**Group Stocks:** Jack Oglesby, Caroline Gaede, Audrey Bristol, Elvis Atitsoabui

**Date:** 03 May 2024

***Analyzing and Predicting Stock Market Behavior Using Machine Learning***

**I: Introduction**

**Motivation**

The main goal of our project was to understand stocks and their behavior and especially their volatility. We wanted to do this using machine learning methods and time series techniques that were adapted to work with time dependent data. From this, we hoped to be able to make accurate predictions on stock price. The motivation for this project came essentially from the core of investment analysis. With the rise of artificial intelligence, we see that many traders are relying on algorithmic trading to predict and gain information on stocks. Not just this, but new methods, models, and techniques are being derived to tell traders and investors what stocks to trade and when. With the help of new effective machine learning techniques, prediction on volatile time dependent data such as stocks has become much easier and much more predictable then it was even 10 years ago. Our project was directed at exploring how powerful these techniques are, what the conditions of their effectiveness are, and just exactly how much these techniques could do. Our resulting models serve as a first step in trying to minimize volatility and maximize return on investment.

**The Dataset**

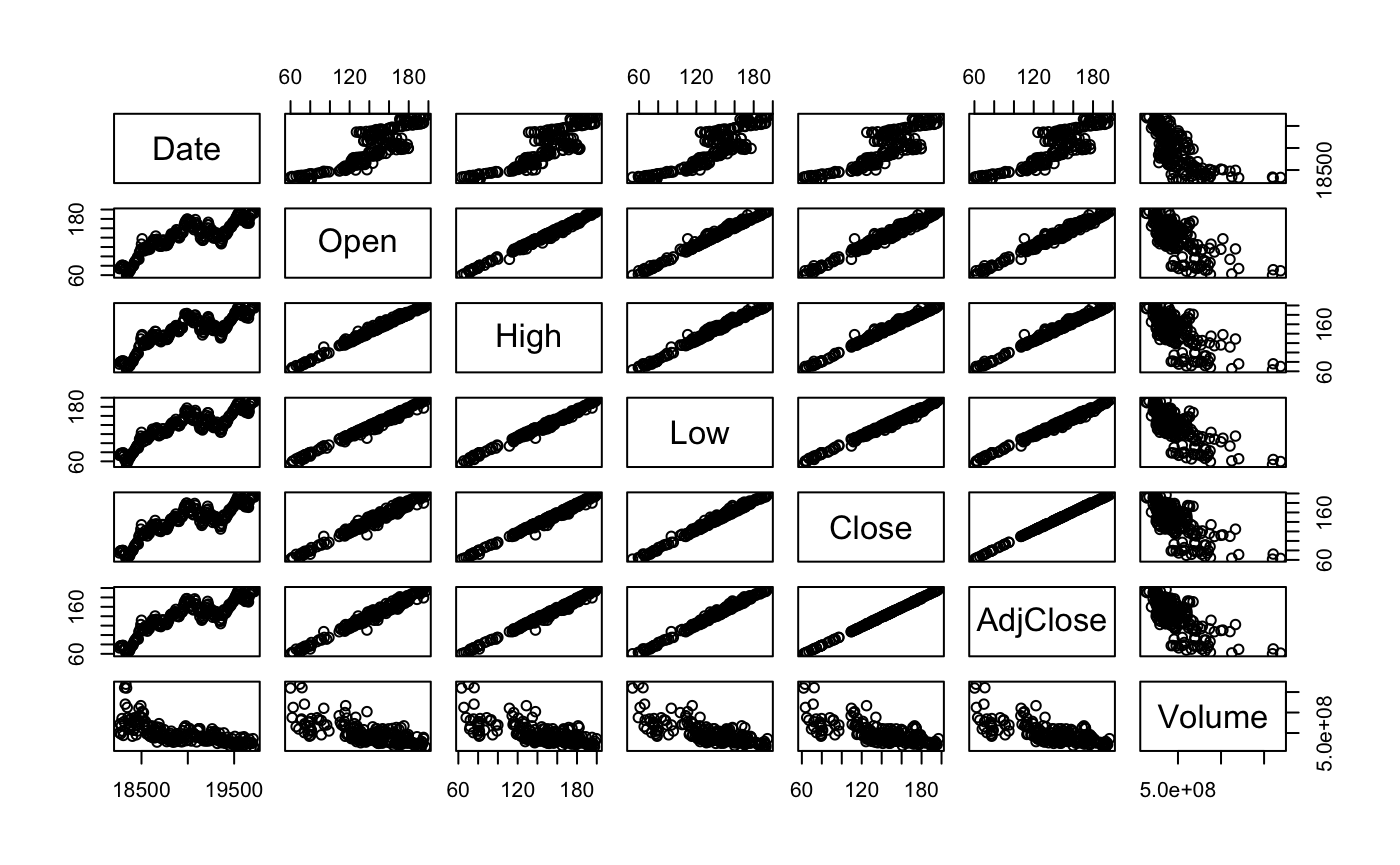
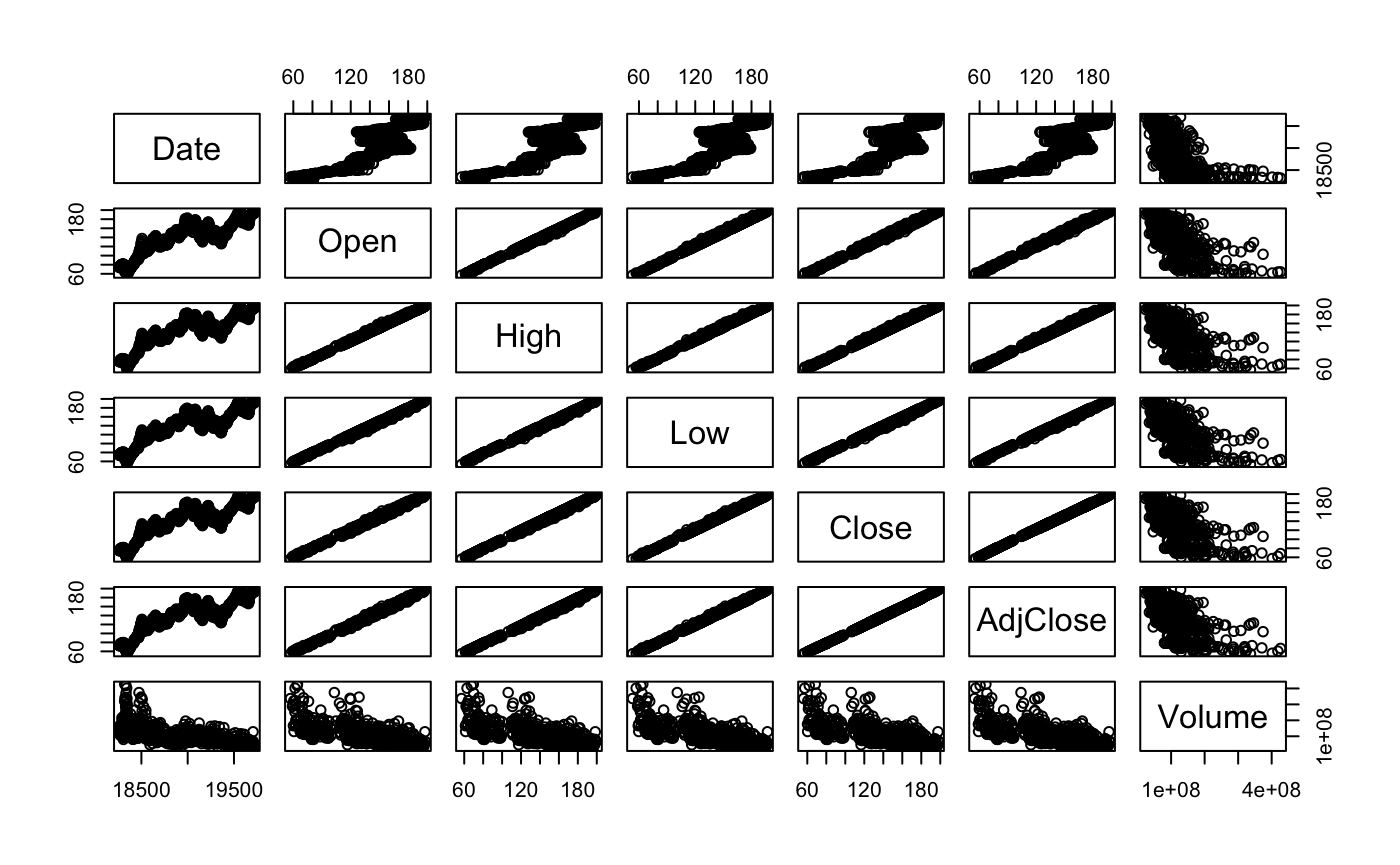
The datasets used for this analysis were taken from Yahoo Finance. It was the Apple (AAPL) stock data from over a four-year timespan (January 2020 - December 2023) and consisted of three separate datasets broken down by timeframe: Daily, Weekly, and Monthly. This way, we could measure the effects of our various methods of predicting average return on different time scales. Each dataset recorded six numerical variables in addition to the date. ‘Open’ was the price of the stock at the beginning of the stock market (Monday 9:30 AM, EDT). ‘Close’ was the value of the stock at market close (same day at 4:00 PM, EDT). Of course, weekends were excluded from the dataframe since the market is closed. The ‘High’ and ‘Low’ were the maximum and the minimum of the stock prices, respectively, over the given timeframe. ‘Volume’ represented the amount of trades made in that same timeframe. Lastly, ‘Adjusted Close’ was the closing price of the stock adjusted for dividends and the amount needed to pay back shareholders.

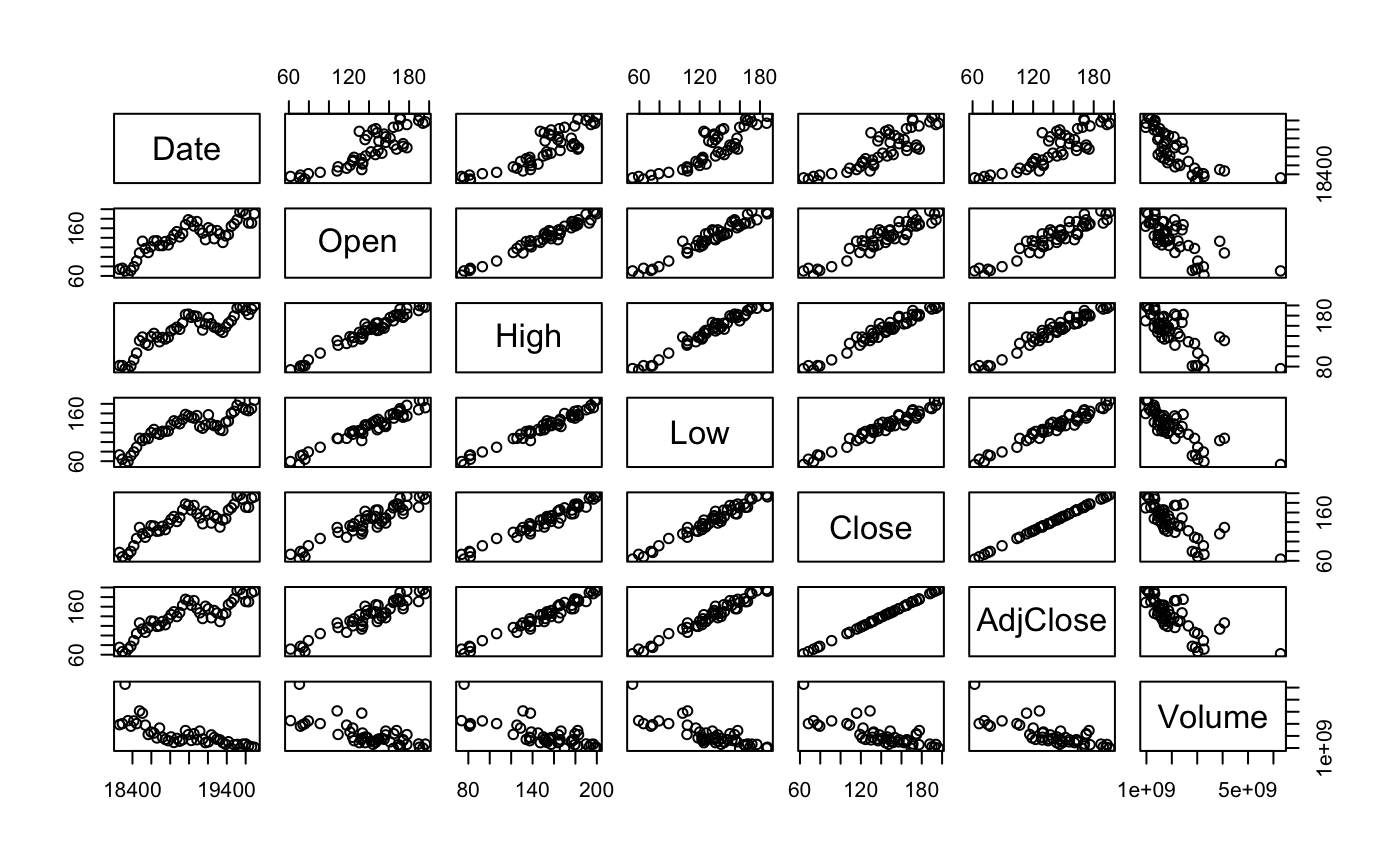
Adding to this dataset, we created an additional column designated for ‘Returns’. We calculated these by subtracting opening stock prices from closing ones, then dividing by those closing values. This reflected the change in the stock for one time-step of its respective timeframe. Should the stock price have gone up in one time-step, then the value would have been positive; had it gone down, the value would have been negative. Understanding the closing values and their role in returns was as close to predicting the stock in real time that we could get without using actual tick data, which is more comprehensive but significantly more expensive and computationally complicated.

**Exploratory Data Analysis**

Since little preprocessing was required given our Daily, Weekly, and Monthly datasets, the next step was to perform an exploratory analysis to pickup on trends in the data and get a better feel for variable interactions and behaviors. The goals of this initial analysis included gathering summary statistics, observing pairwise plots and correlation matrices, investigating multicollinearity, calculating and visualizing distributions of returns, and observing distribution of differences between variables. This last goal was added since variable differences such as those used directly in the computation of returns could tell us more about these relationships.

The first notable finding was seen in the three pairwise plots. The figures below advance from Daily to Weekly to Monthly pairwise plots for the seven variables in the datasets. We see that variability decreased as more data points were added to the set. It was also easy to see high, strong, positive correlations between some of these variables.





Regarding the investigation on multicollinearity, we decided a simple observation of correlation matrices and these pairwise plots was not enough, so we checked the proportion of variance explained by eigenvalues of variables, which is seen below for the Daily, Weekly, and Monthly datasets.



We also calculated condition numbers which told us, along with the values above, that we would need to address the issue of multicollinearity in each of the respective models and we would have to proceed with caution. For the remaining goals, the summary statistics were as expected, we found that return distributions are roughly Normal, and we concluded that features of differences in variables had similar results as those from our multicollinearity investigation.

After this initial analysis, we were ready to proceed with our predictive models. Based on our findings, we had two time series techniques for stochastic modeling, which included an Autoregressive (AR) model and an Autoregressive Integrated Moving Average (ARIMA) model. Next, we explored Long Short-Term Memory (LSTM) Neural Networks which are well-suited for time series data since forget gates and memory cells help capture patterns better than for regular neural networks. Lastly, we explored tree models since they are common in the industry, robust, and interpretable. They were great for picking up nonlinear complex relationships and have built-in functions to identify feature importance which could be useful.

**II: Methodology**

**Autoregressive Model(Jack Oglesby)**

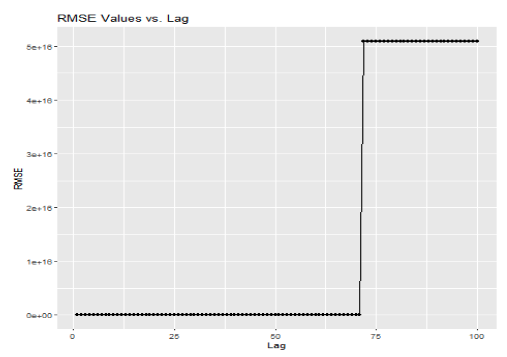
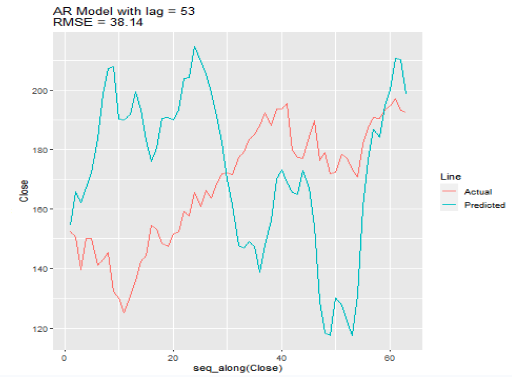
***Why AR?***

We started with an Autoregressive (AR) model because of its usefulness when analyzing time series data where the value of a variable is dependent on its previous values. Essentially, the model uses these previous values to predict its future values. Autoregressive models are relatively simple and easier to interpret compared to other complex machine learning methods we were planning on using for our analysis, which is why we started with it first.

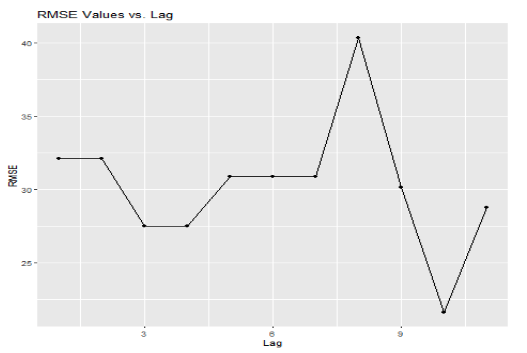
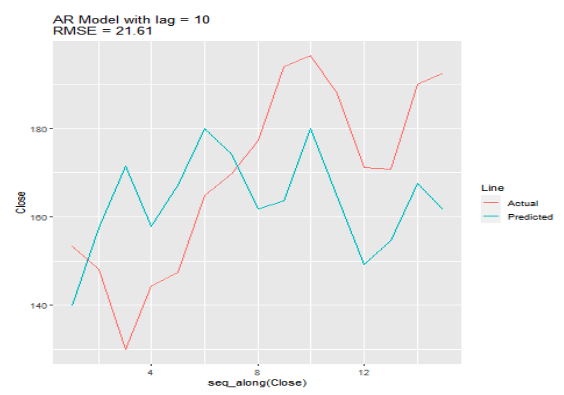
In order for the model to accurately capture the temporal structure of the data, it is essential to select the optimal number of lag terms. An appropriate number of lag terms balances the trade-off between relevant temporal dependencies and avoiding unnecessary complexity. Selecting too few lag terms might result in the overlooking of crucial patterns and lead to inaccurate predictions. On the other hand, too many lag terms can lead to overfitting and limit the predictive ability of the model.

***Results***

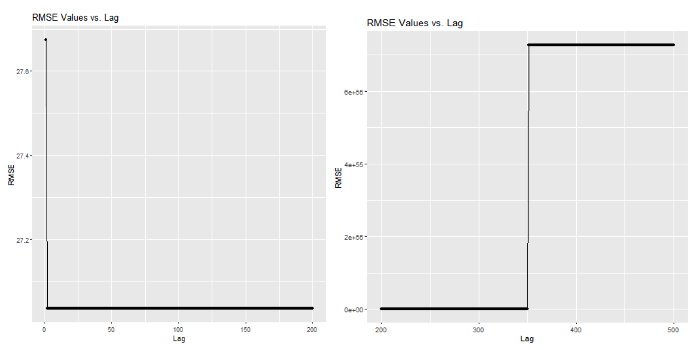
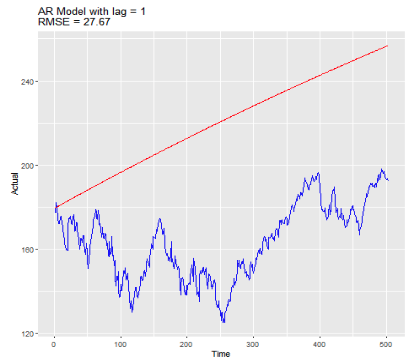
We did not initially expect any of the AR models to perform particularly well. We thought the model would have trouble understanding the time dependency and volatility of the stocks with correlation terms due to the price constantly fluctuating, particularly for the daily data. The AR model was applied to the Daily, Weekly, and Monthly AAPL stock data.

******

Initially, we split the data into training and test sets on a 70/30 split, then fit an AR model to the data and plotted the predictions from the model along with the actual values from the test set. As you can see for the Weekly data above, as the number of lag terms increased, the RMSE fluctuated and eventually spiked at 72 terms which can be attributed to overtraining. The most accurate AR model we got for the Weekly data was with a lag of 53, which gave us a predicted return of 0.002 on an actual of 0.004 and had a RMSE of 38.14. However, that model did not have the best RMSE for Weekly, which actually occurred at lag 2. The predicted return for that model was 0.0003. This phenomena highlights a limitation of AR modeling, which is sensitivity to order. Selecting the appropriate lag can be challenging and required trial and error.



You can see this effect in action when we apply the AR model on the Monthly data, shown above. The plot on the right shows the RMSE for models with 1-11 lag terms. As depicted, the RMSE dips for lags 3-4, then spikes to 40 for lag 8, then back down even lower than before at lag 10. The left shows the actual vs. predicted lines of the lag 10 model, which gave us both the lowest RMSE and best predicted return for the Monthly data of 0.0085 against the actual of 0.0189.



Moving to the Daily data as shown above, we see the AR model struggle. As previously mentioned, we thought the AR model may have issues with the Daily data given how complex the time dependency is, and our thinking proved true. The best predicted return we got for the Daily data was 0.00006, and the actual was 0.001. The plot on the right shows the RMSE spiking after 375 lag terms and the model overfitting. This model performed terribly on the test set.

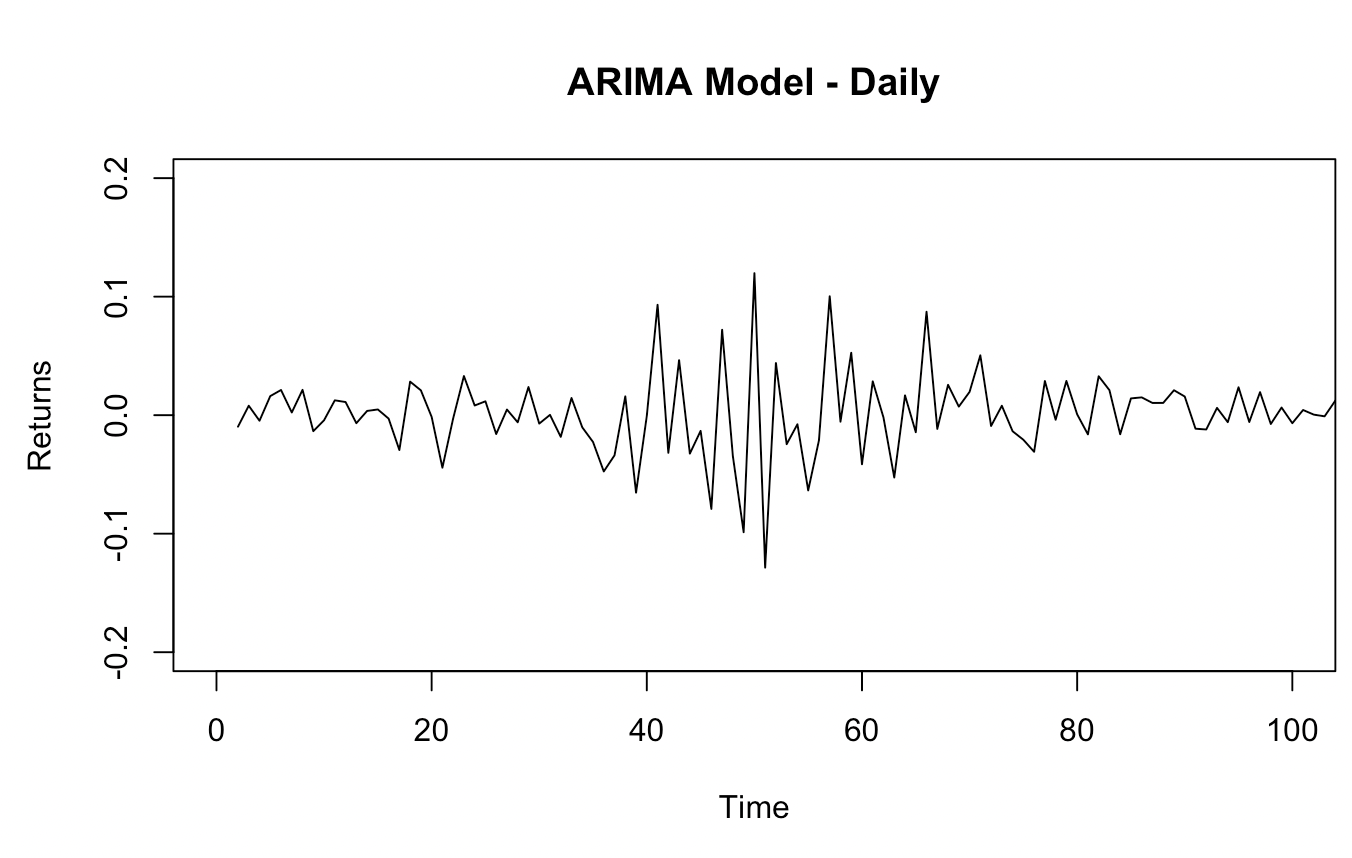
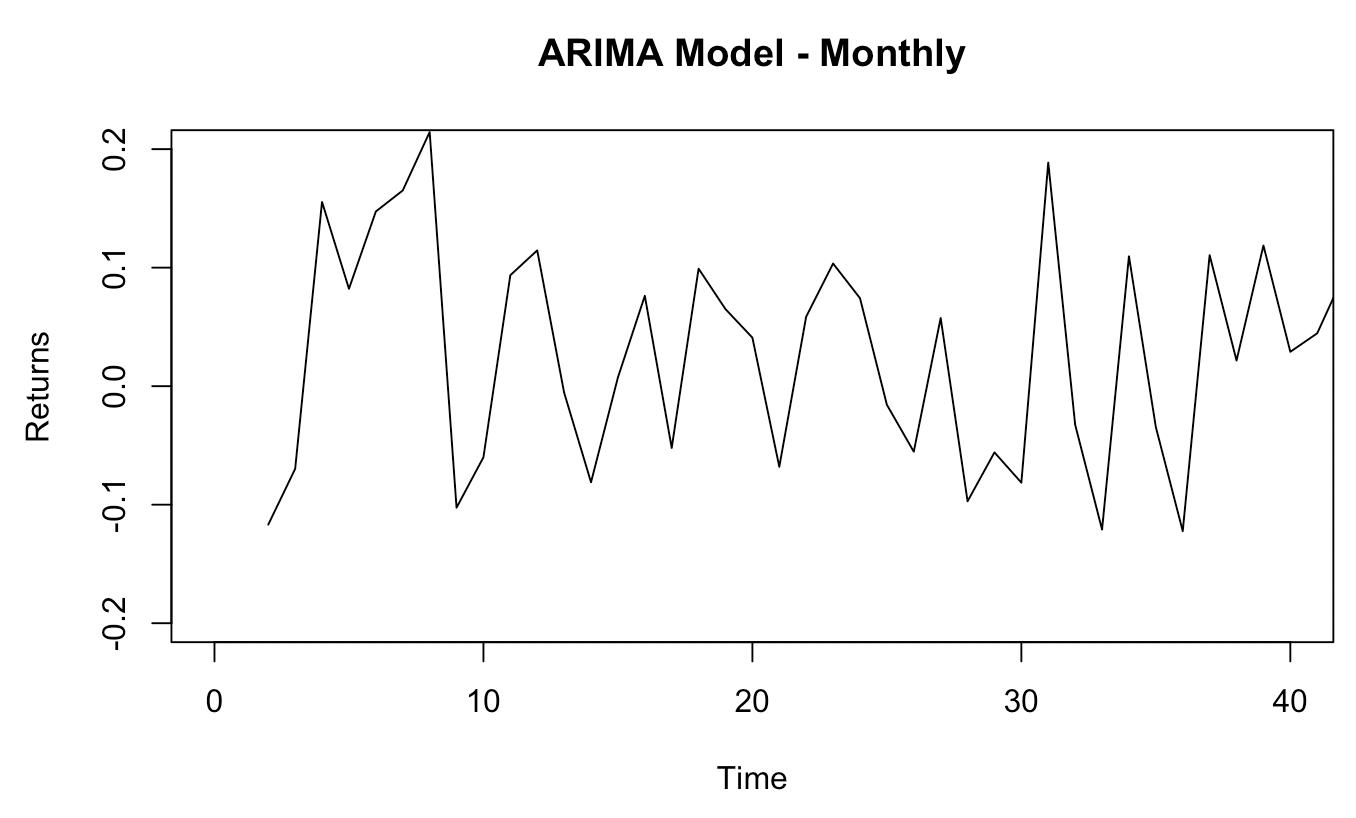
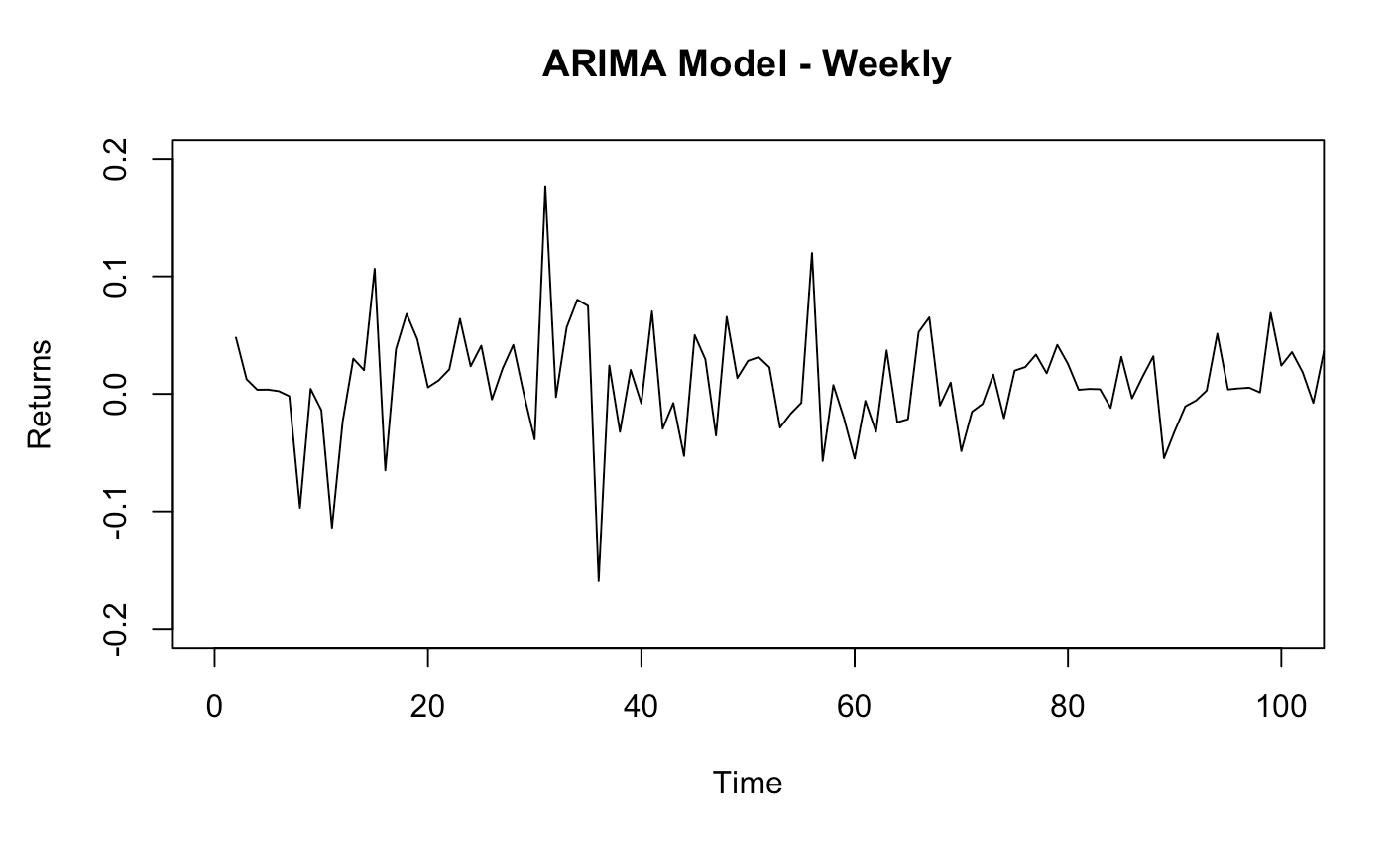
**Autoregressive Integrated Moving Average Model(Caroline Gaede)**

***Why ARIMA?***

Autoregressive Integrated Moving Average (ARIMA) models are known for their flexibility, interpretability, and robustness in modeling non-stationary data. These models make use of the autocorrelation value between time periods in the data while also incorporating the moving average and the difference between values of the stock at times of prediction. Since stock data exhibits time series characteristics like trends, seasonality, and autocorrelation, ARIMA is considered suitable for modeling these features. ARIMA models are also more comprehensive than AR models which is why we have chosen to advance to this method following the AR investigation.

***Results***

The first step in ARIMA model development was to ensure our data was a time series object, and with that, we plotted these objects to check for stationarity. To do so, we looked for randomness and a lack of obvious trends. Next, we checked autocorrelation (ACF) plots for significant autocorrelation and for decay behavior. These plots demonstrate how each observation is correlated with its lagged self. Once we checked these basic setting requirements, it was time to calculate returns using closing prices and extract those values into their own data frame. This step is required because the built-in functions for ARIMA require univariate time series data. Next, we could implement either the *arima()* or *auto.arima()* function, of which we chose the latter since it automatically selects optimal parameters. At this point, it is time to actually create the model and plot the projected predictions. Finally, to interpret the performance of our model, we calculated and obtained accuracy measures and conducted cross validation to depict errors.



Since these results for the Daily, Weekly, and Monthly data frames only plot the predicted returns, we need to investigate error values to see how accurate they really are. The first table below lists our manual calculations of Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) for cross-validation that was performed on a 70/30 split of our data into training and testing sets. The second table represents the automatic output of training set errors as given by the *auto.arima()* function.

|  | **Daily** | **Weekly** | **Monthly** |
| --- | --- | --- | --- |
| **MAE** | 0.01495482 | 0.03129953 | 0.07893488 |
| **RMSE** | 0.02113729 | 0.04203728 | 0.09221766 |

| **MAE** | 0.01494558 | 0.03108545 | 0.07723515 |
| --- | --- | --- | --- |
| **RMSE** | 0.02095271 | 0.04182298 | 0.08944994 |

Since the values in each table are roughly equivalent, it is safe to assume that our manual performance of cross-validation was accurate to that inherently performed by the built-in function. Furthermore, the errors themselves, which are all less than 0.1, are statistically small enough to consider the ARIMA model overall accurate in predicting returns.

A future endeavor to pursue could include the exploration of the *arima()* function and manually tuning parameters to observe their impact on model accuracy. With more time, this could have been an interesting addition to the ARIMA investigation. However, our results using the *auto.arima()* function resulted in a high-performing model with small error values, and thus it will suffice for this analysis.

**Long Short-Term Memory Neural Networks(Elvis Atitsoabui)**

***Why LSTM?***

First, we decided to fit the LSTM model to our due to its ability to pick up on long-term time dependencies. These very crucial facts are needed in determining and predicting stock prices because of the huge amount of volatility involved. The ability of the LSTM to pick up on these time dependencies is due to the fact that it uses various gates to regulate and delete information. This helps to stop the vanishing gradient problem of the RNN and keep accurate predictions and make sure the model does not overtrain. This feature makes the LSTM very adept at understanding complex long term time dependencies.

What we were expecting from our results of fitting the LSTM was the best fit of the data compared to the previous models. We had expected that the LSTM would yield the lowest RMSE and produce visuals that were extremely close to the actual true closing and average return values that we were predicting.

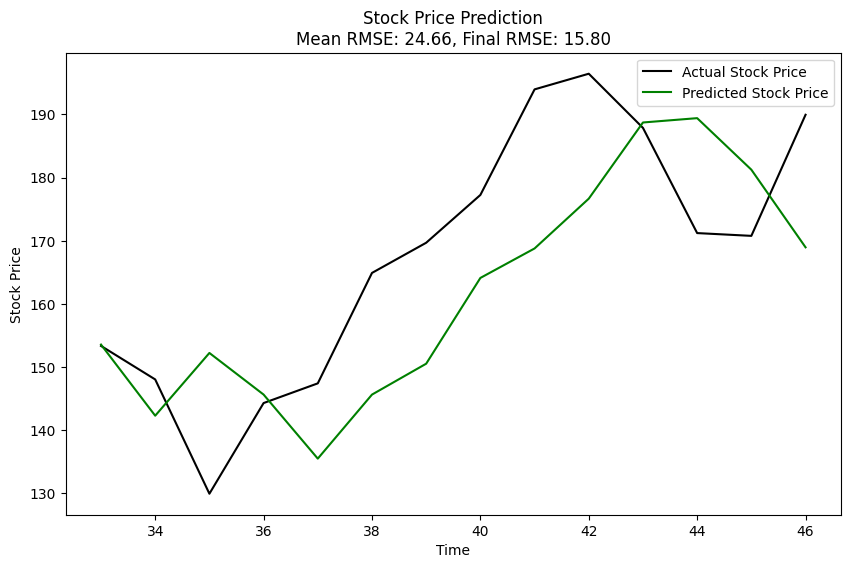
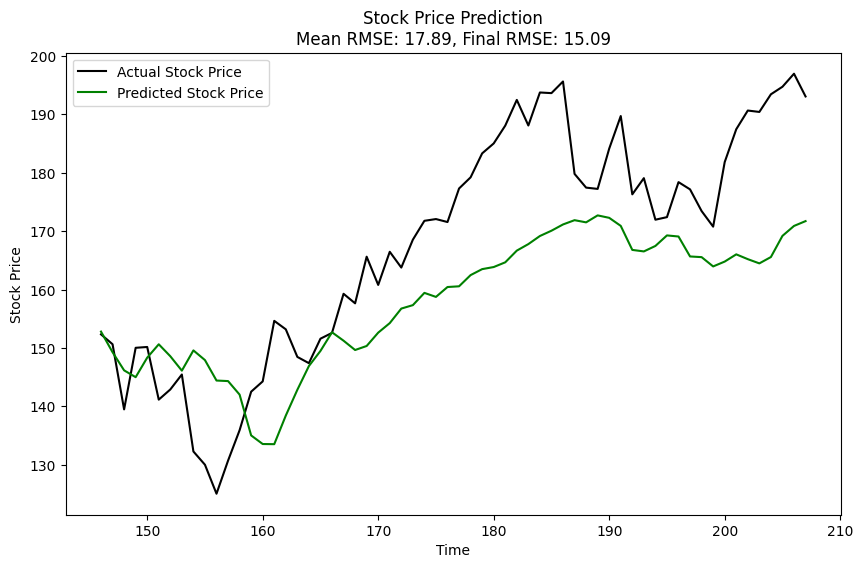
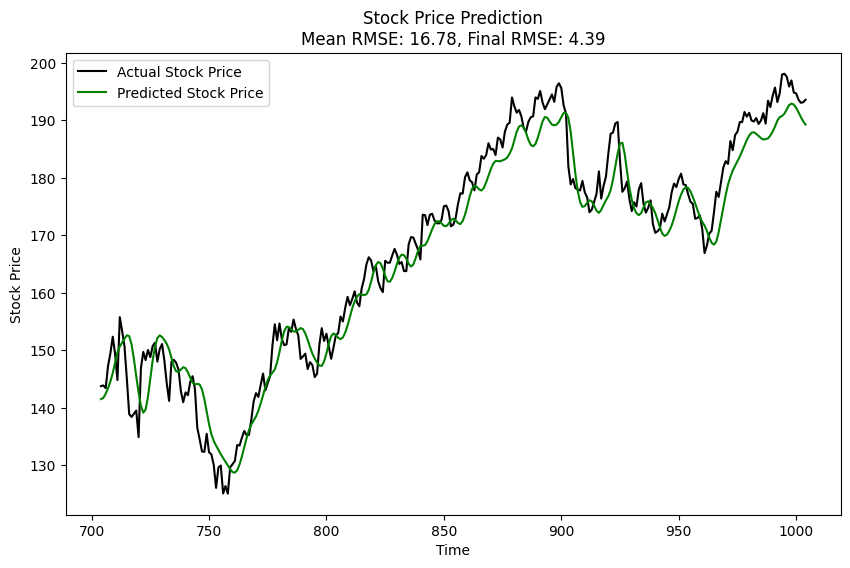
***Results***

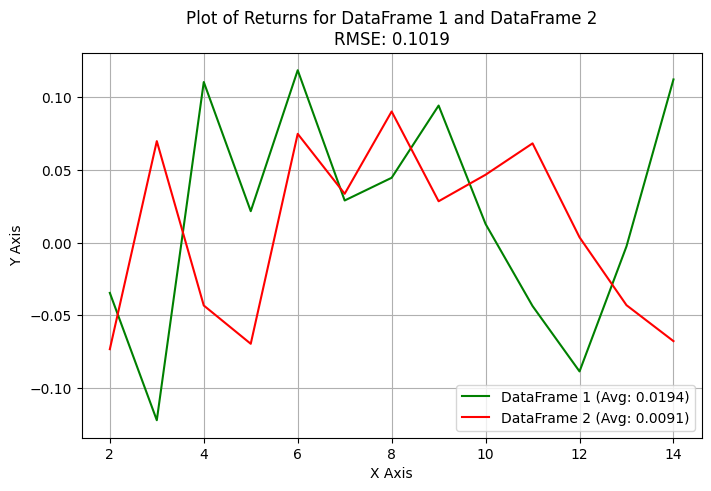
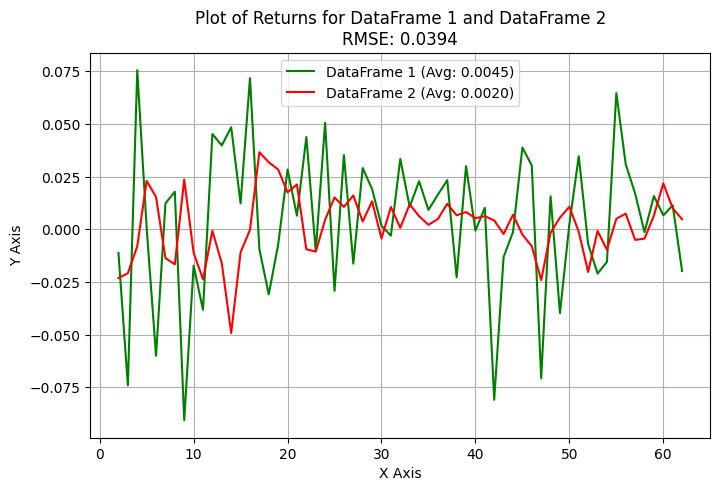
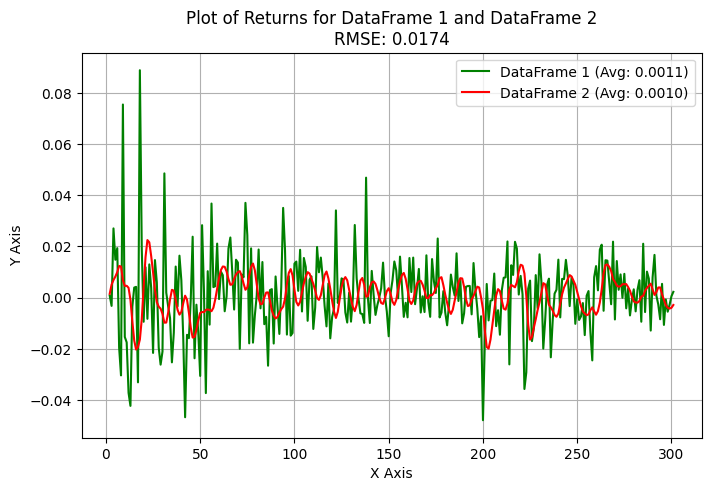
For the Daily dataset, we obtained an RMSE of 4.39 on the prediction of our closing price and a return rate prediction of 0.10 when the actual rate was approximately 0.11. These results were the best for the LSTM model in terms of the other datasets. This makes sense given that the Daily set had 704 observations giving the model more to learn from. After trying multiple parameters, the one that gave us the best results as described were 4 hidden layers, 50 units per layer, 50 time-steps, and 100 epochs.

For the Weekly dataset, the RMSE for the closing prices shot up to 15.09 and the RMSE for the average return rate was 0.0394. The actual rate was 0.0045 while the predicted was 0.0020. The increase in error is due to the fact that we had a decrease in available observations for the model to train on, so the test set had a decrease in accuracy. Despite this, our model was able to follow the general direction of the stock, which is still very valuable as predicting points of increase and decrease can aid in stock prediction and investment. The parameters for the Weekly that gave the best predictions were 5 hidden layers, 50 units per layer, 50 time-steps, and 100 epochs.

Our results for the Monthly dataset were similar to the Weekly results. The RMSE for closing was 15.8 and the RMSE for the return was 0.1019. The predicted average return was 0.0091 and the actual was 0.0194. Again, this increase in error is attributed to the fact that we had less observations in our training set than the Daily. Still, the ability of the model to predict general uptrends and downtrends of the stock is still very beneficial to traders and investors when trying to determine an optimal investment method. This model was trained using 5 hidden layers, 50 units per layer, and 2 time-steps.

The visuals below provide a better understanding of the actual trend of the stock and our predictions. DataFrame 1 represents the actual return rate and DataFrame 2 represents the predicted return rate. The graphs go from Daily to Weekly to Monthly as you advance from left to right.





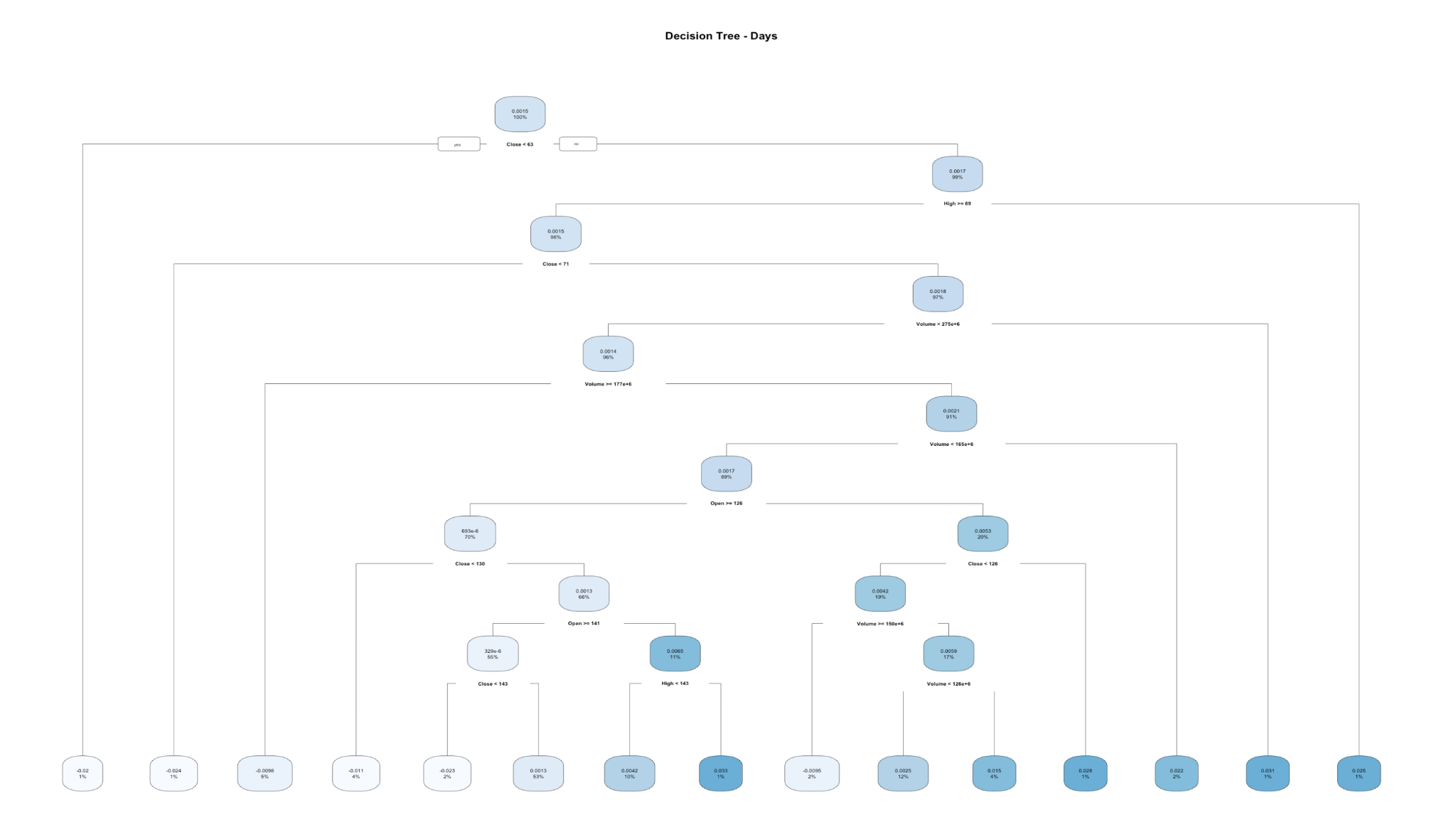
**Tree Model(Audrey Bristol)**

***Why Tree Models?***

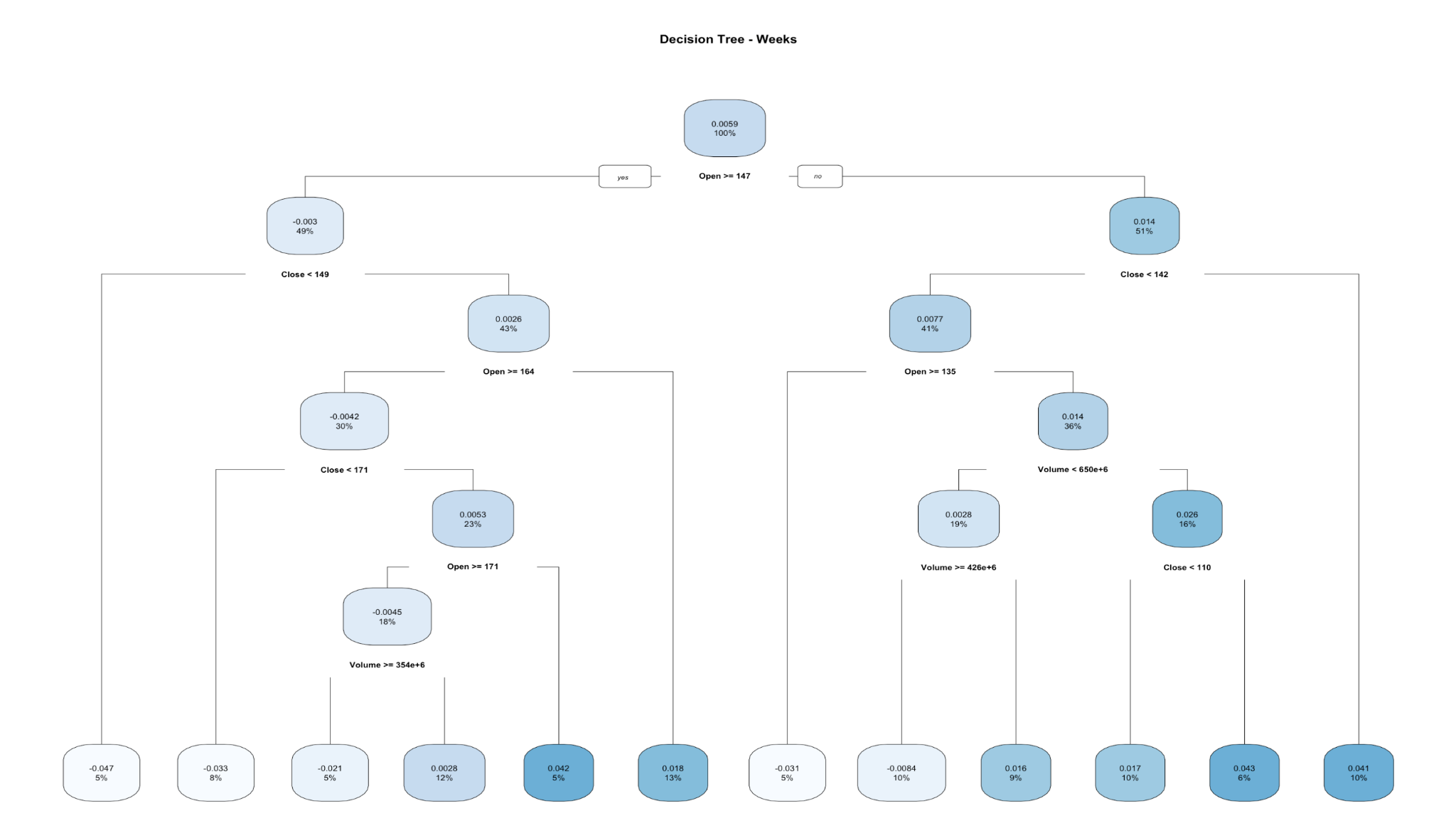
We chose a tree model due to its widespread use in stock data analysis. This model is particularly effective in accounting for multicollinearity and complex interactions of variables. Given that we initially started with only six numerical variables (excluding date and return), we didn’t have to worry too much about overfitting. The simplicity and interpretability of this model are key advantages, as we can easily identify which features are most important in making our predictions.

***Results***

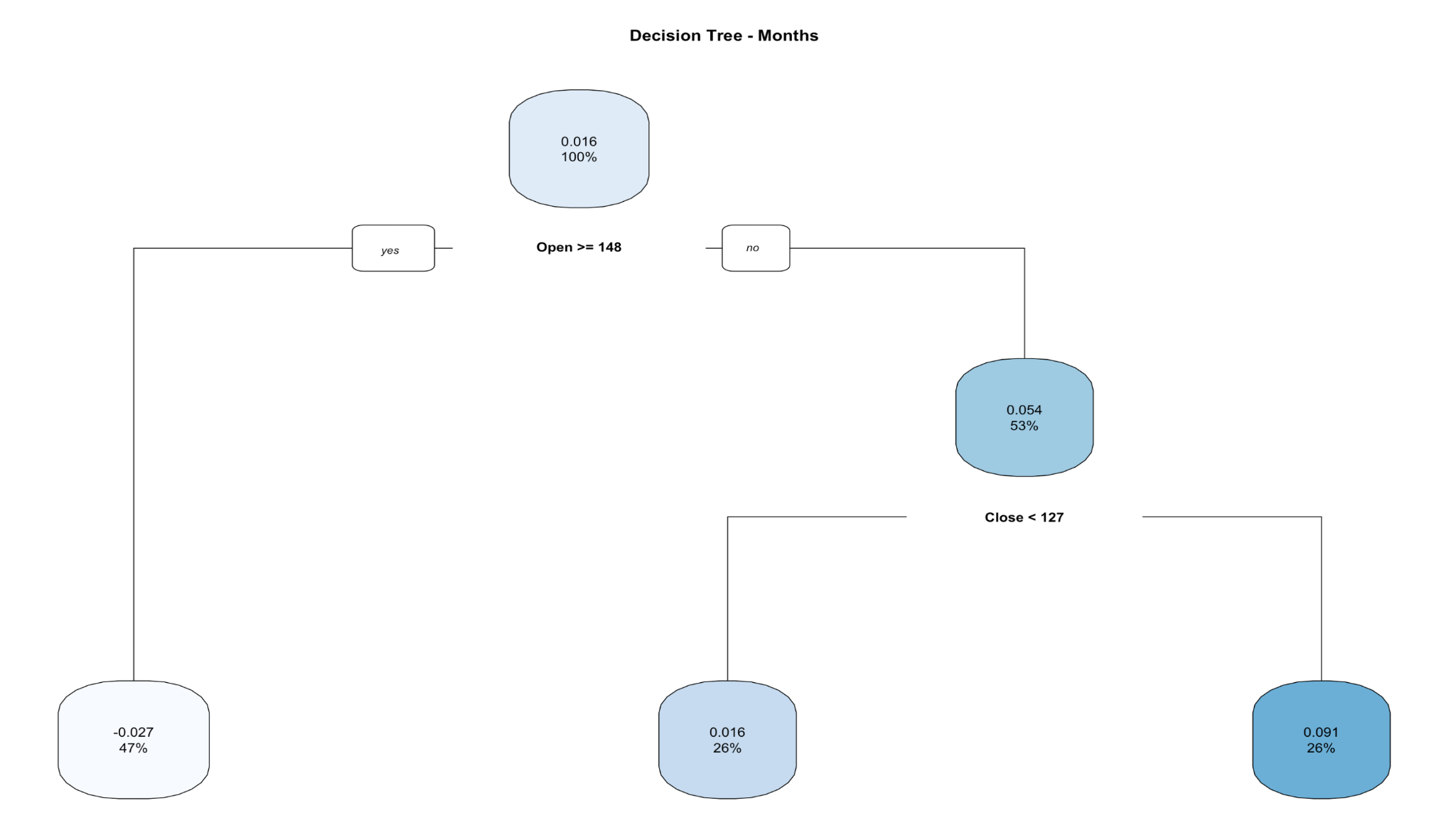
The decision tree model works by splitting the data into subsets based on certain conditions that are determined by the features in the dataset. In our case, the features are ‘Open’, ‘High’, ‘Low’, ‘Close’, and ‘Volume’. The model starts at the root of the tree and makes a decision (split) that results in the largest reduction in variance of the target variable (‘Return’). This process is repeated recursively, resulting in a tree where each leaf node represents a prediction.

We expected the model built on the daily data to make the most accurate predictions, as was true with the previous models. We also expected to see some variables, such as ‘Close’ and ‘Volume’, having more influence on the predictions due to their direct relationship with stock returns. The decision tree model was applied to the Daily, Weekly and Monthly stock data. 

The daily data yielded the best results, with a MSE of 0.0005723163, and the difference between the Predicted Average Return Rate and Actual Average Return Rate was 0.0003849456. This is likely due to the fact that daily data provides the most observations, allowing the model to capture more detailed patterns and fluctuations in the stock returns. The higher frequency of data points allows for a more granular view of the stock’s behavior, which can enhance the predictive power of the model.



The performance on the weekly data was slightly lower than the daily data, with a MSE of 0.0016583531, and the difference between the Predicted Average Return Rate and Actual Average Return Rate was 0.0012082941. While the weekly data has fewer observations than the daily data, it still provides a reasonable amount of data for the model to learn from. This tree is thus slightly less complex than that of the tree for the daily data, but still relatively complex. The weekly data can help smooth out some of the day-to-day volatility and highlight longer-term trends, which can be beneficial for predicting future returns.



The model’s performance was the lowest on the monthly data, with a MSE of 0.0078802762, and the difference between the Predicted Average Return Rate and Actual Average Return Rate was 0.0019289857. The lower frequency of the monthly data means there are fewer data points for the model to learn from, which can limit its ability to capture the complexities of the stock market, which can be seen in the lack of branches on this decision tree. However, monthly data can still provide valuable insights into longer-term trends and cycles in the stock returns.

In all three of these models, the ‘Open’ and ‘Close’ features had the most influence on the predictions, which is what our expectation was.

While the decision tree model provided valuable insights, there’s room for improvement. We could consider ensemble methods like Random Forest or Gradient Boosting, which often yield better results by combining multiple decision trees.

**III: Conclusion**

|  | **Daily RMSE** | **Weekly RMSE** | **Monthly RMSE** |
| --- | --- | --- | --- |
| **AR** | 0.0165 | 0.0633 | 0.1028 |
| **ARIMA** | 0.0211 | 0.0420 | 0.0922 |
| **LSTM** | 0.0174 | 0.0394 | 0.1019 |
| **Tree Model** | 0.0239 | 0.0407 | 0.0888 |

In general, Daily data proved to be the best for our models. This makes sense as it is the most collected data, and the time between each object allows for more subtle variations. The best RMSE was achieved by the Long Short-Term Memory (LSTM) model for Daily data. However, the Autoregressive Integrated Moving Average (ARIMA) and Tree models were close competitors and could be used interchangeably.

In conclusion, if you aim to predict returns for stock data, any of these three models could be applied to your daily data. The choice depends on the complexity of the model you wish to create, with LSTM being the most complex and the Tree model being the least complex. This decision should be based upon your real-life constraints.

Predicting returns, or any other value, is beneficial because the better you can predict stock behavior, the more money you can potentially make in the stock market. However, it’s important to remember that while these models can provide valuable insights, they should not be used as the sole decision-making tool for stock trading due to the inherent risks and uncertainties in the stock market.