ISE 537: Final Project

Professor Papavassilopoulos

By: Lucas (Deuce) Palmer

This analysis of stock price prediction examines the implementation and application of two distinct methodologies: Long Short-Term Memory (LSTM) and Autoregressive Integrated Moving Average (ARIMA). LSTM is a type of recurrent neural network (RNN) which specializes in learning and modeling sequential data, such as stock prices. In comparison to a general RNN, the LSTM better protects against the vanishing gradient problem and therefore can better capture long-term dependencies. ARIMA is a series forecasting model which relies on past data points to predict the future. Specifically, ARIMA is a combination of autoregression (AR), differencing (I), and moving average (MA). AR and MA can be used as standalone models for sequential forecasting but may provide added benefits when combined, especially for complex data. The order of differencing indicates how many times a data point needs to subtract its previous data points in order for the series to be stationary. In this analysis, both models will be examined in depth by attempting to predict stock prices for the two companies; Microsoft Corp and China Evergrande Group. The discussion begins with an introduction to the data and reasoning for company selection. Then continues with a detailed analysis of each model, a comparison of the performances, and a financial interpretation of the findings.

Test and training data collected for the analysis of both stocks and both models were collected from Yahoo's Python finance library, yfinance. This data is stored in a Pandas data frame and includes daily stock information. For the purpose of this study, all data is dropped besides the daily closing price. Preprocessing of the data for the LSTM model includes scaling each of the stock prices with a minimum of 0 and a maximum of 1 in order to achieve normalization. In addition, both models will make one split on the closing price data with the earliest segment of size 80% used for training and the latter 20% used for testing. Selecting one time period is not quite enough to fully understand the performance of each model. Therefore, testing and training will be completed across three different time periods; 6-months, 1-year, and 2-years. Not only will these time periods test the model's ability to predict based on the amount of data implemented, but it will also show how the existence of market conditions across longer periods will contribute to or hinder performance. Each of these periods will end at the last completed month of this year (March 2024) and will only vary based on the start date. Current time periods were chosen to enhance relevancy, assuming that one would look to apply these models in the current market. By choosing market data close to the present, the model will better learn market conditions that are highly likely to also be present in the near future. The goal of the stock selection was to choose stocks that exhibit very different characteristics. In order to better understand how the model performance is affected by the characteristics of a stock, choices included one low-variance, high-priced, high-market cap, and upward trending stock and one high-variance, low-market cap, low-priced, downward trending stock. Utilizing Trading View's stock sorting platform enabled searching for stocks based on these specific attributes. To fulfill the first selection's requirement, Microsoft (MSFT) was chosen which has a current volatility of 1.66%, a market cap of $3 Trillion, and is currently experiencing growth with a strong buy analyst recommendation. For the second selection, China Evergrande Group (EGRNQ) has a volatility of 49.86%, a $300 Million market cap, and has experienced a significant decline in

stock price over the past few years with a recommendation from analysts to sell. By selecting these specific attributes, two very different assets have been chosen, both of which will test the models in their own unique ways. In order to compare the test accuracy of the two models, root-mean-squared error (RMSE) will act as a performance metric as it has specific application benefits to the financial market. For example, RMSE is sensitive to large errors in the same way that large stock price prediction errors could result in significant financial loss. Minimal losses and variance among the stocks are expected and can be suppressed using diversification.

When implementing an ARIMA model, it is vital that the series is stationary. A series is considered stationary if its mean, variance, and autocorrelation remain constant over time. In order to ensure each series was stationary, the pmdarima library's ndiffs function was used to check the order of differencing necessary for stationarity. The pmdarima library's auto_arima function was utilized to optimize the lag values for both the moving average (q) and autoregression (p) components. This function loops through possible values of p, d, and q in order to minimize the Akaike Information Criterion (AIC). While the auto_arima function also optimizes d, comparing its value to the ndiffs results confirms that the series will be stationary. The AIC is a measure of how well the ARIMA model fits the data. In a way, the metric also acts as a regularizer by penalizing models that utilize a larger number of parameters. By favoring models with fewer parameters, the model is more likely to achieve generalization on unseen data and avoid overfitting. It is important to note that choosing the combination of p, d, and q with the lowest AIC does not imply that the combination will also have the lowest test RMSE. AIC only reflects the ability to fit the training data, not the test data. As mentioned, it helps to encourage generalization, but there is no guarantee. The optimal (p, d, q) values for Microsoft are (1,1,0), (0,1,1), and (2,1,2) for the corresponding periods of 6 months, 1 year, and 2 years. Similarly, the optimal orders for China Evergrande Group that minimize AIC are (0,1,2), (3,1,1), and (3,1,2). By choosing these optimal p, d, and q values, the RMSEs across the three time periods for MSFT are 4.85, 5.65, 4.99, and 0.0055, 0.0085, 0.019 for EGRNQ. It is important to note that these values are calculated in a rolling fashion. With each estimate at t+1, the true values are filled in for t, t-1, etc.

Unlike the ARIMA model, an LSTM neural network has many more parameters that require modifications in order to maximize performance. Initially, experimenting with a different number of epochs showed minimal improvement, if any, beyond 100 epochs. If a larger number of epochs would have been used, the model would be very prone to overfitting. Exploration of the number of neurons per LSTM layer revealed, through experimental hyper-tuning, that the optimal value was around 40. Another factor that contributed to improved outcomes was the batch size. While reducing the batch size did extend the training duration, it also enabled the reduction of training and test errors. Throughout further experimentation, the optimal batch size was approximated to be 25. In addition, an analysis was conducted to explore the effect of adding another layer to the neural network. When adding an extra LSTM layer to each period-specific model, there was no significant improvement in training and test error. The concluded hypothesis for the lack of improvement is due to the complexity of the data. Adding

extra layers is crucial when dealing with complex data with many non-linear relationships, concluding that this stock market data is not extremely complex. Similarly to the ARIMA, the model's test predictions are calculated in a rolling fashion, which requires the implementation of a lookback window. In terms of its length, It could be argued that a longer lookback window should be applied to longer periods of data, but by keeping this variable fixed at 10 one can better understand the sole impact of the series length on prediction capabilities. The training loss for each of the three time periods for both MSFT and EGRNQ is plotted over each epoch. Each plot shows strong convergence as training loss rapidly drops during the first few epochs, then stays relatively constant for the remainder of model fitting. This also proves why increasing the number of epochs as discussed previously did not provide any benefit to the performance. For this model and the corresponding data, using 100 epochs was sufficient in order to minimize error and reach convergence. The final RMSEs for the three time periods of 6 months, 1 year, and 2 years for Microsoft were 5.74, 5.78, and 4.92 respectively. Correspondingly for China Evergrande Group, the RMSE values were 0.0061, 0.0062, and 0.015.

When comparing the final RMSE results, there is no clear winner. When examining Microsoft, The ARIMA model outperformed the LSTM for the 6-month and 1-year periods, but underperformed the LSTM for the 2 year period. For China Evergrande Group, ARIMA outperformed the LSTM for only the 6-month period, but not the 1-year or 2-year periods. It seems that for the less-volatile stock, MSFT, the ARIMA model had the upper hand. While for the highly-volatile EGRNQ, the LSTM was better at predicting these complex patterns. In addition, there is evidence that the ARIMA model may possess a slight advantage for shorter periods while the LSTM model strives when given more data. All the optimal lags for the ARIMA models were quite small, indicating that predictions depend only on a few past observations. Therefore, the model does not take into account long-term dependencies. On the other hand, the LSTM model excels at capturing long-term dependencies. The performance advantage most likely correlates with the model's ability to detect strong long-term dependencies across the lengthened series. Another metric to evaluate the two models is interpretability. The ARIMA model is a simple mathematical formula, using past values to predict a future value. This simple architecture increases interpretability, enhancing one's ability to determine how the model works. An LSTM has no clearly defined formula and therefore is not as interpretable as the ARIMA model. One crucial factor when dealing with time period predictions is speed. As we discussed in lecture, a model should produce results faster than the change of the system. In this case, the system changes daily so both models adequately produce results fast enough. But what about the application of these models to day trading, where the fluctuation of prices per second can greatly affect one's profit or loss? The speed of choosing the ARIMA model lags, or the training of the LSTM would be of much larger significance. Iterating through all possible order combinations of lags to see which model minimizes AIC is a timely process. While an ARIMA model is quick to make predictions and re-fit the data when a new data point is received, one could experience issues if the model undergoes constant modifications by testing lag combinations every time a new price entry is received. In comparison, the LSTM model

experienced minor speed setbacks when decreasing the batch size. For applications such as day trading, the batch size along with other parameters may be tuned in order to maximize speed. It is important to note that while the ARIMA model's speed efficiency allowed it to refit every time a new data point was received, an LSTM does not allow for similar modification. Fitting an LSTM model takes adequate time and it would be unreasonable to refit as often as the ARIMA model, especially for day trading.

The financial interpretation and application of these findings greatly relies on the specific use case. The ARIMA model's characteristics of interpretability, speed, and proficiency in generating accurate short-term forecasts make it specifically applicable to certain financial occasions. For example, let's assume a quantitative trader is working at a hedge fund and is building a high-frequency trading model. The stocks in which the model will be deployed on often experience very little volatility. In addition, this trader must be able to clearly explain the model to his/her boss before receiving funds for trading. This specific financial case clearly exemplifies an example where an ARIMA model would outperform an LSTM model. In contrast, take a software engineer who yearns to apply his/her knowledge to the stock market. This individual has extensive knowledge of machine learning models such as the LSTM and is looking to execute trades infrequently to mitigate transactional fees. This individual is comfortable losing the amount of money they will invest and decides to target highly volatile stocks. As the interpretability is not of significant importance, the time horizon is much longer, and the assets in question are highly volatile, an LSTM model would be a great fit for this example. These models, along with others, possess a unique set of characteristics that may help to predict their applicability to different financial use cases.

This analysis explored the application of Long Short-Term Memory and Autoregressive Integrated Moving Average models on time series forecasting. In terms of added efforts beyond the original problem statement, an additional layer was added to the LSTM and three time period lengths were used instead of one with the intent to further understand the performance and correlation of both models. To conclude the findings, The ARIMA model performed better on shorter periods, while the LSTM excelled with longer periods. In addition, the LSTM seemed to be the better choice for highly volatile and complex stocks, while the ARIMA model was favored for the asset with low volatility. It is important to note that these performance hypotheses would require verification across numerous other periods and stocks in order to confirm the findings. Each model has a unique set of characteristics as discussed previously, all of which need to be considered when deciding on real market implementation.