Daily Opening Price Predictor Report

CMPT 353 Project

Aidan Vickars (301255806)

Table of Contents

[Table of Contents 2](#_Toc57377165)

[Objective 3](#_Toc57377166)

[The Code 3](#_Toc57377167)

[The Data 3](#_Toc57377168)

[Model Fitting & Techniques 4](#_Toc57377169)

[Results 5](#_Toc57377170)

[Limitations & Next Steps 5](#_Toc57377171)

[Citations 5](#_Toc57377172)

# Objective

Within this project I endeavored to make a reasonable attempt to predict the stock price of various stocks. Upon researching this, I came across the work done by Yacoub Ahmed in “Predicting stock prices using deep learning” where he uses a Long Short Term Memory Neural Network (LSTM NN) to predict the opening price of a stock using the previous n day opening prices as predictors with a moderate degree of success. Using these predicted opening prices, a decision to buy or sell stock can be made by buying stock if the predicted next day opening stock price is greater than the current opening stock price and selling stock if the predicted next day opening stock price is less than the current opening stock price. Thus, the objective of this project is to make my own first attempt to use deep learning models by applying and subsequently improving Ahmed’s trading methodology and LSTM NN model on the stock price of five different companies. As well as create an original simulation that simulates the expected profits over two years. To be succinct, I will be using Ahmed’s original LSTM NN model, and his ideas as inspiration. However, all code will be original.

# The Code

All code referenced in this report can be found in the GitLab repository here: <https://csil-git1.cs.surrey.sfu.ca/avickars/cmpt-353-stock-market-predictor>. Additional information on how to execute the code and what packages they require can be found in the repository’s README document.

# The Data

To attain the data, I leveraged the Yahoo Finance package in python that is a direct API to the data that can be found on Yahoo’s Finance webpage. To choose the companies to apply the model on, I developed the criterion that each companies stock must be traded in in USD and over Eastern Time to eliminate the additional need to account for different currencies and time zones. Furthermore, companies must come from different industries and must have a reasonably low and stable stock price. The necessity of the latter I will explain in greater detail in Results section of this report. Beyond these criterions, the following companies were chosen arbitrarily: Ford, Pepsi, Nordstrom, Bank of America, and Forward Industries. Note, the reader will almost certainly be familiar or have heard of the first four companies but may not be familiar with the fifth. Forward Industries is a product distributor company that designs, manufactures, and sources various goods.

Twenty years of data was queried from Yahoo Finance from January 1, 2000 to November 17,2020. Note, constant dates were used to ensure the reproducibility of the results and to eliminate any potential errors when running the code. The results of this query returned data sets for each company that attained the following schema: Date, Opening Stock Price, High Stock Price, Low Stock Price, Closing Stock Price, Number of Trades, Dividends, and Stock Splits. However, per our objective only the Date and Opening Stock Price is needed and thus all other values were discarded.

# Model Fitting & Tuning

For simplicity and curiosity, the LSTM NN used was specifically tuned on the data for Forward Industries. To be succinct, the tunable parameters of the LSTM NN as will be discussed shortly were specifically tuned for the data for Forward Industries. This is because tuning a LSTM NN is computationally expensive and takes a significant amount time even with a graphics process unit capable of training a LSTN NN. Thus, tuning five models individually for each company in question is left outside the bounds of this project. However, it is also of interest to ascertain if a LSTM NN specifically tuned on the data for one company could be applied to the data for others. Therefore, for the remainder of this report the data used can be assumed to be for Forward Industries unless otherwise indicated.

Chart

Description automatically generatedNow, before the model can be applied and tuned, some initial formalities and data manipulation must be attended to. The first is to as a formality ensure it is reasonable to even use the past opening prices of a stock to predict the next day opening price. To do this we will plot the sample autocorrelation or ACF below in Figure 1.

**Figure: 1**

Figure 1 is a plot of the correlation between a given day’s opening stock price in the y axis versus some lag k in the x axis. In greater detail, if we let the opening stock price for some arbitrary day t be represented as , we are performing a hypothesis test that tests if the correlation between and is significant where is the opening stock price for some k days in the past relative to t. If the sample ACF values presented in the plot are above or below the shaded area for some lag k we have sufficient evidence to reject the null hypothesis in favor of the alternate hypothesis that there is significant correlation between and for that lag k. We can see that the correlation appears to be significant for many lags k. Thus, in the most basic sense it appears to be reasonable to use the past opening day stock prices as a predictor. To this end, it is arbitrarily chosen to use the past 40 opening day stock prices as predictors of the next day opening stock price.

Now that the model’s predictors have been decided, the data must be manipulated into the proper form to train the model. To do so, the data was first scaled to be between zero and one and then transformed into the new schema that contains the following columns: Date, Opening Price, Open-1,…,Open-40. The columns Open-1,…,Open-40 represent the past opening day stock prices that will be used as predictors. The data was subsequently split into training and validation sets where the validation set contained approximately two years or ten percent of the data. Finally, the data was split into x and y values and converted into NumPy arrays with the required shape for TensorFlow.

Before presenting the model, some terminology of the tuning parameters that were focused on must first be defined. Batch size refers to number of samples that is process by the model while training before its parameters or weights are updated. Epoch refers to the number of times we run the complete training data through the model during training. Finally, the learning rate refers to the rate at which the model learns. Now that these parameters have been defined we leveraged the model used by Yacoub Ahmed to use to make our predictions. The model uses a sequentially defined form beginning with a LSTM layer with 40 nodes as predictors, a dropout layer that will randomly drop twenty percent of the nodes, a dense layer with 64 nodes, lastly a sigmoid activation layer culminating into a layer with a single node that contains our predictions. The learning rate was 0.0005.

To begin the tuning process, the batch size was tuned first. The results are presented below in Figure 2.

|  |  |  |
| --- | --- | --- |
| **Batch Size** | **Training MSE** | **Validation MSE** |
| 10 | 0.05375 | 0.007168 |
| 32 | 0.069077 | 0.006059 |
| 100 | 0.096221 | 0.005149 |
| 500 | 0.135626 | 0.013121 |
| 1000 | 0.181391 | 0.006829 |
| 1500 | 0.24618 | 0.0105 |
| 2000 | 0.363314 | 0.035387 |
| 2500 | 0.421914 | 0.046237 |
| 3000 | 0.215748 | 0.005812 |

**Figure 2**

We can see that as the batch size increases our Training MSE clearly increases except for a batch size of 3000. This trend does appear to also be true for the Validation MSE albeit not as well defined. We conclude that we will continue to use a batch size of 10. To verify the results, lets plot the results of our predictions from our model using a batch size of 10 and 3000 in Figures 3 and 4.

**Chart, histogram

Description automatically generatedGraphical user interface, chart, histogram

Description automatically generatedFigure 3** **Figure 4**

We can clearly see the actual results of the different batch sizes on the test data. The smaller batch size fits the validation data far more closely than the larger batch size. This is not surprising as a smaller batch size typically generalizes better than a larger batch size.

The next parameter that is tuned is the number of epochs. The results are presented below in Figure 5.

|  |  |  |
| --- | --- | --- |
| **Epochs** | **Training MSE** | **Validation MSE** |
| 50 | 0.914686 | 0.362143 |
| 100 | 0.267374 | 0.044504 |
| 500 | 0.128154 | 0.052861 |
| 1000 | 0.054909 | 0.003233 |
| 2000 | 0.026806 | 0.004708 |
| 3000 | 0.139692 | 0.008902 |

**Figure 5**

It can clearly be see in Figure 5 that as the number of epochs increases, so does our accuracy to a point. For 3000 epochs, our accuracy begins to decrease. Thus, we must choose either the 1000 or 2000 epoch model. Because the Validation MSE is almost identical, we choose the 2000 epoch model because its Training MSE is slightly lower.

The final parameters we tune are the learning rate and the number of nodes in the various layers of the model. To this, we recorded the model’s training and validation MSE for a variety of combinations. Due to the number of models tested, the table is not shown here. However, if the reader is interested it can be found in the folder Data-Model Training Results-forwardCombonationTraining.csv.

# Model Diagnostics

Using the model as trained above, the results are shown below in Figure 6.

Chart, histogram

Description automatically generated**Figure 6**

Examining the results, the model appears to fit the data quite well. However, notice that the model appears to systematically overestimate the truth. This presents a significant problem because it causes our predicted next day opening stock price to almost always be larger than the current opening price. Thus, by the methodology used we will almost always buy stock. To account for this, the model’s predictions will be dropped by subtracting the average residual between the model’s predicted opening price and the truth for the previous 180 days from the predicted opening stock price for each day respectively. The results of this are shown in Figure 7.

**Chart, histogram

Description automatically generatedFigure** **7**

It can clearly be seen in Figure 7 that the model overestimates the truth far less. Of course, what matters according to our trading methodology is how often the model correctly choose when to buy or sell stock. This model makes the correct choice approximately 63% of the time. Considering that it could be argued that this model is essentially predicting noise rather than the overall trend in the stock, this result is promising.

Lastly, there does exist a flaw in the model. This can be seen by examining only a small subset of the validation data in a plot in Figure 8.

**Chart, line chart, histogram

Description automatically generatedFigure 8**

It can be seen in Figure 8 that the model makes appears to be simply reacting rather than predicting. This is apparent because the model will predict the oscillation a stock makes almost immediately after the oscillation occurs. This indicates that the model while highly sensitive, likely not picking up any underlying trend. Again, this is not entirely surprising since the argument can be made that model is simply predicting the noise in the stock price.

# Results

The model as tuned above was applied to the other Companies discussed in the Data section. The results were albeit less impressive than when applied to Forward Industries. The results are displayed in Figure 8.

|  |  |
| --- | --- |
| **Company** | **Accuracy** |
| Bank of America | %53 |
| Exxon | %53 |
| Ford | %53 |
| Nordstrom | %55 |

**Figure 8**

The results appear to be little better than guessing. This is not entirely surprising given the complexity of predicting stock prices and the model was tuned specifically for Forward Industries. Furthermore, this also potentially answers one of the original objects that was to ascertain if a single model tuned to a specific company could be applied to others. It appears this is not the case. The reader may also notice that results for Pepsi are not listed in Figure 8. This is because the results in this case were clearly useless and no further investigation was needed. A plot of the predicted values and the truth for Pepsi is shown in Figure 9.

**Chart, line chart

Description automatically generatedFigure 9**

We can clearly see that the model fails to predict with any degree of accuracy on the validation data. This is because it is attempting to predict values that is never seen before. The stock price of Pepsi continued to rise past the separation date between the training and validation data. Thus, the model was not trained on stock prices this high. This also reveals a significant limitation of the model. It is unable to predict with any degree of accuracy when the stock price of a company reaches new heights or new lows.

While the results of our model appear to be little more than guessing when applied to arbitrary companies, the most important way to assess the results of the model is to assess its performance with respect to the profits it generates. To do so, a simulation was created such that for each stock, the decision is made to buy or sell stock based on the calculation that if the next day predicted opening stock price is greater than the current opening stock price we will buy stock, otherwise we will sell. However, because more than one stock is now being considered an integer programming optimization model has been imbedded into the simulation to optimize which and how much of each stock will be bought on a given day according to the parameters we have set. These parameters are the stocks we are considering in a given simulation, the time range we are considering, the seed money that represents the initial investment, the risk that represents the proportion of the total amount of money that we have to spend that can be spent on any one stock and finally the difference in price between the current opening price and the predicted opening price that represents our expected profit. For instance, suppose on a given day we have $10 to spend, we are considering the companies Ford and Nordstrom, and we have set a risk parameter of 0.5. Furthermore, suppose the current opening share prices of Ford and Nordstrom are $4 and $2 respectively and the predicted next day opening prices are $10 and $4, respectively. This gives an expected profit of $6 and $2 per share, respectively. The risk parameter notwithstanding, the optimization model will indicate to maximize the expected profits 2 shares of Ford and 1 share of Nordstrom should be bought assuming they can be bought at their respective opening prices. However, putting 80% of the money available money on any one stock is somewhat risky. This is where the risk parameter comes into play. By setting it to be 0.5, the optimization model will indicate to maximize our expected profits we should buy 1 share of Ford and 2 shares of Nordstrom leaving a total of $2 unspent. While this will certainly limit profits, it could potentially limit losses in the event the stock price of a company takes a significant dip.

As to selling stock, as stated previously if the model predicts the next day opening stock price of a company is less than current opening price it will indicate that we should sell all shares we own of that company. Using the example from above, suppose we bought 2 shares of Ford at $4 per share and a few days later the model indicates that we should now sell. Assuming we can also sell the stocks at the current opening price of $11 per share, the shares of Ford will be sold for a $14 total profit. Note, it is entirely possible that the model will predict that stock for a company should be bought for several days in a row and that the stock should be sold for several days in a row. The simulation will simply continue to buy stock if the current available funds allow and will sell stock if there is stock to sell.

Now, lets first examine the results of the application of the model to Forward Industries created using the simulation. Beginning with an initial investment of $1000 and a risk value of 1, the cumulative profit over time for Forward Industries is shown in Figure 10.

**Chart, line chart

Description automatically generatedFigure 10**

The results are excellent! Over two years, we made a total profit of $11 418.76 this is over eleven times the initial investment. It should be noted that approximately $3000 of the total profit came from a single, highly profitable trade. However, even if this is taken out of the total profit our predictions appear to be highly profitable.

Similarly, if we apply the simulation to all the companies using an initial investment of $1000 and a risk value of 1, the cumulative profits over time are shown in Figure 11.

**Chart, line chart

Description automatically generatedFigure 11**

The results are far less profitable with a total profit of $70.87. In fact, once any combinations of stocks are introduced into the simulation other than Forward Industries, the profits decrease significantly. This is almost certainly due to decrease in accuracy when predicting to buy or sell shares of the respective stocks.

# Limitations & Next Steps

Overall, the model combined with the simulation produced intriguing results. A profit of over $11 000 certainly warrants additional investigations. However, the results must be taken with a degree of caution. This is because in the simulation two highly influential factors were not accounted for. The first being that it was always assumed stocks could be sold. That is, there was someone willing to buy the stock. This may be a negligible as the stocks for all companies used are all traded on daily with a high degree of frequency. The second, and likely far more important is time. To be succinct, it is always assumed that shares can be bought and sold at the opening price. This is not actually the case. In real life, the moment the market opens the stock prices will begin to change and thus the time it takes to make an actual trade is not accounted for. This time could be only a second or two, but depending on internet speed, potential buyers, etc. it may take longer. Unfortunately, the only true way to accurately assess this is to further develop this project into an actual stock trading bot that makes actual trades. However, this is clearly far outside the scope of this project. Although, I will certainly admit if I had unlimited amounts of time I would certainly aim to achieve this.

While the results of predicting the next day opening stock price for Forward Industries appear to produce very good results, the results of other companies were not nearly as good. To this end, given additional time I would certainly aim to fine tune the model specifically to each company. Furthermore, it would also be very interesting to ascertain if there is a minimum degree of accuracy needed to see a worth while amount of profit. For instance, is it necessary to achieve an accuracy of at least 60 percent to see lots of growth in the cumulative profits?

# Project Experience Summary

In the final project for the course Computational Data Science I applied a Long-Short-Term-Memory Neural Network that predicts the oscillation in a stock’s daily opening price. The resulting predictions were translated into a simulation with an embedded optimization model to optimize stock trades that returns the expected daily and cumulative profits over time.