Daily Opening Price Predictor Report

CMPT 353 Project

Aidan Vickars

Table of Contents

Objective

In this project I endeavored to formulate a model that predicts stock prices. While researching this I came across the work done by Yacoub Ahmed in "Predicting stock prices using deep learning". Ahmed uses a Long-Short-Term-Memory Neural Network or LSTM NN to predict the opening price of a stock from the previous n opening prices. Using these predicted opening prices, he formulated a stock trading methodology where he buys stock if the predicted next day opening price is greater than the current opening price and sells otherwise. Thus, the objective of this project can be split into three parts. The first is to use deep learning by applying and improving the LSTM NN model created by Ahmed on the stock price of five different companies. The second, will be to construct a simulation that not only employs Ahmed's trading methodology but also extends it by embedding an optimization model to optimize which stocks should be bought an any given day. The third is to construct a dashboard using Dash that visualizes the results to attain additional insights into the model's performance. 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 Data

To attain the data, I leveraged the Yahoo Finance package in python that is a direct API to Yahoo Finance. To simplify the data wrangling process, possible companies were limited to those traded in USD, in Eastern time and that had a relatively low stock price. Furthermore, companies were also limited to those with a reasonably stable stock price. The necessity of this I will explain in greater detail in the 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 may not be familiar with Forward Industries. 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 that returned several columns such as the date, Opening Stock Price, Closing Stocking price and others. However, per our objective only the Date and Opening Stock Price is necessary and thus all other attributes were discarded.

Model Fitting & Tuning

For simplicity and curiosity, the parameters of the LSTM NN used only tuned on the data for Forward Industries. This is because tuning a LSTM NN is computationally expensive and takes a significant amount time even with a GPU capable of training a deep learning model. 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 for one company could be applied to others. Therefore, for the remainder of this report the data used can be assumed to be for Forward Industries unless otherwise indicated.

Now, before the model can be applied and tuned, we will check that it is reasonable to use past opening prices to predict the next opening price. To do this we will plot the sample autocorrelation or ACF below in Figure 1

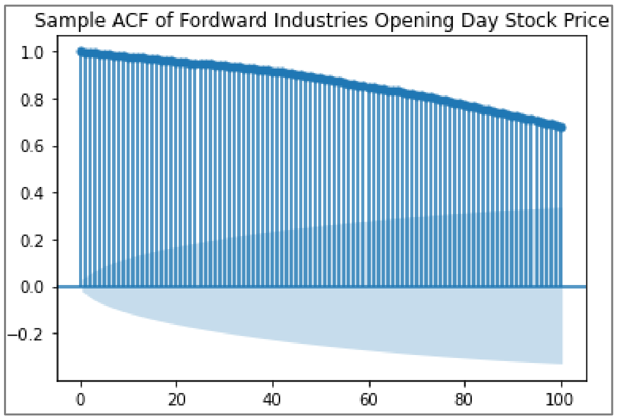

Figure: 1

Figure 1 is a plot of the ACF 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 Y_t, we are performing a hypothesis test that tests if the correlation between Y_t and Y_{t-k} is significant. Note, Y_{t-k} is the opening stock price for some lag k or 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 that there is significant correlation between Y_t and Y_{t-k} for that lag k. We can see in Figure 1 that the correlation appears to be significant for many lags k. Thus, in the most rudimentary sense it appears to be reasonable to use past opening stock prices as a predictor. To this end, it is arbitrarily chosen to use the past 40 opening prices as predictors of the next day opening 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 such that each row contains the following columns: Date, Opening Price, "Open-1",…,"Open-40". The columns "Open-1",…,"Open-40" represent the past opening 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 in the training and validation sets respectively and converted into NumPy arrays with the required shape to be used in the model.

Before presenting the model, some terminology of the tuning parameters that were focused on must be defined. The Batch Size refers to number of samples that are processed by the model while training before its parameters are updated. The 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, the model used can be presented. 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, a sigmoid activation layer culminating into a layer with a single node and finally a linear activation layer 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 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 increases except for a batch size of 3000. This trend also appears to be true for the Validation MSE albeit not as well defined. We conclude that we will use a batch size of 10. To verify, lets plot the results of the model using a batch size of 10 and 3000 in Figures 3 and 4.

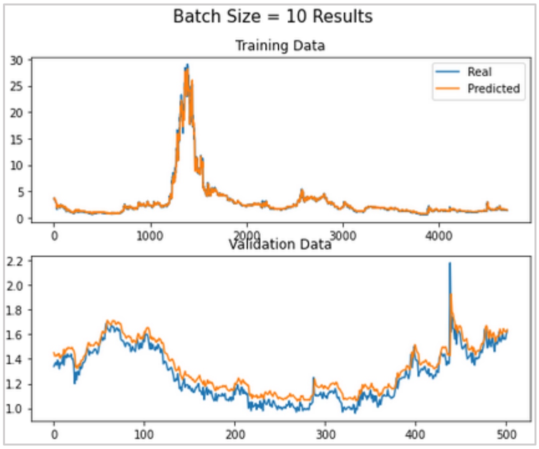

Figure 3

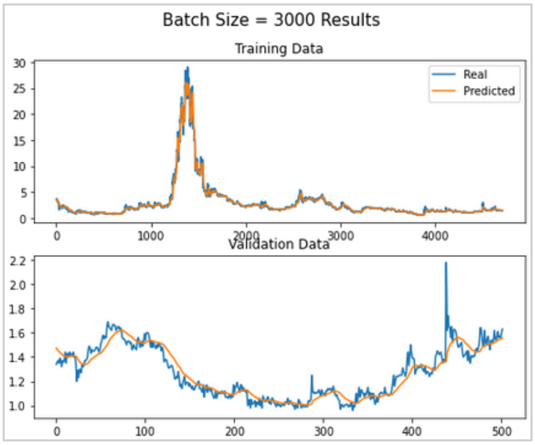

Figure 4

We can see that 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 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

Looking at the results, we can recognize that as the number of epochs increases, so does our accuracy. However, for 3000 epochs, this trend disappears. Thus, we must choose either the 1000 or 2000 epoch model. Because the Validation MSE is almost identical, the 2000 epoch model was chosen do to its slightly lower Training MSE.

The final parameters that were tuned 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 many combinations. However, due to the number of models tested the table is not shown here and can be found in the file 'Data/Model Training Results/cvModelsResult.csv'. While the results of this do give some potential improvements that could be given to the model, in practice they did not improve the accuracy of correctly predicting when to buy or sell stock. In fact, they were worse. Thus, the results of this step were not utilized. The final model is displayed in displayed in the function modelTraining in "project_functions/analysis_functions.py" using the default parameters.

Model Diagnostics

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

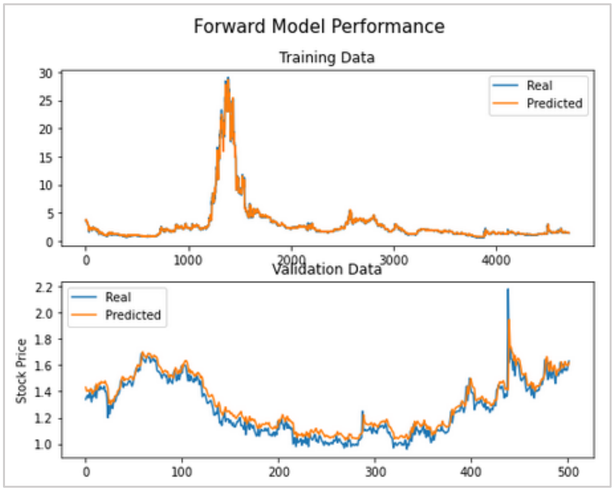

Figure 6

We can see in Figure 6 that the model fits the data well. However, it systematically overestimates the truth. This presents a significant problem because it causes our predicted opening 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. The results of this are shown in Figure 7.

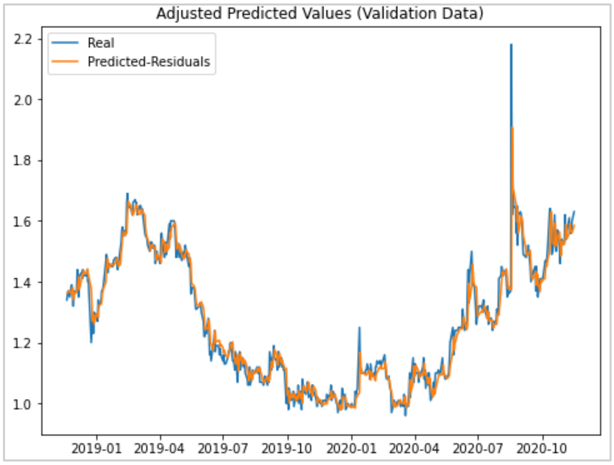

Figure 7

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 chooses 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, it must be noted that there is a flaw in the model. This can be seen by examining only a small subset of the validation data shown in Figure 8.

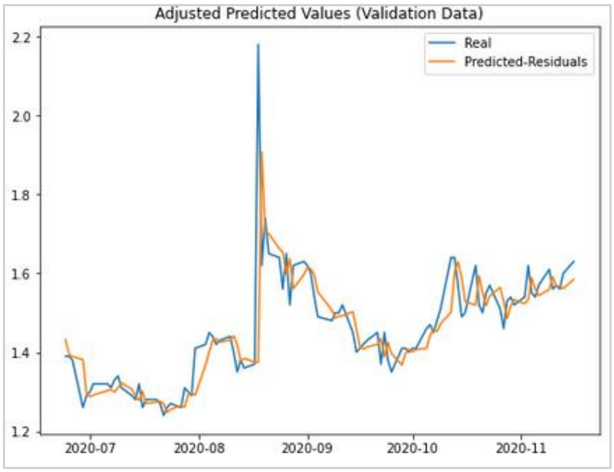

Figure 8

In Figure 8 the model 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 the model is likely not picking up any underlying trend and is simply reacting.

Results

The model as tuned above was applied to the other Companies discussed in the Data section. The results of correctly predicting when to buy or sell stock on the validation data is shown in Figure 9.

Company	Accuracy
Bank of America	%53
Exxon	%53
Ford	%53
Nordstrom	%55

Figure 9

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 answers the question: could a model tuned to one 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 9. This is because the results in this case were useless and no further investigation was needed. To prove this, the results of the model when applied to Pepsi is shown in Figure 10.

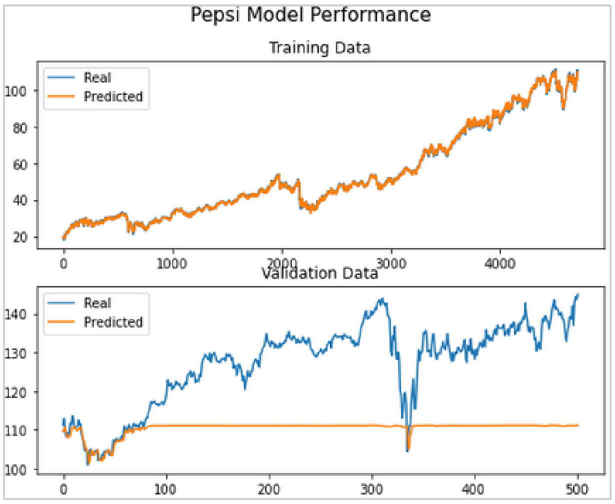

Figure 10

We can see that the model fails to predict with any accuracy on the validation data. This is because 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 with these values and is unable to predict values it has not seen before. This also reveals a limitation of the model. It is unable to accommodate the stock price of a company when it reaches new heights or new lows. This is also why the chosen companies were limited to those with stable stock prices.

While the results of our model appear to be little more than guessing when applied to arbitrary companies, the most critical measure of success is to gauge the model's ability to generate profits. To do this, a simulation was created that buys stock at the current opening price if the predicted opening price is greater than the current opening price and otherwise sells stock at the current opening price. Furthermore, because more than one stock is now being considered an optimization model has been imbedded into the simulation to on a given day select which and how many shares of each company we should buy according to the expected profits and three additional parameters. These parameters are the companies being considered, the initial investment, and the risk we are considering in the simulation. While the affects of the three initial parameters are obvious, the fourth is not. The risk parameter sets the proportion of the money we can spend on a single stock at a time. This reduces risk by forcing the optimization model to diversify the stocks it buys at the expense of maximizing profits. Note, due to the length limitations of this report the optimization model is not described here however it is briefly stated in the comments in the "optimizer" function in "project_functions/dashboard_functions.py".

Now, let us 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 11.

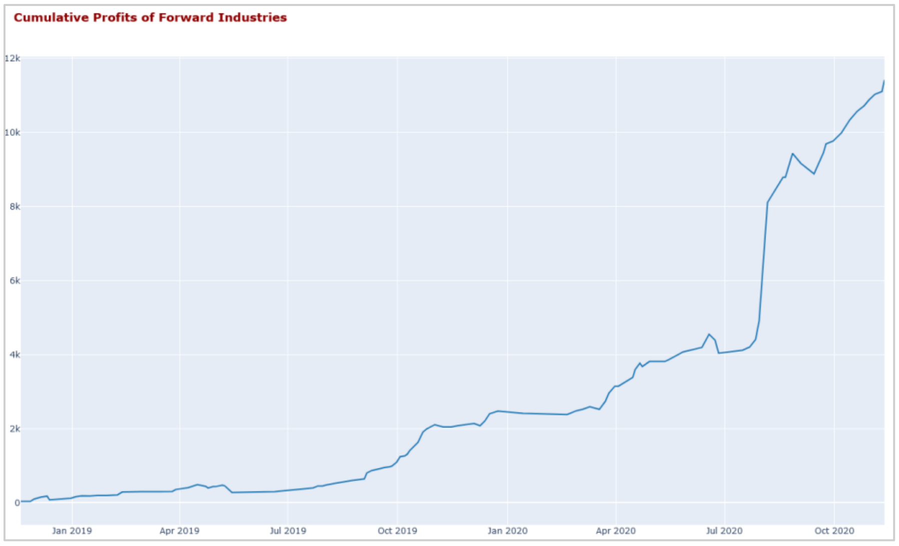

Figure 11

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. However, if we apply the simulation to all the companies the results are not as appealing. Using an initial investment of $1000 and a risk value of 1, the cumulative profits over time are shown in Figure 12.

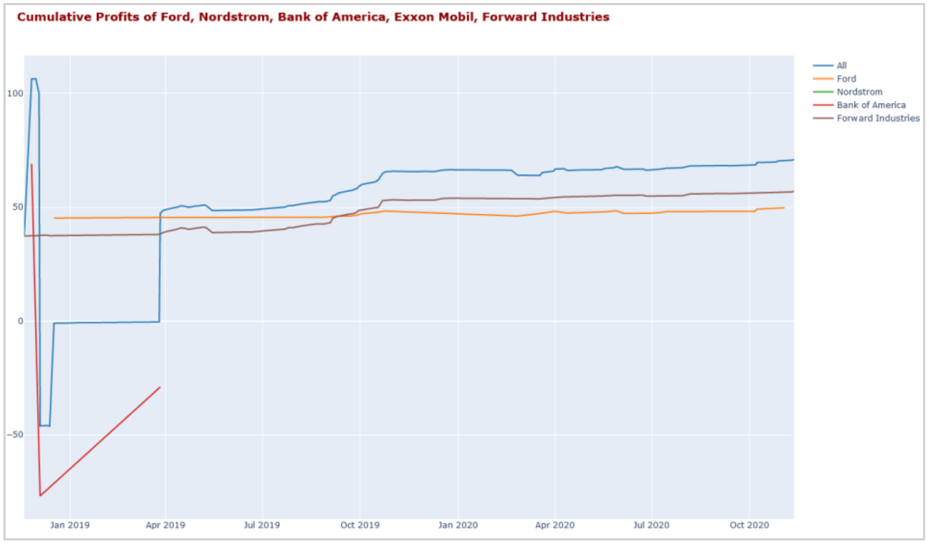

Figure 12

Clearly, the results are far less profitable with a total profit of $70.87. In fact, once any combination of companies are introduced into the simulation other than Forward Industries, the profits decrease significantly. This is almost certainly due to the decrease in model accuracy for these companies.

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 caution because two highly influential factors were not accounted for in the simulation. The first is that it was always assumed stocks could be sold. That is, there was someone willing to buy the stock. However, this may be 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 stocks 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 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, computation time, etc. it may take longer. Unfortunately, the only way to accurately assess the effects of this is to further develop this project into an live stock trading bot that makes actual trades. However, this is far outside the scope of this project. Although, I certainly admit if I had unlimited amounts of time, I would aim to achieve this.

While the results of predicting the opening stock price for Forward Industries appear to produce very good results, the results of the model applied to other companies were far less spectacular. To this end, given additional time I would fine tune the model to each company. Furthermore, it would also be interesting to ascertain if there is a minimum degree of accuracy needed to see a worthwhile amount of profit. For instance, is it necessary to achieve an accuracy of at least 60 percent to see considerable growth in 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 daily opening stock price of various companies. 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. The results of this simulation produced results with the potential to be highly profitable.