Roy Miao

Predicting the S&P 500

Introduction

In this report, I will show the methods used to predict the S&P 500 index. The coding language that was used was Python3, along with Numpy and Pandas to manipulate csv files. TensorFlow was also utilized to create the neural network model to carry out the predictions. All the data was retrieved from yahoo finance. Before starting this project, I had a little knowledge about neural networks, loss functions, and the stock market.

Methods

A long short-term memory (LSTM) neural network was used to predict how much percent change the close price of the S&P 500 index changes on a day to day basis. A LSTM neural network is a type of recurrent neural network (RNN). RNNs are used when maintaining information in memory over time. It can be difficult to train standard RNNs that have long term dependencies like the stock market because predicting the stock market requires looking at trends over long periods of time. Standard RNNs would not work well. However, as LSTMs have a memory cell within them that maintains information for long periods of time, LSTMs can remember the trends of the stock market.

In this project, the data used is from 1/5/88 to 10/17/20 of the S&P 500 and the Dax, a German index. The Dax closes right before the US markets open and it includes companies like Adidas, Volkswagen, etc. The Dax should provide useful information about how the S&P 500 performs that day because there are similar companies in the S&P 500 like Nike and Ford. Other countries' indexes are good predictors for other indexes throughout the world. A recent example of this is in mid February 2020 through March 2020. This time period was when Covid first started becoming a pandemic. Both the Dax and S&P 500 indices, as well as many other indices like the UK FTSE 100 index, crashed during this time period. Additionally, the 5 day average closing price for the S&P 500 and Dax was calculated for each day. Data from 1/5/88 to 10/5/12 was used for training which is 75% of the data, and the testing data is from 10/8/12 to 10/14/20 which is 25% of the data. The testing data tests a little over 8 years which is long enough to see if the model performs well over a long period of time because if the testing data tested only 1 year. The model may perform really well that year, but if the testing data had more than 1 year. We may see that the model only performs well that one year and terribly all the other years. Having more testing data gives better results to the models overall performance.

I started this project with the intention of predicting the closing price of the S&P 500. However, the prices of the S&P 500 kept increasing: the training data did not have any reference to current day dollars and the model doesn't take inflation into account. Because of this, the model was always predicting closing prices lower than the actual closing price. The model was reworked to predict how much the closing price changed everyday in percentage because there is usually less than 5% change in the price each day, and inflation wouldn't affect the percent change. Also, the model now would be able to train on most of the percentages it should encounter in the test data.

Three models were developed using the adam optimizer and mean squared error to calculate loss. With no definitive 'best' starting point for the number of neurons and hidden layers, I decided multiples of 25 would be a good start for the number of neurons with 3 hidden layers because. I've used multiple of 25s in past neural network projects, with success in past projects. The input layers had 60 neurons, the first and second hidden layers had 50 neurons, the third hidden layer had 25 densely connected neurons, and the output layer just had one neuron.

The second model also had 3 hidden layers, but this model had multiples of 32 instead of 25 and added a dropout because traditionally in computer science you pick powers of 2. The input layer had 60

neurons, the first hidden layer had 128 neurons, the second hidden layer had 64 neurons plus a 10% dropout. The third hidden layer still had 25 densely connected neurons, and the output layer just had one neuron. Using multiples of 32 had better results, multiples of 32 at that point but with another layer with dropout to ensure that the model wasn't overfitting.

My third and most successful model had an input layer of 60, the first hidden layer had 1024 neurons, the second hidden layer had 512 neurons with a 30% dropout rate, the third hidden layer had 256 densely connected neurons with a 20% dropout rate, the forth hidden layer had 128 densely connected neurons, and an output layer with 1 neuron. Comparisons were made of the three models with an RMSE score and calculations of the predicted movement direction (i.e. the change in close price percentage from the previous day increased or decreased). The RMSE loss function was used because RMSE gives a higher weight to large errors. High errors are very undesirable when predicting the stock market because if your model makes a large error, you could lose most of your money.

Results

Various batch sizes and numbers of epochs with the three models were tested (Figure 1). It was found that the model with the lowest RMSE score of 1.067 and with a 74.257% directional accuracy was when the third model used a batch size of 64 and had 5 epochs.

Model	Batch Size	Number of Epochs	RMSE	Directional Accuracy
1	32	10	1.113	31.683%
2	32	10	1.107	32.574%
3	32	10	1.083	70.990%
3	64	5	1.067	74.257%
3	256	50	1.076	63.911%

Figure 1: Table comparing different models

In addition, an investing simulation with SPDR S&P 500 ETF Trust (SPY), exchange-traded funds (ETF) that tracks S&P 500, was run in order to track the models performance in the real world. If the model predicted that the change in price would increase the next day, I purchased as many ETFs as I could that day at the opening price. But if it predicted the change in price would decrease the next day, I sold all the SPY I owned. I ran this simulation from 5/7/2012 to 10/14/2020. Following the prediction of my model, I started with $5,000 and at the end of the simulation I ended with $11,246.90. I compared it to a baseline of buying SPY every year and then selling it at the end of the year (figure 2).

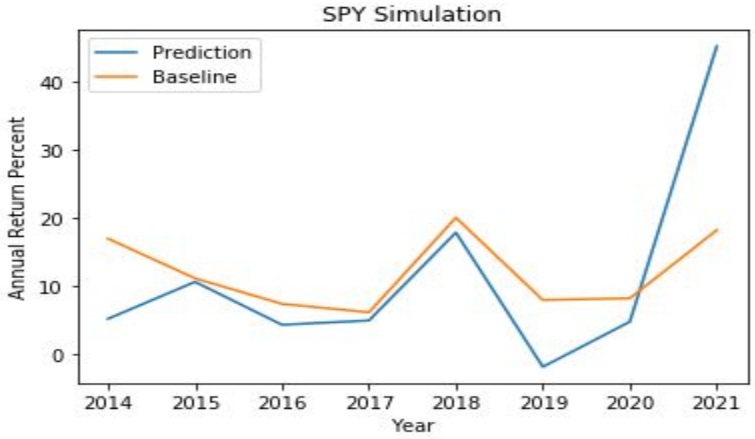

Figure 2: Plot shows annual return rate of my model and baseline

From figure 2, my model made less money per year except for 2020 - 2021. That year my model made about 20% more than the baseline. This is because of the Covid crash that happened in February and March of 2020. My model was able to predict when the S&P 500 would fall, therefore it didn't buy any SPY. However, the S&P 500 also rallied some of those days, and my model was able to accurately predict when the S&P 500 would rise during that crucial time period. Overall the model did not perform well because it returned a smaller annual return rate most of the years than the baseline.

Limitations

The model does not predict the actual percent change accurately. Most of the models predictions were still a concerning amount lower than the actual percent change (figure 3).

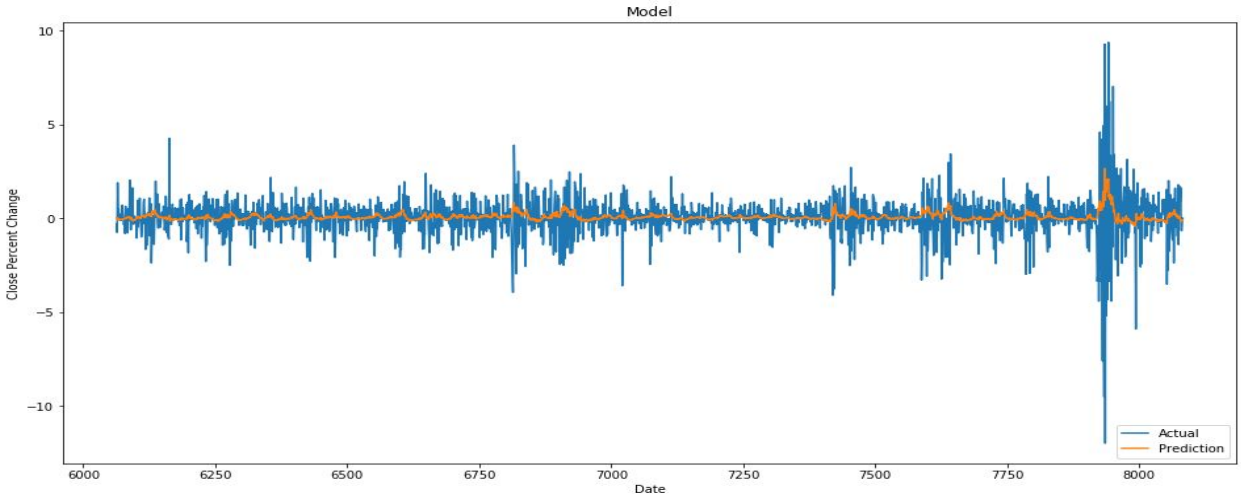

Figure 3: Graph of the actual (blue) vs the predicted (orange) change in percentage

Conclusion

A LSTM neural network was built to predict the day to day percent change of the closing price of the S&P 500. My most successful model had 3 hidden layers, using adam optimization and mean squared error loss, a batch size of 64, and had 5 epochs. The model had a 74.257% accuracy of predicting whether the closing price would increase or decrease. But the model failed to get a close prediction of the actual percent change in closing price. However, just following the accuracy of predicting if the closing price would increase or decrease. My model was able to get an annualized return of 36.5542% compared to the 12.3919% of just holding the stock and selling at the end.

Appendix

CS 373 and CS 490LDA were very helpful because these courses are where machine learning is focused on. From CS 490LDA, I was able to learn different types of neural networks and how they worked. From CS 373, I learned about different loss functions and how an individual neuron worked. By completing this project, I was further able to explore these topics like LSTM neural networks, the basics of TensorFlow, adam optimizer, feature selection, loss functions, and basic financial knowledge. I am hoping to go into the financial industry post graduation, and with this capstone I was able to get a glimpse of what my future may look like, and some of the tools that financial technology companies may use.