Stock Price Forecasting Project

Praj Konanur

**Table of Contents:**

Introduction ………………………………………………………………………………………………………… 3

Data …………………………………………………………………………………………………………………… 4

Methods …………………………………………………………………………………………………………….. 5

Analysis ……………………………………………………………………………………………………………… 6

Results …………………………………………………………………………………………………………….. 11

Next Steps ………………………………………………………………………………………………………… 16

**Introduction:**

As soon as the trading floor opens at 9:30 a.m., millions of stocks and commodities are traded in the market every minute. Most of these transactions and decisions are driven by data science. Businesses maximize profits by employing models that forecast future stock prices based on historical data and utilize algorithmic trading to execute these trades in real time. While data science cannot predict the future with absolute certainty, techniques such as time series analysis can go a long way in providing the key insights needed to make the right decisions.

One of the most popular and powerful models in time series analysis is the Autoregressive Integrated Moving Average (ARIMA) model. The ARIMA model is a statistical model that can predict future values based on past values and is made up of three components:

Autoregression (AR)(p) – Anticipates a time series dependence on its lagged values

Integrated (I)(d) – Represents the differencing of raw observations for the time series to become stationary

Moving Average (MA)(q) – Anticipates a time series dependence on past forecast errors

In this project, I will develop an ARIMA model utilizing a walk forward validation approach that can accurately forecast closing stock prices. I will model both the non-stationary data and the stationary data to demonstrate differences in the errors. Furthermore, I will utilize Prophet, a forecasting tool created by META that can forecast time series data into the future, to create forecasts of closing prices 1 year into the future. I have chosen five companies from a variety of economic sectors to model:

1. McKesson Corporation (MCK)
2. Verizon Communications Inc. (VZ)
3. Exxon Mobil Corporation (XOM)
4. Hilton Worldwide Holdings Inc. (HLT)
5. META Platforms Inc. (META)

**Data:**

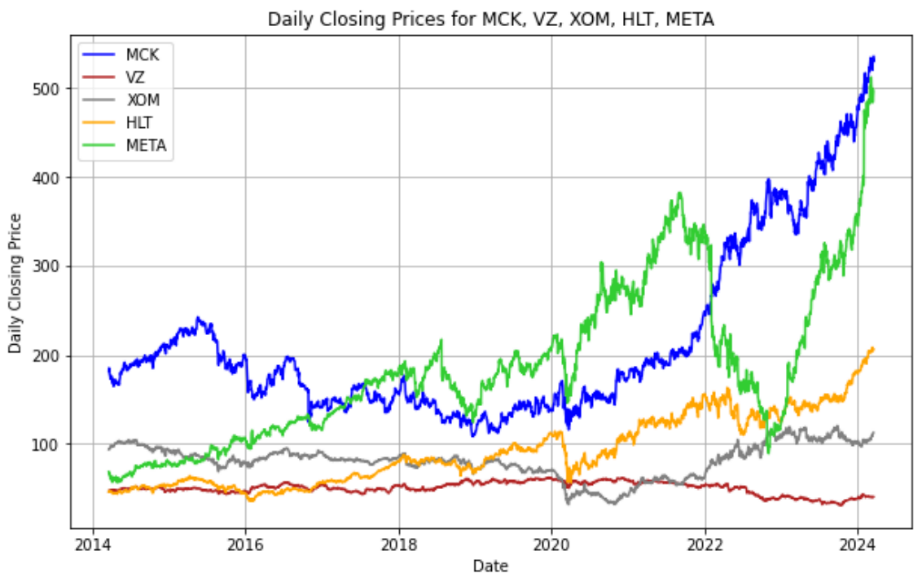
The data for the five stocks (McKesson, Verizon, Exxon Mobil, Hilton, META) was gathered from Yahoo Finance over a period of ten years (3/19/2014 – 3/19/2024). After the time frame was selected, the data was downloaded into a csv file. Each stock’s Data is organized into the following columns:

A screenshot of a graph

Description automatically generated

For the purpose of this project, the models will base predictions off the daily close price (Close), which is the price of a stock before the markets close for the day.

Daily Closing Price for Each Stock:



**Methods:**

The following tools were used for this project:

1. Python (Spyder IDE and Anaconda Prompt)
2. Microsoft Excel

The following Python libraries were used in this project:

1. Pandas – Used to clean, analyze, and manipulate datasets.
2. Numpy – Used to perform numerical operations.
3. Matplotlib – Used to create visualizations (line plots, histograms, etc.)
4. Statsmodels – Used to estimate statistical models and perform tests.
5. Sci-kit Learn – Used to run machine learning algorithms.
6. Pmdarima – Used to determine optimal order of the ARIMA model.
7. Prophet – Library created by META that provides completely automated forecasts.

Utilizing the following libraries, functions were created to execute three tasks:

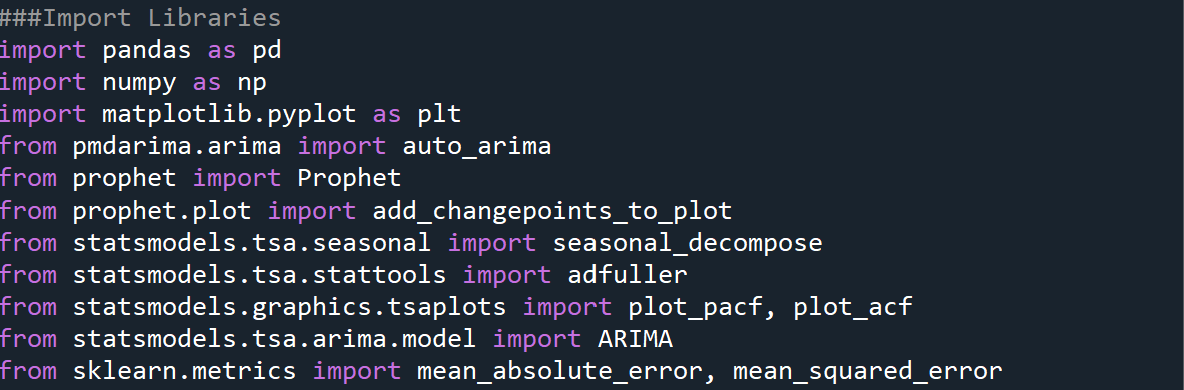
1. Identify Stationarity/Determine Model Parameters
2. Augmented Dickey-Fuller (ADF) Test – A Statistical procedure that determines if a time series is stationary. Stationarity for a time series is achieved when the adf value is negative and the p-value is less than 0.05.
3. Partial Autocorrelation Plot – Used to help validate p in order of ARIMA model.
4. Autocorrelation Plot – Used to help validate q in order of ARIMA model.
5. Auto Arima – A component of pmdarima, the function takes in a timeseries and returns the optimal (p, d, q) parameters to use in the ARIMA model.
6. Create Forecasts
7. ARIMA Model – The function takes in time series data split into train and test sets and the parameters obtained from the Auto Arima function to return predicted closing stock prices for the test set.
8. Prophet Forecast – The function takes in time series data and creates a new data frame with two columns:
9. ds – datestamp column in pandas format
10. y – stock closing price for forecasting

The function then fits the model and returns future predictions based on the period set (ex. period=365). The function also returns upper and lower bounds based on certainty.

1. Evaluate Forecasts
2. To evaluate the validity of the ARIMA model for each stock, the following metrics are calculated:
3. Residuals – The error of each result calculated by subtracting the predicted value from the true value. When the residuals are plotted in a histogram, the distribution should resemble a normal distribution.
4. Mean Absolute Error (MAE) – measure of errors between paired observations where errors are weighted the same.
5. Mean Squared Error (MSE) – calculated by averaging the square of the difference between the predicted and true value. Unlike MAE, larger errors receive a greater weight. MSE is the metric most commonly employed to quantitatively measure the efficiency of an estimate.

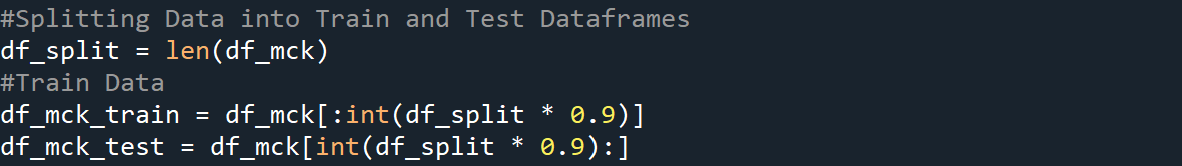
**Analysis:**

For each stock, the following procedure was conducted. For the purpose of this analysis, we will explore predicting McKesson’s closing stock price in depth.

**Step 1 – Import appropriate libraries and modules**

**Step 2 – Upload and pre-process data**

Upload the csv file and split the data into train and test sets where the train set contains data from 3/19/14 to 3/19/23 and the test set contains data from 3/20/23 to 3/19/24.



**Step 3 – Transform time series data into a stationary series**

Before stationary tests are conducted, set the date column as the index, and drop all other columns except the close price for the train and test sets.

A number and numbers on a blue background

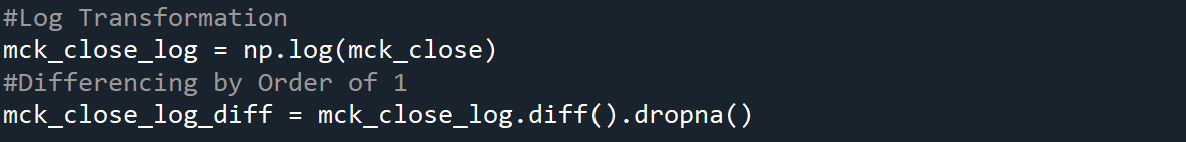
Description automatically generated

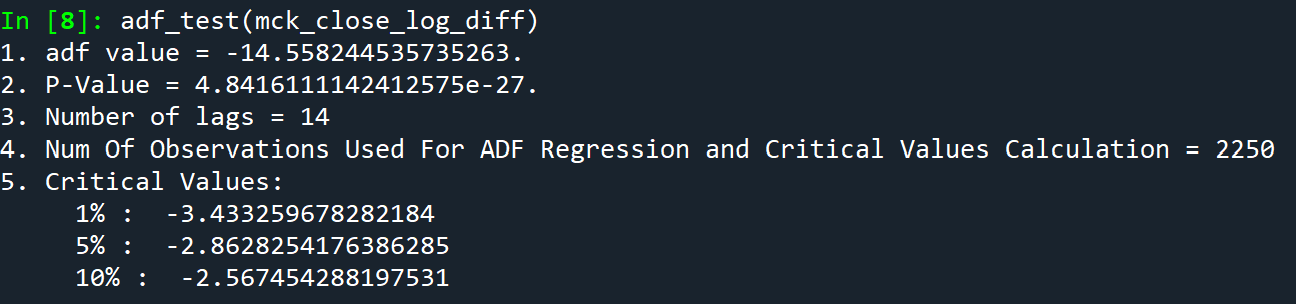
A computer screen shot of a computer error

Description automatically generatedA computer screen shot of a code

Description automatically generatedOnce the data is transformed into this format, we can test for stationarity by calling out adf\_test function and passing the train set.

The adf value is positive and the p-value > 0.05, indicating that there is trend present in the data. To achieve stationarity, we can conduct a log transformation and differencing.





The train series no longer has a trend and is stationary. Conduct the same stationary transformations on the test set.

**Step 4 – Call auto Arima function to obtain optimal model parameters**

A computer screen shot of text

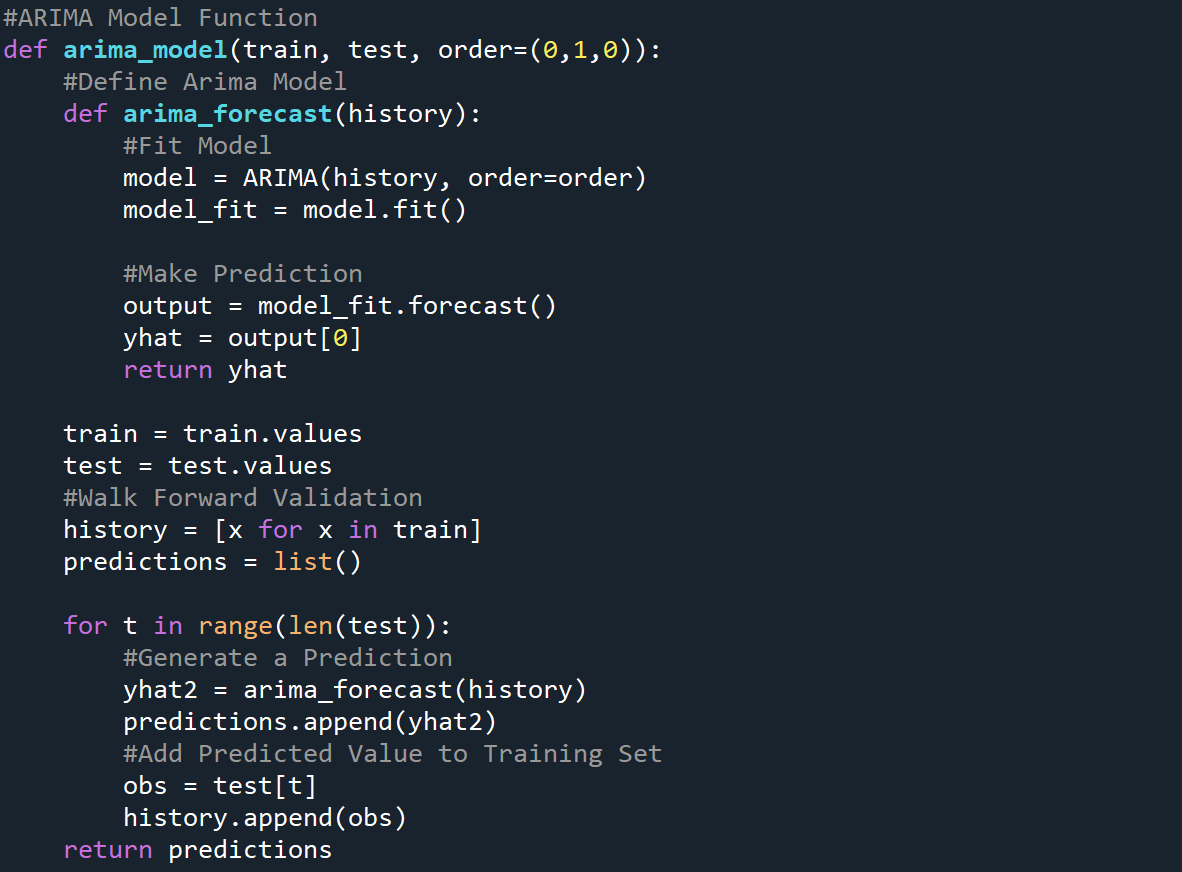
Description automatically generatedNext, we will call the auto\_arima function for our non-stationary and stationary train data to obtain optimal ARIMA model parameters

When we input out non-stationary train data, we obtain the following ARIMA model parameters:

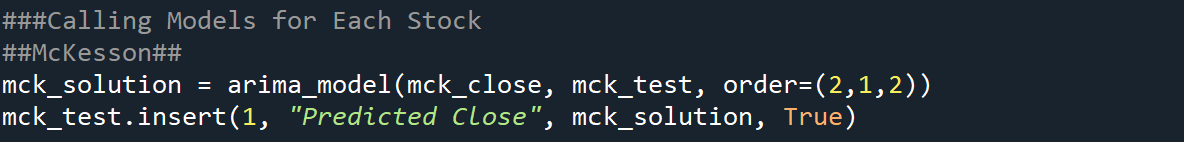
Run the same function to obtain the parameters for the stationary train data.

**Step 5 – Run ARIMA model to obtain predicted values of the test set**

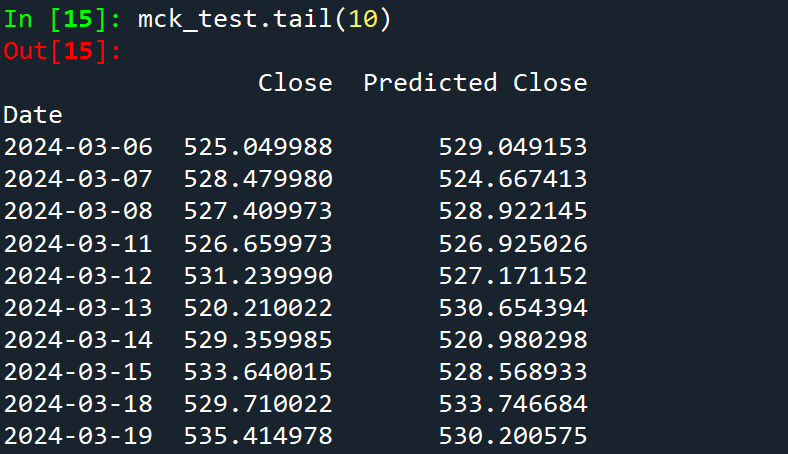
Next, we will call the arima\_model function and pass the train data, test data, and optimal ARIMA parameters. The ARIMA model follows a walk forward validation approach. In this process, the model is initially run on the train set to make a prediction on the first value of the test set. The prediction is added to a separate list while the test value is appended to the train set. This process repeats until the model creates a prediction for every test value.



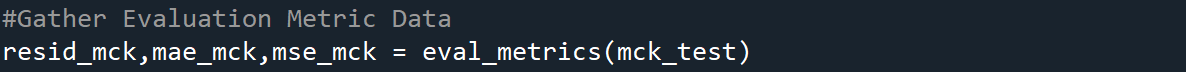
Let us call the ARIMA model on the same data we found the parameters on in step 4.



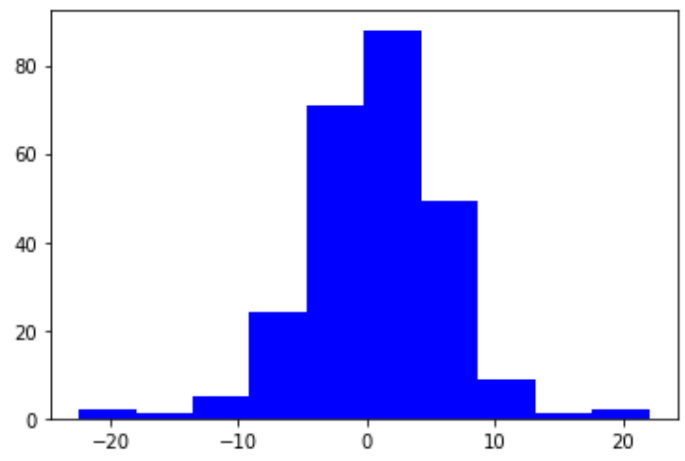
Here are the last ten test and predicted values:



**Step 6 – Evaluate model by calculating residuals, MAE, and MSE**

Next, we will check a histogram of the residuals to make sure they resemble a normal distribution and calculate the MAE and MSE.

Residual Plot:



MAE = 4.079 MSE = 30.157

**Step 7 – Run Prophet model to obtain predicted future values**

Finally, we will run the Prophet model. Created by data scientists at META, it forecasts time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality. Prophet works best with data that has several seasons of historical data and is robust to series with missing data and outliers. Input the original data uploaded from the stock’s csv file and the written function will take care of the rest.

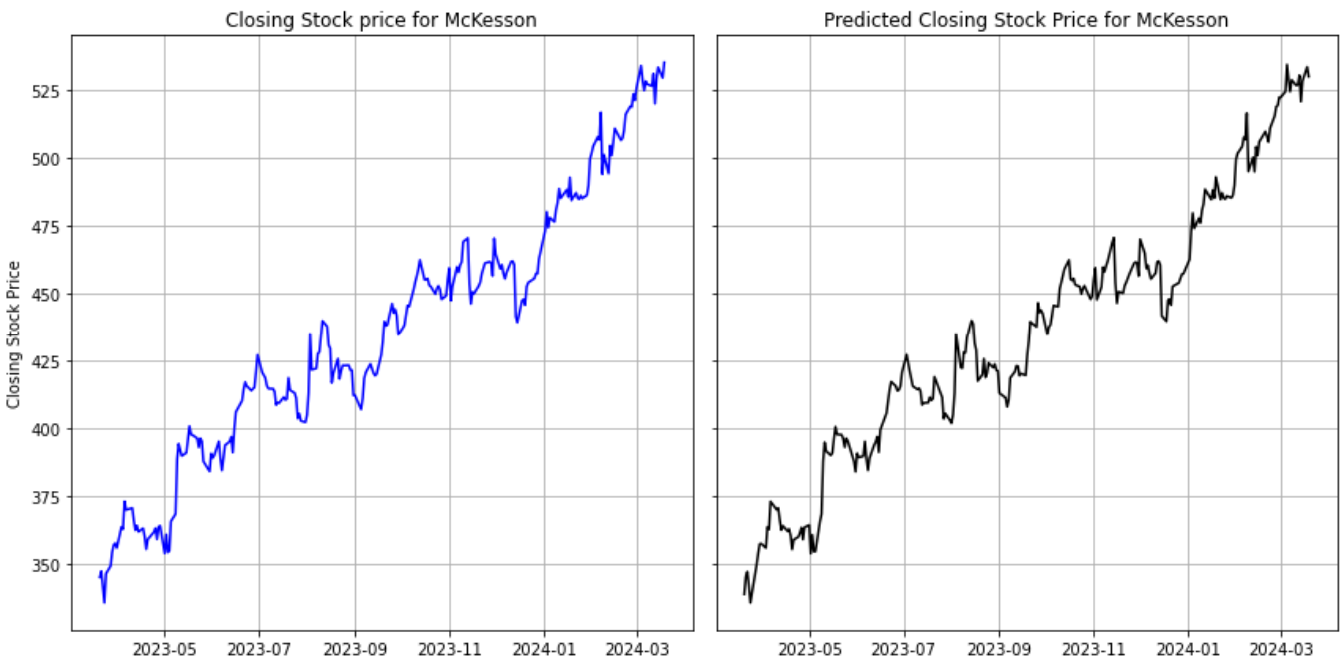
A computer screen shot of a program code

Description automatically generated

**Step 8 – Repeat Steps 2-7 for the other stocks (Verizon, Exxon Mobil, Hilton, META**)

**Results:**

Note: For daily predicted closing stock prices and all future prophet predicted values, consult the python program. The plots shown for each stock are for models run on non-stationary data.

1. McKesson

Non-Stationary: MAE = 4.0799 MSE = 30.1566

Stationary: MAE = 0.0094 MSE = 0.0016

Prophet Forecast:

A graph with lines and numbers

Description automatically generated

1. A graph of a stock market

   Description automatically generatedVerizon

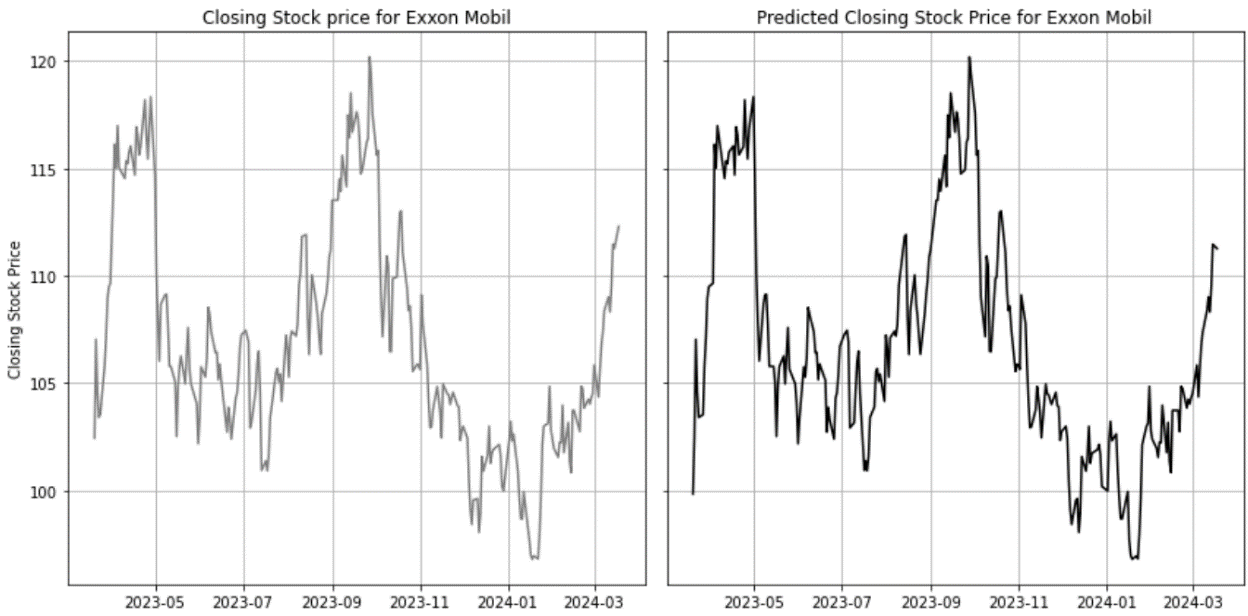
Non-Stationary: MAE = 0.3731 MSE =0.3017

Stationary: MAE = 0.0102 MSE = 0.0002

Prophet Forecast:

A graph with lines and numbers

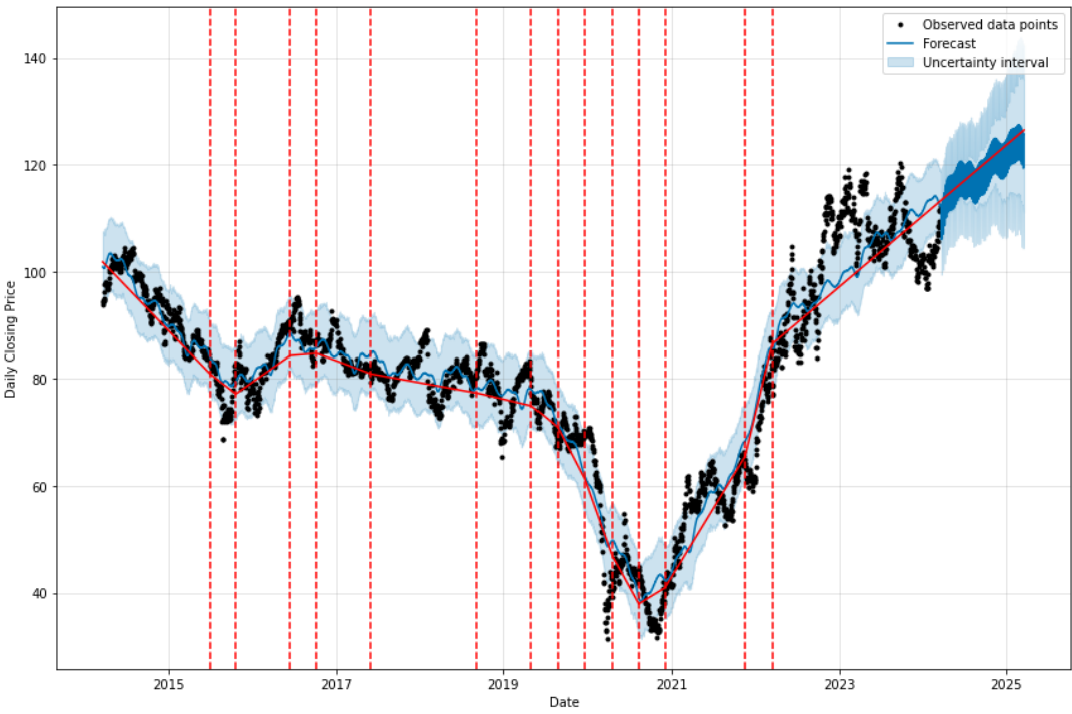
Description automatically generated

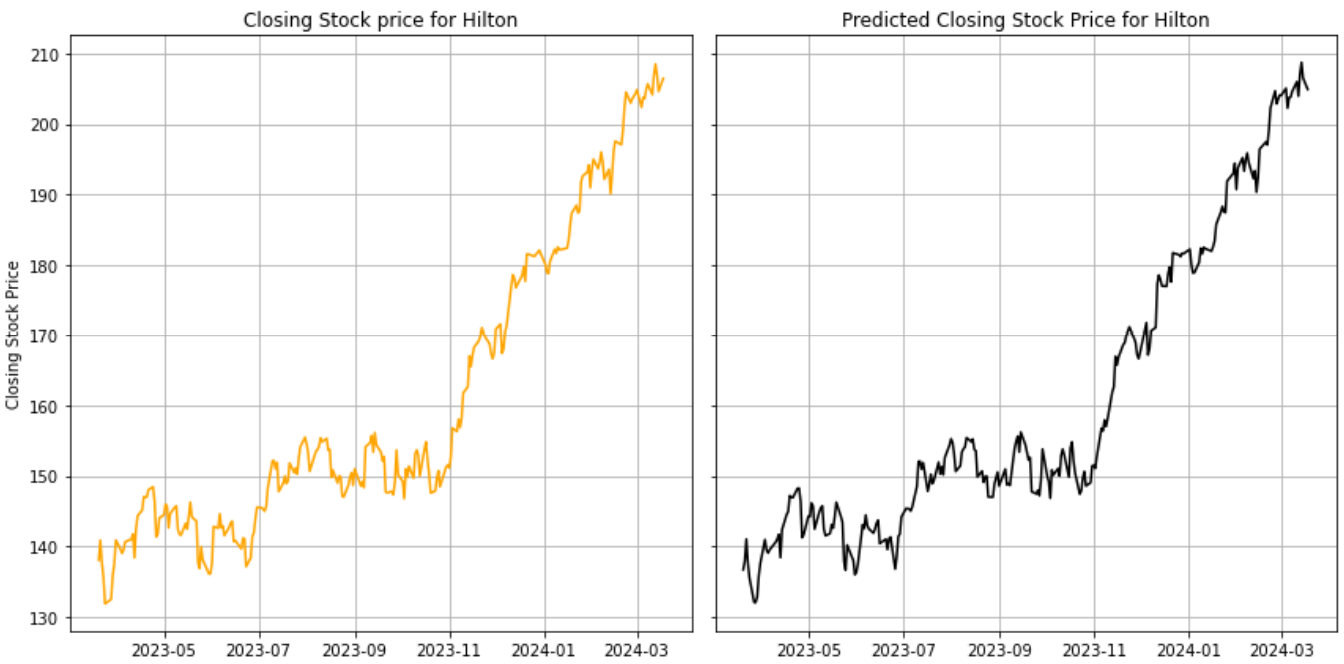
1. Exxon Mobil

Non-Stationary: MAE = 1.2213 MSE = 2.5095

Stationary: MAE = 0.0114 MSE = 0.0002

Prophet Forecast:



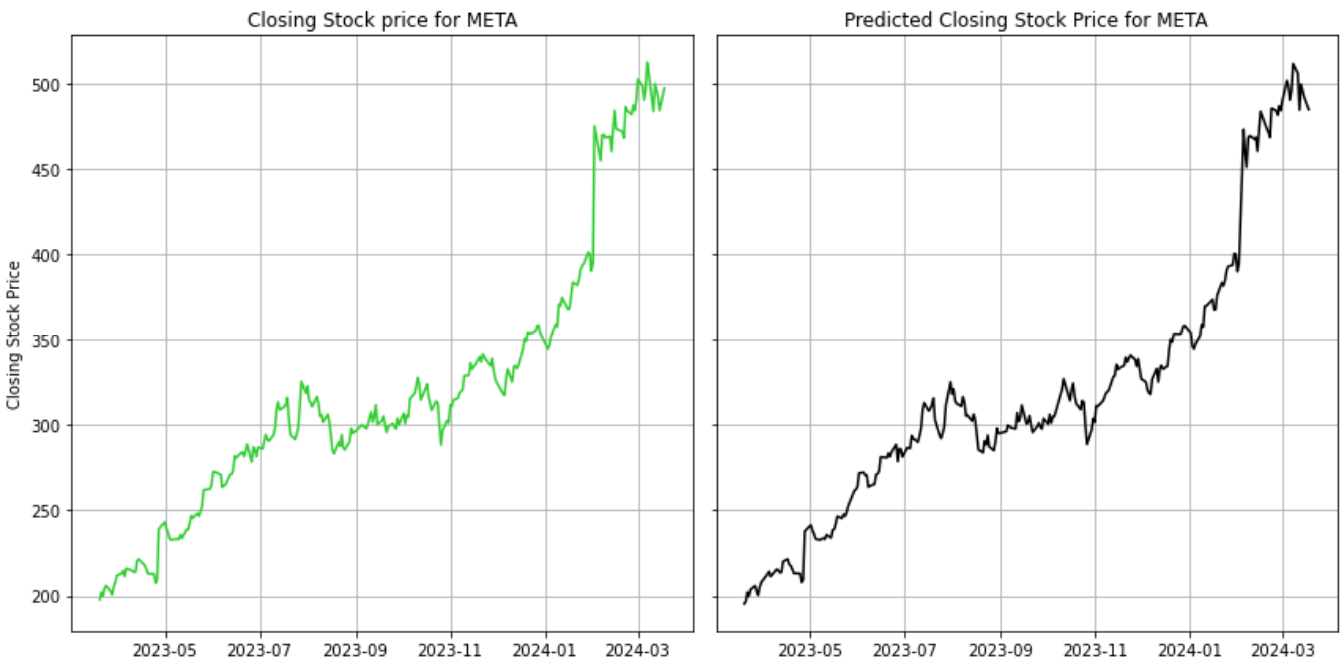
1. Hilton

Non-Stationary: MAE = 1.5697 MSE = 3.9962

Stationary: MAE = 0.0099 MSE = 0.0002

Prophet Forecast:



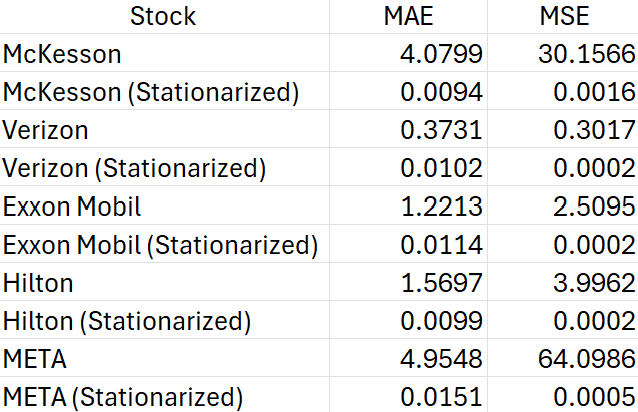
1. META

Non-Stationary: MAE = 4.9548 MSE = 64.0986

Stationary: MAE = 0.0151 MSE = 0.0005

Prophet Forecast:



MAE/MSE value for each stock

As we can see from the table, the MAE and MSE values are smaller for models run with stationary data compared to when the models are run with non-stationary data. However, the difference was often negligible. That is because we use different ARIMA parameters that fit the models better for our non-stationary and stationary data. By selecting the right parameters for the ARIMA model through the pmdarima auto\_arima module and utilizing a walk forward validation approach when modelling, we can obtain accurate closing price predictions for both non-stationary and stationary data.

The models for Verizon, Exxon Mobil, and Hilton provided much smaller MSE’s than the models for McKesson and Meta. We can see from the original closing stock price plot (Refer to Data Section), McKesson and Meta exhibited a very large variance over a ten-year period, which may have led to some inaccuracies in the model.

**Next Steps:**

There are several steps that we can take to elevate this project to the next level. First, we can make the forecast graph for each stock more interactive by utilizing plotly and dash. With these tools, we can build interactive dashboards that present the forecasts. We can also use other visual platforms such as Tableau to execute this task.

We can also enhance the project by expanding the length of the time series data. Data is available beyond the ten-year window (3/19/14-3/19/24), however, by making the time series longer, making ARIMA models becomes more computationally expensive.

Finally, we can heighten the project by setting up the time series to update in real time. ARIMA models are typically not built for long-term forecasting, so by modeling on real time data, we can develop better models. To achieve this, we can implement yfinance, a python library that offers a seamless way to retrieve stock data from Yahoo Finance.

**References:**

Data Collection:

1. <https://finance.yahoo.com/quote/MCK/history>
2. <https://finance.yahoo.com/quote/VZ/history>
3. <https://finance.yahoo.com/quote/XOM/history>
4. <https://finance.yahoo.com/quote/HLT/history>
5. <https://finance.yahoo.com/quote/META/history>