Importing the necessary Libraries

In [3]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
```

Importing the dataset

In [4]:

```
df=pd.read_csv('/Users/shashankpatil/Downloads/stock.csv')
```

Description of the dataset

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1009 entries, 0 to 1008
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       1009 non-null   object 
 1   Open       1009 non-null   float64
 2   High       1009 non-null   float64
 3   Low        1009 non-null   float64
 4   Close      1009 non-null   float64
 5   Adj Close  1009 non-null   float64
 6   Volume     1009 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 55.3+ KB
```

Statistical information of the dataset

In [6]:
```
df.describe()
```

Out[6]:

	Open	High	Low	Close	Adj Close	Volume
count	1009.000000	1009.000000	1009.000000	1009.000000	1009.000000	1.009000e+03
mean	419.059673	425.320703	412.374044	419.000733	419.000733	7.570685e+06
std	108.537532	109.262960	107.555867	108.289999	108.289999	5.465535e+06
min	233.919998	250.649994	231.229996	233.880005	233.880005	1.144000e+06
25%	331.489990	336.299988	326.000000	331.619995	331.619995	4.091900e+06
50%	377.769989	383.010010	370.880005	378.670013	378.670013	5.934500e+06
75%	509.130005	515.630005	502.529999	509.079987	509.079987	9.322400e+06
max	692.349976	700.989990	686.090027	691.690002	691.690002	5.890430e+07

In [7]:
```
df
```

Out[7]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900
...	...	...	...	...	...	...	...
1004	2022-01-31	401.970001	427.700012	398.200012	427.140015	427.140015	20047500
1005	2022-02-01	432.959991	458.480011	425.540009	457.130005	457.130005	22542300
1006	2022-02-02	448.250000	451.980011	426.480011	429.480011	429.480011	14346000
1007	2022-02-03	421.440002	429.260010	404.279999	405.600006	405.600006	9905200
1008	2022-02-04	407.309998	412.769989	396.640015	410.170013	410.170013	7782400

1009 rows × 7 columns

Coverting the date column to datetime format

In [8]:
```
df['Date']=pd.to_datetime(df['Date'])
```

In [9]:
```
df
```

Out[9]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900
...	...	...	...	...	...	...	...
1004	2022-01-31	401.970001	427.700012	398.200012	427.140015	427.140015	20047500
1005	2022-02-01	432.959991	458.480011	425.540009	457.130005	457.130005	22542300
1006	2022-02-02	448.250000	451.980011	426.480011	429.480011	429.480011	14346000
1007	2022-02-03	421.440002	429.260010	404.279999	405.600006	405.600006	9905200
1008	2022-02-04	407.309998	412.769989	396.640015	410.170013	410.170013	7782400

1009 rows × 7 columns

Assigning the features and labels

In [10]:
```
x=df[['Open','High','Low','Volume']]
y=df['Close']
```

Calling and storing LinearRegression function and storing it in lmr variable

In [11]:
```
lmr=LinearRegression()
```

splitting the data into training and testing

In [12]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

fitting the model to training data

In [13]:
```
lmr.fit(x_train,y_train)
```

Out[13]:
```
LinearRegression()
```

predicting the test data

In [14]:
```
predicted=lmr.predict(x_test)
```

Detrmining the model performance

In [15]:
```
lmr.score(x_test,y_test)
```

Out[15]:
```
0.998433742993808
```

In [16]:
```
df2=pd.DataFrame({'Actual':y_test,'predicted':predicted})
```

Actual data vs Predicted data

In [17]:
```
df2
```

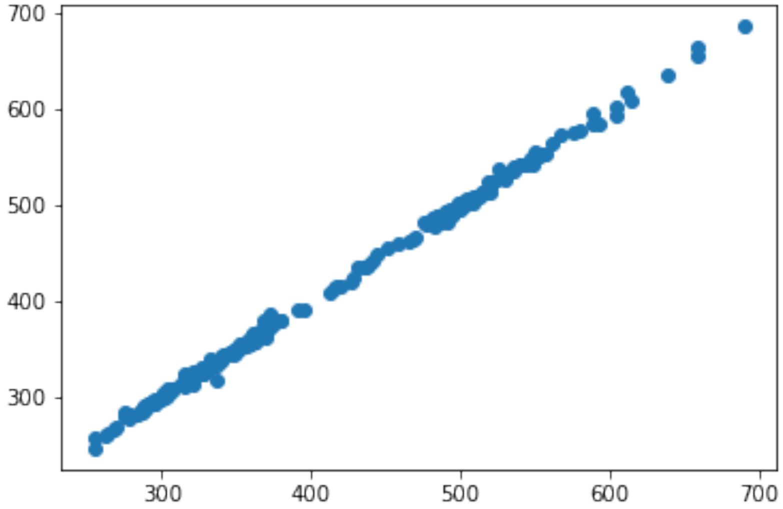
Out[17]:

	Actual	predicted
993	519.200012	525.020417
153	368.149994	370.057806
45	303.670013	310.015756
845	491.899994	492.561802
305	374.230011	375.778946
...	...	...
911	589.349976	584.325210
197	286.730011	285.084969
33	300.940002	303.870805
715	515.780029	512.165472
524	368.970001	362.265091

202 rows × 2 columns

```
In [19]: plt.scatter(y_test,predicted)

Out[19]: <matplotlib.collections.PathCollection at 0x7f7f20796d60>
```



Mean absolute error

```
In [20]: from sklearn.metrics import mean_absolute_error
         print("Mean_absolute_error:",mean_absolute_error(y_test,predicted))

         Mean_absolute_error: 2.9897047333430926
```

Mean squared error

```
In [21]: from sklearn.metrics import mean_squared_error
         print("Mean_squared_error",mean_squared_error(y_test,predicted))

         Mean_squared_error 16.236493786097043
```