

ECE 470
Project Report
Summer 2021



**University
of Victoria**

Predict Stock Prices in New York Stock Exchange

Donovan Polard	V00849484
Atta-ul Nasir	V00879687
Calder Stuade	V00870522
Matt Macleod	V00868600

Submission Date: **July 29, 2021**

Table of Contents

List of Figures	2
Abstract	3
Introduction	4
Related Work	5
Problem Formulation	5
Methodology.....	6
PyTorch	6
Pandas	6
Yahoo Finance	7
NumPy	7
Matplotlib	7
Si Kit Learn (SKLearn)	7
Algorithm Structure	8
Results and Discussion	9
Limitations	12
Application.....	14
Conclusion.....	18
Future Work.....	18
References	19

List of Figures

Figure 1: Long short-term memory data flow [12].....	8
Figure 2: High-level algorithm flow chart	9
Figure 3: Neural network flow diagram	10
Figure 4: Neural network hyperparameters	10
Figure 5: Summary of training model.....	11
Figure 6: Predicted Stock vs. Actual Stock for General Electric	12
Figure 7: Tesla stock prediction	13
Figure 8: GUI opening page.....	14
Figure 9: Stock fundamentals window	14
Figure 10: View stock charts window	15
Figure 11: Stock chart of GE	15
Figure 12: Date options for additional stock	16
Figure 13: General Electric	16
Figure 14: Run AI prediction window	17
Figure 15: AI predicted General Electric stock performance	17

Abstract

Since the Covid-19 pandemic there has been an increasing amount of retail investors entering the stock market. Though risky, the stock market has potential to increase one's net worth due to the return on investment having the potential to be higher than inflation and bank rates. Therefore, being able to accurately predict prices for stocks trades on the New York Stock Exchange (NYSE) would be beneficial for investors who are looking to invest for a better future while reducing the risk that comes with the stock market. A Recurrent Neural Network with Long Short-Term Memory (LSTM) with a many-to-one architecture was implemented to predict the future price action of stocks. The project utilized PyTorch and NumPy to create the AI model and Panda to hold the data used to train the model. Yahoo Finance API was used to retrieve stock data, Matplotlib to create plots that represented the data and Si Kit Learn to create the machine learning model. In addition, an application was created to demonstrate the project and how the model worked. This GUI was created using Tkinter. After just 10 Epoch, the results were relatively accurate with an error of approximately 3.219%. The model continued to produced more accurate results with lower error as training continued. The trained model did an efficient job in predicting the actual stock however, there were slight differences when drastic changes occurred in the stock. Furthermore, the model had significantly poor performance when predicting stocks with high volatility and was incapable of predicting long-term stock performance. It is recommended to explore the implementation of multivariable inputs to the dataset to enable the neural network to make much more accurate predictions.

Introduction

Investing in the stock market is a great way to increase one's net worth where the returns on investments can be higher than the interest rates given by banks. In addition, stock prices reflect the condition of the economy. Being able to accurately predict prices for stock trades on the New York Stock Exchange (NYSE) would be beneficial for professional analysts and retail investors who are looking to invest for a better future. A prediction tool would give a reliable insight on the market ensuring the users are making safe financial decisions when entering, increasing/decreasing or exiting their positions at optimal times that maximizes profits and minimizes losses. It is impossible to predict the future but we can make estimated guesses and informed forecasts based on the information we have in the present and the past regarding any stock. An estimated guess from past movements and patterns in stock price is called Technical Analysis (TA) and can be used to predict a stock's price direction. Therefore, this project involves the research, design, and implementation of this desired software prediction tool. The goal of this project is an algorithm capable of predicting stock prices based on given features within an acceptable error.

The dataset given, Yahoo Finance API, provides financial news, data, and commentary including:

- Stock quotes
- Stock movers
- Press releases
- Financial reports
- Charts

This dataset is then used to train and test the stock price prediction model.

Related Work

Other work that has been performed to solve this problem includes an AI enhanced stock trading bot from Trade Ideas, a pattern recognition and back testing bot from TrendSpider, and an AI from MegaStock used for backtesting and forecasting [1] [2] [3].

Trade Ideas, created in 2003, has been developing AI algorithms to trade stock in the stock market. Currently, the platform utilizes three AI algorithms that back test to analyze every listed company on the Toronto Stock Exchange (TSX) and New York Stock Exchange (NYSE), to determine “high probability trading opportunities [3].” Utilizing these algorithms, the company has claimed to have reached “Enlightenment,” thanks to their Holy Grail (AI), capable of generating a variety of high probability trading opportunities [3].

TrendSpider, has been developing its AI algorithm capable of performing technical analysis on various stocks. The AI has been designed to identify trend lines and perform Fibonacci and Multi-timeframe analysis when determining which stocks should be traded [2].

MegaStock has developed AI algorithms to be “Expert Advisors,” allowing the bots to analyse markets and recommend financial products investment opportunities. The algorithms allow users to access 58 different systems, each performing their own analysis and resending their results to the user [4]. Through this platform users may be given “expert advice” on what stocks to trade.

Problem Formulation

Currently, stock analysts study the stock market to determine when to trade stock with mixed results. To improve stock analyst’s trading performance, it may be advantageous to utilize an AI capable of predicting trends in the stock market. Companies such as Two Sigma have already begun implementing this strategy with great success [5].

The project aims to develop an AI capable of predicting stock market trends. The AI will be trained off a generic daily prices dataset from the New York stock market and then tested to see how well it can predict future New York stock market trends.

The problem search space includes the companies listed on The Standard and Poor's 500 (S&P 500). One may wish to train the model on the 2800 companies listed on the NYSE to further improve the model's performance. The issue with training the model on the entirety of the companies listed on the NYSE is that the team does not have access to sufficient computer hardware to perform the task. Thus, the team has chosen to train the model on a single stock at a time [4]. Additionally, more financial data, such as alpha and beta values, are provided on the companies listed on the S&P 500, making it an ideal data set to work with for the project.

Methodology

The libraries that were used in the project are the following:

PyTorch

PyTorch is a free open source machine learning library developed by Facebook [6]. The project utilized the PyTorch Library to create the AI model. The PyTorch library was chosen over other libraries due to its support for Python and abundant documentation. PyTorch provides high-level features such as Tensor Computing and acceleration via a graphics processing unit. Tensor computing is where the AI model is created from data structures such as matrices and vectors. Linear algebra operations may be performed on these data structures or tensors to create a machine model. Computer graphic processor units are well equipped to perform these linear algebra operations making creating a machine learning model fast and efficient.

Pandas

The Panda's library is a free library used to hold the data used to train the model. Pandas are a library designed for Python programming for data manipulation [7]. The program utilizes pandas by first calling the Yahoo Finance API to retrieve stock data. The data is then taken from

Yahoo and stored locally in a Panda. By storing the data locally, the machine model may be built faster as the program does not have to query the Yahoo Finance server for data.

Yahoo Finance

The Yahoo Finance API was used to retrieve stock data. The API works by passing parameters to Yahoo Finance allowing the Yahoo Finance server to retrieve the data and send it back [8]. If no end date is entered the API will return the least known closing price for the given stock. To prevent overuse of the API, Yahoo Finance limits API calls to 2000 per hour. Because the program stores the retrieved locally in a panda, Yahoo's API limit does not inhibit the performance of the program. The required parameters include the stock's ticker symbol, starting date, and optionally an ending date. The retrieved data is stored locally on the host machine in a panda data structure. The model is then trained using the retrieved data.

NumPy

The NumPy library is a free Python library that supports arithmetic operations on multidimensional data structures [9]. The NumPy library is used with the Pandas library and PyTorch library to create the AI model. The NumPy library was chosen because it supports the Python programming language and the library is well documented. Additionally, it has been specially designed to take advantage of computer hardware to streamline repetitive arithmetic operations.

Matplotlib

The Matplotlib library is a free chart plotting library that supports the Python programming language [10]. It is an extension of the NumPy library and is used to produce plots representing data. The program uses Matplotlib to create stock plots. The plots are displayed in the program's graphical user interface (GUI). Images of the stock charts it produces may be seen in the results section of this report.

Si Kit Learn (SKLearn)

The Si Kit Learn library is a free machine learning library that supports the Python programming language [11]. The library features various machine learning tools such as classification,

regression, and clustering algorithms. These features are used with the PyTorch to create the machine learning model.

Algorithm Structure

The algorithm uses a Recurrent Neural Network (RNN) that is designed based on a many-to-one architecture. The data that was fed into the neural network was the closing price of the stock of the previous day. Based on the data used, the many-to-one architecture only forms a one-to-one architecture, but having the many-to-one architecture will allow for future data to be added.

The neural network follows a long short-term memory (LSTM) type. Figure 1 gives a general overview for how data is transferred through the different states.

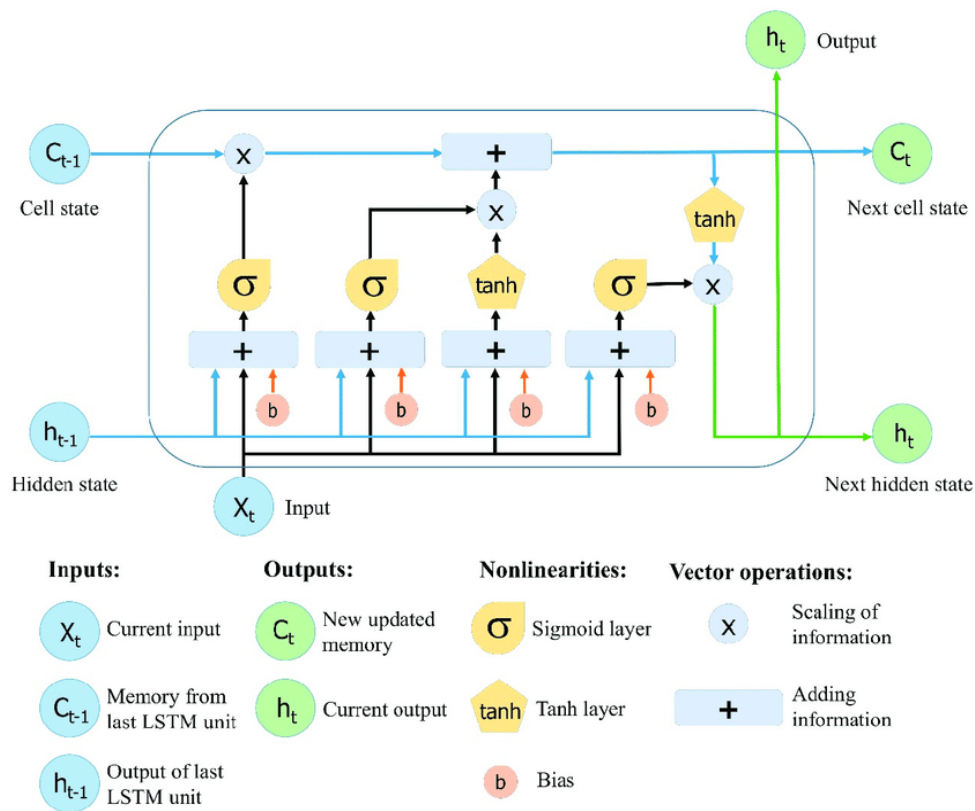


Figure 1: Long short-term memory data flow [12].

This RNN type was selected because of its feedforward and feedback connections. Single data points and entire sequences of data can be processed, making this neural network type well suited for classifying, processing, and making predictions based on time series data. In addition,

this long short-term memory also resolves common vanishing gradient problems too. A high-level algorithm flow chart can be found Figure 2.

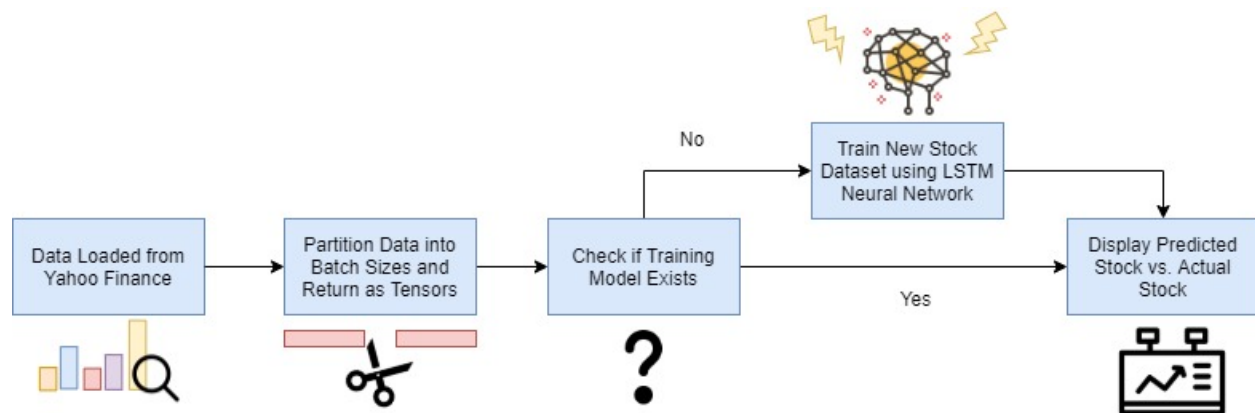


Figure 2: High-level algorithm flow chart

The algorithm will take data directly from Yahoo Finance and partition it into the proper batch sizes. The data is then returned as tensors for the LSTM neural network to use. The program will determine if a training model for that stock already exists, if so, the program will display the predicted stock price versus the actual stock price. Otherwise, the program will train a new stock dataset using the LSTM neural network and display those results against the actual stock.

Results and Discussion

The following diagram is a high-level description of the neural network architecture. Below represented in green, orange, blue and grey are the input, hidden, fully connected and predictive layers, respectively.

The following figure is a flow diagram for the RNN used in the project.

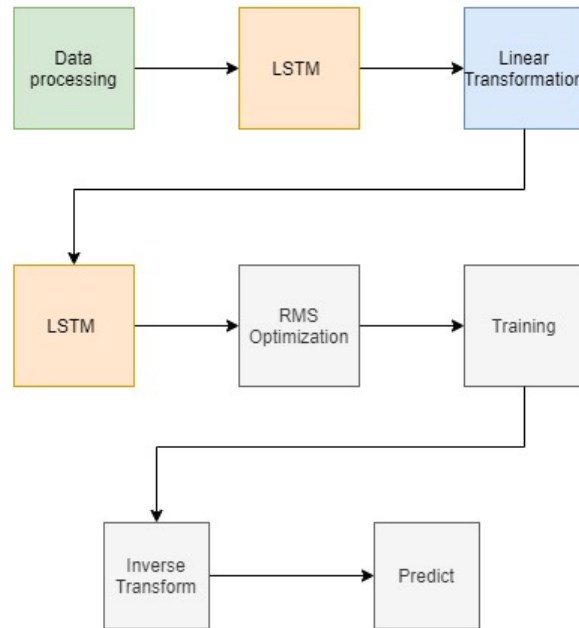
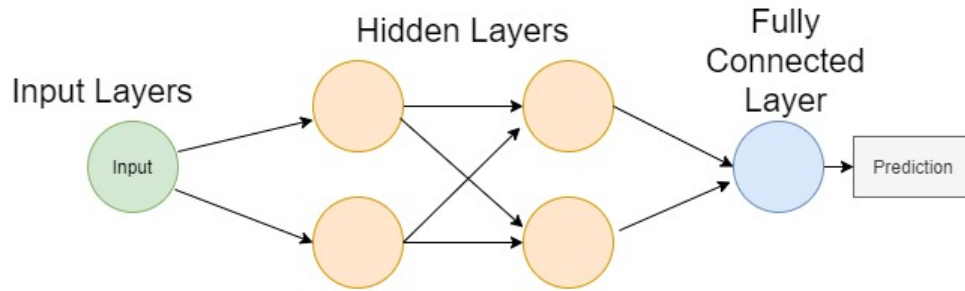


Figure 3: Neural network flow diagram

A high-level description of each of these layers can be realized in the Table 1. Below are the intrinsic hyperparameters of the neural network. These values were chosen based on empirical testing and were determined to provide the best results for our model.

```

input_dim = 1
hidden_dim = 32
num_layers = 2
output_dim = 1
learning_rate = 0.001
num_epochs = 100

```

Figure 4: Neural network hyperparameters

Table 1: Layer Descriptions

Layer	Functionality	Input	Output
Input	API responsible for processing the input data and converting it to a recognizable format for the NN	CSV Data	Tensor
Hidden	Processing the data to fit a mathematical model (LSTM) for the predictive state	Tensor	Weighted tensor
Fully Connected	Fitting the weighted LSTM tensors to a trend that can be extrapolated based on granularity parameters	Weighted Tensor	Predictive subset
Predictive Layer	Using an optimizing function to trend the predictive subset into a single result	Predictive subset	Numerical Value

After training the data on 100 epochs and using the stocks daily closing price we achieved the following results in Figure 5.

```

***** Training Model For Stock Data *****
EPOCH: [10] | ERROR: [0.032192520797252655]
EPOCH: [20] | ERROR: [0.008733400143682957]
EPOCH: [30] | ERROR: [0.005679776892066002]
EPOCH: [40] | ERROR: [0.0038692450616508722]
EPOCH: [50] | ERROR: [0.0025680679827928543]
EPOCH: [60] | ERROR: [0.002090298105031252]
EPOCH: [70] | ERROR: [0.0020080851390957832]
EPOCH: [80] | ERROR: [0.0019791298545897007]
EPOCH: [90] | ERROR: [0.0019553760066628456]
Train RMSE: 98.237

```

Figure 5: Summary of training model

The results obtained were relatively accurate with just after 10 Epoch and produced an error of approximately 3.219%. The error continued to decrease as the training continued, resulting in a root mean square error of 98.237.

Figure 6 shows a screenshot of the predicted stock price versus actual stock price for General Electric.

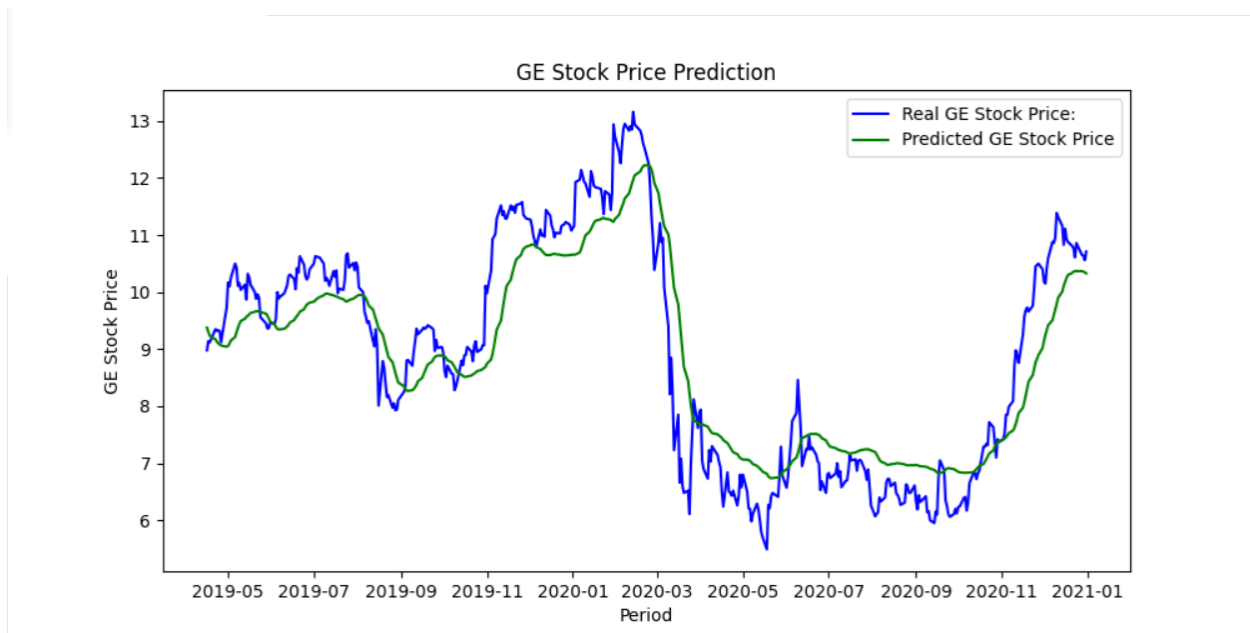


Figure 6: Predicted Stock vs. Actual Stock for General Electric

The trained model does an efficient job in predicting the actual stock and follows the general trend line. There are slight differences when drastic changes occur in the actual stock, but the predicted stock can mostly to produce similar results.

Limitations

Limitations to the project include poor performance when predicting stocks with high volatility, using a single parameter to train the model, and being incapable of making long-term stock performance predictions.

The AI model is trained off the previous performance of the stock. This design works well for predicting stocks that have high validity, often seen with blue-chip stocks or index funds. Consequently, If the stock begins to act abnormally, the model will not be able to accurately predict the stock's performance. An example of a poor prediction may be seen in Figure 7 when the model is used to predict the performance of Tesla.

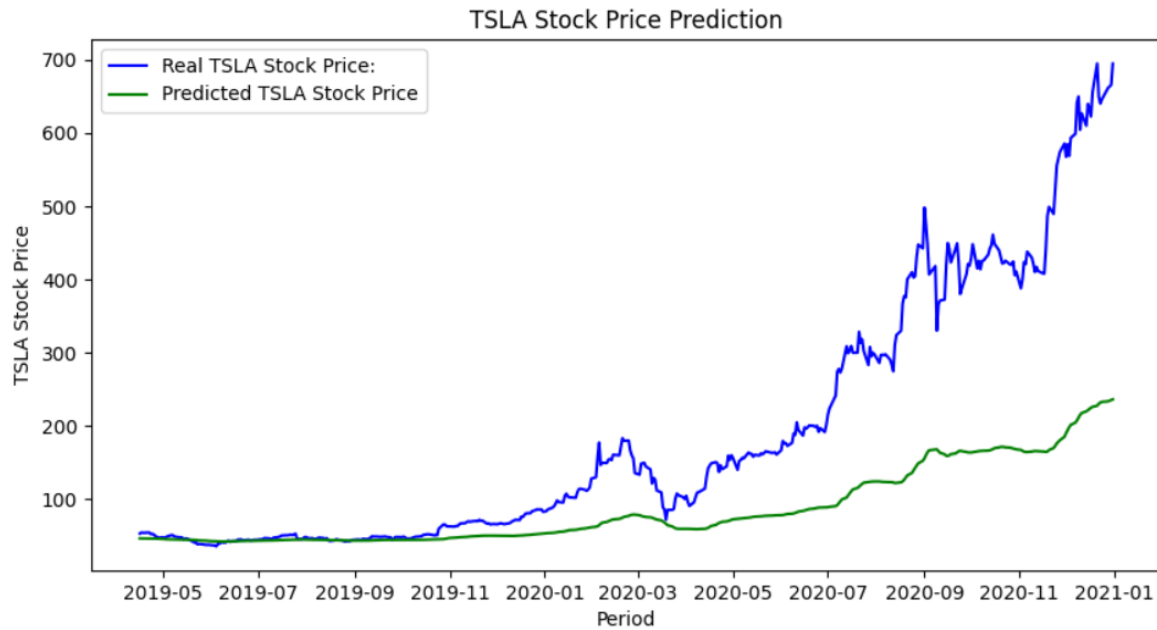


Figure 7: Tesla stock prediction

The LSTM model was chosen because it is well designed to solve time series forecasting problems. The stock market is an ideal application for this model and is a popular model choice for programmers looking to make a stock predicting program. Despite having a many-to-one architecture, the program only takes one parameter (the previous day's closing price) and uses this data to train the model. Hence, this model lacks accuracy when compared to a model trained using multiple parameters including technical analysis indicators such as volume and moving averages as well.

Furthermore, the model has been designed to make real-time stock predictions, limiting its ability and performance when making long-term predictions. To make long-term stock predictions, one may look at the trajectory of the real-time predicted stock performance. Ideally, the program should be run in real-time using current data and based on the model's prediction, one may extrapolate a few minutes into the future to predict the stock price. However, long-term future predictions are nearly impossible to implement as the stock market has an inverse relationship to inflation and bond rates -- i.e., the market reacts negatively to higher inflation and bond rates. Predicting when and what the government decides to do in regards to inflation and bond rates is impossible to predict.

Application

An application was created to demonstrate the project and how the model works. The graphical user interface (GUI) was created using the Tkinter library. The Tkinter library is a free Python library that offers a GUI toolkit to developers for creating applications.

The application features multiple windows. This allows a user to view data side by side to compare stocks and other fundamentals. When the program is started, the user is greeted with the window shown in Figure 8. To interface with the application, the user begins by entering a ticker symbol in the textbox in the top right of the window. Following the entry, the user has three options: see stock fundamentals, run AI prediction and view stock charts.

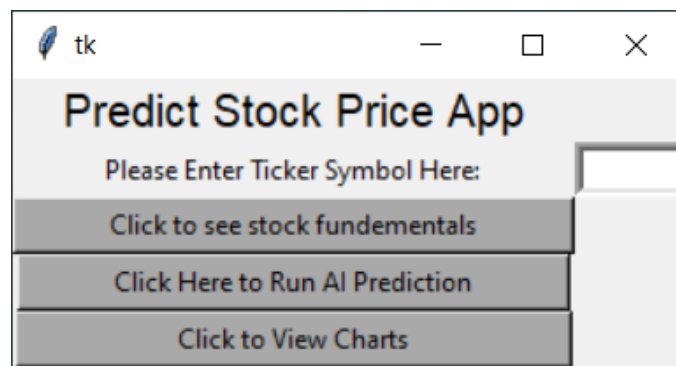


Figure 8: GUI opening page

After clicking the “Click to see stock fundamentals,” the user is presented with Figure 9, showing stock fundamentals.

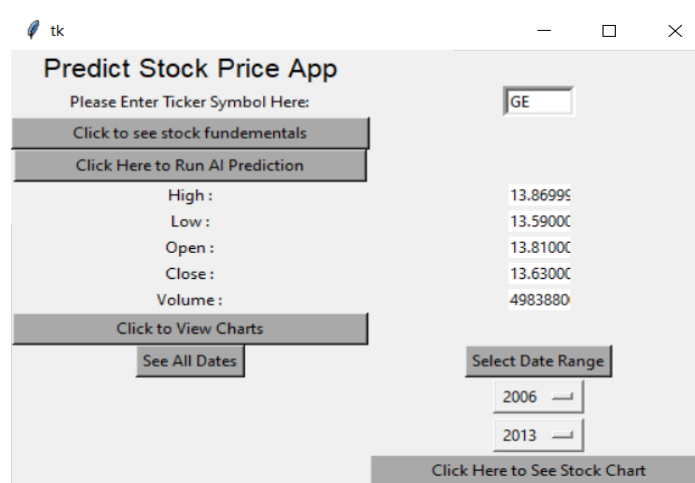
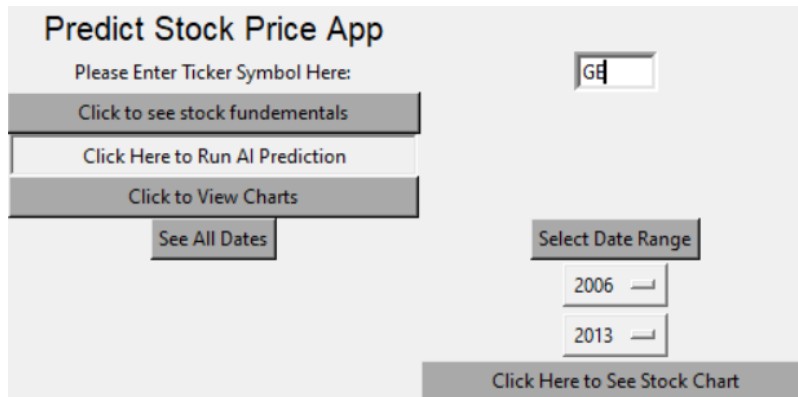


Figure 9: Stock fundamentals window

To view stock charts the user may click the “Click to View Charts” button. Next, the user may either choose to see all dates or select a specific date range. The window showing these various options can be seen in Figure 10.



Predict Stock Price App

Please Enter Ticker Symbol Here:

[Click to see stock fundamentals](#)

[Click Here to Run AI Prediction](#)

[Click to View Charts](#)

[See All Dates](#)

[Select Date Range](#)

[Click Here to See Stock Chart](#)

Figure 10: View stock charts window

After selecting from the desired date options, the user is shown with the stock chart. Figure 11 shows stock chart for General Electric (GE) with the selected date ranges of 2006 and 2013.

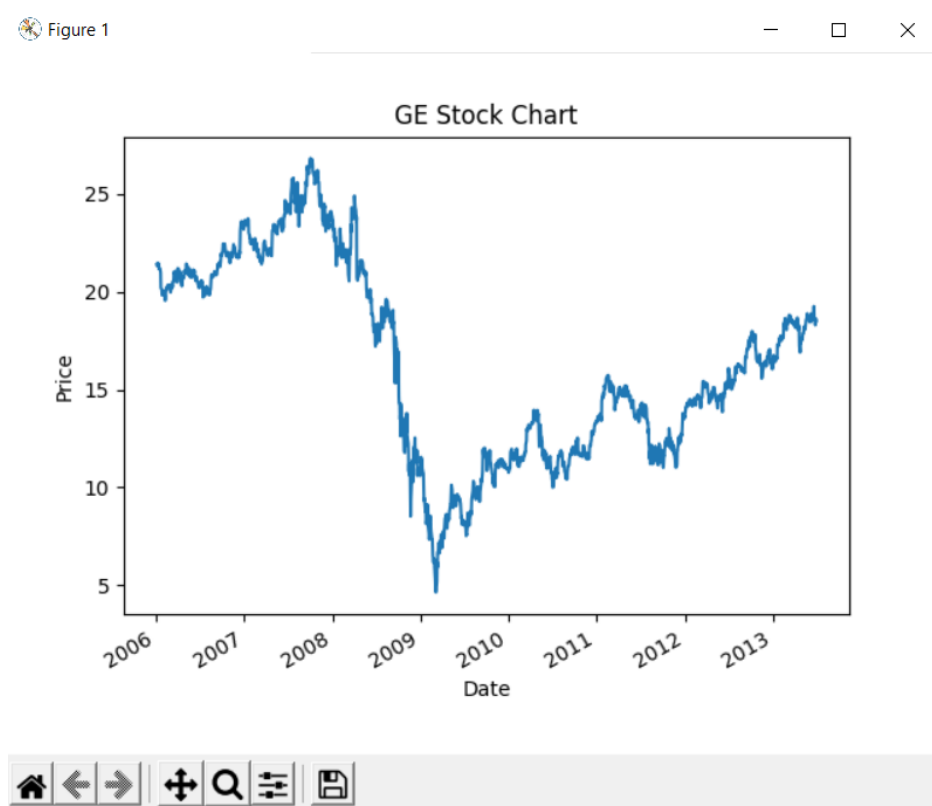


Figure 11: Stock chart of GE

If a user wishes to compare the performances between two stocks, they can enter the ticker symbol of another stock in the top right textbox and select from the desired date options.

Figure 12 shows the window displayed to the user to select another stock.

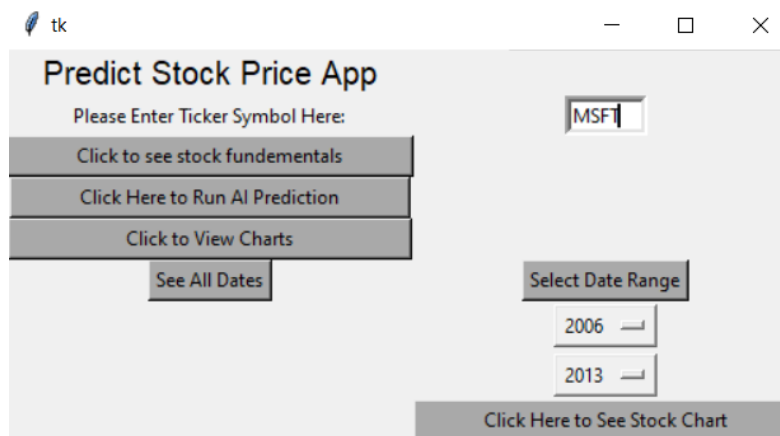


Figure 12: Date options for additional stock

Figure 13 shows the resulting chart when comparing the performance of two stocks.

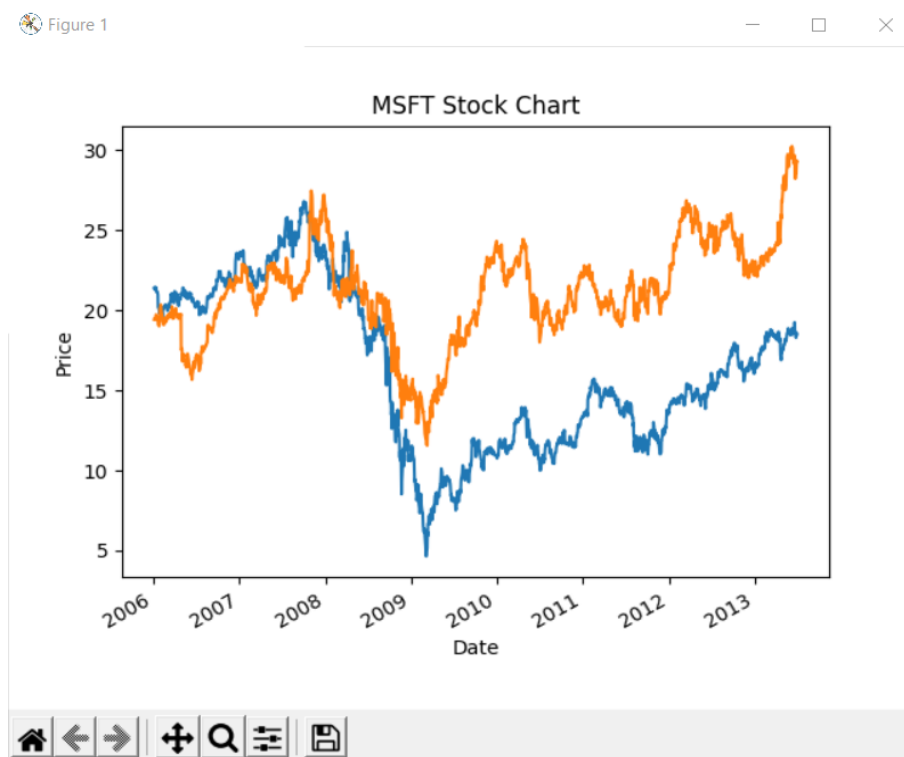


Figure 13: General Electric

To get an AI stock prediction of a stock, the user starts by entering the stock ticker symbol in the top right textbox. The user then clicks the “Click Here to Run AI Prediction” to generate the AI stock prediction.

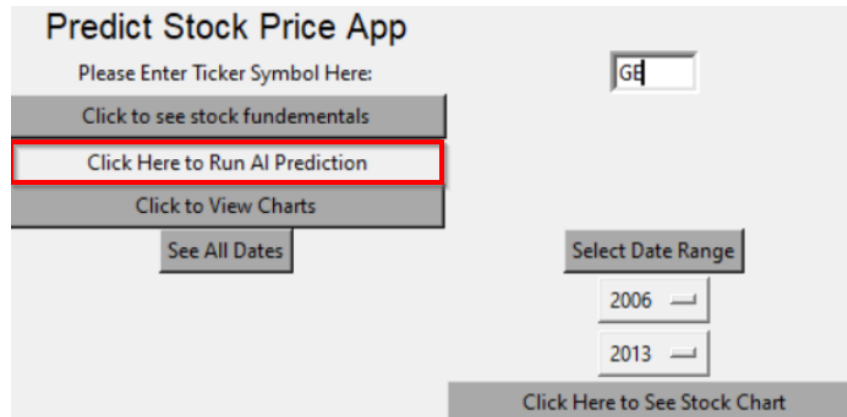


Figure 14: Run AI prediction window

Figure 15 shows the AI stock prediction chart. Here, the AI closely matches the stock trend of General Electric (GE), demonstrating the model’s accuracy.

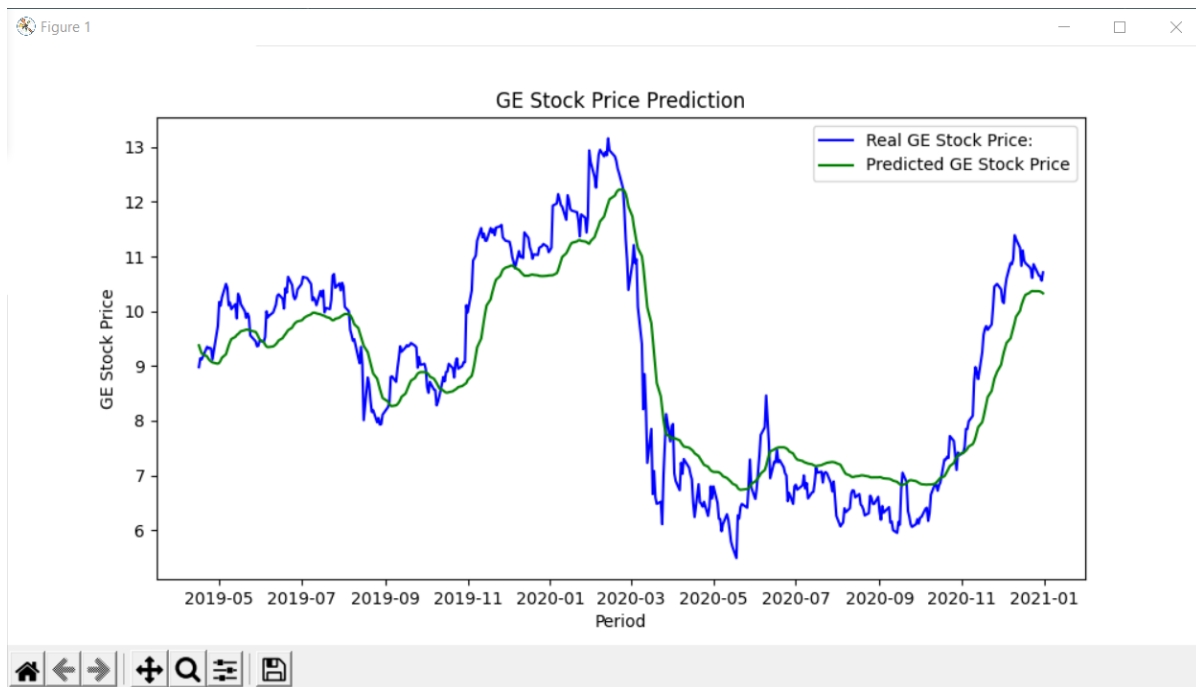


Figure 15: AI predicted General Electric stock performance

Conclusion

An AI model was successfully constructed using a Recurrent Neural Network of Long Short-Term Memory layers that is able to take input data from Yahoo Finance API. A root mean squared error of 98.237 was achieved. The model shows accurate predictions when applied to stocks with low volatility but severely suffers when applied to high volatility stocks. The quality of the model can be improved by using multiple inputs rather than just one. The predictions the model produces can be extremely useful for those wishing to gain insight into the future price movement of a stock even though predicting the future is impossible. However, this approach is significantly better than randomly guessing.

Future Work

Since the Recurrent Neural Network was constructed to only take in one sequence (past closing prices) to predict the future, it may be beneficial to investigate the results of additional data to the neural network. It would be intriguing to see if multivariable input could enhance the price forecasts from the current model. Perhaps by adding in technical analysis indicators such as volume and moving averages to the dataset, the neural network might be able to make much more accurate predictions.

References

- [1] Jeff G., "Trade Ideas," The only stock market technology that teaches you how to trade and invest. [Online]. 2021.Available: trade ideas,<https://www.trade-ideas.com/> [Accessed: May 15, 2021].
- [2] MetaStock, "It's Time to Trade," The Explorer – Focus on Today's Best Opportunities. [Online]. 2021.Available: MetaStock,<https://www.metastock.com/offer/trial/3for1/?whc=Liberated3for1&pc=eq-liberated> [Accessed: May 15, 2021].
- [3] Trend Spider, "Don't Settle for Ordinary Trading Software," "See how TrendSpider technical analysis software can help you make smarter, more efficient trading decisions". [Online]. 2021. Available: Trend Spider,https://trendspider.com/?fp_ref=barry75 [Accessed: May 15, 2021].
- [4] ADVFN, "New Alerts," New York Stock Exchange : Company Listings. [Online]. 2021.Available: free Real-time News Alerts,<https://www.advfn.com/> [Accessed: May 15, 2021].
- [5] Kaggle, "New York Stock Exchange," S&P 500 companies historical prices with fundamental data. [Online]. 2015. Available: Kaggel Datasets, <https://www.kaggle.com/dgawlik/nyse> [Accessed: May 15, 2021].
- [6] Pytorch.org. 2021. PyTorch. [Online]. Available at: <<https://pytorch.org/>> [Accessed 16 June, 2021].
- [7] Pandas.pydata.org. 2021. pandas - Python Data Analysis Library. [Online]. Available at: <https://pandas.pydata.org/> [Accessed 16 June, 2021].
- [8] Ca.finance.yahoo.com. 2021. Verizon Media. [Online]. Available at: <https://ca.finance.yahoo.com/> [Accessed 26 June, 2021].
- [9] Numpy.org. 2021. NumPy. [Online]. Available at: <https://numpy.org/> [Accessed 4 Jul, 2021].
- [10] Matplotlib.org. 2021. Matplotlib: Python plotting — Matplotlib 3.4.2 documentation. [Online]. Available at: <https://matplotlib.org/> [Accessed 6 July, 2021].
- [11] Scikit-learn.org. 2021. scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation. [Online]. Available at: <https://scikit-learn.org/stable/> [Accessed 6 July, 2021].
- [12] Hochreiter, S. and Schmidhuber, J., 2021. Long Short-Term Memory.