TATA MOTORS

Importing Required Libraries

In [106…

```
import numpy as np
import pandas as pd
import seaborn as sns
sns.set(context="notebook", palette="Spectral", style = 'darkgrid' ,font_scale = 1.5, c
from sklearn import preprocessing
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

Loading Data for TATA MOTORS

In [107…

```
data = pd.read_csv("C:/Users/Galaxy star Gopi/Downloads/TTM.csv")
```

In [108…

```
data.head()
```

Out[108…

	Date	Open	High	Low	Close	Adj Close	Volume
0	2020-01-02	13.48	13.57	13.45	13.52	13.52	1649500
1	2020-01-03	13.45	13.47	13.13	13.18	13.18	1058800
2	2020-01-06	12.94	13.03	12.85	13.03	13.03	1203400
3	2020-01-07	12.86	12.89	12.75	12.82	12.82	1008300
4	2020-01-08	12.79	12.90	12.66	12.86	12.86	1052500

In [109…

```
data.tail()
```

Out[109…

	Date	Open	High	Low	Close	Adj Close	Volume
735	2022-12-01	27.000000	27.240000	26.760000	26.809999	26.809999	650500
736	2022-12-02	26.100000	26.570000	26.100000	26.549999	26.549999	1104100
737	2022-12-05	25.760000	26.000000	25.400000	25.549999	25.549999	717300
738	2022-12-06	25.450001	25.520000	24.790001	25.190001	25.190001	1662400
739	2022-12-07	24.809999	25.200001	24.809999	24.910000	24.910000	752700

In [110…

```
data.isnull().sum()
```

```
Date         0
```

Out[110…
```
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

In [111…
```
data = data.dropna()
```

In [112…
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 740 entries, 0 to 739
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       740 non-null    object 
 1   Open       740 non-null    float64
 2   High       740 non-null    float64
 3   Low        740 non-null    float64
 4   Close      740 non-null    float64
 5   Adj Close  740 non-null    float64
 6   Volume     740 non-null    int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 46.2+ KB
```

In [113…
```
data.describe()
```

Out[113…

	Open	High	Low	Close	Adj Close	Volume
count	740.000000	740.000000	740.000000	740.000000	740.000000	7.400000e+02
mean	19.837081	20.078973	19.611635	19.858419	19.858419	1.572788e+06
std	9.059902	9.155982	8.992796	9.080184	9.080184	1.092182e+06
min	4.200000	4.240000	3.920000	4.080000	4.080000	2.191000e+05
25%	10.352500	10.485000	10.225000	10.315000	10.315000	7.800000e+05
50%	21.119999	21.350001	20.930000	21.269999	21.269999	1.234800e+06
75%	27.582500	27.909999	27.342500	27.677500	27.677500	2.018625e+06
max	35.099998	35.380001	34.910000	34.939999	34.939999	7.612300e+06

Visualize close value from 2020 to 2022

In [114…
```
plt.figure(figsize=[12, 5]); # Set dimensions for figure
data.plot(x='Date', y='Close', figsize = (14, 6), legend = True, color='g')
plt.title('Close')
plt.ylabel('Price')
plt.xlabel('Date')
plt.xticks(rotation=90)
```

```
plt.grid(True)
plt.show()
```

```
<Figure size 864x360 with 0 Axes>
```

In [115…

```
import plotly.graph_objs as go
```

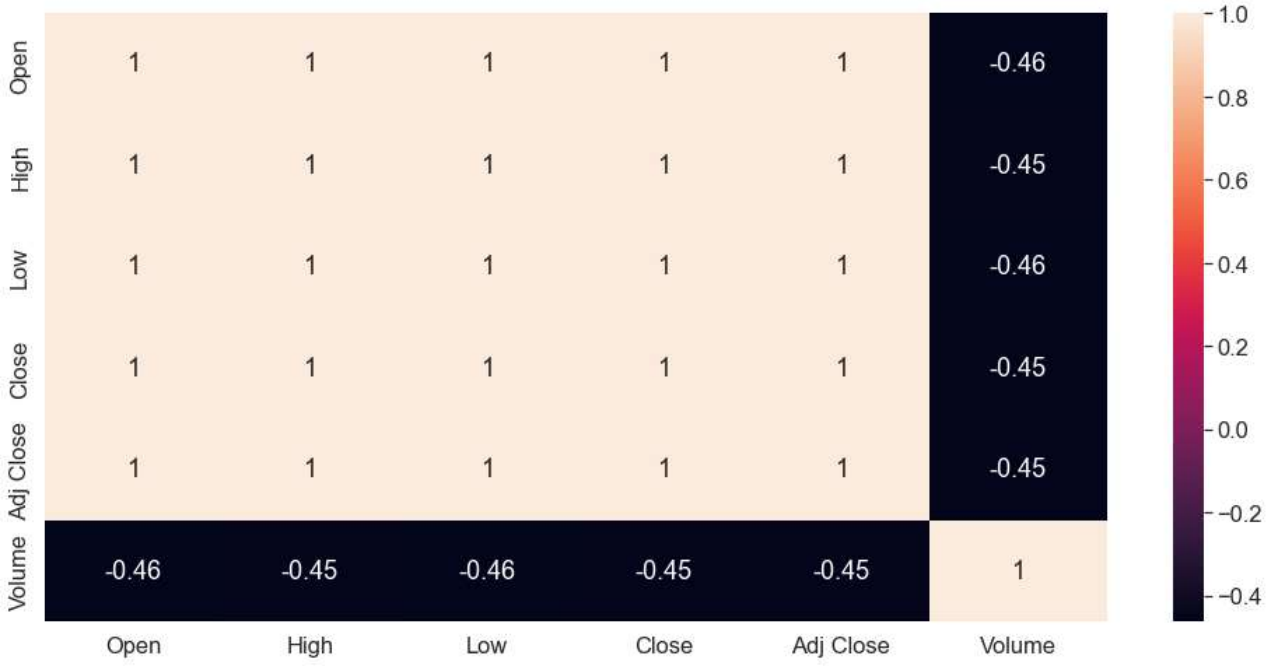
In [116…

```
#Declare figure
fig = go.Figure()

#add a trace
fig.add_trace(go.Scatter(x=data.Date, y=data['Close']))
#Update X and Y axis with title
fig.update_xaxes(
    title = 'Date',rangeslider_visible=True
)

fig.update_yaxes(
    title = 'Stock price'
)

#Show
fig.show()
```

Now see some correalations between data

In [117…

```
import seaborn as sns
plt.figure(1 , figsize = (17 , 8))
cor = sns.heatmap(data.corr(), annot = True)
```



Separate the Input and Output Columns

In [118…

```
X = data[['High','Low','Open','Volume']].values
y = data['Close'].values
```

In [119…
```
X
```

Out[119…
```
array([[1.3570000e+01, 1.3450000e+01, 1.3480000e+01, 1.6495000e+06],
       [1.3470000e+01, 1.3130000e+01, 1.3450000e+01, 1.0588000e+06],
       [1.3030000e+01, 1.2850000e+01, 1.2940000e+01, 1.2034000e+06],
       ...,
       [2.6000000e+01, 2.5400000e+01, 2.5760000e+01, 7.1730000e+05],
       [2.5520000e+01, 2.4790001e+01, 2.5450001e+01, 1.6624000e+06],
       [2.5200001e+01, 2.4809999e+01, 2.4809999e+01, 7.5270000e+05]])
```

In [120…
```
y
```

Out[120…
```
array([13.52    , 13.18    , 13.03    , 12.82    , 12.86    , 13.33    ,
       13.61    , 13.72    , 13.75    , 13.95    , 13.82    , 13.85    ,
       13.3     , 12.99    , 13.12    , 12.94    , 12.69    , 12.54    ,
       13.11    , 13.09    , 12.13    , 11.68    , 11.73    , 12.64    ,
       12.36    , 11.91    , 11.81    , 12.01    , 12.03    , 11.83    ,
       11.71    , 11.21    , 11.12    , 11.05    , 10.88    , 10.33    ,
       10.06    , 10.06    ,  9.72    ,  9.16    ,  8.96    ,  8.69    ,
        8.57    ,  8.02    ,  7.77    ,  6.71    ,  7.2     ,  6.45    ,
        5.29    ,  6.09    ,  5.08    ,  5.31    ,  4.91    ,  4.78    ,
        4.72    ,  4.33    ,  4.71    ,  4.84    ,  5.05    ,  4.6     ,
        4.55    ,  4.72    ,  4.11    ,  4.12    ,  4.08    ,  4.56    ,
        4.34    ,  4.61    ,  5.14    ,  4.86    ,  5.04    ,  4.69    ,
        4.93    ,  5.12    ,  5.11    ,  4.7     ,  5.01    ,  5.01    ,
        4.87    ,  5.      ,  5.08    ,  5.88    ,  5.92    ,  5.6     ,
        5.61    ,  5.47    ,  5.39    ,  5.5     ,  5.55    ,  5.85    ,
        6.04    ,  5.55    ,  5.66    ,  5.48    ,  5.58    ,  5.32    ,
        5.51    ,  5.57    ,  5.46    ,  5.61    ,  5.81    ,  5.75    ,
        5.71    ,  6.06    ,  6.61    ,  6.81    ,  6.68    ,  7.45    ,
        7.8     ,  7.32    ,  7.21    ,  6.55    ,  7.      ,  6.71    ,
        6.31    ,  6.19    ,  6.37    ,  6.59    ,  6.79    ,  6.89    ,
        6.71    ,  6.95    ,  6.6     ,  6.71    ,  6.57    ,  6.68    ,
        6.87    ,  7.53    ,  7.32    ,  7.06    ,  7.23    ,  7.35    ,
        7.09    ,  7.02    ,  7.02    ,  6.91    ,  6.95    ,  7.12    ,
        7.35    ,  7.14    ,  7.      ,  6.87    ,  6.79    ,  7.24    ,
        7.12    ,  7.03    ,  6.82    ,  7.45    ,  7.54    ,  7.82    ,
        7.85    ,  7.92    ,  8.21    ,  8.1     ,  8.38    ,  8.72    ,
        8.27    ,  8.21    ,  8.39    ,  8.33    ,  8.22    ,  8.12    ,
        8.24    ,  8.67    ,  9.65    ,  9.65    ,  9.77    ,  9.74    ,
        9.77    , 10.1     ,  9.98    , 10.1     ,  9.63    ,  9.67    ,
        9.57    ,  9.71    , 10.07    , 10.09    , 10.27    , 10.      ,
        9.88    ,  9.33    ,  9.11    ,  8.74    ,  8.43    ,  8.65    ,
        8.98    ,  8.87    ,  9.09    ,  9.15    ,  9.07    ,  9.2     ,
        9.7     ,  9.73    ,  9.67    ,  9.58    ,  9.34    ,  9.08    ,
        8.92    ,  8.81    ,  8.8     ,  8.61    ,  8.92    ,  8.84    ,
        9.12    ,  9.32    ,  9.06    ,  9.48    ,  8.85    ,  8.93    ,
        8.77    ,  9.08    ,  9.29    ,  9.28    ,  9.5     ,  9.49    ,
       10.2     , 10.03    , 10.21    , 10.07    , 10.03    , 10.49    ,
       10.9     , 11.63    , 11.58    , 11.29    , 11.46    , 11.7     ,
       11.78    , 12.38    , 11.94    , 12.57    , 12.64    , 12.66    ,
       12.51    , 12.52    , 12.45    , 12.38    , 11.99    , 12.12    ,
       11.99    , 12.34    , 12.57    , 12.31    , 12.18    , 11.19    ,
       11.06    , 11.65    , 11.8     , 12.5     , 12.36    , 12.6     ,
       12.6     , 12.76    , 13.3     , 13.34    , 13.45    , 13.67    ,
       15.9     , 16.549999, 16.27    , 16.889999, 17.559999, 17.68    ,
       18.940001, 20.059999, 19.57    , 18.549999, 18.450001, 17.82    ,
       18.559999, 17.809999, 19.52    , 22.17    , 22.620001, 22.309999,
       21.540001, 23.370001, 22.4     , 22.24    , 22.25    , 22.58    ,
```

```
22.209999, 22.360001, 21.870001, 21.43    , 20.959999, 21.93    ,
22.129999, 22.120001, 21.99    , 22.57    , 23.35    , 23.42    ,
22.5     , 22.32    , 21.67    , 21.74    , 21.91    , 22.469999,
21.969999, 21.889999, 21.809999, 21.33    , 20.709999, 21.219999,
20.93    , 20.67    , 19.92    , 19.84    , 20.4     , 20.370001,
20.52    , 20.790001, 21.030001, 20.93    , 20.84    , 20.450001,
20.940001, 21.24    , 18.559999, 19.66    , 20.049999, 20.17    ,
20.51    , 20.02    , 19.32    , 19.629999, 19.200001, 19.700001,
19.809999, 20.23    , 20.43    , 20.1     , 19.35    , 19.66    ,
19.459999, 19.709999, 20.389999, 20.58    , 21.049999, 21.35    ,
21.4     , 21.66    , 21.24    , 21.9     , 21.91    , 20.85    ,
21.17    , 21.299999, 21.52    , 21.370001, 21.76    , 22.18    ,
21.73    , 21.790001, 22.34    , 22.24    , 23.6     , 23.799999,
24.110001, 23.42    , 23.559999, 23.9     , 23.98    , 23.75    ,
23.620001, 22.98    , 22.18    , 22.6     , 22.68    , 22.41    ,
22.49    , 22.65    , 22.9     , 22.83    , 22.780001, 22.969999,
23.049999, 20.33    , 20.51    , 19.92    , 20.379999, 20.32    ,
20.4     , 20.559999, 20.52    , 20.379999, 20.09    , 20.440001,
20.52    , 19.889999, 19.610001, 19.700001, 19.27    , 19.34    ,
19.82    , 19.68    , 19.84    , 20.559999, 19.719999, 20.1     ,
20.110001, 19.870001, 19.74    , 20.040001, 20.5     , 20.5     ,
20.219999, 19.48    , 19.440001, 19.1     , 19.15    , 18.73    ,
19.120001, 19.17    , 18.9     , 19.75    , 19.76    , 19.68    ,
20.18    , 19.91    , 20.200001, 19.76    , 19.82    , 20.209999,
20.290001, 20.440001, 20.549999, 21.110001, 21.030001, 20.68    ,
19.889999, 20.469999, 21.200001, 22.07    , 21.799999, 22.809999,
22.07    , 22.33    , 22.41    , 22.780001, 22.719999, 23.129999,
22.34    , 25.75    , 26.35    , 28.059999, 31.68    , 34.200001,
33.93    , 34.189999, 33.790001, 32.       , 32.360001, 33.57    ,
32.27    , 32.200001, 33.169998, 32.360001, 32.040001, 31.58    ,
33.529999, 31.77    , 32.880001, 32.48    , 32.759998, 34.080002,
33.610001, 33.459999, 33.59    , 33.57    , 33.57    , 34.66    ,
34.939999, 33.139999, 33.450001, 32.240002, 32.84    , 32.880001,
30.360001, 30.959999, 30.540001, 31.280001, 31.969999, 31.450001,
30.799999, 31.85    , 32.610001, 32.139999, 32.75    , 31.950001,
31.690001, 32.150002, 31.629999, 30.309999, 29.469999, 30.379999,
31.5     , 31.209999, 31.780001, 31.719999, 31.549999, 31.33    ,
32.09    , 33.48    , 32.23    , 31.950001, 32.849998, 32.880001,
33.630001, 34.130001, 34.32    , 34.       , 34.5     , 33.779999,
34.299999, 34.029999, 32.619999, 31.51    , 31.9     , 31.75    ,
32.700001, 32.77    , 33.759998, 33.400002, 33.66    , 33.110001,
33.099998, 32.799999, 32.990002, 34.029999, 33.360001, 32.200001,
31.110001, 33.200001, 33.049999, 32.970001, 32.669998, 31.549999,
31.030001, 29.360001, 30.799999, 30.290001, 28.799999, 29.889999,
28.08    , 26.33    , 24.75    , 24.9     , 26.639999, 26.120001,
26.200001, 25.77    , 27.6     , 28.459999, 28.08    , 28.59    ,
27.82    , 28.610001, 28.15    , 28.58    , 28.4     , 28.540001,
29.15    , 28.25    , 27.950001, 29.219999, 29.690001, 29.51    ,
29.65    , 29.219999, 30.15    , 29.32    , 28.309999, 27.98    ,
27.75    , 28.549999, 28.190001, 28.83    , 28.91    , 28.09    ,
27.969999, 27.67    , 27.75    , 28.450001, 28.209999, 28.290001,
28.549999, 28.07    , 27.299999, 26.530001, 25.41    , 25.459999,
24.459999, 24.120001, 26.389999, 26.09    , 27.530001, 26.110001,
26.01    , 26.76    , 27.379999, 27.26    , 27.24    , 28.01    ,
28.190001, 28.5     , 28.629999, 28.799999, 27.5     , 27.799999,
28.17    , 27.889999, 27.15    , 26.65    , 25.379999, 25.99    ,
26.809999, 24.780001, 25.17    , 26.08    , 25.360001, 25.879999,
26.67    , 26.379999, 26.24    , 26.309999, 25.870001, 26.32    ,
26.139999, 26.42    , 27.700001, 27.9     , 27.26    , 27.129999,
26.620001, 26.92    , 27.82    , 27.99    , 28.629999, 28.190001,
```

```
28.719999, 28.190001, 28.01    , 27.549999, 27.51    , 28.190001,
28.52    , 30.58    , 30.360001, 29.940001, 29.57    , 29.219999,
29.59    , 29.540001, 30.84    , 29.76    , 30.41    , 30.360001,
31.02    , 30.41    , 30.629999, 29.280001, 28.25    , 28.889999,
29.24    , 29.200001, 28.58    , 28.6     , 28.99    , 29.      ,
29.26    , 28.59    , 28.26    , 28.360001, 27.879999, 28.139999,
28.75    , 28.02    , 28.09    , 27.610001, 26.77    , 27.040001,
26.889999, 26.25    , 26.25    , 25.469999, 24.280001, 24.370001,
24.809999, 23.73    , 24.26    , 24.360001, 25.450001, 25.25    ,
24.959999, 24.27    , 23.610001, 23.469999, 23.67    , 24.309999,
23.49    , 24.209999, 24.290001, 23.76    , 23.58    , 24.1     ,
24.51    , 24.370001, 24.75    , 24.76    , 25.26    , 25.01    ,
25.65    , 25.27    , 25.32    , 26.440001, 26.959999, 27.4     ,
26.4     , 26.33    , 26.559999, 27.1     , 27.030001, 25.85    ,
25.58    , 25.690001, 25.540001, 25.940001, 25.92    , 26.389999,
26.26    , 26.389999, 27.299999, 26.809999, 26.549999, 25.549999,
25.190001, 24.91    ])
```

Spliting the Train and Test data

In [121…
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=1)
```

Linear Regression Model

In [122…
```
regressor = LinearRegression()
```

Fitting the data in the Model

In [123…
```
regressor.fit(X_train, y_train)
```

Out[123… ▾ LinearRegression

LinearRegression()

In [124…
```
print(regressor.coef_)
```

```
[ 8.02652317e-01  7.61298177e-01 -5.64618505e-01  8.42314631e-09]
```

In [125…
```
print(regressor.intercept_)
```

```
-0.004670328871611673
```

Prediction

In [126…
```
predicted = regressor.predict(X_test)
```

In [127…
```
print(predicted)
```

```
[27.85204889 13.60755375 33.49643255 26.74641226 21.01479556 21.43644878
 19.02905216  9.03533775 29.14428868 19.05675419 32.42793425  6.66135376
 26.10698769 24.06552085 20.44154439 13.67455739 21.12679618 32.77983008
 26.57792703 27.41429744 30.82011011  6.54648294 28.7302381  13.00925117
 22.25731964 28.19747164 34.48099466  9.21976264 20.10864034 24.29410683
 23.84043903 19.97178836 19.84284987  4.83341686  4.42901872 23.57580459
 20.36256066  5.8586617  13.3668827   4.79628289 30.35557898 26.95568155
 21.75749026 29.63843737 32.73971559 12.72354534 31.52088745  6.96908777
  5.56599273  9.51141486  5.58522383 25.66052424 20.99643546  6.66083552
 28.82519838 33.72536873 27.65063005  8.39698699 28.73649311  6.65981466
 27.87719023  4.6736772   9.89085542 21.7293343   8.8395071  28.21320111
 23.44691371  5.31737784 19.40721423 28.05975704 20.07738202  5.79810563
 31.90126286 21.86017993 22.70119214 17.75655865 28.01594304  4.71658257
 24.64354621 20.2526525  20.75837026  8.70964987 12.79556888 28.55048282
  7.51399804 23.01443975 32.54176689  4.53120727 22.19630561  8.60468407
 12.29054678 25.31963994  4.72341759 25.94015639 26.52538228 12.50309734
 20.11285719 26.92200946 30.43138421 34.2243431  10.26945868 22.49252598
  8.97630225 30.76604518 12.68147785  8.73287253 25.61747792 33.69928683
 12.68924794 28.24325863  7.74397266 27.96411414 25.66273284 21.74669971
 32.39014295 13.52961862 11.84374796 26.99269762  6.73756943  5.18025354
 22.26626353 23.95968625  4.54703684 20.37539144 25.87755509 19.57850079
 22.35120253 28.16299771 19.10795515  8.12774245 26.39981373 28.38147697
 34.68776987 33.8780724  28.91432707 12.98049578 27.28676071 21.80442674
 34.73836965  5.55673402 34.67427361 35.16300462 28.00539014 26.46444117
 29.2092218  23.77864207 26.38617072  7.19163275 29.5075463  23.50877881
 19.47702274 25.02201388 20.37328874  8.73389578 23.13650912  5.18712343
  4.96943681  9.49743817  8.37413039  7.20431984  9.6870441   9.48936123
  6.84204839 22.27045562 22.82047185 10.95798306 27.98152245 32.63370785
 26.45281032 24.18628696 24.41385742 33.97505352 30.59297091  6.94724435
  6.09605636 21.39933511 21.3442231   9.61881121  7.68175923 12.51792395
 27.75257362  9.64376118 27.65055557 11.13441262 10.25773392 19.6878079
 23.47954663 28.21167724  6.80247958 10.17924857 30.59112662 30.2693992
 19.8671498  13.82691061  5.54254447 20.3377499  33.10896727  7.45292415
 22.79192782 18.38770478 19.89576618 27.20676582 29.80599031 28.49149614
 26.32595437 20.81290214  9.72425358  6.97642856  5.7100614  11.17806998
 23.1804103  25.86024742 22.91457362 11.57507586 25.54107732 28.16169684
 20.6218749   6.30460576 22.63801401 11.55260226 19.86053156  5.3977922 ]
```

Combine the Actual and Predicted data

In [128…
```
data1 = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted' : predicted.flatten()})
```

In [129…
```
data1.head(20)
```

Out[129…

	Actual	Predicted
0	27.610001	27.852049
1	13.670000	13.607554
2	33.110001	33.496433
3	26.920000	26.746412
4	21.110001	21.014796

	Actual	Predicted
5	21.520000	21.436449
6	19.150000	19.029052
7	9.150000	9.035338
8	28.990000	29.144289
9	19.100000	19.056754
10	32.360001	32.427934
11	6.610000	6.661354
12	26.090000	26.106988
13	24.100000	24.065521
14	20.370001	20.441544
15	13.750000	13.674557
16	21.219999	21.126796
17	32.619999	32.779830
18	26.670000	26.577927
19	27.400000	27.414297

Mean Absolute Error

In [130…

```
import math
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test,predicted))
print('Mean Squared Error:', metrics.mean_squared_error(y_test,predicted))
print('Root Mean Squared Error:', math.sqrt(metrics.mean_squared_error(y_test,predicted
```

```
Mean Absolute Error: 0.10708610080860481
Mean Squared Error: 0.024500188221460183
Root Mean Squared Error: 0.1565253596752302
```

Plotting Graph

In [131…

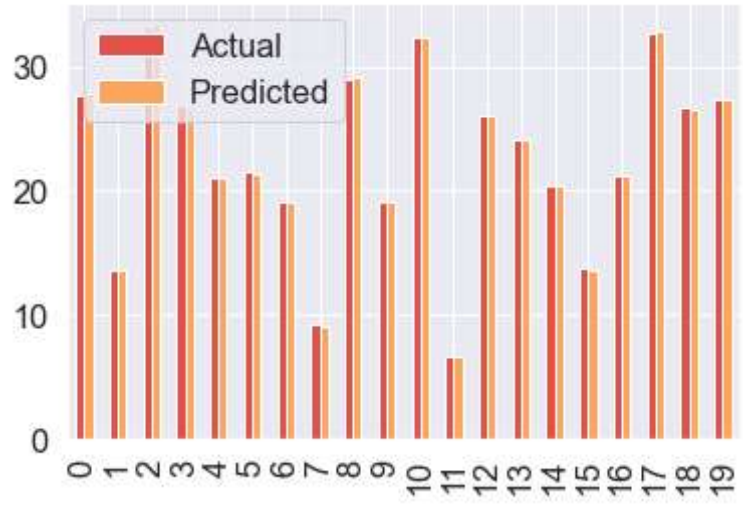
```
graph = data1.head(20)
```

In [132…

```
graph.plot(kind='bar')
```

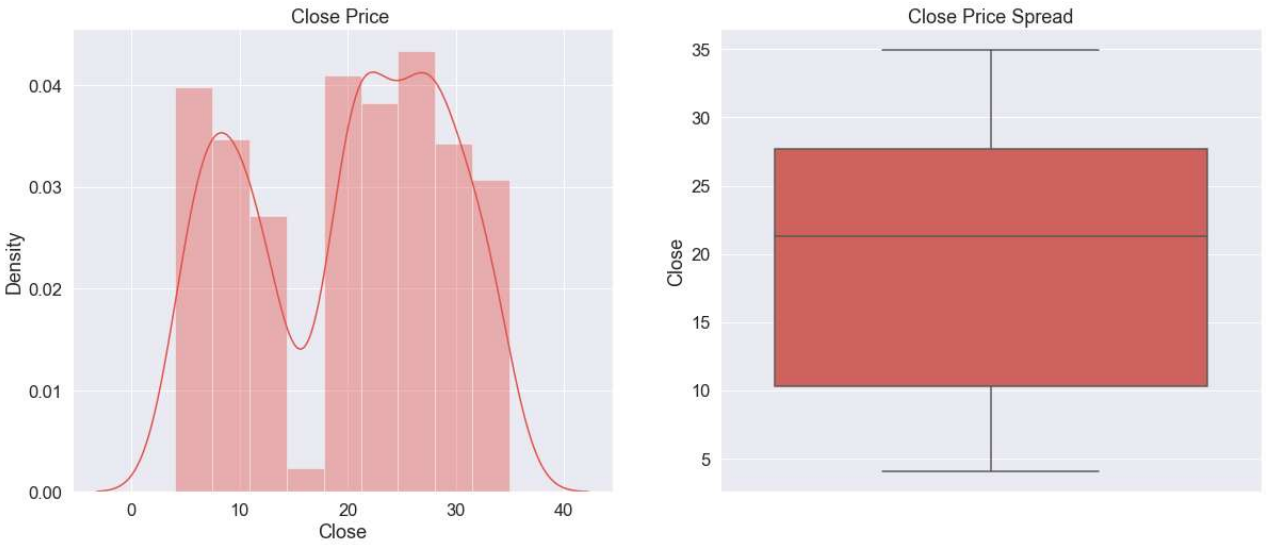
Out[132…

```
<AxesSubplot:>
```



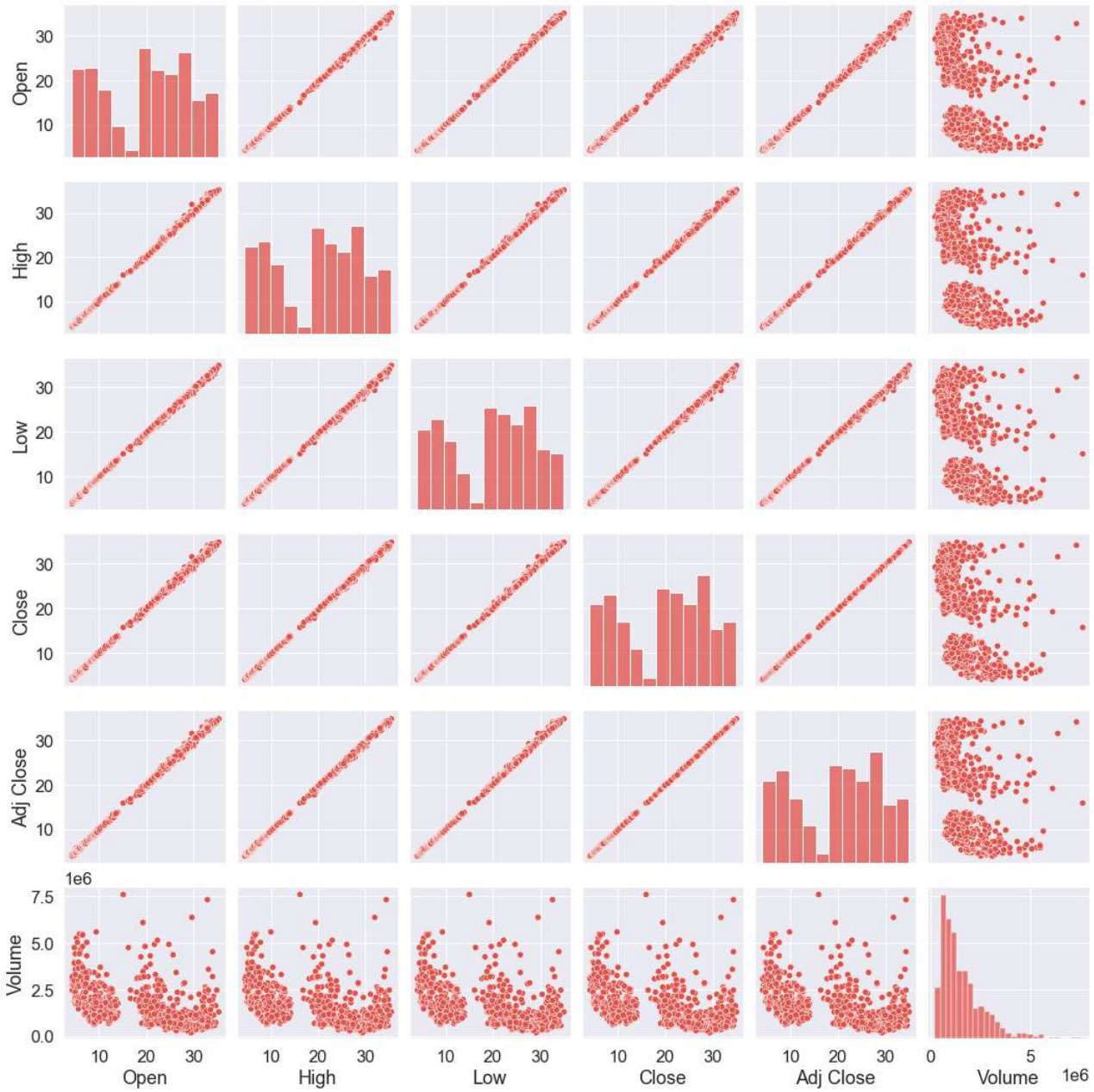
In [133…

```
plt.figure(figsize=(20,8))

plt.subplot(1,2,1)
plt.title('Close Price ')
sns.distplot(data.Close)

plt.subplot(1,2,2)
plt.title('Close Price Spread')
sns.boxplot(y=data.Close)

plt.show()
```



In [134…

```
data = sns.pairplot(data)
```



In [144…

```
predictions = regressor.predict(X_test)
print ("Type of the predicted object:", type(predictions))
print ("Size of the predicted object:", predictions.shape)
```

```
Type of the predicted object: <class 'numpy.ndarray'>
Size of the predicted object: (222,)
```

In [145…

```
plt.figure(figsize=(10,7))
plt.title("Actual vs. predicted prices",fontsize=25)
plt.xlabel("Actual test set Close prices",fontsize=18)
plt.ylabel("Predicted Close prices", fontsize=18)
plt.scatter(x=y_test,y=predictions)
```

Out[145…

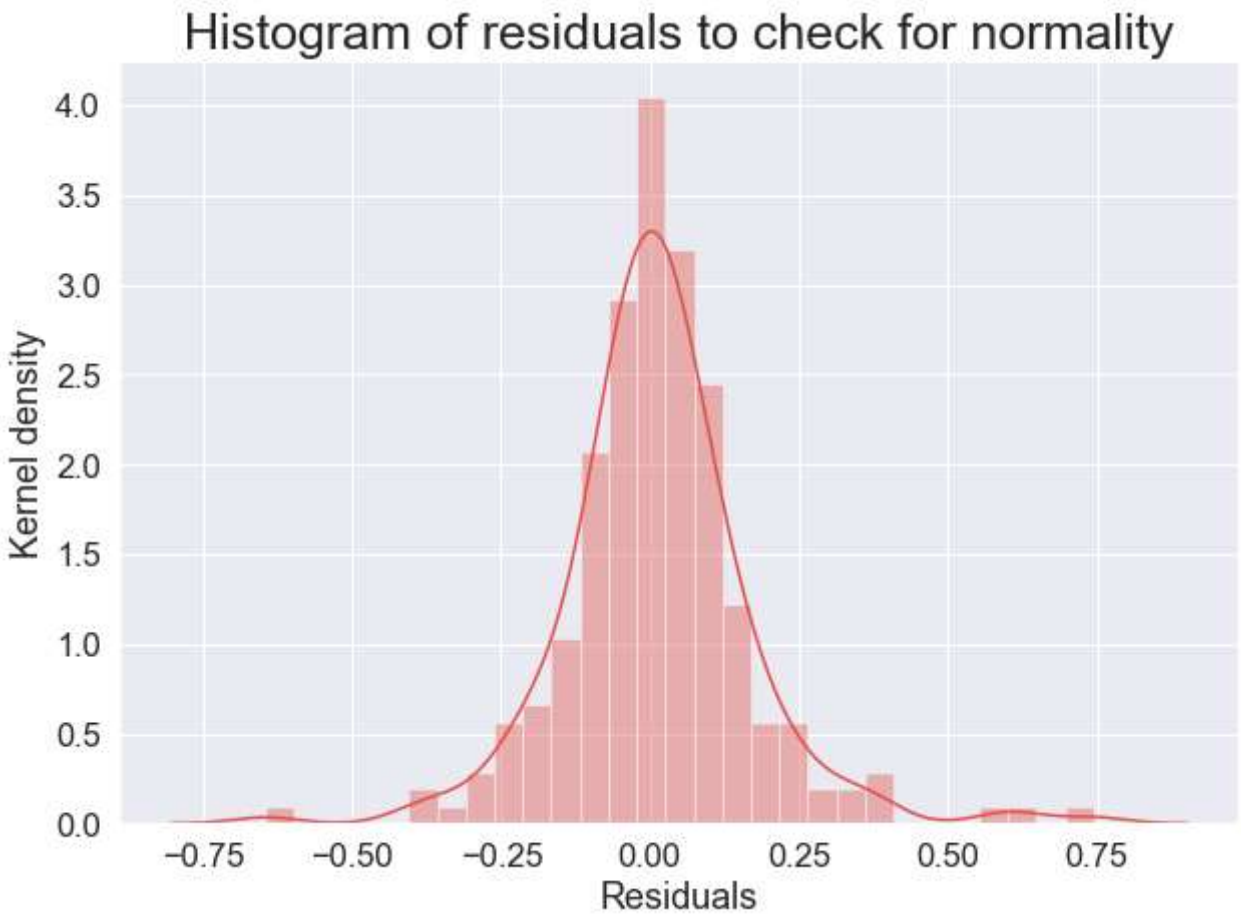
```
<matplotlib.collections.PathCollection at 0x2d10fff1760>
```

In [147…

```
plt.figure(figsize=(10,7))
plt.title("Histogram of residuals to check for normality",fontsize=25)
plt.xlabel("Residuals",fontsize=18)
plt.ylabel("Kernel density", fontsize=18)
sns.distplot([y_test-predictions])
```

Out[147…

```
<AxesSubplot:title={'center':'Histogram of residuals to check for normality'}, xlabel='R
esiduals', ylabel='Kernel density'>
```

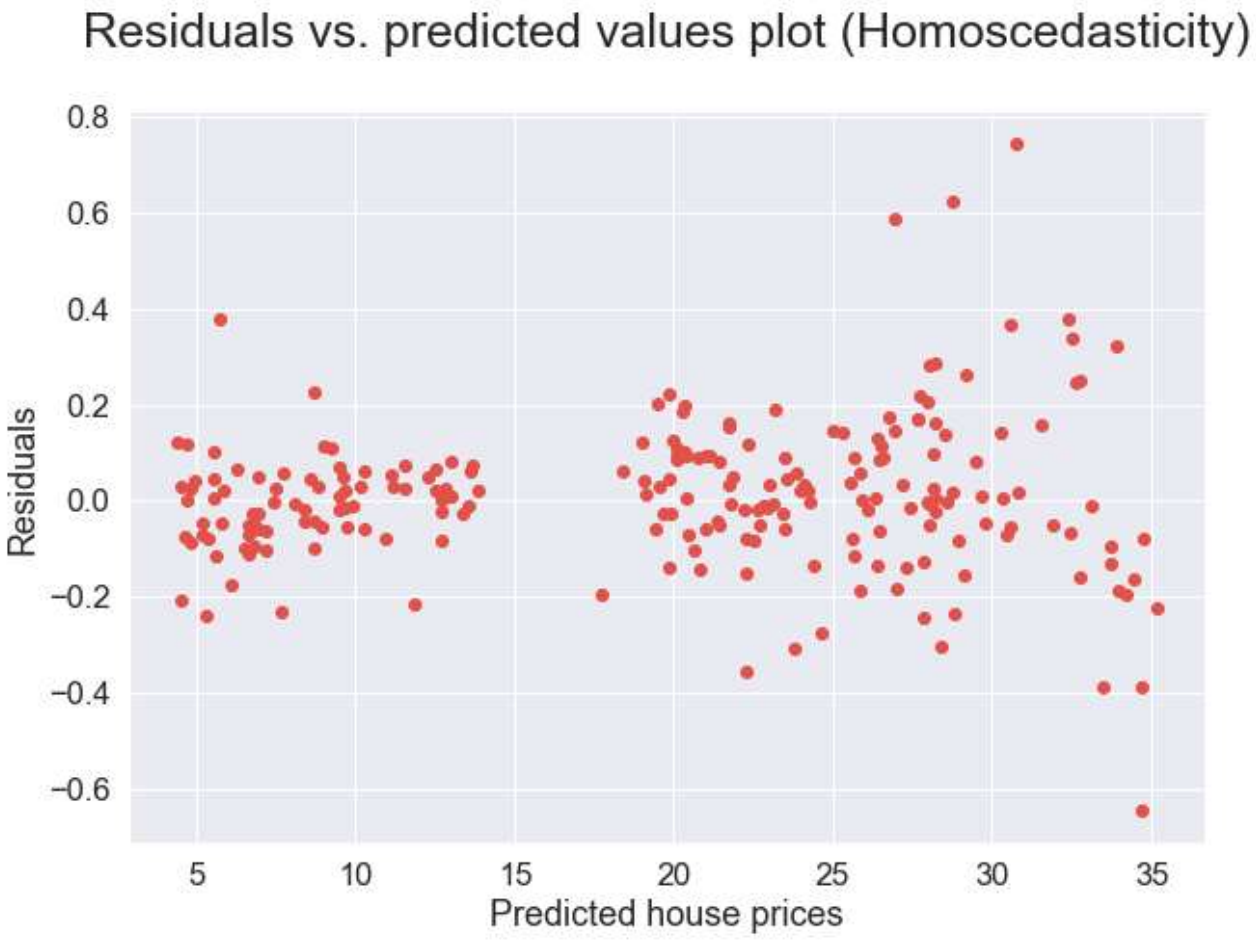


In [146…

```
plt.figure(figsize=(10,7))
plt.title("Residuals vs. predicted values plot (Homoscedasticity)\n",fontsize=25)
plt.xlabel("Predicted house prices",fontsize=18)
plt.ylabel("Residuals", fontsize=18)
plt.scatter(x=predictions,y=y_test-predictions)
```

Out[146…

```
<matplotlib.collections.PathCollection at 0x2d10e95e4f0>
```



In [153…

```
plt.figure(figsize=(10,7))
plt.title("Actual vs. predicted prices",fontsize=25)
plt.xlabel("Actual test set Close prices",fontsize=18)
plt.ylabel("Predicted Close prices", fontsize=18)
plt.scatter(x=y_test,y=predictions)
```

Out[153…

```
<matplotlib.collections.PathCollection at 0x2d10ee39340>
```

In [143…

```
data1.head(20)
```

Out[143…

	Actual	Predicted
0	27.610001	27.852049
1	13.670000	13.607554
2	33.110001	33.496433
3	26.920000	26.746412
4	21.110001	21.014796
5	21.520000	21.436449
6	19.150000	19.029052
7	9.150000	9.035338
8	28.990000	29.144289
9	19.100000	19.056754
10	32.360001	32.427934
11	6.610000	6.661354
12	26.090000	26.106988
13	24.100000	24.065521
14	20.370001	20.441544

	Actual	Predicted
15	13.750000	13.674557
16	21.219999	21.126796
17	32.619999	32.779830
18	26.670000	26.577927
19	27.400000	27.414297

In [ ]: