EXXONMOBIL CORPORATION (XOM) STOCK PRICE ANALYSIS AND PREDICTION

INTRODUCTION

Exxon Mobil Corporation is an American multinational oil and gas corporation headquartered in Irving, Texas.

The company explores for and produces crude oil and natural gas in the United States, Canada/South America, Europe, Africa, Asia, and Australia/Oceania. It also manufactures and markets commodity petrochemicals, including olefins, aromatics, polyethylene and polypropylene plastics, and specialty products; and transports and sells crude oil, natural gas, and petroleum products. Exxon Mobil Corporation was formerly known as Exxon Corporation and changed its name to Exxon Mobil Corporation in November 1999.

A trend of bearish technical signs has been illustrated by Exxon Mobil Corp. (XOM). We are careful as global oil prices continue to fall and production is still at a very high historical level.

In the second quarter, Exxon Mobil Corp.'s profit dropped by half on sharply lower oil and gas prices around the world, but the company's oil and gas production, which has typically declined in recent years, increased.

In this project, we analyze the ExxonMobil's stock price with respect to its competitors and predict the Exxon's stock price.

DATA DICTIONARY

The data is obtained from Yahoo finance. The dataset consists of stock market data of XOM from Nov 30, 2010 to Nov 30, 2020.

Attribute Name	Description
Date	**Date in yyyy-mm-dd**
High	Highest price at which a stock traded during a period
Low	lowest price at which a stock traded during a period
Close	Price of an individual stock before the stock closes for the day
Volume	total number of shares traded in a security over a period
Adj Close	**Adjusted closing** price amends a stock's **closing** price to reflect that stock's value after accounting for any corporate actions

Snapshot of the dataset:

Date	High	Low	Open	Close	Volume	Adj Close
2010-11-30	69.750000	68.320000	68.550003	69.559998	27524300.0	47.884735
2010-12-01	71.550003	70.379997	70.379997	71.330002	26590200.0	49.103195
2010-12-02	71.660004	70.949997	71.190002	71.480003	21274800.0	49.206451
2010-12-03	71.300003	70.889999	71.010002	71.190002	19457100.0	49.006813
2010-12-06	71.599998	71.059998	71.199997	71.309998	15985400.0	49.089413
...	...	...	...	...	...	...
2020-11-23	39.430000	37.279999	37.500000	39.360001	29367600.0	39.360001
2020-11-24	42.080002	40.220001	40.509998	41.980000	46979300.0	41.980000
2020-11-25	41.709999	40.650002	41.669998	40.810001	28178200.0	40.810001
2020-11-27	40.939999	39.869999	40.700001	40.189999	14971900.0	40.189999
2020-11-30	39.880001	37.970001	39.799999	38.130001	45614300.0	38.130001

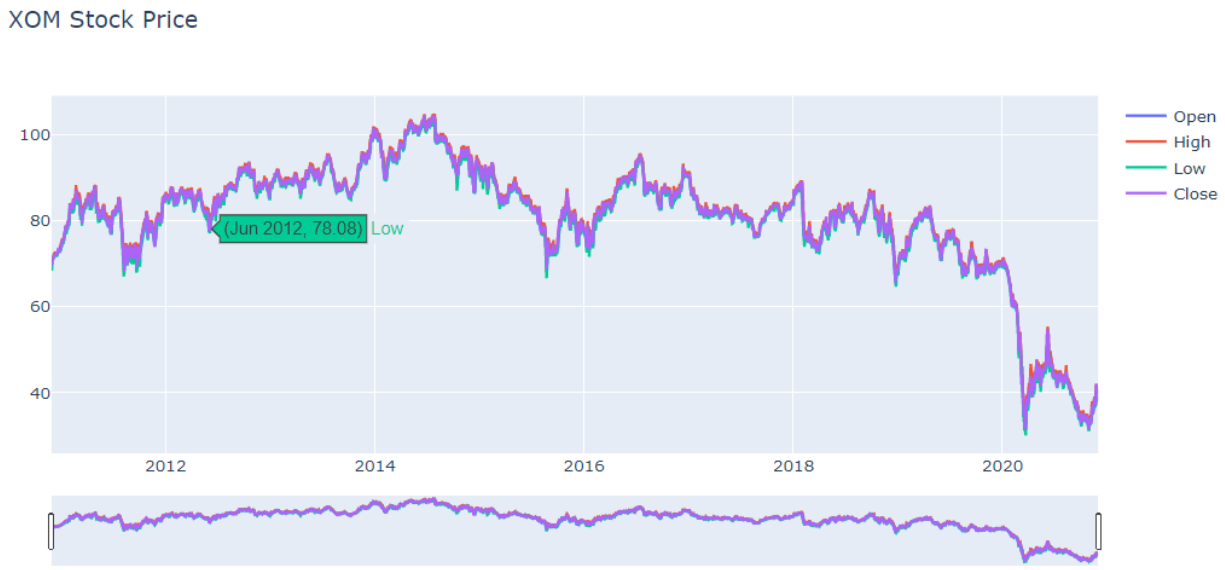
2518 rows × 6 columns

EXPLORATORY DATA ANALYSIS

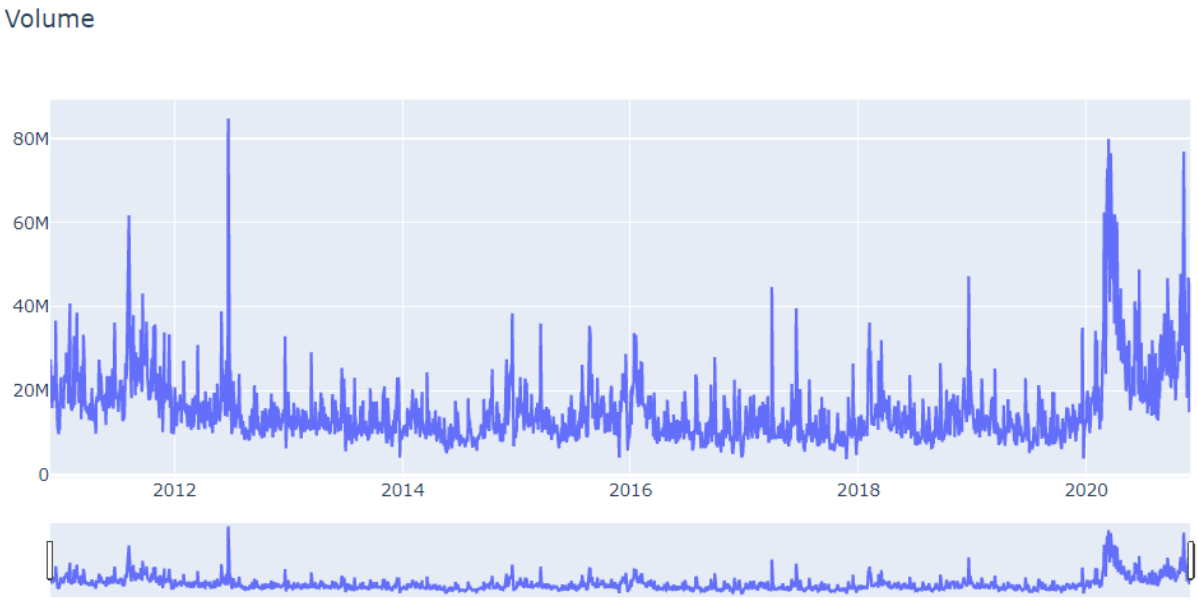
We reset the index to get the Date as a new column, then we check the for null values and data types of our attributes.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2518 entries, 0 to 2517
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype         
---  ------     --------------  -----         
 0   Date       2518 non-null   datetime64[ns]
 1   High       2518 non-null   float64       
 2   Low        2518 non-null   float64       
 3   Open       2518 non-null   float64       
 4   Close      2518 non-null   float64       
 5   Volume     2518 non-null   float64       
 6   Adj Close  2518 non-null   float64       
dtypes: datetime64[ns](1), float64(6)
memory usage: 137.8 KB
```

Let us now plot the attributes against date to see how the stock prices change over the given period.

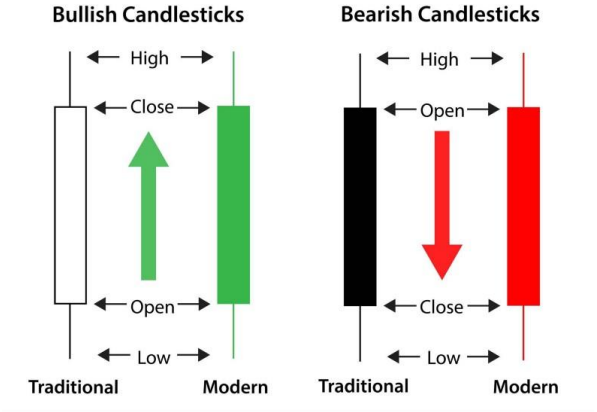


Plotting the volume of the stocks sold for the give period.



Candlesticks Interpretation

Perhaps the most common graphical way of displaying commodities price behavior in a giving time frame, candlestick graphs allows us to get a quick and intuitive perception on the stock's performance. Each candle represents a specific period of analysis and informs the opening and closing prices, as well as its highs and lows.



Plotting the closing price using scatter plot.

A given time series is thought to consist of three systematic components including level, trend, seasonality, and one non-systematic component called noise.

These components are defined as follows:

- **Level**: The average value in the series.
- **Trend**: The increasing or decreasing value in the series.
- **Seasonality**: The repeating short-term cycle in the series.
- **Noise**: The random variation in the series. First, we need to check if a series is stationary or not because time series analysis only works with stationary data.

ADF (Augmented Dickey-Fuller) Test:

The Dickey-Fuller test is one of the most popular statistical tests. It can be used to determine the presence of unit root in the series, and hence help us understand if the series is stationary or not. The null and alternate hypothesis of this test is:

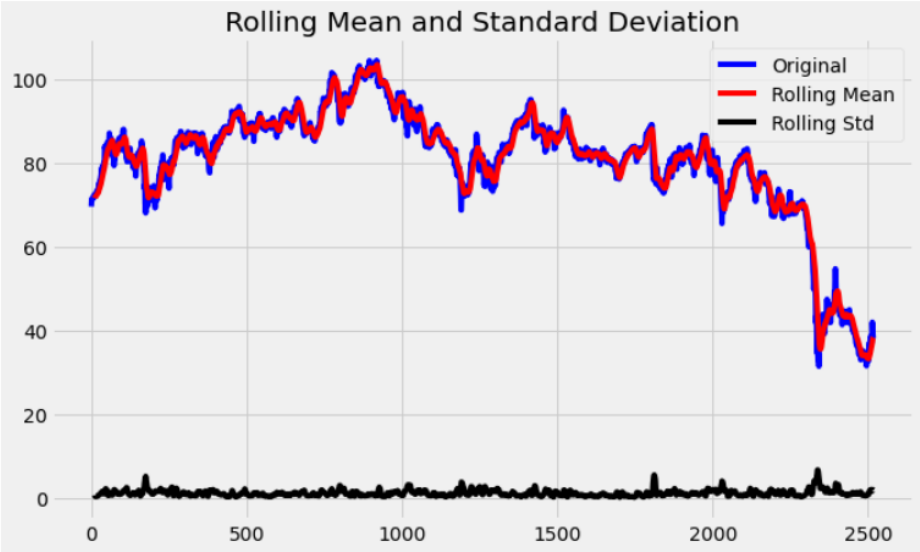
Null Hypothesis: The series has a unit root (value of a =1)

Alternate Hypothesis: The series has no unit root.

If we fail to reject the null hypothesis, we can say that the series is non-stationary. This means that the series can be linear or difference stationary.

If both mean and standard deviation are flat lines (constant mean and constant variance), the series becomes stationary.

So, let us check for stationarity:

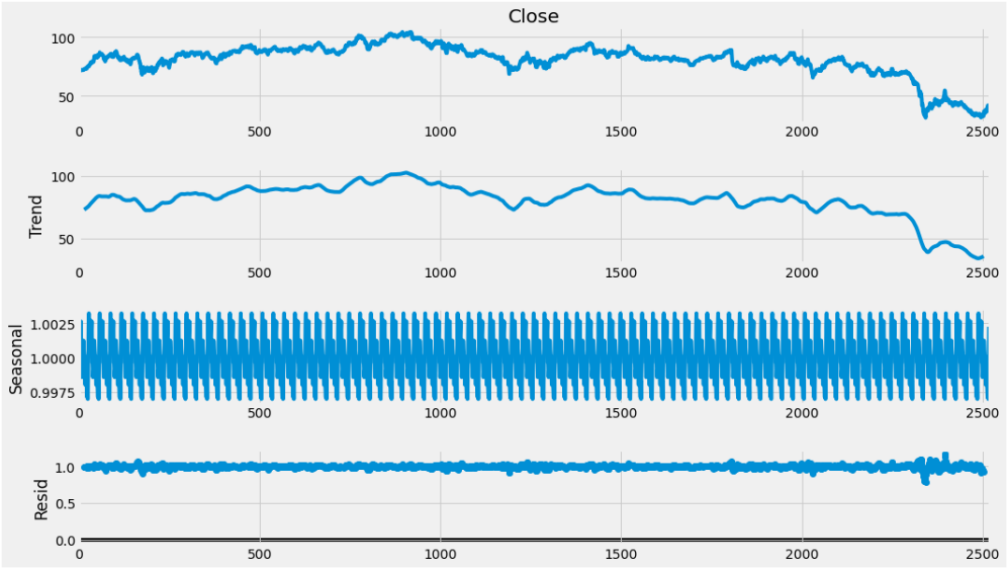



```
Results of dickey fuller test
Test Statistics                  -0.499313
p-value                           0.892151
No. of lags used                  3.000000
Number of observations used    2514.000000
critical value (1%)              -3.432954
critical value (5%)              -2.862690
critical value (10%)             -2.567382
dtype: float64
```

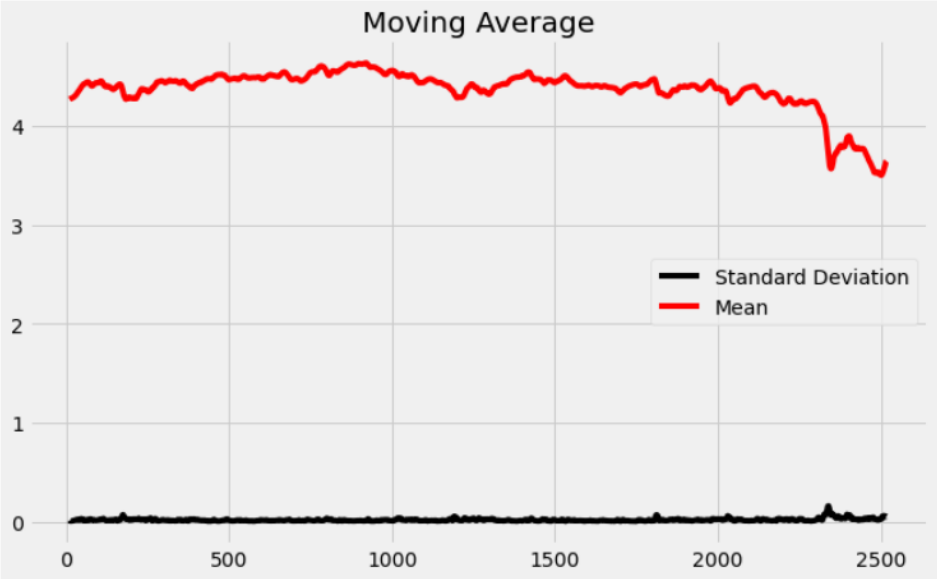
Through the above graph, we can see the **increasing mean and standard deviation** and hence our series is **not stationary.**
We see that the **p-value is greater than 0.05** so we cannot reject the Null hypothesis. Also, the test statistics is greater than the critical values. Hence, the data is non-stationary.

In order to perform a time series analysis, we may need to separate seasonality and trend from our series. The resultant series will become stationary through this process.

So, let us separate Trend and Seasonality from the time series.



We start by taking a log of the series to reduce the magnitude of the values and reduce the rising trend in the series. Then after getting the log of the series, we find the rolling average of the series. A rolling average is calculated by taking input for the past 12 months and giving a mean consumption value at every point further ahead in series.



LINEAR REGRESSION

We are creating a variable called forecast_out, to store the number of days (30 days) into the future that we want to predict. This variable will be used throughout the program so that we can simply change the number and the rest of the program will correspond accordingly.

Pros:

- Simple to implement.
- Used to predict numeric values.

Cons:

- Prone to overfitting.
- Cannot be used when the relation between independent and dependent variable are nonlinear.

We need to create a variable to predict 'x' days out into the future. Then create a new column to store the target or dependent variable. This is essentially the close price shifted 'x' days up. So, we will create a new column called 'Prediction' and populate it with data from the Close column but shifted 30 rows up to get the price.

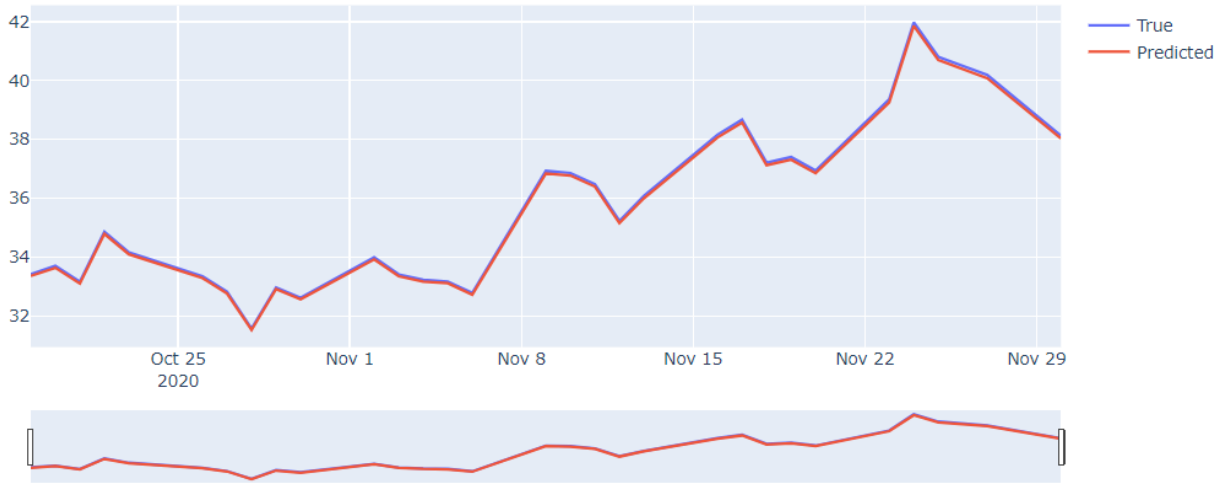
Printing the predictions:

```
# Print linear regression model predictions for the next '30' days
lr_prediction = lr.predict(x_forecast)
print(lr_prediction)
```

```
[33.35815413 33.63624679 33.09992929 34.78833412 34.09310816 33.28863191
 32.76224833 31.52077475 32.91122667 32.5636118  33.92426957 33.34822401
 33.16945151 33.10985941 32.72252026 36.84421406 36.77469184 36.39728282
 35.15580924 36.00001354 38.06582361 38.57234316 37.12230293 37.31100934
 36.85414418 39.257639   41.85976656 40.69774911 40.08197549 38.03602945]
```

Plotting the results:

Linear Regression Model



Testing the model:

```
# Testing Model: Score returns the coefficient of determination R^2 of the prediction.
# The best possible score is 1.0
lr_confidence = lr.score(x_test, y_test)
print("lr confidence: ", lr_confidence)
```

```
lr confidence:  0.8444568655585621
```

SUPPORT VECTOR REGRESSION

Support Vector Regression (SVR) is a type of Support Vector Machine and is a type of supervised learning algorithm that analyzes data for regression analysis.
Pros:

- It is effective in high dimensional spaces.
- It works well with clear margin of separation.
- It is effective in cases where number of dimensions is greater than the number of samples.

Cons:

- It does not perform well, when we have large data set.
- Low performance if the data set is noisy (a large amount of additional meaningless information).

We use the previously created dataset with for linear regression for predictions and train the Support Vector Machine.

```
# Create and train the Support Vector Machine (Regressor)
svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)
svr_rbf.fit(x_train, y_train)
```

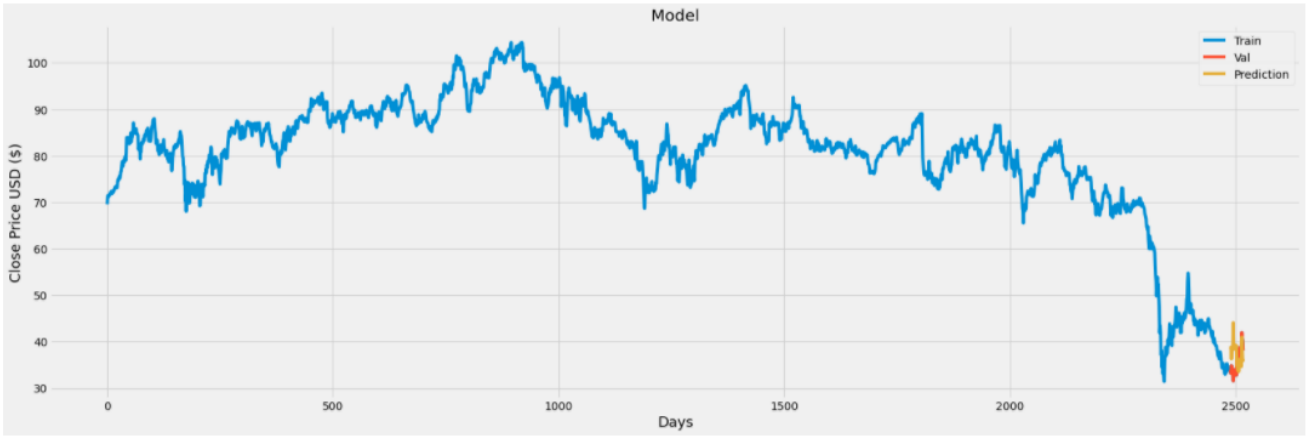
```
SVR(C=1000.0, gamma=0.1)
```

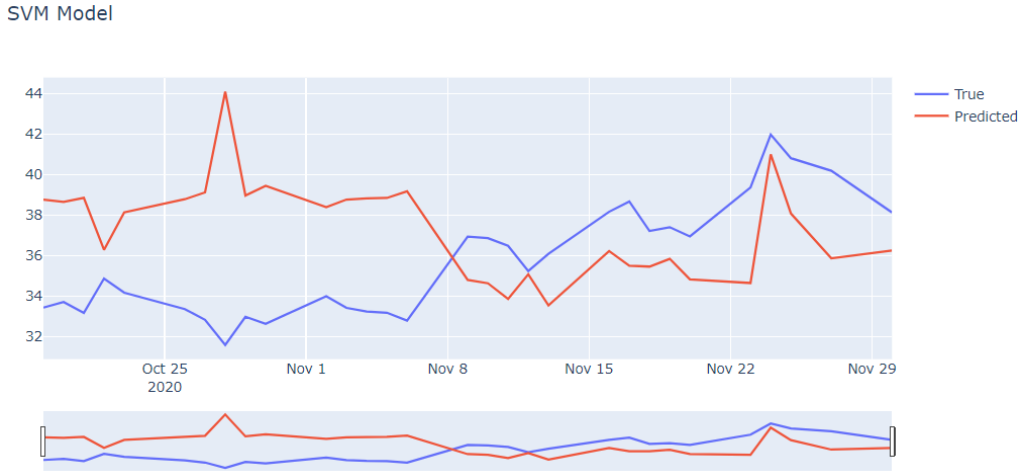
Printing the predictions:

```
# Print support vector regressor model predictions for the next '30' days
svm_prediction = svr_rbf.predict(x_forecast)
print(svm_prediction)
```

```
[38.76084316 38.64592305 38.84883164 36.2798587  38.13051776 38.78168188
 39.11840922 44.10101143 38.96504041 39.44088186 38.38500112 38.76384498
 38.82063907 38.84444583 39.17114455 34.79191569 34.62528132 33.8515703
 35.06773525 33.53368388 36.21581026 35.49114891 35.44966464 35.83355481
 34.81590141 34.63650767 41.00205856 38.0748608  35.85866034 36.23976605]
```

Plotting the results:





Testing the model:

```
# Testing Model: Score returns the coefficient of determination R^2 of the prediction.
# The best possible score is 1.0
svm_confidence = svr_rbf.score(x_test, y_test)
print("svm confidence: ", svm_confidence)
```

```
svm confidence:  0.8982096933271526
```

ARIMA

In an ARIMA model there are 3 parameters that are used to help model the major aspects of a times series: **seasonality, trend, and noise**. These parameters are labeled p,d,and q.

Auto ARIMA: Automatically discover the optimal order for an ARIMA model. We perform Auto ARIMA to get the best values of p, d, q.

The auto_arima function seeks to identify the most optimal parameters for an ARIMA model and returns a fitted ARIMA model. This function is based on the commonly-used R function, forecast::auto.arima.

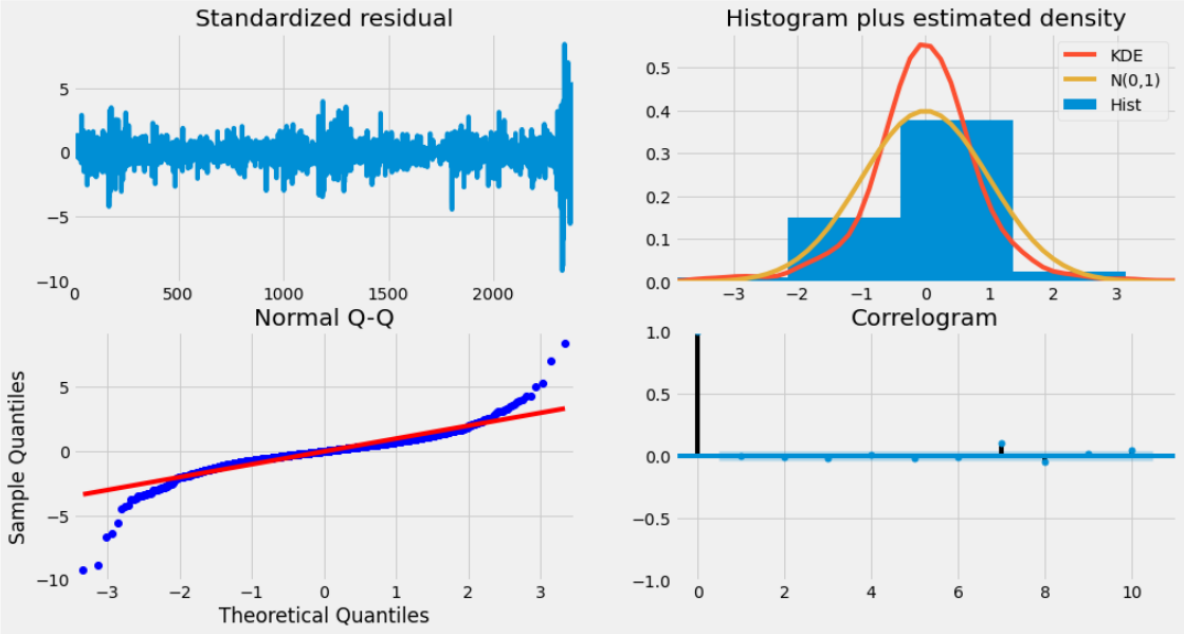
The auro_arima function works by conducting differencing tests (i.e., Kwiatkowski–Phillips–Schmidt–Shin, Augmented Dickey-Fuller or Phillips–Perron) to determine the order of differencing, d, and then fitting models within ranges of defined start_p, max_p, start_q, max_q ranges. If the seasonal optional is enabled, auto_arima also seeks to identify the optimal P and Q hyper- parameters after conducting the Canova-Hansen to determine the optimal order of seasonal differencing, D.

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-13507.192, Time=0.31 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-13516.492, Time=0.24 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-13515.270, Time=0.20 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-13508.850, Time=0.15 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=-13522.919, Time=0.52 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=-13522.119, Time=1.47 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-13522.264, Time=0.59 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-13520.357, Time=1.61 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=-13520.076, Time=0.49 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=-13524.572, Time=0.37 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=-13518.097, Time=0.13 sec
 ARIMA(3,1,0)(0,0,0)[0]             : AIC=-13523.754, Time=0.42 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=-13523.912, Time=0.26 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=-13521.924, Time=0.18 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=-13521.711, Time=0.36 sec

Best model:  ARIMA(2,1,0)(0,0,0)[0]
Total fit time: 7.307 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 2389
Model:               SARIMAX(2, 1, 0)   Log Likelihood                6765.286
Date:                Tue, 15 Dec 2020   AIC                         -13524.572
Time:                        20:46:31   BIC                         -13507.238
Sample:                             0   HQIC                        -13518.264
                                - 2389
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0645      0.010     -6.526      0.000      -0.084      -0.045
ar.L2          0.0596      0.009      6.918      0.000       0.043       0.076
sigma2         0.0002    2.2e-06     92.185      0.000       0.000       0.000
===================================================================================
Ljung-Box (Q):                      128.15   Jarque-Bera (JB):             15210.62
Prob(Q):                              0.00   Prob(JB):                         0.00
Heteroskedasticity (H):               2.39   Skew:                            -0.50
Prob(H) (two-sided):                  0.00   Kurtosis:                        15.32
===================================================================================
```

So the Auto ARIMA model provided the value of **p,d, and q as 2,1 and 0** respectively.

Before moving forward, let us review the residual plots from auto ARIMA.



Top left: The residual errors seem to fluctuate around a mean of zero and have a uniform variance.
Top Right: The density plot suggests normal distribution with mean zero.
Bottom left: All the dots should fall perfectly in line with the red line. Any significant deviations would imply the distribution is skewed.
Bottom Right: The Correlogram, aka, ACF plot shows the residual errors are not autocorrelated. Any autocorrelation would imply that there is some pattern in the residual errors which are not explained in the model. So you will need to look for more X's (predictors) to the model.

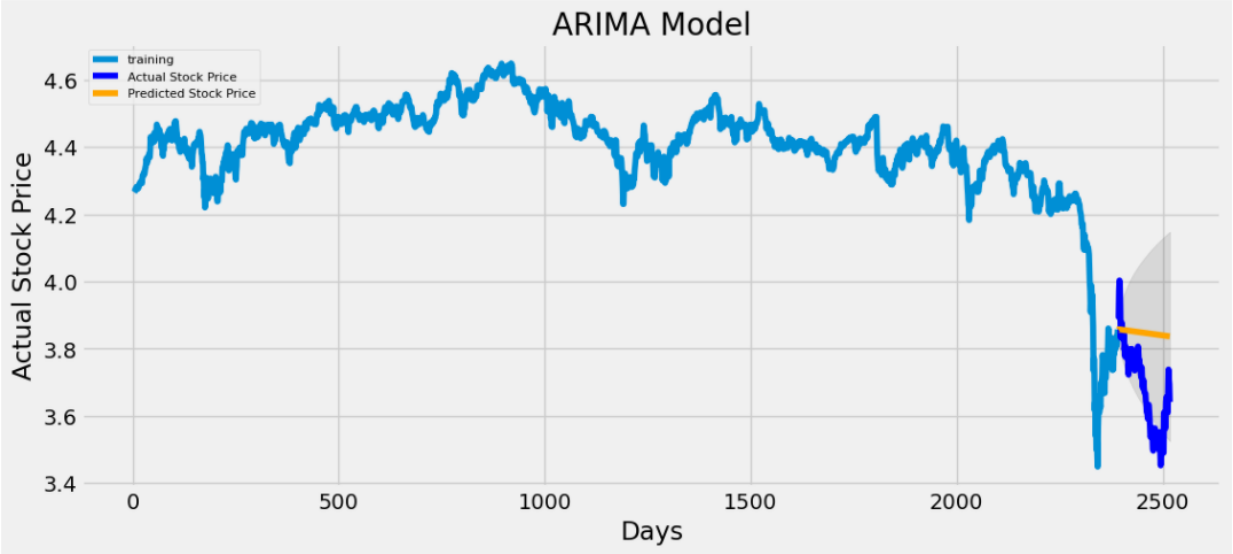
Overall, it seems to be a good fit. Let us start forecasting the stock prices.

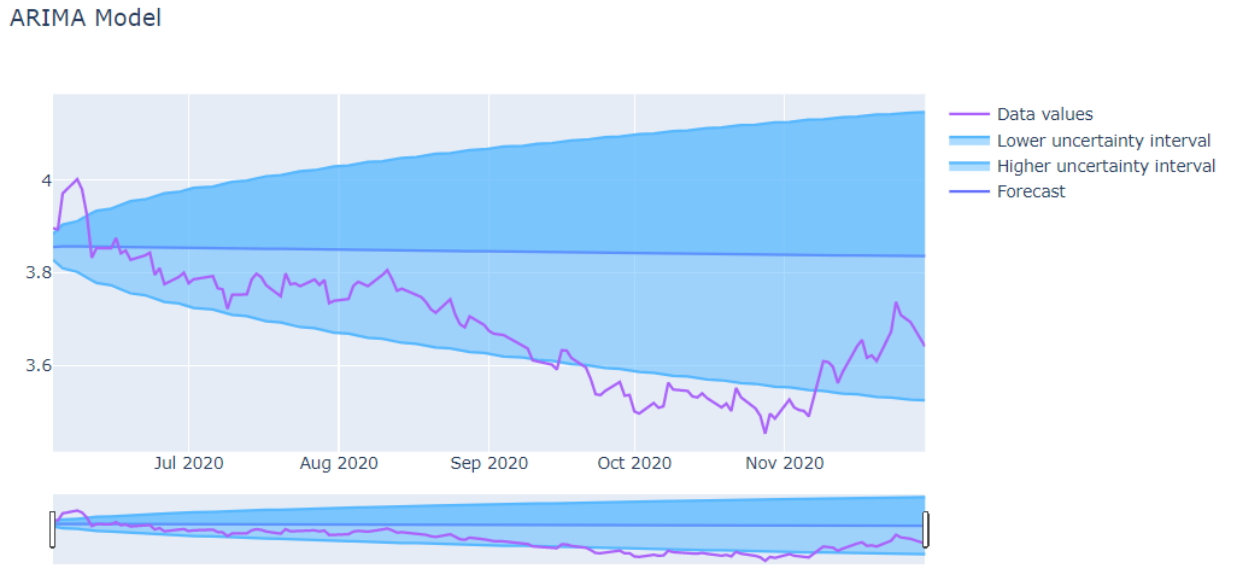
Next, create an ARIMA model with provided optimal parameters p, d and q.

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                 D.Close   No. Observations:                 2388
Model:                 ARIMA(2, 1, 0)   Log Likelihood                6765.460
Method:                       css-mle   S.D. of innovations              0.014
Date:                Tue, 15 Dec 2020   AIC                         -13522.919
Time:                        20:46:32   BIC                         -13499.806
Sample:                             1   HQIC                        -13514.509

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0002      0.000     -0.589      0.556      -0.001       0.000
ar.L1.D.Close -0.0646      0.020     -3.161      0.002      -0.105      -0.025
ar.L2.D.Close  0.0594      0.020      2.905      0.004       0.019       0.099
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -3.5957           +0.0000j            3.5957            0.5000
AR.2            4.6837           +0.0000j            4.6837            0.0000
-----------------------------------------------------------------------------
```

Plotting the ARIMA results:





Let us check the commonly used accuracy metrics to judge forecast results:

```
# report performance
mse = mean_squared_error(test_data, fc)
print('MSE: '+str(mse))
mae = mean_absolute_error(test_data, fc)
print('MAE: '+str(mae))
rmse = math.sqrt(mean_squared_error(test_data, fc))
print('RMSE: '+str(rmse))
mape = np.mean(np.abs(fc - test_data)/np.abs(test_data))
print('MAPE: '+str(mape))
```

```
MSE: 0.04192005827108994
MAE: 0.17312737616677829
RMSE: 0.20474388457555928
MAPE: 0.04804420167389875
```

PROPHET

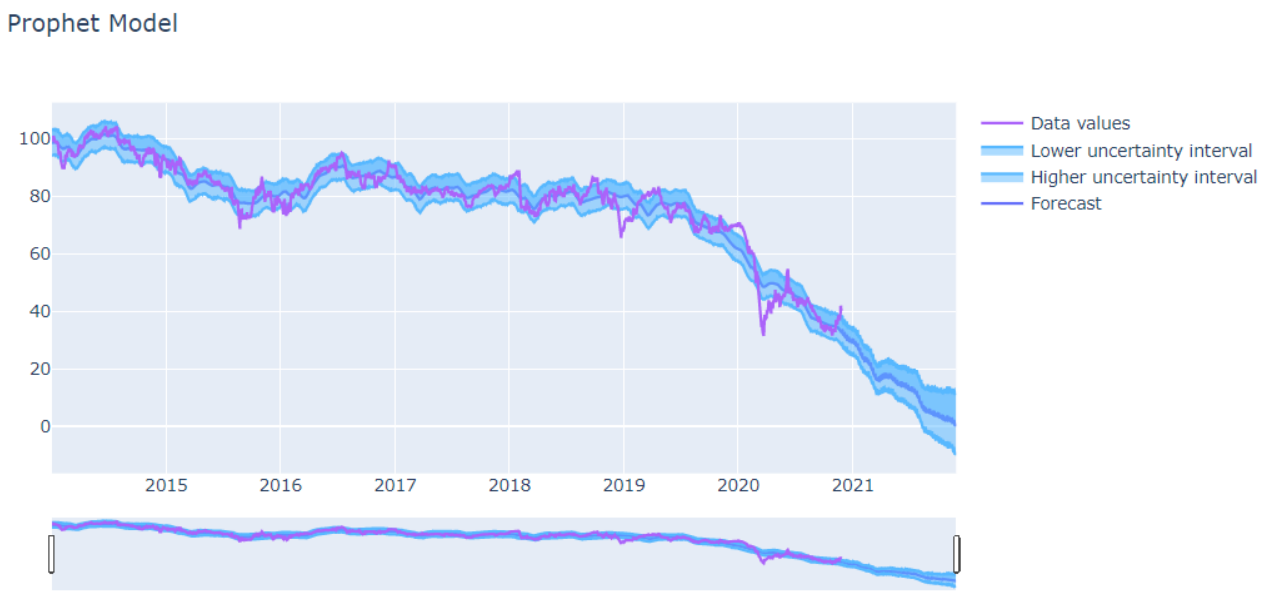
Prophet is a forecasting model which allows to deal with multiple seasonalities. It is an open-source software released by Facebook's Core Data Science team.

Forecasting the price for next 365 days:

```
future_prices = model.make_future_dataframe(periods=365)
forecast = model.predict(future_prices)
df_forecast = forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']]
df_forecast.head()
```

	ds	yhat	yhat_lower	yhat_upper
0	2014-01-02	98.631202	94.085913	102.742127
1	2014-01-03	98.651046	94.163165	103.055245
2	2014-01-06	98.847121	94.136195	103.147529
3	2014-01-07	99.061925	94.590950	103.566339
4	2014-01-08	98.978871	94.594691	103.230918

Plotting the results:



LONG-SHORT TERM MEMORY (LSTM)

Unlike regression predictive modeling, time series also adds the complexity of a sequence dependence among the input variables.

A powerful type of neural network designed to handle sequence dependence is called recurrent neural networks. The Long Short-Term Memory network or LSTM network is a type of **recurrent neural network** used in deep learning because very large architectures can be successfully trained.

We will build the LSTM with **50 neurons and 4 hidden layers**. Finally, we will assign 1 neuron in the output layer for predicting the normalized stock price. We will use the **MSE loss function** and the **Adam stochastic gradient descent optimizer**.

```
model = Sequential()
#Adding the first LSTM layer and some Dropout regularisation
model.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
model.add(Dropout(0.2))
# Adding a second LSTM layer and some Dropout regularisation
model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))
# Adding a third LSTM layer and some Dropout regularisation
model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))
# Adding a fourth LSTM layer and some Dropout regularisation
model.add(LSTM(units = 50))
model.add(Dropout(0.2))
# Adding the output layer
model.add(Dense(units = 1))

# Compiling the RNN
model.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
model.fit(X_train, y_train, epochs = 100, batch_size = 32)
```

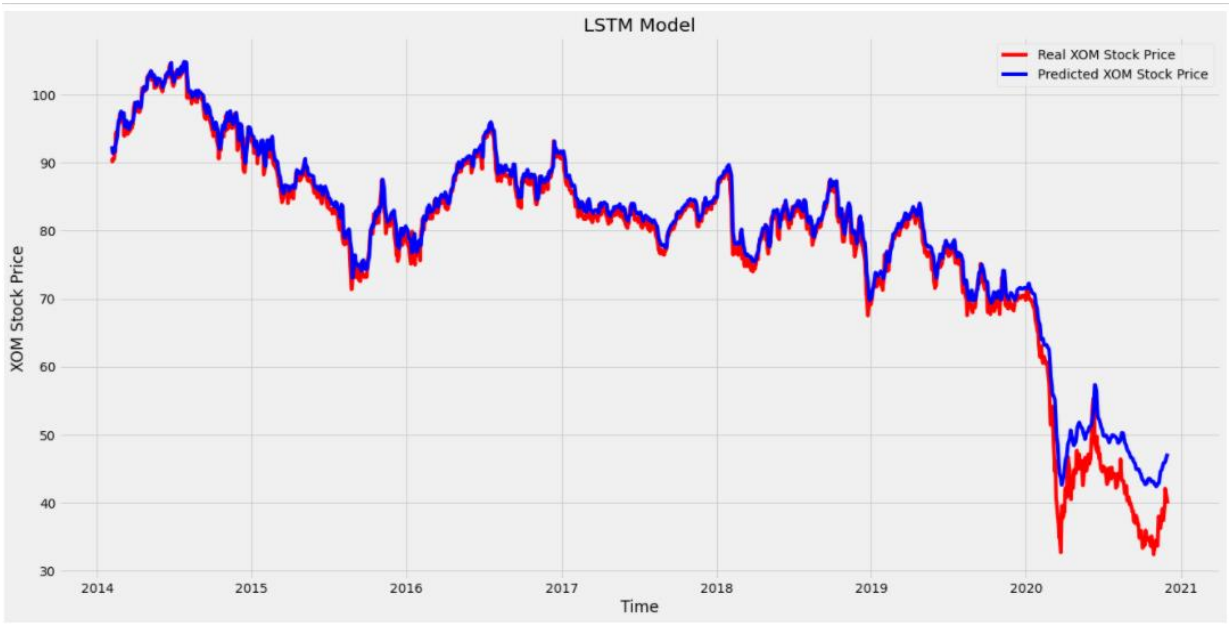
```
Epoch 1/100
30/30 [==============================] - 2s 66ms/step - loss: 0.0613
Epoch 2/100
30/30 [==============================] - 2s 70ms/step - loss: 0.0129
Epoch 3/100
30/30 [==============================] - 2s 70ms/step - loss: 0.0099
Epoch 4/100
30/30 [==============================] - 2s 69ms/step - loss: 0.0091
Epoch 5/100
30/30 [==============================] - 2s 69ms/step - loss: 0.0086
Epoch 6/100
30/30 [==============================] - 2s 70ms/step - loss: 0.0087
Epoch 7/100
30/30 [==============================] - 2s 72ms/step - loss: 0.0096
Epoch 8/100
30/30 [==============================] - 2s 72ms/step - loss: 0.0084
Epoch 9/100
30/30 [==============================] - 2s 72ms/step - loss: 0.0083
Epoch 10/100
```

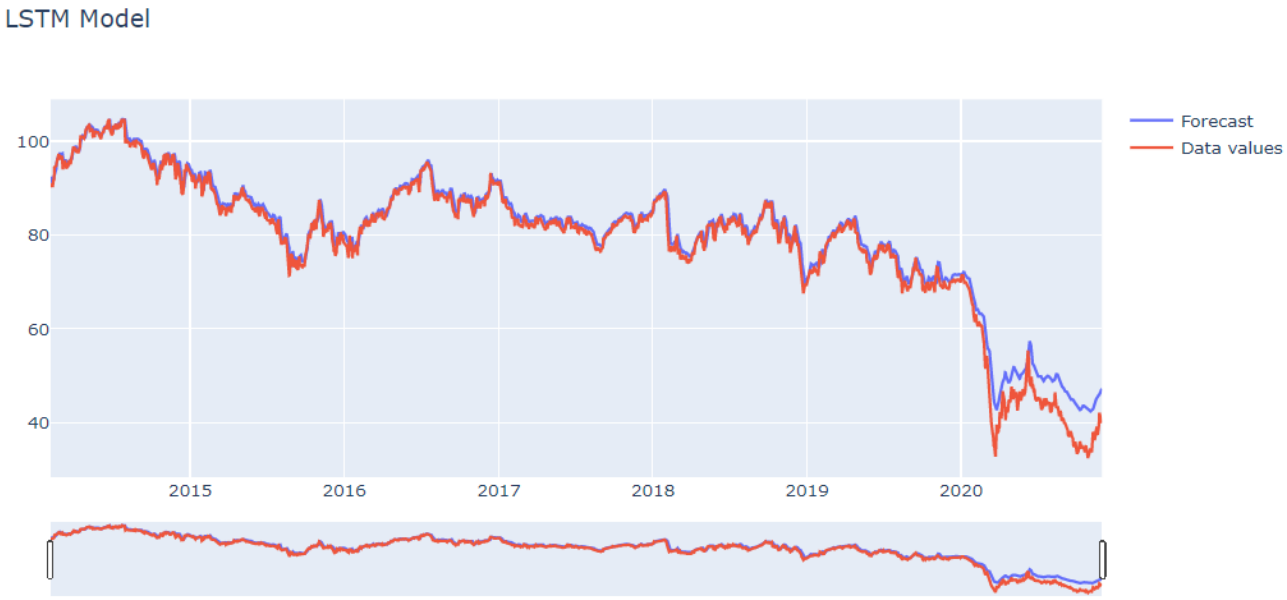
Printing the predicted price:

```
print(predicted_stock_price)
```

```
[[91.64334 ]
 [91.0077  ]
 [90.75813 ]
 ...
 [42.664513]
 [43.16031 ]
 [43.740532]]
```

Plotting the results:



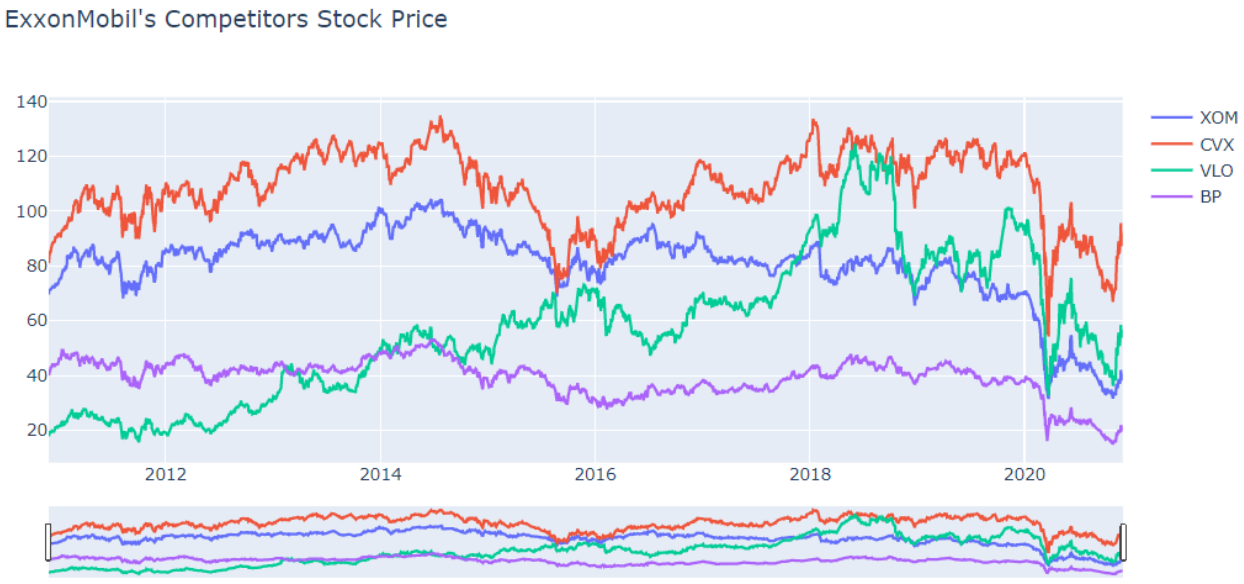


Huge drop in March 2020 due to the COVID-19 lockdown. We can clearly see that our model performed very good. It can accurately follow most of the unexcepted jumps/drops however, for the most recent date stamps, we can see that the model expected (predicted) higher values compared to the real values of the stock price.

EXXONMOBIL BUSINESS ANALYSIS

Competitors

Let us analyze ExxonMobil's competitors. Some of the major competitors are: Chevron Corporation (CVX), Valero Energy (VLO), BP (BP).



We can see that **Chevron and Valero** are performing better than Exxon. This is because of various factors, let us see what some of the factors are:

ExxonMobil's correlation with oil prices

- Some of Exxon's competitors, including Shell and France's Total, have been **pivoting towards electricity**. The rationale of this transition is very simple: carbon-based fuels are being replaced by cleaner substitutes, impacting everything from transport to electricity generation.
- In the early part of 2020, oil prices literally dropped to zero, implying that things are not right in energy space. The fact that oil drillers paid consumers to take their oil for a brief period is terrifying. However, prices are back up to $40 a barrel of oil, indicating that the near-death of oil is greatly exaggerated. As a commodity, oil is subject to supply and demand, with current low prices (driven by COVID-19 demand dislocations) triggering a material stir in the industry.

Is Exxon going Bankrupt?

An Exxon bankruptcy is **extremely unlikely** in the near term. As it stands, the organization has one of the strongest balance sheets among its integrated peers. The **debt-to-equity ratio** of Exxon is approximately **0.38** times. While Chevron (NYSE:CVX) is 0.25 times lower, all European players have a much higher debt-to-equity ratio. At the top of the graph, BP is around 1.1 times. The fact is that 0.38 times is a reasonably respectable amount, leaving Exxon with plenty of space on its balance sheet to handle more adversity.

CONCLUSION

- One crucial aspect that has led to the longevity of ExxonMobil for more than 100 years is its financial discipline. However, with **rising debt levels** recently, investors are worried that ExxonMobil is not keeping up on this front.
- Consistently **low oil and gas prices** have had an impact on the company's profits over the last few years, but their massive growth plans have persisted. It has placed a burden on the balance sheet.
- In addition, an unprecedented **decline in oil demand** due to the Coronavirus pandemic severely impacted ExxonMobil's success in the second quarter, preventing them from making any debt repayments from earnings.
- **Rising debt levels and mega-capital spending plans** in an unfavorable oil price setting have contributed to a rapid decline in stock prices over the last few years. Its **exclusion from the Dow Jones Industrial Average Index** further exacerbated investor fears about the oil giant, which at one time was the largest listed firm by market capitalization. ExxonMobil's stock dropped by **more than 40% in 2020**.

It is not wise for one to invest in ExxonMobil's stock right now, considering all the above factors.

CITATION

- https://www.kdnuggets.com/2020/01/tutorials.html
- https://medium.com/
- https://plotly.com/python/time-series/
- https://www.fool.com/investing/stock-market/market-sectors/energy/oil-stocks/
- https://www.fool.com/investing-news/