than normal feedforward neural networks, LSTM also features feedback connections. Along with individual data points, it can also handle streams of data. Examples of this scenario include activities like handwriting identification, speech recognition, etc.

2.3. ROOT MEAN SQUARED ERROR

Root mean square error (RMSE) is a commonly used measure of the difference between an observed value and a value predicted by a model or estimator (a sample or population value).

3. THEORETICAL AND CONCEPTUAL STUDY

RNNs are a type of neural network design that is used for context-based learning and memory. It learns from past inputs via a hidden state. When unrolled in time, it resembles a chain of perceptrons with similar weights.

$$h_t = f(h_{t-1}, x_t)$$

$$y_t = g(h_t)$$

Fig 1.

h_t – depicts a basic RNN cell.

y_t – specifies a more specific RNN cell for regression.

$$h_t = \tanh(W^T h_{t-1} + U^T x_t)$$

$$y_t = V^T h_t$$

Fig 2. RNN of Equation 2, unrolled through time

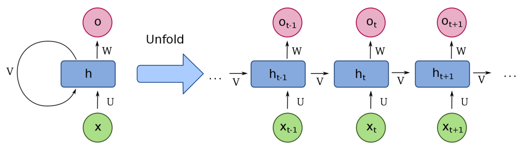


Fig 3.

RNNs are excellent at learning sequential data, but they suffer from disappearing and exploding gradients in deep time sequences. It may be tweaked somewhat by adjusting the activation function, shortened propagation, and gradient clipping. There are RNN architectural alternatives that can address these difficulties. Some of the most common are Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU).

LSTM is a more sophisticated RNN variation. It, along with others, overcomes the problem of gradient descent in RNN. It differs from a standard RNN cell in its makeup. It has both long-term and short-term memory. As a result, the term LSTM was coined.

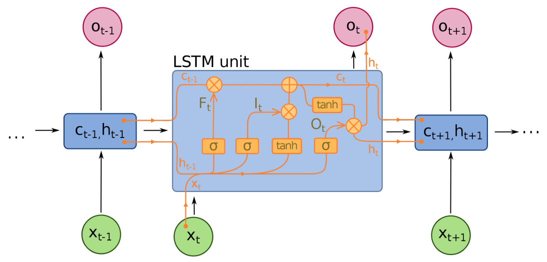


Fig 4.

It has a cell state as well as a hidden state to efficiently store long-term sequence memory. It also makes use of the concept of gates for efficient calculation. Gates are basically neural networks that are integrated within the cell.

LSTM's components are as follows:

Cell State: The network's long-term memory.

Forget Gate: Is in charge of deleting information that is no longer useful to the network.

Input Gate: This gate regulates the information that is loaded into the cell state.

Output Gate: Determines the next concealed state based on the cell state.

The above cell state formulation assures that gradients will not vanish fully over extended sequences, hence addressing the vanishing gradient problem.

3.1. FORWARD PROPAGATION

Propagating the input sequence across LSTM cells for each timestep of the sequence and lastly employing hidden state to create output for the outcome of forward propagation.

Internal state:

$$state_t = a_t \odot i_t + f_t \odot state_{t-1}$$

Output:

$$out_t = \tanh(state_t) \odot o_t$$

Input activation:

$$a_t = \tanh(W_a \cdot x_t + U_a \cdot out_{t-1} + b_a)$$

Input gate:

$$i_t = \sigma(W_i \cdot x_t + U_i \cdot out_{t-1} + b_i)$$

Forget gate:

$$f_t = \sigma(W_f \cdot x_t + U_f \cdot out_{t-1} + b_f)$$

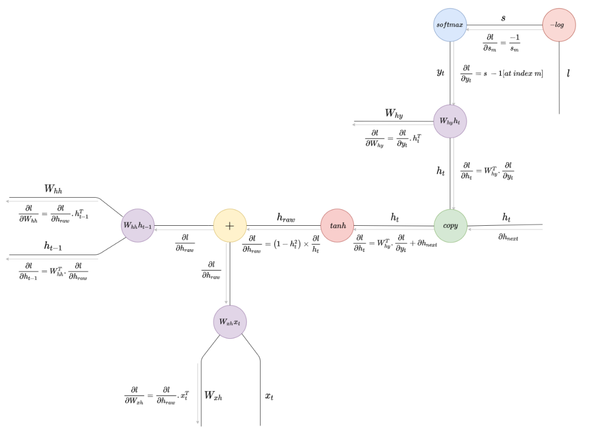
Output gate:

$$o_t = \sigma(W_o \cdot x_t + U_o \cdot out_{t-1} + b_o)$$

Fig 5.

3.2 BACKWARD PROPAGATION

Backpropagation in RNN's can be defined as essentially backpropagation across time in an unrolled RNN. As a result, this process is known as backpropagation across time.



softmax
s
-log
$\frac{\partial l}{\partial s_m} = \frac{-1}{s_m}$
y_t
$\frac{\partial l}{\partial y_t} = s - 1[at\ index\ m]$
l
W_{hy}
$W_{hy}h_t$
$\frac{\partial l}{\partial W_{hy}} = \frac{\partial l}{\partial y_t} . h_t^T$
h_t
$\frac{\partial l}{\partial h_t} = W_{hy}^T \cdot \frac{\partial l}{\partial y_t}$
W_{hh}
$\frac{\partial l}{\partial W_{hh}} = \frac{\partial l}{\partial h_{raw}} . h_{t-1}^T$
$W_{hh}h_{t-1}$
+
h_{raw}
tanh
copy
∂h_{next}
h_{t-1}
$\frac{\partial l}{\partial h_{t-1}} = W_{hh}^T \cdot \frac{\partial l}{\partial h_{raw}}$
$\frac{\partial l}{\partial h_{raw}}$
$\frac{\partial l}{\partial h_{raw}} = (1 - h_t^2) \times \frac{\partial l}{h_t}$
$\frac{\partial l}{\partial h_t} = W_{hy}^T \cdot \frac{\partial l}{\partial y_t} + \partial h_{next}$
$W_{xh}x_t$
$\frac{\partial l}{\partial W_{xh}} = \frac{\partial l}{\partial h_{raw}} . x_t^T$
W_{xh}
x_t

Fig 5.

$$\delta out_t = \Delta_t + \Delta out_t$$

$$\delta state_t = \delta out_t \odot o_t \odot (1 - \tanh^2(state_t)) + \delta state_{t+1} \odot f_{t+1}$$

$$\delta a_t = \delta state_t \odot i_t \odot (1 - a_t^2)$$

$$\delta i_t = \delta state_t \odot a_t \odot i_t \odot (1 - i_t)$$

$$\delta f_t = \delta state_t \odot state_{t-1} \odot f_t \odot (1 - f_t)$$

$$\delta o_t = \delta out_t \odot \tanh(state_t) \odot o_t \odot (1 - o_t)$$

$$\delta x_t = W^T \cdot \delta gates_t$$

$$\Delta out_{t-1} = U^T \cdot \delta gates_t$$

Fig 6.

Truncated backpropagation via time is that confines propagation to few previous timesteps from the current timestep.

4. DATASET DETAILS

This dataset provides daily price history for all NASDAQ tickers. nasdaqtrader.com has the most up-to-date list. The yfinance python module is used to retrieve historical data from Yahoo Finance. It includes rates that are valid until April 1, 2020.

Data Structure

a) *Date* - date of trading
b) *Open* - opening price of a stock
c) *High* - max price in the day
d) *Low* - min price in the day
e) *Close* - for splits, the adjusted close price
f) *Adj Close* - for both dividends and splits, the adjusted close price adjusted.
g) *Volume* - the number of shares that changed hands during a given day.

Depending on the kind, all of the ticker data is then saved in an ETFs or stocks folder. In addition, each filename corresponds to a ticker symbol. Finally, symbols valid meta.csv provides some more metadata for each ticker, such as the entire name.

4.1. PREPROCESSING

The data is subjected to the following pre-processing steps:

1. The feature is chosen from one of the dataset's columns.
2. Using a sliding window approach, the data in the specified column is divided into seven days.
3. The first two days' data are stored in a numpy array, the following two days' data in another numpy array, the next two days' data in yet another numpy array, and the last two days' data in yet another numpy array.
4. By dividing the data by 1000, the data is normalized.

5. RESULTS AND ANALYSIS

5.1. META Platforms Inc Stock Prediction Results

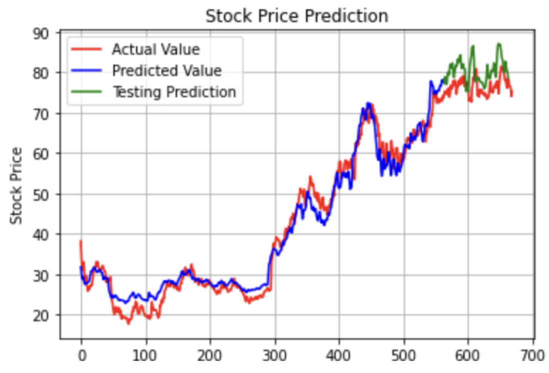


Fig . FB_AdjClose

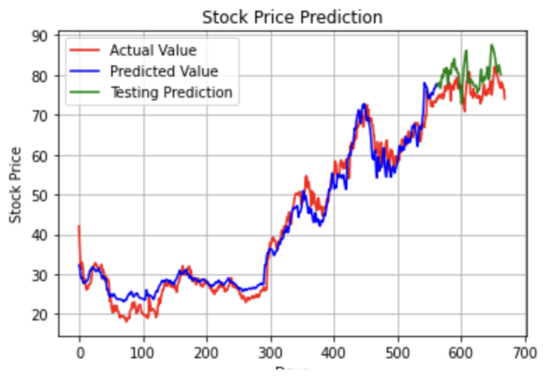


Fig . FB_Open

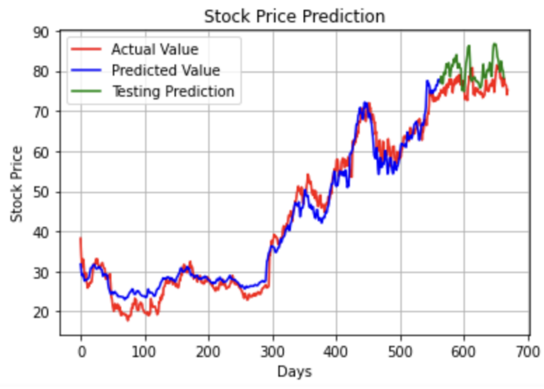


Fig . FB_Close

5.2. SAGE Therapeutics Inc. Stock Prediction Results

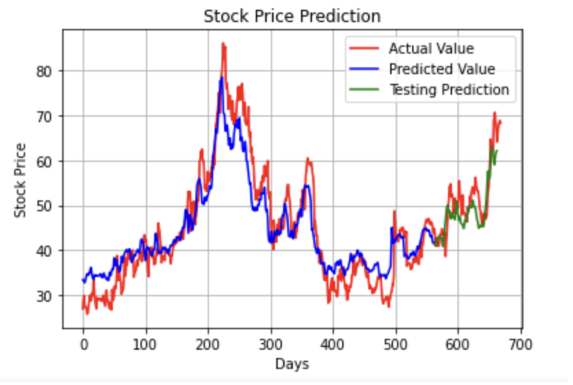


Fig . SAGE_Open

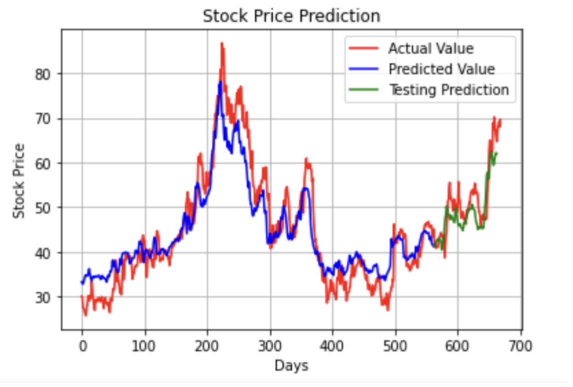


Fig . SAGE_Close

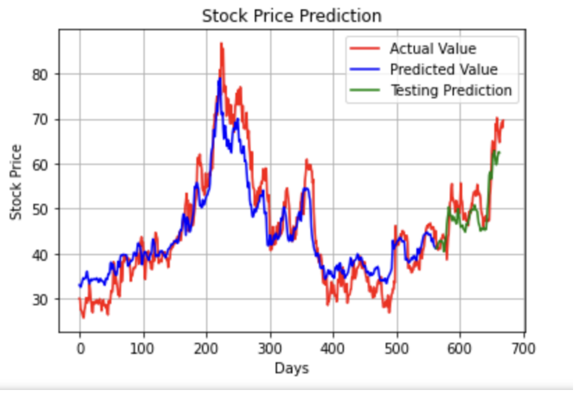


Fig . SAGE_AdjClose

5.3. Sapiens International Corporation N.V. Stock Prediction Results

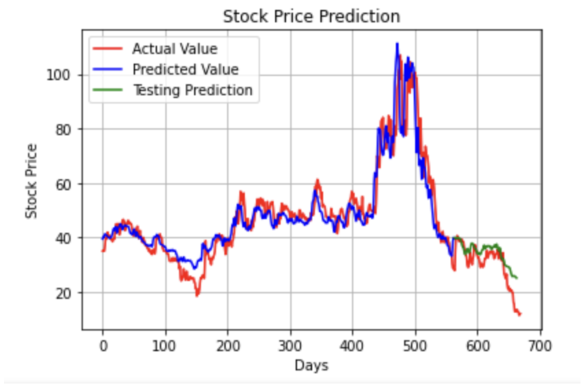


Fig . SPNS_Open

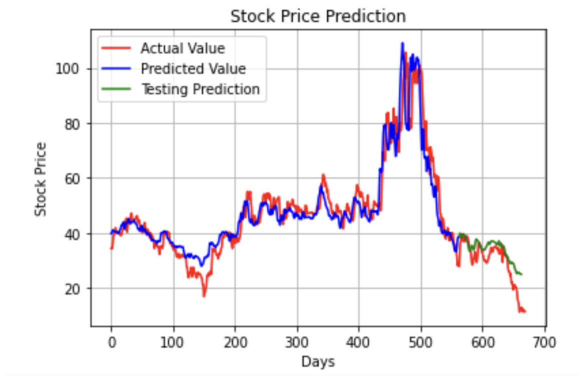


Fig . SPNS_Close

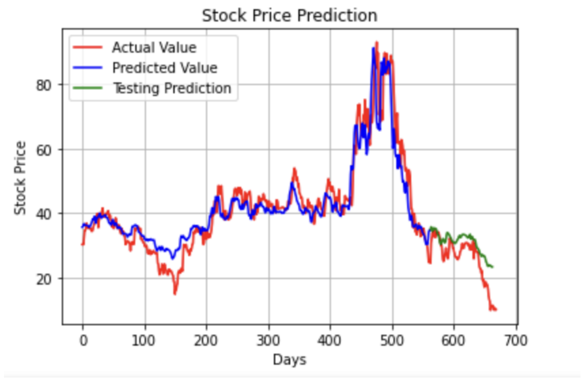


Fig . SPNS_AdjClose

6. CONCLUSION AND FUTURE WORK

Stock prices, as we know, are a time series data. Recurrent neural networks are the model used to make predictions. Using a dataset from Kaggle, we apply RNN. Stock price data necessitates a large memory content in order to predict. As a result, we used the LSTM. Applying this to the Meta Platform Inc. data, we predicted the R2 value (coefficient of determination) to be 0.9782.

Similarly, we forecasted for SAGE Therapeutics Inc. and Sapiens International Corporation N.V.

For the future enhancements, we can use auto - encoders for feature engineering, improve the current model's variations neural network architecture by using drop-out gradient clipping, truncated backpropagation, and other techniques. Furthermore, one can always train on more data.

7. REFERENCES

1. https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/stock-market/
2. https://www.investopedia.com/terms/s/stockmarket.asp
3. https://www.sciencedirect.com/topics/engineering/recurrent-neural-network
4. https://en.wikipedia.org/wiki/Long_short-term_memory
5. https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9
6. **Original Dataset:** https://www.kaggle.com/datasets/jacksoncrow/stock-market-dataset?select=symbols_valid_meta.csv
7. **Datasets Used:** https://personal.utdallas.edu/~axs210036/FB.csv https://personal.utdallas.edu/~axs210036/SAGE.csv https://personal.utdallas.edu/~axs210036/SPNS.csv