CS 4262: Foundations of Machine Learning
Spring 2023 (Undergraduate)

Machine Learning Approaches for Predicting Meta Stock Movement: A Comparison of SVR, LSTM, and SVC Techniques

Zachary Gruber * **Umphaipun Atipunumphai** * [1]

Abstract

The scope of our project is to understand the limitations and benefits of using machine learning for predicting stock prices. We examine two methods, SVR and LSTM, to predict the next sequence in the time series. We also added SVC to predict whether the stock price would increase or decrease daily. Some of the questions we are trying to answer are: how can we predict Meta stock price movements in the short term using machine learning techniques? What variables can we use for the multivariate prediction? How can we select a model using a hyper parameter search for LSTM? How does sequence prediction performance perform? We obtain daily stock price data through the "AlphaVantage" API under a student access code, Kaggle, and Statista.

We limit our prediction framework to a tech stock like Meta, since it has high volatility. This leads to greater divergence for predictions, and thus has better potential to distinguish the differences between SVR and LSTM. We are also motivated by the fact that tech stocks have gotten significant news attention recently as retail traders have become more interested in the markets. Thus, we explore avenues through which novices can implement machine learning strategies to hone programming skills and market insights.

1. Data Wrangling and Data Visualisation

1.1. Visualizing Meta Stock Prices Over time

For visualization we graph META daily prices using a library called `plotly.graph_objects`. The output from the API is daily price data with open high low and close prices. These can be formed into "candlesticks" on graphs which traders typically use to look at price trends. Plotly helped us transform the data into the candlesticks with color coded visualizations *(Figure 11)*.

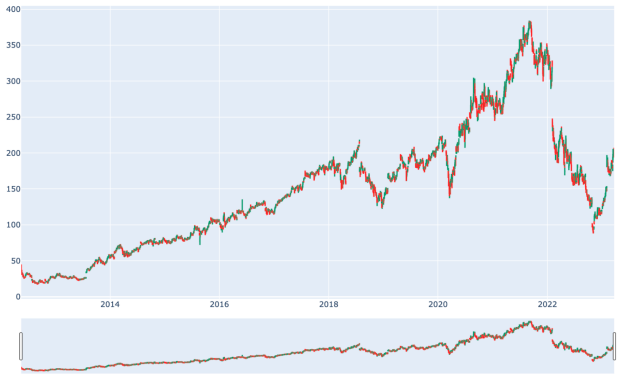

Figure 1. Meta Daily Closing Stock Price

1.2. Prepping the Data for LSTM:

To get the data we use a wrapper and call the method `get_daily_adjusted`. This returns a time series of daily stock prices for a given symbol. Adjusted prices are typically employed by researchers to account for corporate actions that affect the close. We choose the output size as full which returns roughly 2700 data points starting from May, 2012. To preprocess the data we split the dataset into a 2D frame where days 0 to 19 are the x values for the first row. The second row contains values 1 to 20. We continue with these batches of 20 until the day is reached. The last column in each row are the y values or the outputs that the model can train on. For the first row this would be the 20th day. For the second row this would be the 21st day.

Thus, we train on the first 80% of days and validate on the last 20% of days. We see this as a first round approach, and for future research one can train using fold cross validation to get better results across periods. We then put the data into a 3D tensor by adding an extra dimension using the `expand_dims` function. This allows us to feed it into the LSTM. We also use a batch size of 64, 25 epochs, and the Adam optimizer. For tailoring the hyperparameters, we use a grid search to try all the permutations from learning rate [.01,.001,.0001] and a dropout rate of [.1,.2,.3]. This allows us to select the best model from the validation results.

1.3. Prepping the Data for Multivariate SVM/SVC

For the features and output, we start with the most obvious ones: using the date as the input feature and Meta's closing price as the output. Apart from that we also brainstormed other features that might be able to help predict Meta's stock price. For example, features directly related to the company's performance like their monthly active user and net income. In addition, we also look at the stock price of other social media and the stock price of other major tech companies.

We gather and merge all the input variables (Meta's Monthly Active User, Twitter's stock prices, Snapchat's stock prices, Apple's stock prices, Google's stock prices, Meta's net income, and Meta's Lobbying Expense) and output (Meta's closing price) into one dataframe. Since some data is not on the daily timeframe, we extrapolate less frequent data to fill the gaps. For example, stock prices are daily and MAU is monthly.

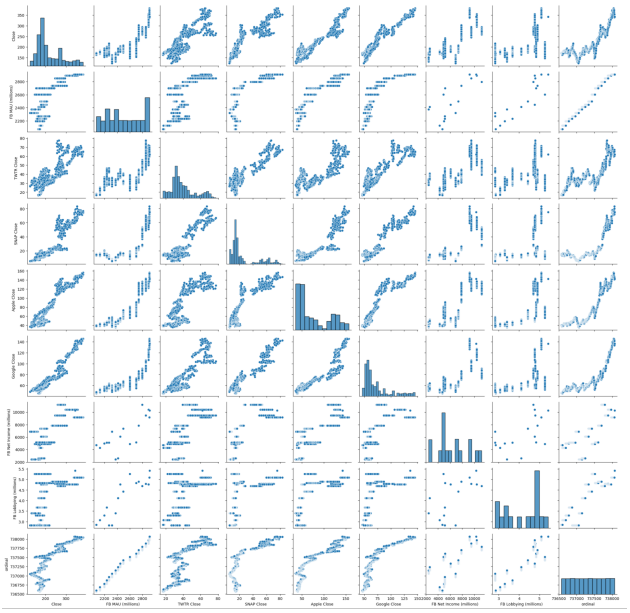

Figure 2. Input Feature Pair Grid

Then once we have all of that we plot it using the pair grid to see which features are highly correlated with each other. This will be able to guide us in selecting the best features.

2. Machine Learning Methods

2.1. Support Vector Regression

We ran four SVR models using different combinations of input features to predict Meta's Closing Stock Price:

Table 1. Root Mean Squared Error for Support Vector Regression on various input features.

Model	Featuers	RMSE
1	Date	7.91
2	Date, Meta's Net Income, Meta's Monthly Active User	4.22
3	Date, Apple's Closing Stock Price, Google's Closing Stock Price	3.61
4	Every Input Feature: Meta's Monthly Active User, Twitter's closing prices, Snapchat's closing prices, Apple's closing prices, Google's closing prices, Meta's net income, and Meta's Lobbying Expense	4.71

- Model 1: Date
- Model 2: Date, Meta's Net Income, Meta's Monthly Active User
- Model 3: Date, Apple's Closing Stock Price, Google's Closing Stock Price *(Figure 3)*
- Model 4: Meta's Monthly Active User, Twitter's closing prices, Snapchat's closing prices, Apple's closing prices, Google's closing prices, Meta's net income, and Meta's Lobbying Expense

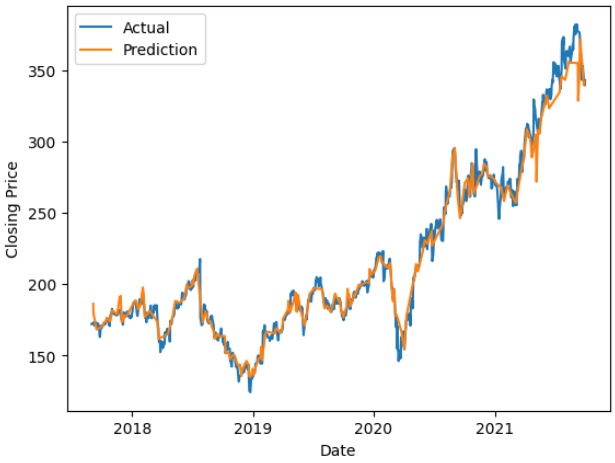

Figure 3. [Model 3] Actual vs. Predicted Price using SVR

We ran the four combinations of features and calculated the root mean square error *(Table 1)*. Looking at the RMSE there seems to be a sweet spot for choosing highly correlated features which are the stock prices of similar-scale tech companies like Apple and Google as input features. It's also not the best to use every single input feature we can find because it'll actually increase the RMSE.

The good thing about using SVR is it pretty easy to implement and takes a much shorter time to run than neural networks like LSTM. In this case, SVR fits the data extremely well, but it is most likely overfitting. Therefore, it'll have quite a hard time generating unseen data like future stock prices compared to LSTM.

2.2. Long Short Term Memory

We chose LSTM (RNN) for the second method for a number of reasons. While it is typically employed for language models like predicting the next word in a sequence, the LSTM can also be used for time series prediction. Since stock values over time are considered a time series we figured this would be an appropriate application of the LSTM. This meant we could have a window of memory to choose how many points of data to base our predictions on. We chose 20 for this. This represented a good trade-off between accuracy and speed.

LSTM is also useful for solving the vanishing gradient problem. It uses a set of gates to selectively retain or discard information at each time step. This ensures that gradients do not become too small over time and approach 0 to update the next sequence of weights. For example, the forget gate determines which information to discard from the previous state. This makes it particularly useful for long strings of data. We have attached a diagram from CS 230 at Stanford that goes into greater detail about the LSTM. One can observe the structure of the input, forget, and output. These are hooked together using a combination of arithmetic operators, the tanh function, and the sigmoid function.

To construct the model we import sequential from keras as well as the optimizer and layers. Code sample *(Figure 12)* at the end of the document illustrates the construction of the model. We have an input layer, followed by an LSTM layer, a dense layer, a dropout layer, a dense layer, and a final dense layer to obtain a single output value. With this structure we can dynamically optimize for different values of the dropout rate. In the compilation part of the model we select 'mse' as the loss and select the 'meanabsolutepercentageerror' as the metric. We were motivated to use a mean absolute percentage error for the metric as opposed to a binary classification, since many trading strategies rely on momentum. We figured that the percentage change would be a useful metric to understand the magnitude of the move in a stock instead of it simply going up or down. This is particularly useful for implementing strategies with moving averages, for example. Importantly, we give the `learning_rate` a variable that can also be used in the hyperparameter optimization technique.

2.3. Support Vector Classification

We also tried a third method that uses SVC to predict whether Meta's stock prices will decrease or increase in order to implement a simple trading strategy. For SVC we ran two models with different input feature combinations:

- Model 1: Meta's Stock Price(Open-Close, High-Low)
- Model 2: Meta's Stock Price(Open-Close, High-Low), Apple's Stock Price(Open-Close, High-Low), Google's Stock Price(Open-Close, High-Low)

The second combination of features was chosen based on the model without the lowest RMSE using SVR.

There are two evaluation metrics for this model training/testing accuracy and cumulative return from trading. The trading strategy is pretty simple. If tomorrow's price is higher than today's price(output = 1) buy the stock, and for anything else(output = 0) we do nothing.

The first metric used to evaluate the model is the accuracy. Model 1 has a training accuracy of 0.53 and a testing accuracy of 0.51. Model 2 has a training accuracy of 0.55 and a testing accuracy of 0.50. The training accuracy is higher than the testing accuracy so this indicates some amount of overfitting. In addition, both models have an accuracy of just over 50%, which doesn't look very good.

However, when trading we are looking to optimize cumulative return not accuracy. Therefore, we also added a second evaluation metric comparing the cumulative return(always buy) against the cumulative strategy return(buy based on SVR prediction). The cumulative return is 95%, Model 1 has a cumulative strategy return of 119%, and Model 2 has a cumulative strategy return of 134%. As expected every model had a great return, because Meta's stock price has increased significantly since 2017. However, Model 2 which takes in a combination of the best features has the highest return.

3. Results and Updates

From our initial presentation, we got feedback that we should look at other evaluation methods that focus on returns from trading instead of RMSE. Therefore, we added a new method, which is the trading strategy based on Support Vector Classification. With SVC we compared the cumulative return of the strategy in addition to the accuracy. We also changed the hyperparameter search from the presentation.

We originally used a grid search on learning rate and epochs. We now grid search on learning rate and dropout rate. This we realized is an important change, since manipulations in the epochs are probably not as interesting or useful as changes in the dropout rate. The dropout rate is especially important in the LSTM model for stock price prediction to reduce overfitting. A higher dropout rate can be used for layers that are prone to overfitting.

Iteration	Dropout rate	Learning rate	MAPE	Loss*100	Prediction for Tomorrow (4/26)
1	.1	.0001	59.11%	7.04	214.23
2	.1	.001	27.12%	1.83	210.81
3	.1	.01	22.01%	5.46	203.02
4	.2	.0001	60.32%	5.01	210.65
5	.2	.001	42.98%	12.92	199.97
6	.2	.01	26.77%	11.08	198.28
7	.3	.0001	66.64%	15.23	210.79
8	.3	.001	35.12%	18.11	197.87
9	.3	.01	32.76%	5.91	205.10

Figure 4. LSTM Hyperparameter Results

We successfully met a number of goals originally laid out in the project proposal. We selected a number of dynamic variables for the SVR and did an analysis to choose which ones might be best to use in the regression platform. For the LSTM we attach a table*(Figure 4)* that features the mse and metric score for each iteration of the hyperparameter search. We highlight in green the best performing model with the lowest mean absolute percentage error (MAPE) for the validation data as this would represent the most accurate model for predicting movement magnitude. The best model under this framework is a dropout rate of .1 and a learning rate of .01. This results in a MAPE of 22.01% and a loss (*100) of 5.46. The worst performing model has a dropout rate of .3 and a learning rate of .0001. This results in a MAPE of 66.64% and a loss (*100) of 15.23. Interestingly, the stock price of the best model predicts a 203.02 close price for today April 26 using data from the past 20 days. The worst model predicts a closing price of 210.79 for April 26. At the time of writing this report the stock price sits at 212.49 and the stock closed yesterday (4/25) at a price of 208.64. Both LSTM models suggested a higher price for today. Lastly, the results indicate that the model training time takes about 7 seconds. We realize that more features add complexity, which would increase this duration in a hypothetical scenario.

4. Discussions and Conclusions

4.1. Findings and Class Concepts

For methods predicting stock prices, we've implemented two methods SVR and LSTM. The two methods are able to predict stock prices fairly well with SVR having an RMSE of 3.61 and the optimal LSTM model having an RMSE of 0.135. SVR takes a significantly shorter time to run compared to a Neural Network like LSTM. However, SVR tends to overfit and isn't able to predict future values as well as LSTM. Our third method SVC doesn't predict the stock price but whether the stock increases or decreases. With SVC we are able to implement a trading strategy that produces a net positive cumulative return despite a low accuracy.

We also compare some LSTM model in graphs 1-6 in the appendix *(Figures 5-10)*. The odd number graphs represent the optimal model we selected from the results section above. The even number graphs represent the least optimal model we selected from the results section above. Graphs 1 and 2 compare the MAPE scores, 3 and 4 compare the MSE scores, and 5 and 6 compare the prediction graphs for the validation data. Despite the MAPE being significantly lower for graph 1, graph 2 exhibits a nicely shaped reduction over time. We attribute the difference in MAPE variability to mainly the learning rate. A graph of a dropout rate of .1 and a learning rate of .0001 *(Figure 15, Control Graph 1)* shows that the variability is still low and the reduction over time is constant despite holding dropout rate constant at .1 when comparing to graph 1. This indicates that learning rate plays a significant role in the variability of the MAPE over time in epochs.

We see similar behavior with graphs 3 and 4. This makes sense since the losses and the MAPE scores are highly correlated. Graph 3 has a significantly better MSE over time as compared to graph 4. However, graph 3 exhibits significant variability over the epochs. This evidence indicates that loss scores do not improve "too much" beyond about 10 epochs. Again, holding dropout rate constant at .1, we see that comparing control graph 2 *(Figure 16)* to graph 3 *(Figure 7)* indicates that learning rate accounts for a significant portion of the variability in loss over the epochs. Moreover, the magnitude of loss is significantly higher when adjusting the learning rate from .01 to .0001 and holding the dropout rate constant at .1.

Graphs 5 and 6 really illustrate the predictive power of the two models. Most interestingly, both models consistently underpredict META's price on the validation data until about January of 2022. We consider this interesting, since the COVID and the subsequent stimulus might have affected the way in which the stock trades. Thus, we consider this as a possible flaw of the LSTM and other predictive models.

Unforeseen actions taken by the government or other actors can greatly affect the predictive power of such models. That said, graph 5 shows significantly better performance on the validation data compared to graph 6.

We find some very interesting results when comparing graphs 5 and 6 to control graphs 3 and 4 *(Figure 17-18)*. We find that a lower learning rate results in less variability in predicted volatility of stock price. This holds true when comparing graph 6 *(Figure 10)* to control graph 4 *(Figure 18)* and graph 5 *(Figure 9)* to control graph 3 *(Figure 17)*. In both instances the dropout rates are held constant and the learning rates vary. The lower learning rate shows significantly less volatility on a day to day basis. Moreover, when holding learning rate constant at .0001 (graph 6 vs control graph 3) we see that a lower dropout rate (.3 vs .1) displays significantly better predictive power as indicated by a better underestimation prior to the January 2022 period. When holding learning rate constant at .01 the differences are even more pronounced between graph 5 *(Figure 9)* and control graph 4 *(Figure 18)*. In fact, the control graph with the higher dropout rate of .3 looks like a vertical transformation downward (in the 'less correct' direction) of the graph with the lower dropout rate of .1.

Aside from the grid search, another thing we did related to class is the SVR. However, instead of classifying points, it finds a best-fit line. The best-fit line is the hyperplane with the maximum number of points within a certain threshold. We also implemented a pair grid similar to the wine problem set from class to look at the correlation between input variables in order to aid feature selection.

4.2. Challenges

One challenge was finding and formatting data for multiple input variable models. This is because each feature came from different sources, different frequencies, and different time frames. To solve this we had to find as much data as possible followed by a lot of data wrangling.

Another challenge was preprocessing the data into the correct format for the LSTM to use. This required some data manipulations to fit into the correct window size and dimensionality increases. Otherwise, the process was smooth to plot the data and create the model thanks to the built in functionality in the keras library. A significant portion of the time was thus spent preprocessing and running/testing the model for accuracy.

For future work it would be interesting to incorporate more features into the LSTM model by increasing tensor size through a sequential forward search or a tree search built into keras. This would be useful to compare accuracy when predicting META stock price as compared to just the META price alone.

Another interesting extension of the project would be to design a trading strategy based on a momentum indicator instead of stock price prediction. Using an LSTM or GRU academic research in the area has explored designing a strategy to trade based on the momentum as indicated by the prediction of a ratio of moving averages. This could be especially interesting if one can then backtest the strategy across a number of equities to see performance over time. Unfortunately, a lot of the alpha derived from such strategies tends to lose its effectiveness over time in generating profits. Markets are highly efficient, but it may still prove to be a useful thought exercise given enough time.

5. Teamwork

We divided the work into two parts. The first part (Panda's) uses multiple input features and methods we've learned in class such as Support Vector Regression and Support Vector Classification to predict Meta's stock prices. The second part (Zach's) uses LSTM to predict Meta's stock prices and includes the hyperparameter grid search. We worked on the bulk of the coding independently. We met in person for progress updates and to discuss what to work on next.

6. References and Acknowledgements

6.1. Data Sources

1. *"Number of Monthly Active Meta Users Worldwide." Statista, https://www.statista.com/statistics/264810/number-of-monthly-active-Meta-users-worldwide/. Accessed 25 Apr. 2023.*

2. *Prasertk. "Major Social Media Stock Prices 2012-2022." Kaggle, https://www.kaggle.com/datasets/prasertk/major-social-media-stock-prices-20122022. Accessed 25 Apr. 2023.*

3. *Vainero. "Google, Apple, Facebook Stock Price." Kaggle, https://www.kaggle.com/datasets/vainero/google-apple-facebook-stock-price. Accessed 25 Apr. 2023.*

4. *"Facebook's Quarterly Net Income." Statista, https://www.statista.com/statistics/223289/facebooks-quarterly-net-income/?locale=en. Accessed 25 Apr. 2023.*

5. *"Quarterly Lobbying Expenses of Facebook." Statista, https://www.statista.com/statistics/236969/quarterly-lobbying-expenses-of-facebook/?locale=en. Accessed 25 Apr. 2023.*

6.2. Related Literature

1. *Stanford CS230. "A Deep Learning Approach for Stock Market Prediction." Course Project, 2023.*

7. Appendix

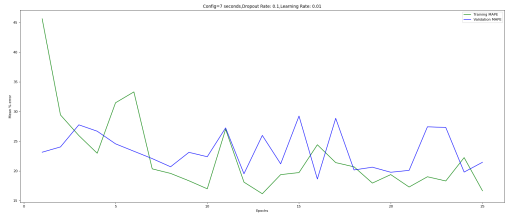

Figure 5. Graph 1: MAPE where DR = .1, LR = .01

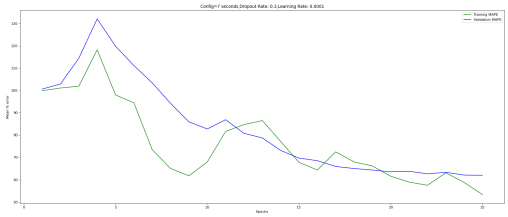

Figure 6. Graph 2: MAPE where DR = .3, LR = .0001

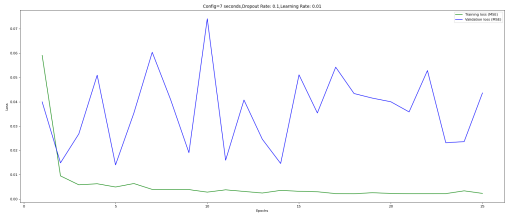

Figure 7. Graph 3: MSE where DR = .1, LR = .01

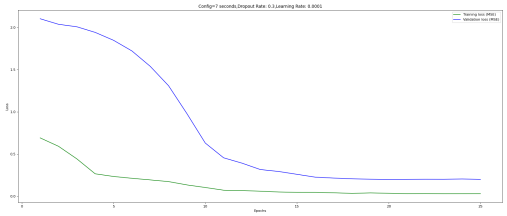

Figure 8. Graph 4: MSE where DR = .3, LR = .0001

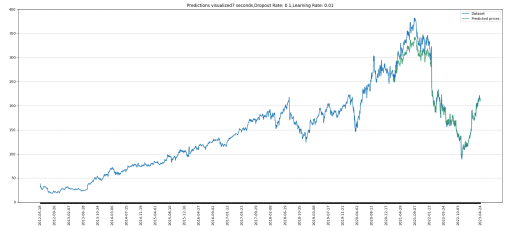

Figure 9. Graph 5: predictions where DR = .1, LR = .01

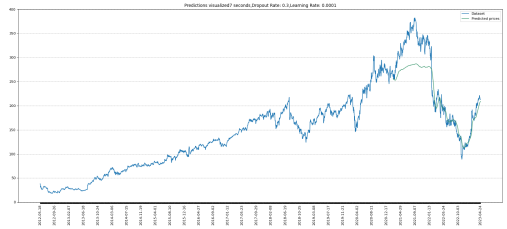

Figure 10. Graph 6: predictions where DR = .3, LR = .0001

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$\tilde{S}_t = tanh(W_s[h_{t-1}, x_t] + b_s]$$

$$S_t = f_t * S_{t-1} + i_t * \tilde{S}_t$$

$$h_t = o_t * tan(\tilde{S}_t)$$

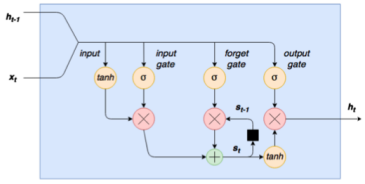

Fig. LSTM Memory Unit [9]

Figure 11. Code Visualisation

```
model = Sequential([layers.Input((20, 1)),
                    layers.LSTM(64),
                    layers.Dense(32, activation='relu'),
                    layers.Dropout(dr),
                    layers.Dense(32, activation='relu'),
                    layers.Dense(1)])

model.compile(loss='mse',
              optimizer=Adam(learning_rate=lr),
              metrics=['MeanAbsolutePercentageError'])
```

Figure 12. Code Sample 1

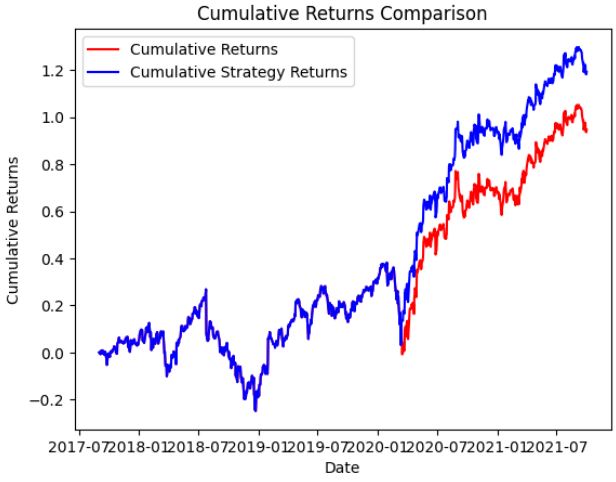

Figure 13. Model 1: Cumulative Returns Comparison

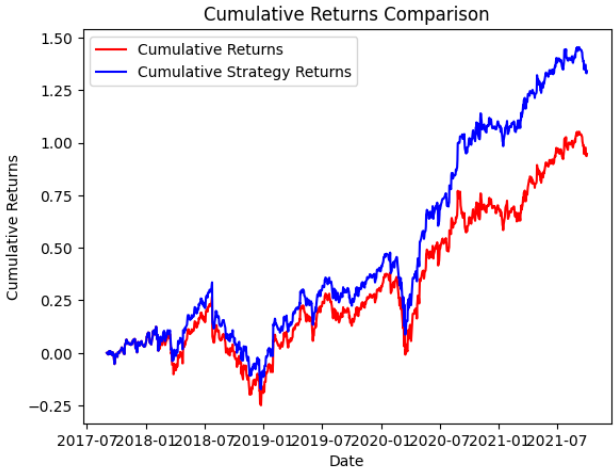

Figure 14. Model 2: Cumulative Returns Comparison

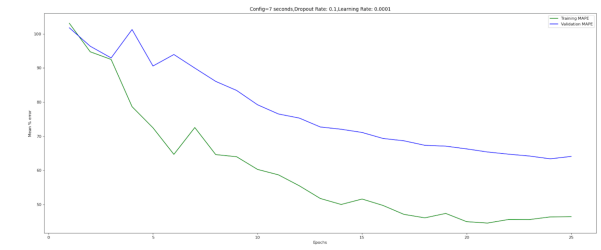

Figure 15. Control Graph 1

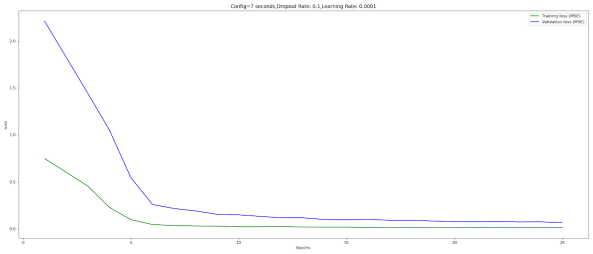

Figure 16. Control Graph 2

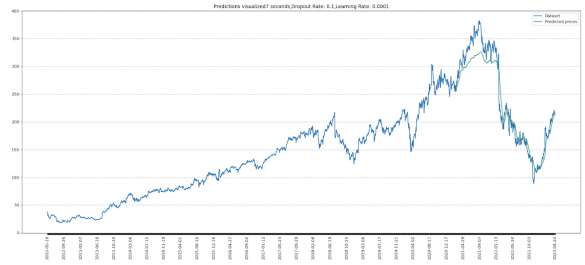

Figure 17. Control Graph 3

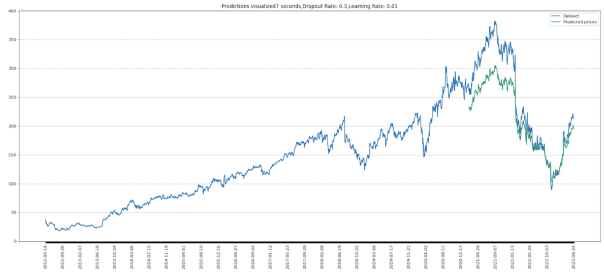

Figure 18. Control Graph 4