**URL TO GITHUB REPOSITORY:** <https://github.com/JWVivs/GDAT626>

**Section I. Introduction**

Although it is difficult, being able to predict future stock prices is an extremely valuable tool that can help guide individuals towards profitable stock purchases, and to gauge what their current stocks’ values will be. With this in mind, I decided to retrieve historical data on McDonald’s stock prices from 12/7/2014 to 12/7/2019 in an attempt to create a model capable of predicting future stock prices. Assuming I can get anything meaningful from this analysis (e.g. model that shows accurate or at the least promising prediction capabilities), then it would serve as a very useful model going forward that may provide further insight on how a predictive model should be built in order to accurately assess stock prices. The dataset is multivariate; however, I will be focusing on being able to assess the closing stock price, making this a univariate time-series analysis. The reason I am focusing on the closing stock price is because it is generally regarded as the most important price when analyzing how a stock performed during a trading day.

**Section II. Exploratory Data Analysis**

The dataset consisted of the date, the open, high, low, close, and adjusted prices, along with volume. As mentioned earlier, I wanted to assess the closing stock prices, which means that I will have to filter my dataset to feature only the closing price and date.

A screenshot of a social media post

Description automatically generated

**Figure 1.** Daily Closing Prices of McDonald’s Stock

After converting the date to the proper format, I needed to normalize the stock values. The reason for this is so that the stock prices from five years ago are on a common scale with the stock prices from the current year (2019). By having these values fall between 0 and 1, they become comparable. The normalized closing stock prices and the date were added to a new data frame (mcd\_norm). Figure 1 features the daily closing prices before normalization for the sake of data visualization and seeing if there is any notable trend. This plot hints towards potential stationarity, but it is not certain, so it was something to keep in mind as I progressed.

Following normalization, I wanted to make a new column in my normalized dataset that has a weekly moving average for the closing stock prices.

A close up of a map

Description automatically generated**Figure 2.** Daily Closing Stock Prices Versus Weekly Moving Average

Figure 2 features the newly created, normalized weekly moving average compared to the normalized daily closing stock prices. It is evident from the plot that the weekly moving average is smoother than the daily closing stock prices. This is beneficial in that it allows me to easier spot a trend in the data; however, I had to be careful not to over smooth the data as that will sacrifice my eventual model’s accuracy. The exploratory data analysis conducted allowed me to better determine what model should be created in order to predict stock prices by observing the stationarity. Stationarity indicates that the mean and variance is constant over time, which is vital to an ARIMA model. In this case, it seemed that an ARIMA of the moving average would be a possible option to predict future stock prices, so I invested my time in to building a proper ARIMA model.

**Section III. Decomposing the Data**

Now that I have a general direction of what kind of model I want to build, I need to decompose the data and verify that the direction I am headed is justified. I built a weekly moving average featuring a frequency of 252, as the stock market is typically open 252 days each year. I decomposed that weekly moving average and removed the seasonality from the data.

A screenshot of a social media post

Description automatically generated

**Figure 3.** Decomposition of the Weekly Moving Average

Figure 3 shows the decomposition plot of the weekly moving average. There appears to be a steady, positive trend, but there also appears to be a daily seasonal pattern. It is important to remove the seasonality so we can better understand the underlying trend that may be hiding in the data. By doing so, we eradicate noise that may have been masking a trend we would not have seen prior to removing seasonality, which in turn smooths the data. I saved the non-seasonal weekly moving average data into a vector named ‘deseasonal\_cnt’, which will be used when the ARIMA model is being built.

A screenshot of a cell phone

Description automatically generated

**Figure 4.**  Autocorrelation Plot of the Seasonal Weekly Moving Average

The autocorrelation plot in Figure 4 is part of the reason that I was not completely sold on the data being stationary. The lags all extend far beyond the blue lines, indicating that they are all dependent on each other. I opted to create an autocorrelation plot of the non-seasonal data (‘deseasonal\_cnt’) after I conducted differencing.

A screenshot of a cell phone

Description automatically generated

**Figure 5.** Autocorrelation Plot of the Non-Seasonal Weekly Moving Average

Figure 5 looks much more promising, as the initial lags decrease, indicating that the vectors are becoming less dependent on one another. However, I wanted to confirm that the data I now have was actually stationary. This led me to using the Augmented Dickey-Fuller test in order to screen for stationarity across the original, differenced, and weekly moving average time-series. By comparing all of these, I hoped to get some insight as to which data is actually stationary.

The Dickey-Fuller test yielded a p-value of 0.4011 for the original time-series, indicating that the null hypothesis of non-stationarity cannot be rejected. This confirmed my earlier hunch that the data might not be stationary. This was also the case on the weekly moving average, which yielded a p-value of 0.3309. This emphasizes the importance of conducting deeper analysis and testing in order to make a more accurate assumption.

A p-value of 0.01 was yielded from the Dickey-Fuller test on the differenced time-series of the normalized closing stock prices. This allows us to reject the null hypothesis of non-stationarity; therefore, the differenced time-series is stationary. This was also the case on the differenced weekly moving average (not to be confused with the non-differenced weekly moving average from earlier, which was rejected).

**Section IV. Building an ARIMA Model**

The stationary data can now be used to start building an ARIMA model. I started off by seeing what was suggested by the auto.arima function when applying it to the non-seasonal weekly moving average data. It suggested a 4,1,5 model; however, I was not entirely sold on trusting the auto ARIMA. I opted to display the residuals of the auto ARIMA in order to visualize what I was working with.

A screenshot of a cell phone

Description automatically generated

**Figure 6.** Autocorrelation and Partial Autocorrelation Plots of the Residuals of an Auto ARIMA conducted on the Non-Seasonal Weekly Moving Average

The lags seem relatively noisy (lags going outside of the blue lines) in Figure 6, which drove me towards trying to manually build an ARIMA model with much more appealing ACF and PCF plots.

A screenshot of a cell phone

Description automatically generated

**Figure 7.** Autocorrelation and Partial Autocorrelation Plots of the Residuals of 4,1,5 ARIMA conducted on the Non-Seasonal Weekly Moving Average

The plots in Figure 7 refer to a previous model order suggested by auto.arima, which was a 4,1,5 model. I wanted to revisit this to see what the ACF and PACF plots looked like. Compared to Figure 6, there are less lags that extend beyond the blue lines, indicating that the data we have here is potentially more significant. However, I did not stop here, as I noticed there was a pretty large line around lag 8 in the ACF plot of Figure 7. This led me to manually creating an 8th order moving average model.

A screenshot of a cell phone

Description automatically generated

**Figure 8.** Autocorrelation and Partial Autocorrelation Plots of the Residuals of a 2,1,8 ARIMA conducted on the Non-Seasonal Weekly Moving Average

Notice how all of the lags are within the blue lines in the ACF and PACF plots in Figure 8. These were the best looking ACF and PACF plots I was able to achieve. This led me to believe that a 2,1,8 ARIMA model was my optimal model going forward. Now that I am satisfied with the model, I can move forward in assessing its predictive capabilities.

**Section V. Analysis of the Model**

I started off by making a subset of 30 days of data from the non-seasonal weekly moving average that I wanted to predict. The subset was from 1/2/2018 to 2/13/2018. Then I created a subset of data that would be considered my training data from 1/3/2017 to 12/29/2017. I used the ARIMA order of 2,1,8, which I believed to be my optimal model. I applied a forecast of 30 periods, since I am trying to predict the next 30 days of stock prices, and plotted the results.

A close up of a map

Description automatically generated**Figure 9.** Forecast of a 2,1,8 ARIMA: Predicting the Closing Stock Prices from 1/2/2018 to 2/13/2018

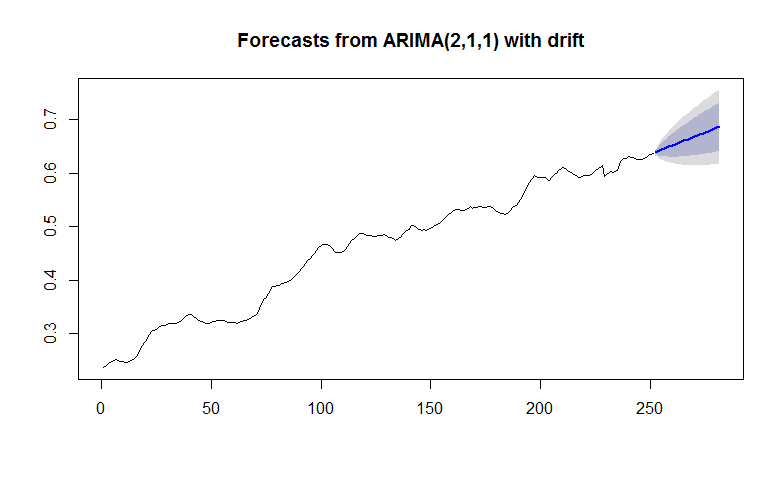
As seen from Figure 9, the ARIMA model performs poorly, as it predicts that the closing stock prices will steadily increase over the next 30 days (blue line). However, as seen by the actual plot of the next 30 days, it steadily increases only briefly before plummeting. Confidence intervals of 80% (darker region) and 95% were used.

A screenshot of a cell phone

Description automatically generated

**Figure 10.** Forecasts of Three Different ARIMAs: Predicting the Closing Stock Prices from 1/2/2018 to 2/13/2018

Figure 10 compares the 2,1,8 ARIMA that I decided to use to the other ARIMAs that were suggested by the auto.arima function. As seen in the figure, each of the models perform poorly, and seem to assume a continuous trend. The 2,1,1 ARIMA and the 2,1,8 ARIMA assume positive trends, whereas the 4,1,5 ARIMA assumes a straight, horizontal trend. A summary of the statistics on the 2,1,8 ARIMA showed a mean absolute scaled error (MASE) of 0.48, which was much higher than the MASE of the 4,1,5 ARIMA (generated through auto.arima) of 0.008. MASE is a measure of forecast accuracy, so the 2,1,8 ARIMA still appeared to be the best performing model of the ones created, though it still performed poorly in attempting to predict the closing stock prices from 1/2/2018 to 2/13/2018 (30 trading days).



**Figure 11.** Forecast of a Seasonal 2,1,1 ARIMA: Predicting the Closing Stock Prices from 1/2/2018 to 2/13/2018

I tried putting seasonality back into the 2,1,1 ARIMA in an attempt to get a more realistic prediction, considering the non-seasonal 2,1,1 ARIMA only assumed the closing stock prices would increase at a moderate, positive rate (Figure 10). However, as seen in Figure 11, it still assumes a positive, linear trend when predicting the next 30 trading days of closing stock prices.

**Section VI. Summary of Findings**

The analysis conducted provided me with ARIMA models that were not capable of accurately predicting future closing stock prices (Figure 9). Two other ARIMA models were created in an attempt to yield a prediction that better matched the actual trend of the 30 days of closing stock prices that were trying to be predicted; however, they still performed poorly (Figure 10). Even after adding seasonality back in to the 2,1,1 ARIMA model it still served as an inaccurate predictor of closing stock prices.

As expected, it was difficult to achieve a model that was capable of accurately predicting future stock prices. It’s possible that the window of time I chose to predict was problematic in that it started right after New Year’s Day. There may be certain fluctuations that occur during the time after the holiday that may be particularly difficult to predict. Perhaps trying to predict during a timeframe in the spring or summer would have produced a more accurate model. Alongside this, I may have been overzealous in attempting to predict 30 days of closing stock prices, whereas I could have limited it down to just a week, or even a few days. The problem I had with making the window to predict very small is that I did not want to fall into the trap of assuming my model is accurately predicting a few days in advance perfectly, when in reality it could be due to chance. This is a dangerous assumption in this context, as an individual could potentially lose a lot of money if they were to invest on stocks based off of a model that has a deceiving forecast plot.

**References**

McDonald’s Corporation. (MCD). Historical Data, December 7, 2019. *Yahoo! Finance*. Retrieved from <https://finance.yahoo.com/quote/MCD/history?p=MCD>