Stock Price Prediction Using Linear Regression

Importing Required Libraries

In [1]:
```
import numpy as np
import pandas as pd
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

In [2]:
```
data = pd.read_csv("TSLA.csv")
```

Let's See The Data

In [3]:
```
data.head()
```

Out[3]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2010-06-30	5.158	6.084	4.660	4.766	4.766	85935500
1	2010-07-01	5.000	5.184	4.054	4.392	4.392	41094000
2	2010-07-02	4.600	4.620	3.742	3.840	3.840	25699000
3	2010-07-06	4.000	4.000	3.166	3.222	3.222	34334500
4	2010-07-07	3.280	3.326	2.996	3.160	3.160	34608500

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2579 entries, 0 to 2578
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       2579 non-null   object 
 1   Open       2579 non-null   float64
 2   High       2579 non-null   float64
 3   Low        2579 non-null   float64
 4   Close      2579 non-null   float64
 5   Adj Close  2579 non-null   float64
 6   Volume     2579 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 141.2+ KB
```

In [5]:
```
data.describe()
```

Out[5]:

	Open	High	Low	Close	Adj Close	Volume
count	2579.000000	2579.000000	2579.000000	2579.000000	2579.000000	2.579000e+03
mean	49.206686	50.301806	48.073117	49.253279	49.253279	3.078217e+07
std	57.934102	59.888383	55.852349	58.119783	58.119783	2.855717e+07

min	3.228000	3.326000	2.996000	3.160000	3.160000	5.925000e+05
25%	7.159000	7.268000	6.989000	7.153000	7.153000	1.047400e+07
50%	44.001999	44.660000	43.301998	43.924000	43.924000	2.413100e+07
75%	59.339000	60.171000	57.841000	59.020000	59.020000	3.979150e+07
max	502.140015	502.489990	470.510010	498.320007	498.320007	3.046940e+08

Separate the Input and Output Columns

In [6]:
```
X = data[['High','Low','Open','Volume']].values
y = data['Close'].values
```

In [7]:
```
X
```

Out[7]:
```
array([[6.08400000e+00, 4.66000000e+00, 5.15800000e+00, 8.59355000e+07],
       [5.18400000e+00, 4.05400000e+00, 5.00000000e+00, 4.10940000e+07],
       [4.62000000e+00, 3.74200000e+00, 4.60000000e+00, 2.56990000e+07],
       ...,
       [4.12149994e+02, 3.75880005e+02, 4.05160004e+02, 9.50742000e+07],
       [3.99500000e+02, 3.51299988e+02, 3.63799988e+02, 9.65611000e+07],
       [4.08730011e+02, 3.91299988e+02, 3.93470001e+02, 6.70684000e+07]])
```

In [8]:
```
y
```

Out[8]:
```
array([  4.766    ,   4.392    ,   3.84     , ..., 380.359985, 387.790009,
       407.339996])
```

Spliting the Train and Test data

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=1)
```

Linear Regression Model

In [10]:
```
regressor = LinearRegression()
```

Fitting the data in the Model

In [11]:
```
regressor.fit(X_train, y_train)
```

Out[11]:
```
LinearRegression()
```

In [12]:
```
print(regressor.coef_)
```

```
[ 8.87095614e-01  5.85513083e-01 -4.76088446e-01 -1.00579987e-08]
```

In [13]:
```
print(regressor.intercept_)
```

```
0.21562505873461646
```

Prediction

In [14]:
```
predicted = regressor.predict(X_test)
```

In [15]:
```
print(predicted)
```

```
[ 48.28710377   5.46796473  67.81519558   6.68986367  50.57475389
  50.15802018  41.85593245  24.42517644  49.54803051  43.90230937
 298.79298494  37.51402064   7.85499937  43.74164349  54.54668474
   4.86374123  43.05771831  62.23745654   6.62671136  50.86453905
 162.1777844   54.01208147  26.89490796  17.34867617  67.53108095
  45.30737392  72.83821008  49.06133251 143.27239735  51.92928264
  50.70256477  29.52666431  49.57760378  47.45345367  32.76661723
 155.05339912  62.70618124   4.36559772  43.69638043  27.58513271
  58.69012648  38.24953676  38.95889794  38.59116287  23.66760728
  39.00734263   3.45184404  40.98080408   8.01183052  52.44962283
   3.97031791   5.74775757   3.01063992  42.26288089  33.17485593
   7.30407958   7.21164416  49.53701516 449.58142716   8.24903154
 130.87602639  47.09386717  70.10875987  41.83007476  43.81371492
  55.29270419 126.86476192  64.01792297  48.65801981  67.90733784
   5.13970988  58.15189852  39.66296045  41.28471972   6.80734247
  37.21190136  34.89873433   5.68303249  53.94065467  45.24036802
   5.59574717  72.51882286  54.85363885  38.41369166   5.77993698
 201.50246088  10.49318868   4.24774584  43.8332091   39.75951296
  44.98932812   5.88711956   4.19142633   5.02904587  65.04301966
  68.16580518  53.37973517   4.82523607  45.56905848   6.09063904
  51.27468185   7.01232326  46.50660146  71.36054459 308.39589612
  44.63208319  61.10491501  43.33698154   5.98953677   4.92801574
  30.72113072   5.62975748  50.09910505  63.97150802   7.11351677
 300.54945572  30.19549818  24.6952598   50.23800452   6.30575923
  46.66445875   7.06632817  70.79974214  50.49665861  46.29016504
  63.62603447   5.72607689  45.90564671  68.92889813  34.99666314
  54.33543259  45.59205896  46.86903562  57.37367338  69.35395171
  39.67672604  52.34205959  28.89503738   7.49530193  45.75050982
  68.36235385   5.26352945  10.88658662  39.04239158  43.76939747
  46.44753976  36.96739673   6.3825037   43.46867372  44.0258889
  51.31506025  50.20852664  85.11410737  66.02233883  44.11538723
  65.6870657   61.18894811  53.64226664  35.721883    37.90672908
  50.53029385  51.62788713  61.52553214   6.54836495   4.27531368
   5.57215464  53.19905099  35.78783817  59.67787761   5.88243989
  48.75391875  32.91963387  69.86835041   7.18737541  24.21586148
  94.88349896   4.99253578  44.02207694   5.02363691  37.9593755
  41.73904034   5.7099585   73.53263696   6.22703768  62.70575856
  42.36896264  11.22338419  46.75236641  55.43794569  40.32449359
   6.47946342   5.15101866   6.12050196   6.00587183  67.60047025
 156.8572698   29.7772174   67.91447378  41.17466198  49.75003014
  34.74542313  76.11571414  42.30415196  39.40000595  42.13138816
  25.3296039   20.4311526   60.92933154  42.34977762   5.48890057
  45.12281417  33.57163573  46.39095547  42.40619638  38.66283614
  24.10790803  30.63575912  44.38516243  45.43514034   6.05576601
  37.48482133   6.61541699   7.68802739  96.87504246  45.45246971
   4.31376157   5.77404102  61.81397849 300.1517157  274.75209856
  50.88468118   6.91935624   6.7972283   56.4099336    6.06126022
  59.87809239   5.85518282  36.34798413 148.93643243  71.83133876
  56.02688617  35.88657035  46.16952348   7.32600944  71.19768244
  64.33220475  49.57469666  47.50285061  44.3908241   56.55273171
   6.20710463  58.84881358  42.42351881  42.32888816  33.79906077
  40.52050217  44.98289835   6.78837191   4.93507452  50.03157384
   4.7791613   63.67528454  24.69326783  45.56105165  58.23923032
  49.98367156   6.20645211  40.99012515  32.59549225  40.28142481
  48.210379    37.7258859   44.24753206  43.94696277   6.38263741
   5.53765952   5.43401002   7.13967432   4.21557009  42.3590692
   6.96036166  54.35871237  40.99290124  54.41512257  66.14130242
  68.96534624   6.71970313  68.27675162  51.00555646  47.80580795
   4.34851268   6.31249272  57.67209568  24.25854604   7.30210767
  59.60287711   6.66879824  65.89300636  47.33020272  53.15459932
  49.11186422  40.65826265  38.61650508  64.10405616  47.31160495
 161.0851126   96.81737733  55.81687524  18.73922495  50.0086286
   5.1190465   27.25923787  40.99186997  44.21567703 111.87874818
  50.23087312   6.10474673  55.47359654   6.95837619   5.68623921
   4.36735312  51.00448614  42.87988799  56.02430377  44.8875339
   7.14752034  61.91614989   6.51380566  65.64461888   5.6488913
```

63.46135626	174.48509297	52.86368823	39.49085284	5.7106024
48.78646358	6.98441958	6.33551307	6.69531357	73.17449733
6.46086338	40.89598739	8.8972896	49.72011737	5.99093983
5.97688798	37.87278174	5.79731365	5.54769735	49.76899916
31.27901185	52.24035437	5.39424905	62.48943706	7.53006185
55.56160269	6.19646971	6.00174752	46.95723617	6.74389152
44.74517209	69.80595999	6.8895122	7.97384323	36.6061583
5.66663795	5.57390848	61.88879002	66.71533813	44.44211138
48.8076207	68.33240872	44.08661939	47.31928654	41.99607457
59.18248087	6.81323195	51.67802495	51.21723337	38.64391845
46.20856502	53.91637856	39.28584027	162.81680926	64.35789451
5.93234801	45.48336601	38.93843042	61.33584282	59.00907146
67.75634059	8.98904355	69.82680424	7.01005313	44.55583799
51.89439057	65.99134578	163.3340212	5.08506302	427.73285437
49.71142281	39.50363919	62.49049671	37.57260788	69.72910014
114.15740808	134.13202754	154.97330435	392.06648505	6.69083514
58.01481686	43.16570753	5.73774379	46.40350962	71.75168545
45.56817943	199.68556895	50.60254767	39.39679134	62.00193509
67.53711091	52.71799032	6.12783714	35.35837833	42.00428665
113.18222912	40.27562075	9.14484009	49.40719631	67.38432021
57.03812349	43.50872332	52.37730854	5.66909542	39.14094088
40.23348284	27.49553674	36.38684954	40.62444249	6.34251977
56.53811592	62.40689061	20.39560423	46.4570898	6.980966
44.38936804	6.02810051	66.17552034	41.30263767	41.47382496
44.84962262	63.67054757	7.33776537	7.74080435	6.03293165
7.62508581	45.93402176	40.19201639	5.55418229	43.11271021
5.10662287	50.43540445	44.67402488	5.86147006	62.41708948
41.72760607	10.02971438	40.47144541	62.92459273	57.21948474
66.835943	6.40990561	62.64812367	5.72543684	68.16215023
68.99941552	63.77066033	362.90121359	44.93878025	52.43081158
105.52239734	58.17932873	83.22175786	23.27675523	39.61928582
39.92990382	69.0806285	63.14353057	4.52099517	49.25074386
6.21152374	46.90890893	47.39144014	46.45525918	5.48454754
7.89526755	5.27452811	5.6999415	5.9097257	68.67796817
61.11347369	70.43183066	4.61254281	6.07207872	6.08496809
322.27387873	45.47900781	59.97785203	41.14573768	4.29159763
67.22943221	5.44586257	6.99935477	7.78537903	5.86435255
74.08374864	453.36127078	5.77428442	270.32059317	76.55776995
45.2448615	46.62232039	45.13871515	101.53871298	50.1782705
6.77729661	32.95199243	66.8016726	38.92311392	51.55866639
38.77508337	42.26798186	6.59605961	5.65753582	5.91875722
41.43601135	32.38917	50.89996876	89.40602696	57.03412306
65.68298811	49.75277177	63.26506016	6.44453868	5.82318382
69.5748789	63.3786198	61.46532182	187.01090398	386.12887981
66.12029702	70.92043771	67.92861883	45.98236966	7.80113862
5.87882777	5.9222337	69.63113677	70.95810391	59.85286798
45.64980053	43.29630282	39.0279016	40.56678783	65.17828517
28.41281928	69.3617471	7.73594301	45.45688815	5.41660361
37.49412232	3.69821661	5.95819996	5.83892722	40.40428005
44.85362571	9.90682117	5.04171305	41.89020597	5.00076375
49.79052041	39.32694338	7.62378312	25.54147569	45.57220041
18.86584526	41.29948664	92.69678339	6.42732325	7.683705
17.45430922	59.37317746	67.83365063	327.60744723	49.45927674
5.34625285	4.75506601	7.65115676	42.04705055	152.04026234
67.18083712	44.31756409	37.70621921	5.77991338	4.45643008
33.17881488	60.45228063	37.7030823	72.77463582	40.08592778
50.4541062	46.09215209	39.24598939	45.95682924	7.91173309
33.22857578	7.05927443	50.43437775	40.48741999	58.27734915
7.01114918	6.3126144	53.93271661	37.17792641	52.16983129
6.46558556	45.48907972	66.12972857	4.28331647	5.30528769
52.26592976	102.76342819	70.1451074	68.35235522	4.39423314
5.58805072	66.12173623	5.71036195	49.79567556	403.90839258
48.07862116	47.15429132	4.80932624	5.58751351	47.99722749
7.04591435	49.67250556	48.07835112	23.66911347	4.48202752
28.57331602	41.35965258	4.23972851	6.46639666	51.21055917
64.71491883	5.1171447	47.30915047	44.22130975	40.67558871

```
 20.9335329   80.54423996  44.65118968  57.14556567  73.39007474
 42.68637339  62.26023607  61.74475715   7.11481121  54.17312703
 50.38022693  19.56809054   5.55986987  68.51619142  37.61581892
  6.37882344  48.2404555   23.51872495  29.41196554  45.4953433
 61.69529705  48.02691169  49.90104417   4.72326172   6.51870721
  6.98410915   3.3680518   35.23714328  68.96903845  42.54651975
 27.32224598  66.16995194  58.88323634  51.2284521    5.71737386
 45.87418892  47.62210068  42.54361467  70.50746758  47.46421845
  6.03777376  40.28139642 174.9567629   45.6033153   45.73374321
  6.45526916  49.01475024  52.4247576   43.0765485  162.23323062
  6.91240293  29.04055217  37.85235665   6.46258865   4.99026354
 50.3008805   73.19788408 160.25155235  52.91328659   5.01891412
  4.94247585 150.33808487   4.74791589  41.41850515   8.29441923
 25.54364412  68.14767868   5.86259032   6.03689519  44.31436614
 95.19809737  41.80396784   6.13342568  28.79711128  41.63914212
 43.17472417  37.91828389   4.45642772  39.54119397   6.75607609
  6.87828356   6.84523505  45.5076838    7.03254758  33.98774092
 49.70483556  64.08894678  40.5328481   34.87792315   7.03985488
  3.7111339   23.50356178  38.63978447   7.10243704  68.07540437
  6.02384503  74.29261713  66.37067193  49.50928026   5.95697306
 56.58131571   4.86994298  54.43700589  58.97986316  59.52562449
 71.51047009  45.70017427   5.81449043  51.98934658  38.23657466
 45.36271566  39.32858094   5.53560936 151.06458873]
```

Combine the Actual and Predicted data

In [16]:
```
data1 = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted' : predicted.flatten()})
```

In [17]:
```
data1.head(20)
```

Out[17]:

	Actual	Predicted
0	48.598000	48.287104
1	5.348000	5.467965
2	68.570000	67.815196
3	6.430000	6.689864
4	49.812000	50.574754
5	50.004002	50.158020
6	41.400002	41.855932
7	24.690001	24.425176
8	49.785999	49.548031
9	43.472000	43.902309
10	297.000000	298.792985
11	38.782001	37.514021
12	7.708000	7.854999
13	43.888000	43.741643
14	53.790001	54.546685
15	4.650000	4.863741
16	43.529999	43.057718
17	62.924000	62.237457

18	6.426000	6.626711
19	51.400002	50.864539

Mean Absolute Error

In [18]:
```
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: 0.4691606803432726
Mean Squared Error: 0.9033937289051058
Root Mean Squared Error: 0.9504702672388579
```

Plotting Graph

In [19]:
```
graph = data1.head(20)
```

In [20]:
```
graph.plot(kind='bar')
```

Out[20]:
```
<AxesSubplot:>
```