# Tesla Stock Prediction


In [30]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import pandas_datareader as data
from keras.models import load_model
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
from sklearn import metrics
```

In [9]:
```
TESLA=pd.read_csv("C:\\Users\\Pranav\\Desktop\\DATA SCIENCE DATA\\CVC file\\TSLA.csv")
TESLA.head()
```

Out[9]:

| | Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|---|
| **0** | 2010-06-29 | 19.000000 | 25.00 | 17.540001 | 23.889999 | 23.889999 | 18766300 |
| **1** | 2010-06-30 | 25.790001 | 30.42 | 23.299999 | 23.830000 | 23.830000 | 17187100 |
| **2** | 2010-07-01 | 25.000000 | 25.92 | 20.270000 | 21.959999 | 21.959999 | 8218800 |
| **3** | 2010-07-02 | 23.000000 | 23.10 | 18.709999 | 19.200001 | 19.200001 | 5139800 |
| **4** | 2010-07-06 | 20.000000 | 20.00 | 15.830000 | 16.110001 | 16.110001 | 6866900 |

In [10]:
```
# shape of dataset
TESLA.shape
```

Out[10]:
```
(2416, 7)
```

In [11]:
```
# information about dataset
TESLA.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2416 entries, 0 to 2415
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       2416 non-null   object 
 1   Open       2416 non-null   float64
 2   High       2416 non-null   float64
 3   Low        2416 non-null   float64
 4   Close      2416 non-null   float64
 5   Adj Close  2416 non-null   float64
 6   Volume     2416 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 132.2+ KB
```

In [12]:
```
#Describe of dataset
TESLA.describe()
```

Out[12]:

| | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| **count** | 2416.000000 | 2416.000000 | 2416.000000 | 2416.000000 | 2416.000000 | 2.416000e+03 |
| **mean** | 186.271147 | 189.578224 | 182.916639 | 186.403651 | 186.403651 | 5.572722e+06 |
| **std** | 118.740163 | 120.892329 | 116.857591 | 119.136020 | 119.136020 | 4.987809e+06 |
| **min** | 16.139999 | 16.629999 | 14.980000 | 15.800000 | 15.800000 | 1.185000e+05 |
| **25%** | 34.342498 | 34.897501 | 33.587501 | 34.400002 | 34.400002 | 1.899275e+06 |
| **50%** | 213.035004 | 216.745002 | 208.870002 | 212.960007 | 212.960007 | 4.578400e+06 |
| **75%** | 266.450012 | 270.927513 | 262.102501 | 266.774994 | 266.774994 | 7.361150e+06 |
| **max** | 673.690002 | 786.140015 | 673.520020 | 780.000000 | 780.000000 | 4.706500e+07 |

In [13]:
```
# find ot null value in dataset
TESLA.isnull().sum()
```

```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

In [14]:
```
# index of dataset
TESLA.index
```

Out[14]:
```
RangeIndex(start=0, stop=2416, step=1)
```

In [15]:
```
# columns in dataset
TESLA.columns
```

Out[15]:
```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

In [16]:
```
# Dtypes of dataset
TESLA.dtypes
```

Out[16]:
```
Date          object
Open         float64
High         float64
Low          float64
Close        float64
Adj Close    float64
Volume         int64
dtype: object
```

In [17]:
```
#nunique of dataset
TESLA.nunique()
```

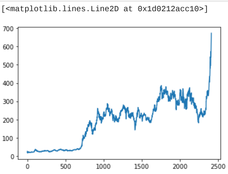
Out[17]:
```
Date         2416
Open         2132
High         2128
Low          2136
Close        2225
Adj Close    2225
Volume       2391
dtype: int64
```

In [18]:
```
#value counts in dataset
TESLA.value_counts()
```

Out[18]:
```
Date        Open        High        Low         Close       Adj Close   Volume    
2010-06-29  19.000000   25.000000   17.540001   23.889999   23.889999   18766300  1
2016-11-10  191.050003  191.610001  180.419998  185.350006  185.350006  6750300   1
2016-11-14  188.000000  188.250000  178.190002  181.449997  181.449997  6552200   1
2016-11-15  182.779999  186.429993  182.050003  183.770004  183.770004  3902000   1
2016-11-16  182.649994  184.729996  181.210007  183.929993  183.929993  3434400   1
                                                                                  ..
2013-09-10  161.449997  167.500000  160.630005  166.369995  166.369995  8967800   1
2013-09-11  166.410004  167.899994  162.130005  163.520004  163.520004  5832500   1
2013-09-12  164.000000  166.759995  160.509995  164.929993  164.929993  6160000   1
2013-09-13  162.770004  166.369995  162.160004  165.539993  165.539993  5401200   1
2020-02-03  673.690002  786.140015  673.520020  780.000000  780.000000  47065000  1
Length: 2416, dtype: int64
```

In [55]:
```
#line plot
plt.plot(TESLA['Open'])
```

Out[55]:
```
[<matplotlib.lines.Line2D at 0x1d9212acc10>]
```




In [22]:
```
#separting data and labels
X=TESLA[['High','Open','Low','Volume']].values
y=TESLA['Close'].values
```

In [47]:
```
print(X)
```

```
[[2.50000000e+01 1.90000000e+01 1.75400010e+01 1.87663000e+07]
 [3.04200000e+01 2.57900010e+01 2.32999990e+01 1.71871000e+07]
 [2.59200000e+01 2.50000000e+01 2.02700000e+01 8.21880000e+06]
 ...
 [6.50880005e+02 6.32419983e+02 6.18000000e+02 2.90057000e+07]
 [6.53000000e+02 6.40000000e+02 6.32520020e+02 1.57193000e+07]
 [7.86140015e+02 6.73690002e+02 6.73520020e+02 4.70650000e+07]]
```

In [25]:
```
print(y)
```

```
[ 23.889999  23.83      21.959999 ... 640.809998 650.570007 780.       ]
```

In [28]:
```
#training and test data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2)
print("shape of X_train= ",X_train.shape)
print("shape of X_test= ",X_test.shape)
print("shape of y_train= ",y_train.shape)
print("shape of y_test= ",y_test.shape)
```

```
shape of X_train=  (1932, 4)
shape of X_test=  (484, 4)
shape of y_train=  (1932,)
shape of y_test=  (484,)
```

In [31]:
```
#Model training
model=LinearRegression()
model
```

Out[31]:
```
LinearRegression()
```

In [32]:
```
model.fit(X_train,y_train)
```

Out[32]:
```
LinearRegression()
```

In [36]:
```
X_train_prediction=model.predict(X_test)
```

In [43]:
```
#model evalution
#Accuracy score
#accuracy of testing data
model.score(X_test,y_test)*100
```

Out[43]:
```
99.96721977026714
```

In [44]:
```
#accurcy of Training data
model.score(X_train,y_train)*100
```

Out[44]:
```
99.97176672587295
```

In [52]:
```
input_data1=(25.00,19.000000,17.540001,18766300)
input_data_numpy_array1=np.asarray(input_data1)
```

In [53]:
```
#reshape the np array as we are predicting for one instance
input_data_reshaped1=input_data_numpy_array1.reshape(1,-1)
input_data_reshaped1
```

Out[53]:
```
array([[2.5000000e+01, 1.9000000e+01, 1.7540001e+01, 1.8766300e+07]])
```

In [54]:
```
prediction=model.predict(input_data_reshaped1)
print(prediction)
```

```
[22.47424046]
```

In [ ]: