William Zhang

Deliverable 3 Report

I’ve changed a lot of things from the last deliverable. From the dataset to the way I process my data all the way to the model I’ve decided to go with. The only consistency from the last deliverable is the goal of the project has remained the same, which is to predict stock prices based on past prices. I am still working with the SNP500 data set, however I’m still tinkering around and will probably add a couple of world indexes in order to represent the world economy, as the united states is strongly influenced by it.

Data Set:

For my data, I’ve decided to go with yahoo finance. I’m using pandas DataReader and collecting the yahoo prices. Currently, I’ve decided to start the date at 1980 however that does not mean I will be training will data starting from 1980. This is due to the various stages the economy goes in, and training on all the prices from 1980 might not be representative of the current economy.

Data Processing & Model:

I’ve decided to go with a multi-layer LSTM. This is because stock-prices are inherently time-dependent, and I believe an LSTM would be able to grasp the time dependence and use that to predict the future price. For right now, I would only be predicting the stocks for the very next day. Although I might experiment and try to predict a week in the future or maybe even a month, but the very nature of predicting more time in the future means that the uncertainties are compounded.

For the specifics of my model, I’m currently having a four-layer LSTM, with 150 nodes at each layer1. The input layer would have an array of size ~9000,60,505. The first dimension is simply the amount of days I currently have in my data-set, which is 9000 days, but once again, I’m not certain if I should be training will all the data. The second dimension is the amount of days in the past I’m using to predict the next day, in this case I would be using 60 days. The last dimension is simply all the different features I’m using, so there are about 505 stock prices in the SNP500, hence why there are 505 layers.

Since I’m currently running my model with the mean squared error loss function, the model is simply trying to predict the next day’s price with the smallest difference in value from the real price. This is good as the distance from the current price indicates some form of confidence in the model’s prediction, however this doesn’t really capture fast non-linear changes in the prices. This can lead to terrible results even though the predicts are mostly accurate because the current stock market can change incredibly quickly, with situations such as flash crashes having already happened in the past. To minimize this, if I were to attempt to trade with this model, I should probably seek to apply some form of stop-loss in case the market changes drastically.

Currently, when I graph my predictions with the actual prices, the results seem very promising and align closely with the trends of the actual stock. However, this can be very misleading since the model could simply be following the same trend as the day before, but simply acting with a delay of 1 day. Therefore, in order to fully test this model, I am currently attempting to calculate the return that the model would produce if it was trading with its predictions. This would give a more reasonable metric on how the model would perform when applied.

Graph of Predictions:

Final Project Web App:

For the web application, my goal is to create allow the user to trade against the model for a week. This would be a fun way to showcase how the model predicts stocks and what information it has. I currently have an idea to make it very interactive and showcase graphs on various stocks, however I don’t know if I’ll have the time to do it all.

My list of priorities is as follows:

1. showcase graphs of past stock prices and the prediction for the next day,
2. allow the user to interact with the graphs
3. Allow the user to trade against the model with a simple buy and sell model
4. Allow the user to short and long stocks and calculate various parameters such as Sharpe and Sortino ratio.

Conclusions:

The model is indeed accomplishing my objective to a reasonable degree, it is predicting the stock price of the next day given the previous 60 days of stock prices for that stock, and for all stocks in the SNP500. I’m also attempting to back-test the model, however this is a bit difficult due to its time-series nature. I’m thinking of performing some form of walk-forward validation, where I train the model on various time periods of the past and test all their predictions on more recent data. After that, I could either use the best performing model to predict stock prices for a week and then retest which model performed best for the last week. This would lead to a lot of testing and computation on a regular basis, and I’m not even sure if this would help the model, but it’s something I would like to see if I have the time.

Also, initially I wanted to add some NLP into the model, although I won’t have the time to include it for this project, I’ll most likely seek to include twitter sentiment analysis as well as news analysis using something like rapidAPI.

References:

1. Yue Deng Automation Department; Youyong Kong ; Zhiquan Ren ; Qionghai Dai, Deep Direct Reinforcement Learning for Financial Signal Representation and Trading, IEEE 2016, https://ieeexplore.ieee.org/abstract/document/7407387

In this reference, it is recommended to use something between 128 and 256 nodes. I’ve decided to with 150 to make the computation load a bit lighter.