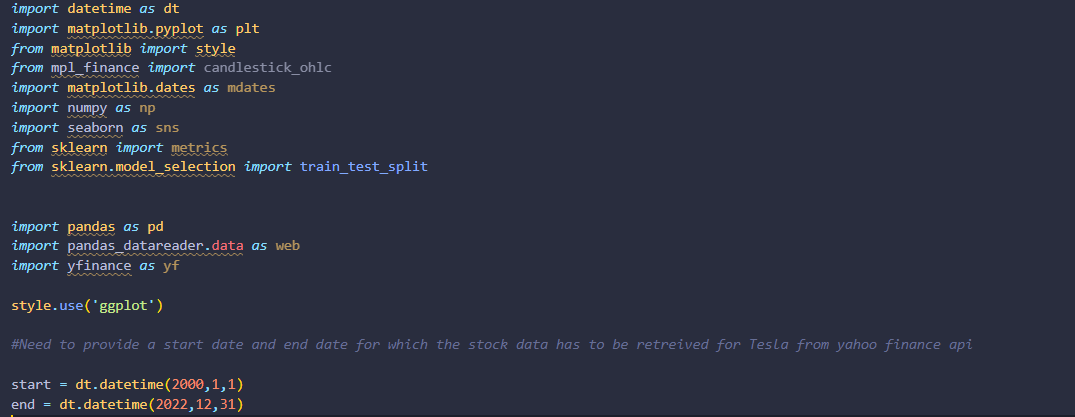
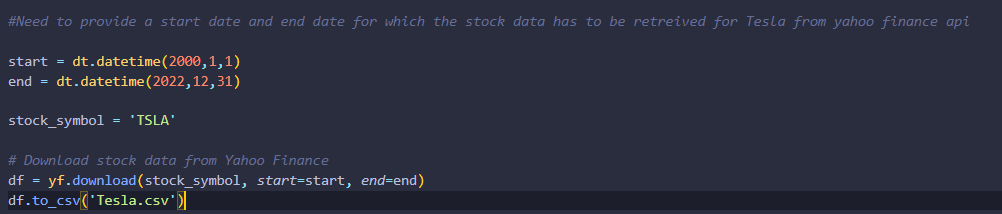
1. Importing all the necessary libraries. We would be majorly Scikit-learn ,Pandas and Matplotlib libraries for creating dataframes, analyzing the data, plotting the charts and for predictive analytics. We have used the mpl\_finance library to get the charts used in financial markets. The yfinance library uses the yahoo Api to obtain stock prices from yahoo finance.



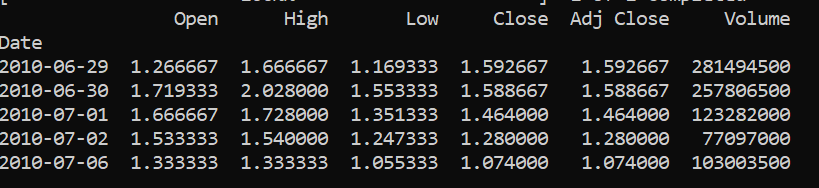
1. Next step would be to retrieve the stock prices from yahoo finance with the date and stock code arguments. We then save the csv file in the project directory with the name “Tesla.csv”.



1. We can read the csv file with the ‘.read\_csv’ built-in pandas function. We check what the dataframe df looks like.We will use the .head() method of the dataframe to view the first few rows of the data.



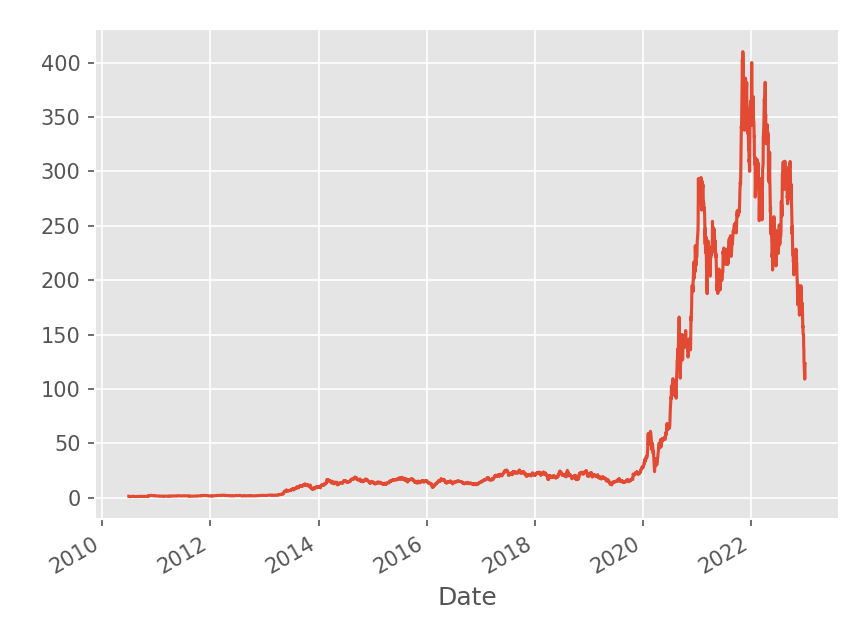
Output:



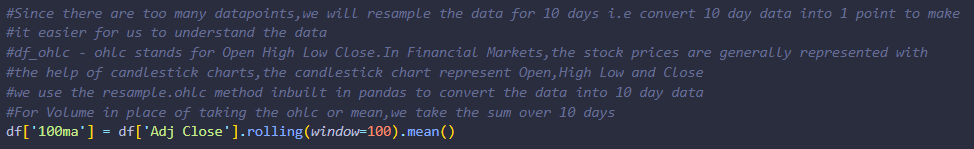
1. We can now plot Tesla’s closing prices since inception with just 2 lines of code.



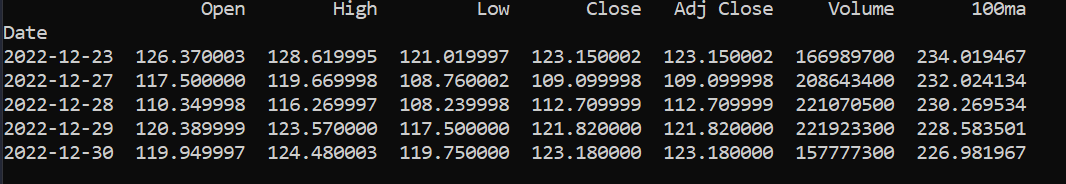
Output:



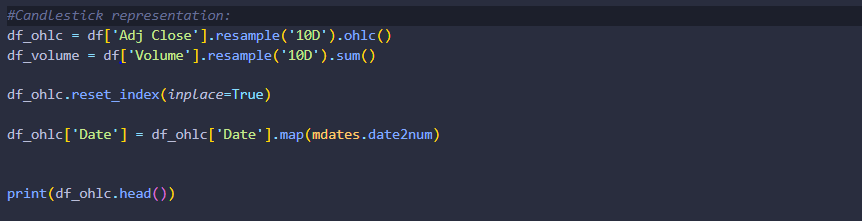
1. We will create a new column in our dataframe titled ‘100ma’,just to make it easier to do analysis on the prices and reduce the data points substantially.The new column gets added to the end of the current df.



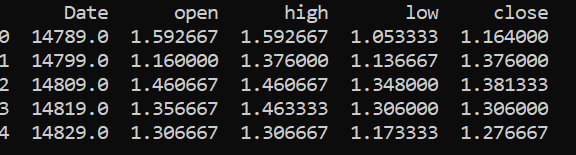
Output:



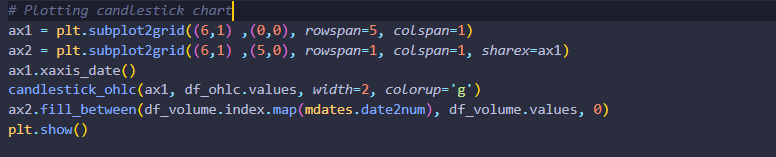
1. The line chart does not explain how a stock performed on a given day. At what price did the stock open? What price did the stock close? What was the peak price during the day? What was the lowest price? These questions are easily answered with a candlestick chart. We would plot the candlesticks with 10-day average, since the data size is massive. To represent the data in the candlestick chart we create a new dataframe df\_ohlc which would have the columns date, open, high, low and close.



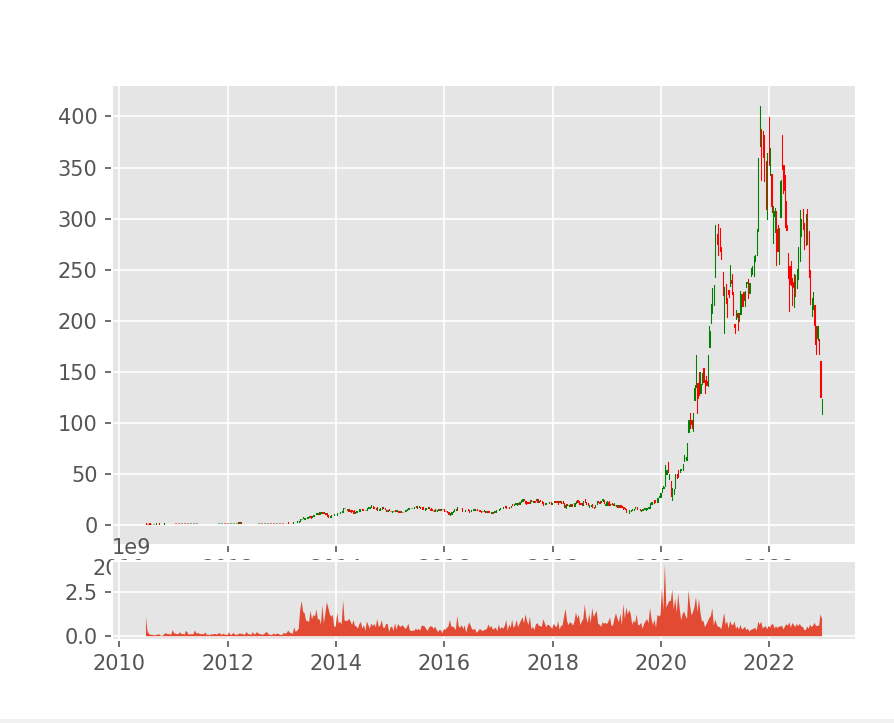
Output:



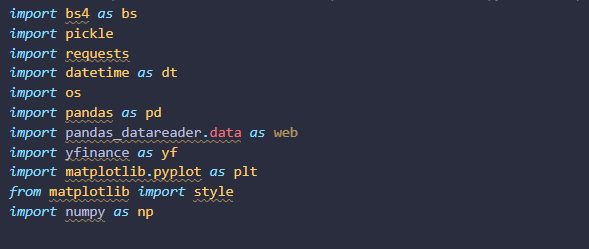
1. We will plot the data with matplotlib using the below code. The candles represent the price movements and the red bar at the bottom is the total volume.



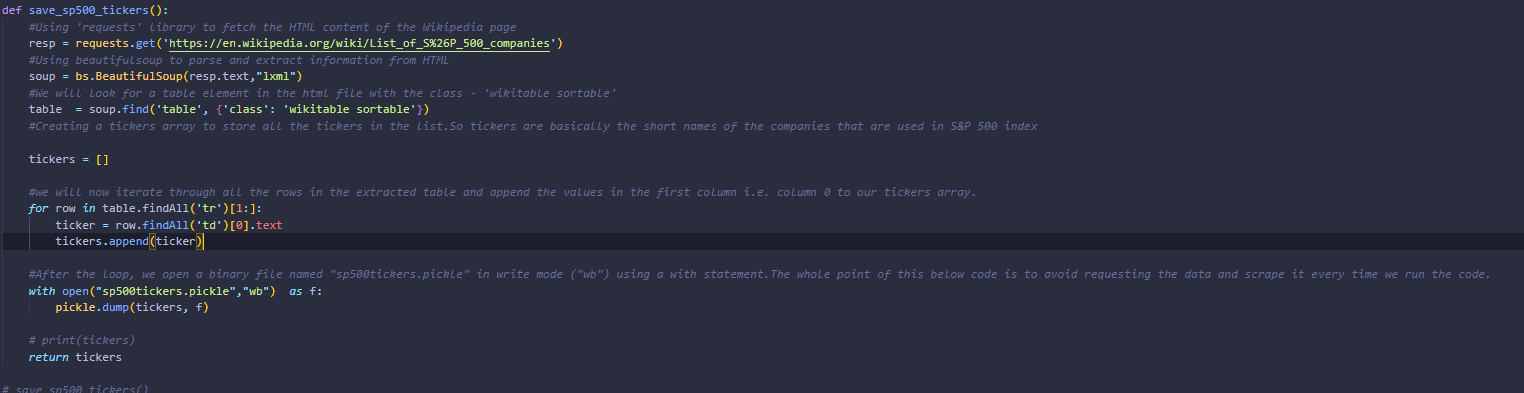
Output:



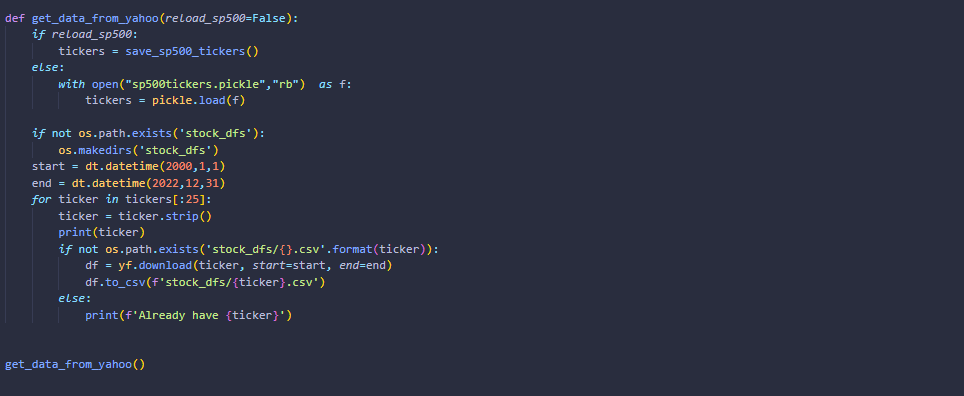
1. We now move on to the next part of the project. We will scrape the stock codes of 25 companies listed on the S&P Index to compare how Tesla is doing. Similar to the first part, we will import the necessary libraries. We have created a different python file titled “S&PData.py” to code this part for simplicity purposes. Most of the libraries we have already mentioned in the first part.The new addition to that would be the beautifulsoup library which scrapes the latest stock codes from Wikipedia. The only reason to scrape the data is there are constant additions and deletions of stock codes in the S&P Index. The pickle library in Python is used for serializing and deserializing Python objects. Serialization is the process of converting a Python object into a byte stream, and deserialization is the process of reconstructing the original object from that byte stream.



1. The below function would save the stock codes also known as tickers to a .pickle file. We do that to ensure every time when we run the code we do not download the data every time if we already have it. We only download the new additions be it a new stock code that has been added or a new row for another day.



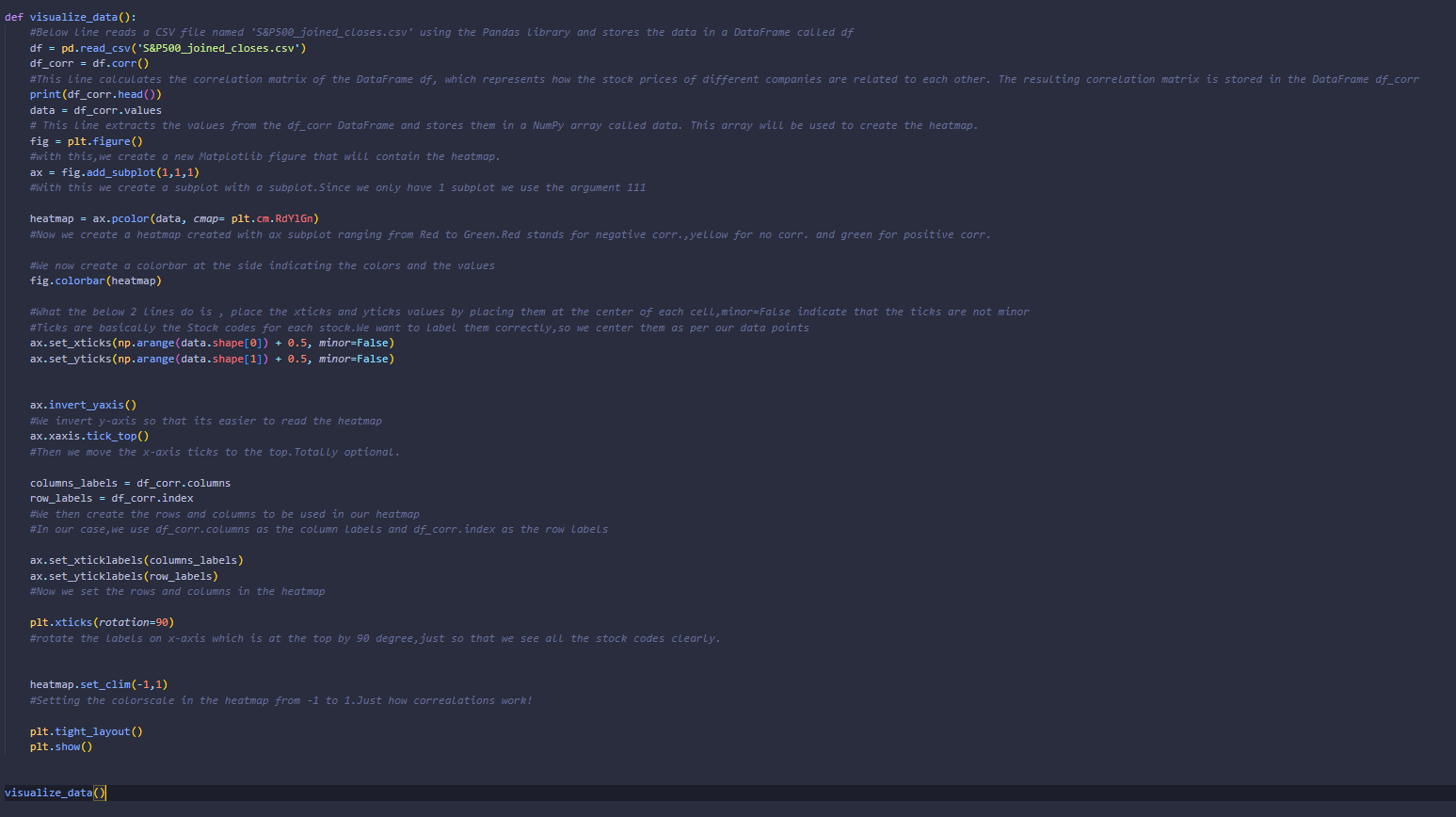
1. The function get\_data\_from\_yahoo() would iterate through the .pickle file that we saved earlier.We set reload\_sp500 to False in the argument to ensure it does not scrape the data again.But we can set it to true if required. The function then check if the directory ‘stock\_dfs’ exists or not?If it does not exist it will create a new directory with the same name. After setting the start and end dates for downloading the prices of the stock codes,it will create separate csv files with the stock code names e.g. AMZN.csv in the stock\_dfs folder.



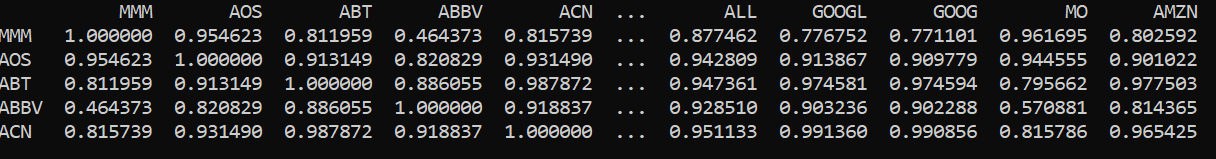
1. Now we will compile all the prices of the 25 Stocks that we extracted into a single csv file with the function compile\_data()

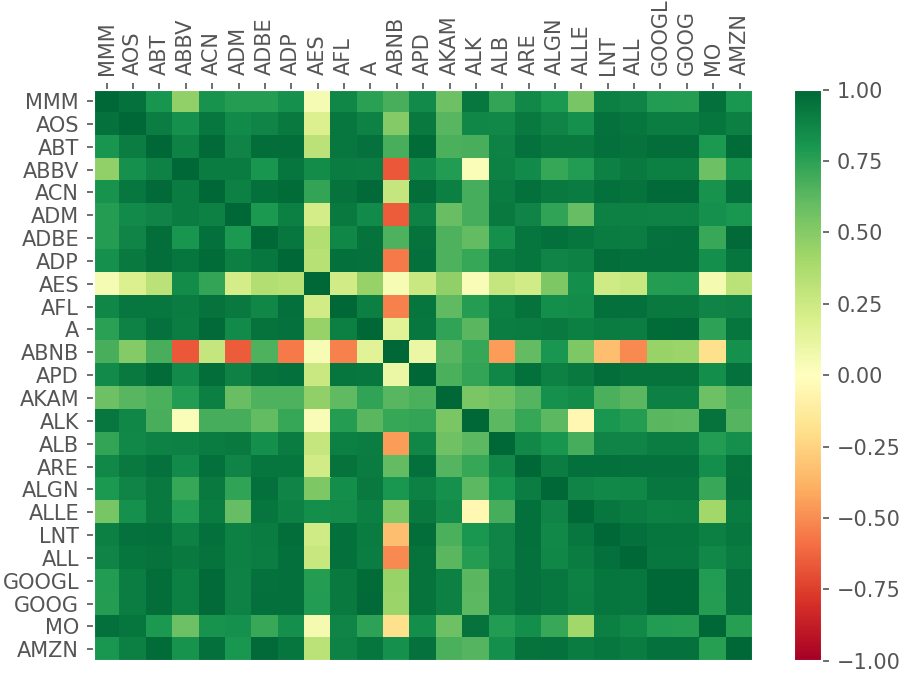


1. One of the plots we can create with the extracted stocks is a correlation heatmap explaining how the stocks we have in our dataframe are correlated to each other. The heatmap has a lot of use cases.

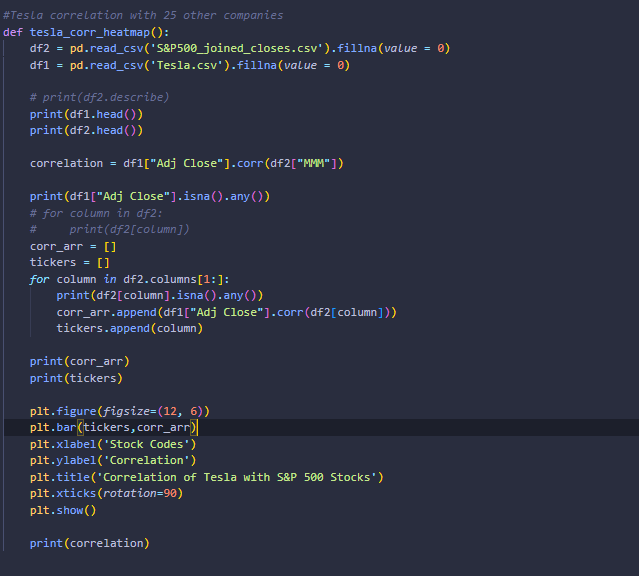


Output:

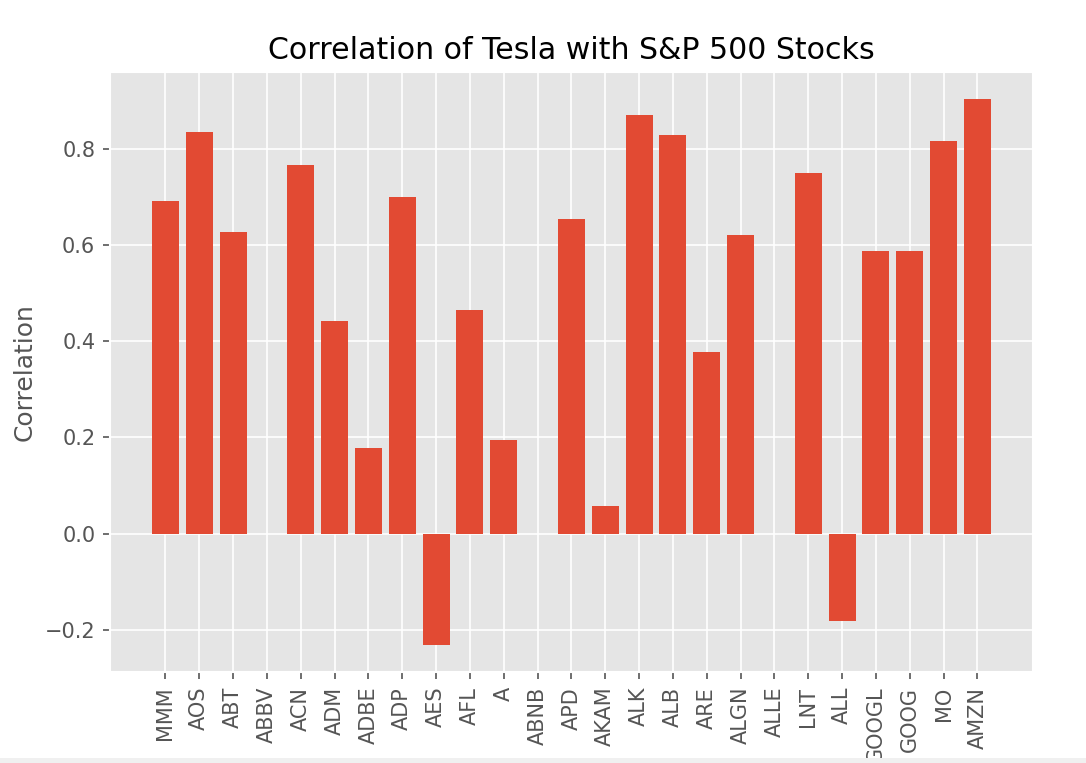




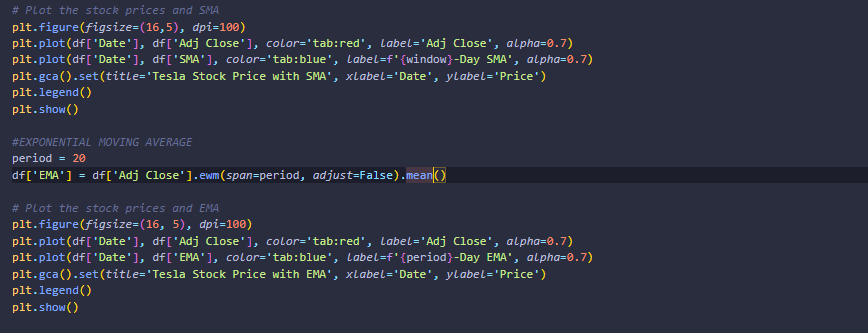
1. Coming back to “main.py” file. Since we now have the stock prices of 25 different stocks. We can now check how Tesla’s Stock prices are correlated to the other stocks with the function tesla\_corr\_heatmap()



Output:

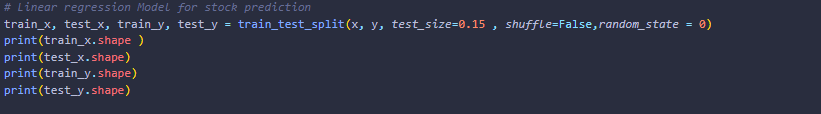


1. We can plot the simple moving average of the closing prices also the Exponential moving average which indicate the price range and are used to figure a direction of the price movements.



1. This code splits the data and labels into training and testing sets, prints the shapes of the resulting arrays to check the sizes, and ensures that the split is deterministic by using a fixed random seed.We split the total datapoints(3150) into training and test features and lables.

train\_x which is the training data has 2677 rows and 4 columns.test\_x is the test dataset consisting of 473 rows and 4 columns.The training label train\_y has 2677 rows while the test label test\_y has 473 rows.



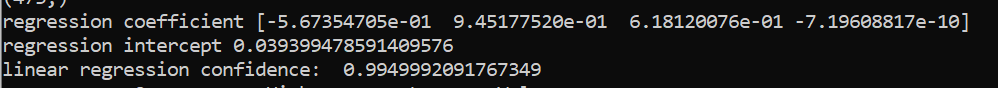
Output:



1. We employs Scikit-learn's Linear Regression model to train on a given dataset (train\_x and train\_y). It prints the coefficients and intercept of the linear regression equation, computes the coefficient of determination (R²) to assess the model's performance on a separate test set (test\_x and test\_y), makes predictions on the test set, and creates a Pandas DataFrame (dfr) to compare actual and predicted stock prices. Dates are added to this DataFrame for clarity, and the first 10 rows along with summary statistics are printed. The primary goal is to evaluate the model's ability to explain the variation in stock prices and visually compare its predictions with actual values.



Output:



\*Regression Coefficients:

For each feature in your input data, there is a corresponding coefficient. In your case, you have four coefficients: -5.67354705e-01, 9.45177520e-01, 6.18120076e-01, and -7.19608817e-10.These coefficients represent the weights assigned to each feature in the linear regression equation. For instance, if X1 is the first feature, the coefficient -5.67354705e-01 indicates that a one-unit increase in X1 is associated with a decrease of approximately 0.5673 units in the predicted output.

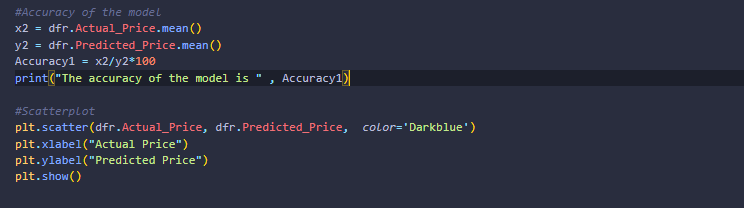
\*Regression Intercept:

The intercept of 0.03939… is the value of the predicted output when all features are zero. In the context of your data, it represents the base value of the stock prices when all other factors are zero.

\*Coefficient of Determination (R²):

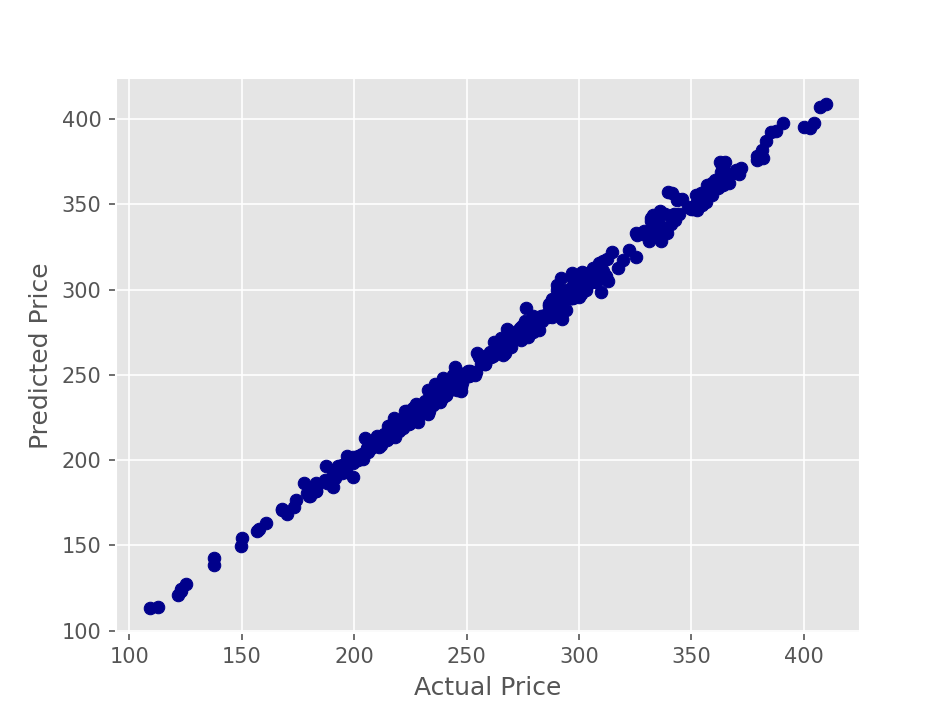
The R² value of 0.9949992091767349 is very close to 1, indicating that the linear regression model is highly successful at explaining the variation in the target variable (stock prices). R² is a measure of how well the model's predictions match the actual values. In this case, approximately 99.5% of the variation in the test data is explained by the model.

1. Checking the accuracy of the model and plotting the scatter plot with actual and predicted prices.

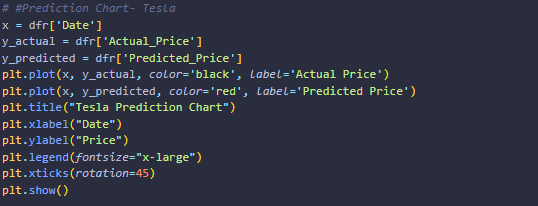


Output:

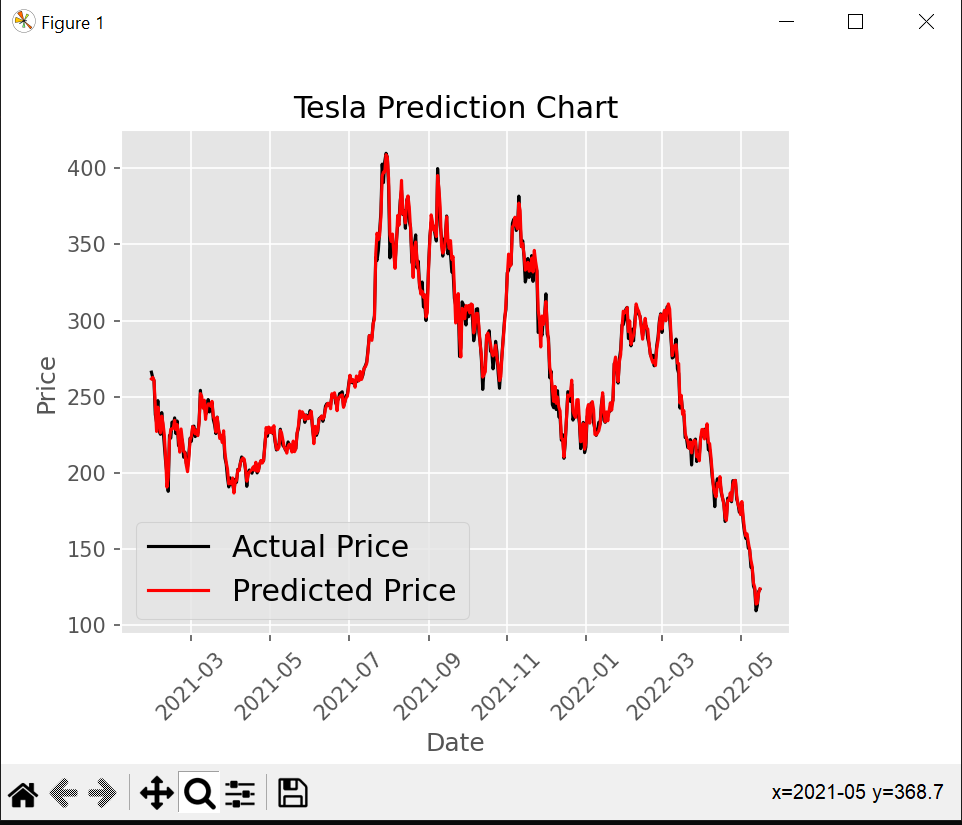




1. We will plot a line chart with matplotlib comparing the actual prices and the predicted prices we derived using linear regression.

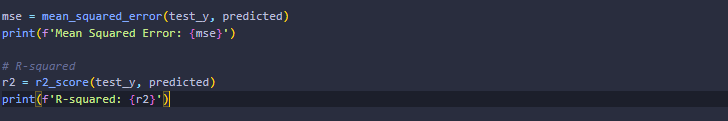


Output:



1. MSE measures the average squared difference between the actual and predicted values. A lower MSE indicates better model performance. In our case, an MSE of 16.37 is relatively low, suggesting that, on average, the model's predictions are close to the actual values.

R-squared is a measure of how well the independent variable(s) explain the variance in the dependent variable. It ranges from 0 to 1, where 1 indicates a perfect fit. Your R-squared value of 0.995 is very high, indicating that the model explains a significant portion of the variance in the test data.



Output:

