Price Forecasting using Machine Learning Analysis

Alexander Cimini

ECNS 460

Introduction

Understanding and predicting stock price movements is a critical challenge in financial markets. This project seeks to address the research question: Can an LSTM accurately forecast the day ahead Microsoft stock closing price using historical price data and additional market indicators? To answer this, a machine-learning model was developed to predict the closing price of Microsoft stock one day ahead. The predictive model chosen is a Long Short-Term Memory (LSTM) model, a specialized type of recurrent neural network designed for applications with time series data. To enhance the model's predictive power, additional variables were incorporated into the analysis. By leveraging these features and a robust machine learning framework, the goal is to develop a reliable and accurate model for predicting stock prices.

Data

Data Source

The data used for this analysis includes historical information on Microsoft stock, obtained from a variety of different data sources. Microsoft stock was chosen for the analysis because of the extensive historical data, with information on it dating back to its initial public offering in 1986, and the high trading volume which in theory minimizes the market volatility / reduces the noise of the data. The dataset spans from January 1st, 2000 to October 10th, 2024, providing a very large timeframe for the training and testing of the model. Figure 1 shows the daily close for the stock price across the entire time horizon. The data set includes daily metrics of certain variables below and the source of each variable is shown as well:

- Closing Price of Microsoft stock : Barchart
- Trading Volume : Barchart
- S&P 500 Indicators : Alpaca
- Volatility Index (VIX) : Barchart
- Corporate Events Markers : Nasdaq

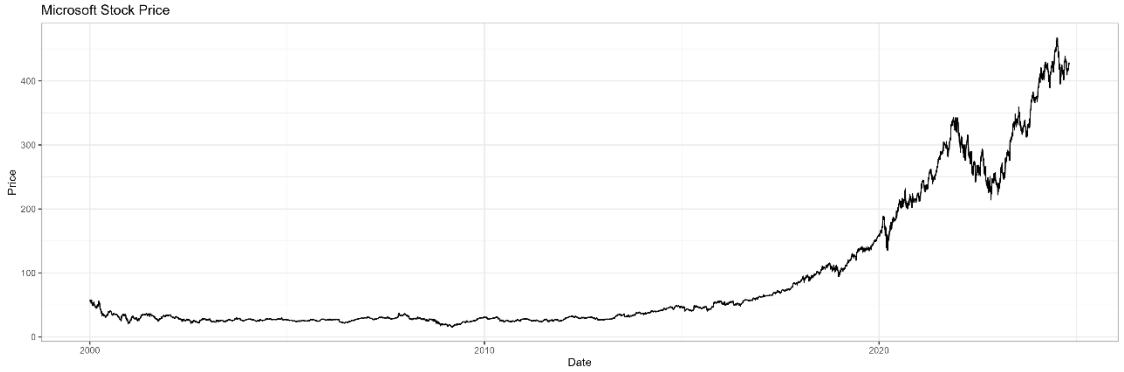

Figure 1

Preprocessing

To prepare the data for analysis, several processing steps were taken to ensure the data was clean and formatted in a way that would make model implementation easy. First cleaning of the data was done, which was not very hard as most financial time series data comes in mostly the same format and rarely has missing values. Additional feature engineering was performed to enhance the dataset by introducing variables such as the percent changes in key metrics and rolling averages of volatility. The methodology behind it is that the features may capture some dynamic patterns and trends that could potentially assist in improving the predictive power of the model. Once the dataset was complete and merged a normalization of all the variables was done, R made this very easy as they have a simple one-line function to turn all numeric variables into scaled versions of themselves.

Finally, the data was divided into three subsets. The first set consisted of the first 65% of the observed data which will be the set of data to train the model on (Training Set). The next set consisted of the next 20% of the observed data which will be used to validate the model and it's hyperparameters (Validation Set). The final set consists of the final 15% of the data which will be used once the model has been tuned, and we will use this to evaluate the model's performance on unseen data (Test Set). Figure 2 shows a graphical interpretation of the splits. All of this is done to remain unbiased during the model creation process and validation.

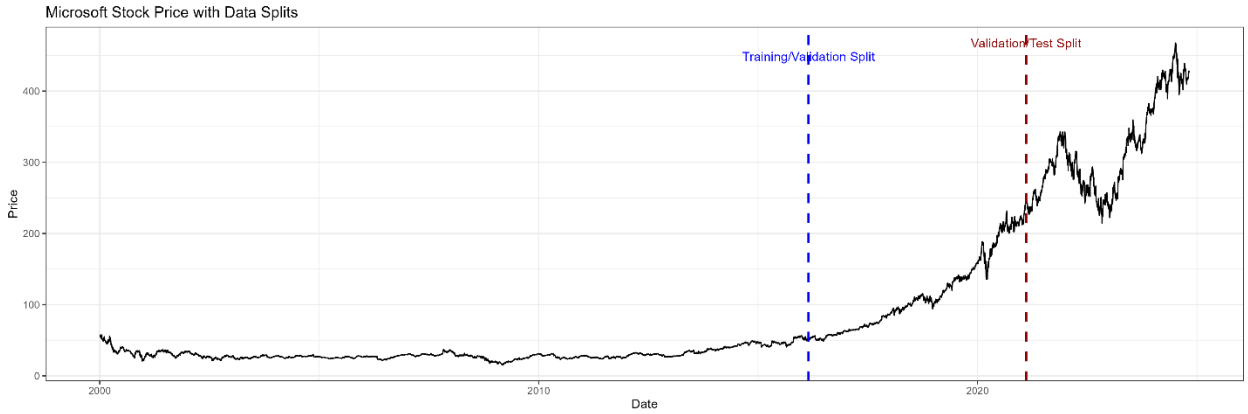

Figure 2

Methods

Modeling Language

All of the data wrangling, processing, and modeling was done in R as many good libraries within R can streamline the process. The main packages that were used were the tidyverse package for the data wrangling and processing, the keras3, and the TensorFlow package were used for the implementation and testing of the LSTM.

Model Structure

The underlying idea for the Long Short-Term Memory (LSTM) network is that it is a very specialized type of recurrent neural network that is designed to learn and model more of the temporal dependencies of sequential data. It incorporates a very unique structure that includes a "memory cell" and gating mechanisms that allow the model to selectively retain or discard

information as sequences are processed. This is why I chose to use the LSTM for price prediction, as it effectively leverages both long-term and short-term trends in the data, making it a more robust modeling approach.

The model optimizer chosen was the Adam optimizer, which is a very efficient algorithm for optimization techniques for gradient descent. Because it is very good when working with large problems involving a lot of data, I chose to go with it. For the loss function of the model, the metric mean squared error (MSE) was evaluated, as this seems to be common industry practice.

Implementation

The model was trained using a single LSTM layer, which is a component of the neural network responsible for capturing temporal dependencies in the data by processing sequences of stock prices and predictors. The training process consisted of 50 epochs, where one epoch represents a complete pass through the entire training dataset. To ensure computational efficiency and stable updates to the model weights, the data was divided into mini-batches of size 32. Each batch is a subset of the data used to compute gradients and update the model parameters during training. Performance was evaluated on the validation set after each epoch to monitor progress and mitigate the risk of overfitting.

Once the model achieved a Mean Squared Error (MSE) that I found balanced the overfitting and dependency of the data, it was applied to the test set for forecasting. The predictions and actual values from the test set were then denormalized, allowing for direct interpretation and comparison of the predicted stock prices against the actual values.

Results

Validation Model

After working with the LSTM model (honestly just playing around with it for a while), the model of choice to move forward with was single-layered with 50 epochs and a batch size of 32. This was ultimately landed upon because it was a good middle-ground between a low MSE that was not overfitting the data and a not-very-good predictive model. This gave an MSE of 0.138. Again, this was a self-made choice and does not have much mathematical backing to it, but it seemed to be the best way forward.

Test Set

After the validation was completed, it was applied to the testing set, we can see the results in Figure 3, where the black line is the actual stock price, and the red line is the predicted stock price. The error of the model over time is shown in Figure 4.

Figure 3

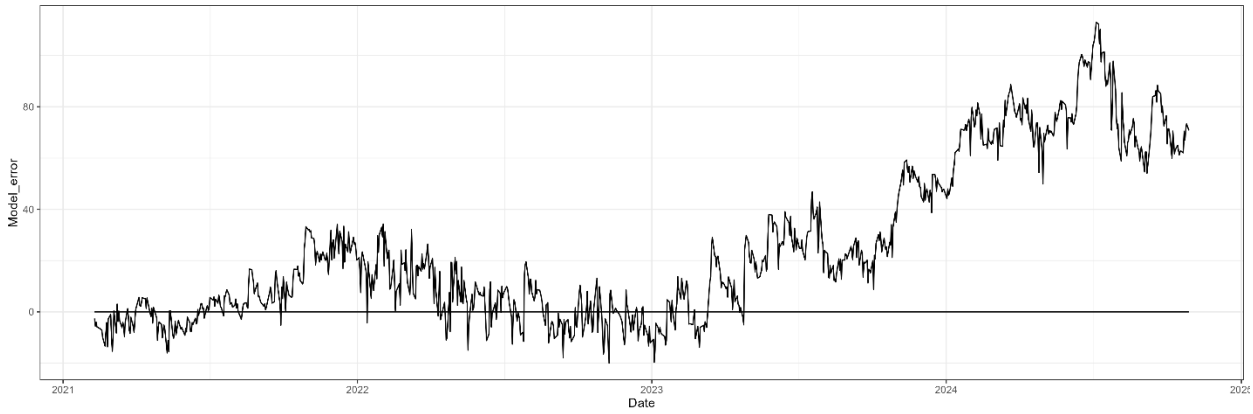

Figure 4

Interestingly we see that the errors look a lot like the underlying price, even with the divergence in the middle of the data, and that the errors do not look like white noise. An autocorrelation function was applied to the errors to see how tightly correlated they were, and the output is in Figure 5. We can see that the autocorrelation of the errors is very high and reflects what we would see out of the original stock.

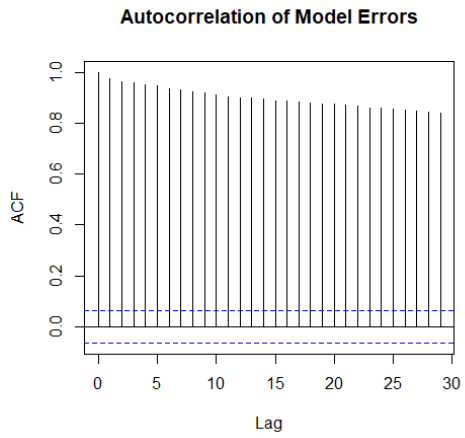

Figure 5

Discussion

Results Analysis

From the output and analysis of the results we can see that the model at the beginning visually does well and then diverges, but further analysis of the errors reveals that the model has a lot of large complications. Because the errors are so tightly correlated and are non-stationary it points to the model just being flat garbage. The problem with the residuals being this way is that it indicates that the model is failing to fully capture the patterns in the data, and it is systematically missing some key information or trends. It could also be because the model is overfitting the data, so it is not suitable for use.

Going forward

I believe the errors made when creating and using this model to be very important in going forward and using them for this type of data again. It is now too late to go back and introduce new features to the data or attempt to change the model since I now have a bias towards the data and doing that would be data leakage or more formally called test set contamination, which can lead to misleading results and even more potential overfitting of the data. But for a different predictor model (maybe one for a different stock or a variable symmetric to it) I believe this type of model can be improved and may be useable.

For the next iteration of this model, I would take several steps to enhance its performance and address potential limitations observed in this analysis. First, I would gather additional data that could provide more context or capture dependencies within the dataset, such as external economic indicators, sector-specific metrics, or other market variables. Expanding the dataset could help the model identify patterns and relationships that may not have been evident in the current data. Second, I would dedicate more time to feature engineering to refine the input variables. By hopefully carefully designing these variables, the model could be better equipped to learn meaningful patterns in the data. Finally, I would explore variations in the data preprocessing methods. For example, I might apply differencing to make the data more stationary or experiment with smoothing techniques to reduce noise and emphasize trends. Doing this could help the model focus on the most relevant signals, improving its ability to generalize and in theory make accurate predictions.

Conclusion

This project shows just how challenging it can be to predict stock prices using machine learning, even with a model like an LSTM. While the model initially seemed to work well in capturing some trends, the results exposed a lot of flaws, including highly correlated errors and systematic issues that suggest it struggled to fully understand the patterns in the data. These findings show the need to put more thought into how the data is prepared, and how some features are selected or built.

Even with its shortfalls, the model provided a lot of useful takeaways for future attempts. Adding more data, creating better features, and trying different ways to preprocess the data could help make a similar model much better.

References

R Core Team. (2023). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Wickham, H., Averick, M., Bryan, J., Chang, W., D'Agostino McGowan, L., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., … Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software, 4*(43), 1686. https://doi.org/10.21105/joss.01686

Allaire, J., & Chollet, F. (2023). *keras: R interface to Keras*. R package version 2.12.0. https://keras.rstudio.com/

Allaire, J., Tang, Y., & Ostblom, J. (2023). *tensorflow: R interface to TensorFlow*. R package version 2.12.0. https://tensorflow.rstudio.com/

GeeksforGeeks. (n.d.). Adam optimizer. *GeeksforGeeks*. https://www.geeksforgeeks.org/adam-optimizer/

GeeksforGeeks. (n.d.). Deep learning: Introduction to long short-term memory. *GeeksforGeeks*. https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/

Nasdaq. (n.d.). *Microsoft Corporation (MSFT) press releases*. Nasdaq. Retrieved December 3, 2024, from https://www.nasdaq.com/market-activity/stocks/msft/press-releases

Alpaca. (n.d.). *Alpaca: Commission-free API-first stock brokerage*. Retrieved December 3, 2024, from https://alpaca.markets/

Barchart. (n.d.). *Market data APIs, feeds, and digital solutions*. Retrieved December 3, 2024, from https://www.barchart.com/

OpenAI. (2024). *ChatGPT: Language model for natural language processing assistance*. OpenAI. https://openai.com/

-AI was used as a coding assistant for error correction with the model creation.