**import** pandas **as** pd

df**=**pd**.**read\_csv('C:\\Users\\Praneeth\\Downloads\\ADANIPORTS.csv')

df

| **Date** | **Symbol** | **Series** | **Prev Close** | **Open** | **High** | **Low** | **Last** | **Close** | **VWAP** | **Volume** | **Turnover** | **Trades** | **Deliverable Volume** | **%Deliverble** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2007-11-27 | MUNDRAPORT | EQ | 440.00 | 770.00 | 1050.00 | 770.00 | 959.0 | 962.90 | 984.72 | 27294366 | 2.687719e+15 | NaN | 9859619 | 0.3612 |
| **1** | 2007-11-28 | MUNDRAPORT | EQ | 962.90 | 984.00 | 990.00 | 874.00 | 885.0 | 893.90 | 941.38 | 4581338 | 4.312765e+14 | NaN | 1453278 | 0.3172 |
| **2** | 2007-11-29 | MUNDRAPORT | EQ | 893.90 | 909.00 | 914.75 | 841.00 | 887.0 | 884.20 | 888.09 | 5124121 | 4.550658e+14 | NaN | 1069678 | 0.2088 |
| **3** | 2007-11-30 | MUNDRAPORT | EQ | 884.20 | 890.00 | 958.00 | 890.00 | 929.0 | 921.55 | 929.17 | 4609762 | 4.283257e+14 | NaN | 1260913 | 0.2735 |
| **4** | 2007-12-03 | MUNDRAPORT | EQ | 921.55 | 939.75 | 995.00 | 922.00 | 980.0 | 969.30 | 965.65 | 2977470 | 2.875200e+14 | NaN | 816123 | 0.2741 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **3317** | 2021-04-26 | ADANIPORTS | EQ | 725.35 | 733.00 | 739.65 | 728.90 | 729.2 | 730.75 | 733.25 | 9390549 | 6.885658e+14 | 116457.0 | 838079 | 0.0892 |
| **3318** | 2021-04-27 | ADANIPORTS | EQ | 730.75 | 735.00 | 757.50 | 727.35 | 748.6 | 749.15 | 747.67 | 20573107 | 1.538191e+15 | 236896.0 | 1779639 | 0.0865 |
| **3319** | 2021-04-28 | ADANIPORTS | EQ | 749.15 | 755.00 | 760.00 | 741.10 | 743.4 | 746.25 | 751.02 | 11156977 | 8.379106e+14 | 130847.0 | 1342353 | 0.1203 |
| **3320** | 2021-04-29 | ADANIPORTS | EQ | 746.25 | 753.20 | 765.85 | 743.40 | 746.4 | 746.75 | 753.06 | 13851910 | 1.043139e+15 | 153293.0 | 1304895 | 0.0942 |
| **3321** | 2021-04-30 | ADANIPORTS | EQ | 746.75 | 739.00 | 759.45 | 724.50 | 726.4 | 730.05 | 743.35 | 12600934 | 9.366911e+14 | 132141.0 | 3514692 | 0.2789 |

3322 rows × 15 columns

In [3]:

df**.**head()

| **Date** | **Symbol** | **Series** | **Prev Close** | **Open** | **High** | **Low** | **Last** | **Close** | **VWAP** | **Volume** | **Turnover** | **Trades** | **Deliverable Volume** | **%Deliverble** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2007-11-27 | MUNDRAPORT | EQ | 440.00 | 770.00 | 1050.00 | 770.0 | 959.0 | 962.90 | 984.72 | 27294366 | 2.687719e+15 | NaN | 9859619 | 0.3612 |
| **1** | 2007-11-28 | MUNDRAPORT | EQ | 962.90 | 984.00 | 990.00 | 874.0 | 885.0 | 893.90 | 941.38 | 4581338 | 4.312765e+14 | NaN | 1453278 | 0.3172 |
| **2** | 2007-11-29 | MUNDRAPORT | EQ | 893.90 | 909.00 | 914.75 | 841.0 | 887.0 | 884.20 | 888.09 | 5124121 | 4.550658e+14 | NaN | 1069678 | 0.2088 |
| **3** | 2007-11-30 | MUNDRAPORT | EQ | 884.20 | 890.00 | 958.00 | 890.0 | 929.0 | 921.55 | 929.17 | 4609762 | 4.283257e+14 | NaN | 1260913 | 0.2735 |
| **4** | 2007-12-03 | MUNDRAPORT | EQ | 921.55 | 939.75 | 995.00 | 922.0 | 980.0 | 969.30 | 965.65 | 2977470 | 2.875200e+14 | NaN |  |  |

df**.**shape

(3322, 15)

df**.**isnull()**.**sum()

Date 0

Symbol 0

Series 0

Prev Close 0

Open 0

High 0

Low 0

Last 0

Close 0

VWAP 0

Volume 0

Turnover 0

Trades 866

Deliverable Volume 0

%Deliverble 0

dtype: int64

df**.**isnull()**.**any()

Date False

Symbol False

Series False

Prev Close False

Open False

High False

Low False

Last False

Close False

VWAP False

Volume False

Turnover False

Trades True

Deliverable Volume False

%Deliverble False

dtype: bool

df**.**info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 3322 entries, 0 to 3321

Data columns (total 15 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Date 3322 non-null object

1 Symbol 3322 non-null object

2 Series 3322 non-null object

3 Prev Close 3322 non-null float64

4 Open 3322 non-null float64

5 High 3322 non-null float64

6 Low 3322 non-null float64

7 Last 3322 non-null float64

8 Close 3322 non-null float64

9 VWAP 3322 non-null float64

10 Volume 3322 non-null int64

11 Turnover 3322 non-null float64

12 Trades 2456 non-null float64

13 Deliverable Volume 3322 non-null int64

14 %Deliverble 3322 non-null float64

dtypes: float64(10), int64(2), object(3)

memory usage: 389.4+ KB

df**.**describe()

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

I

print('lenghth of dataset:',len(df))

lenghth of dataset: 3322

**import** matplotlib.pyplot **as** plt

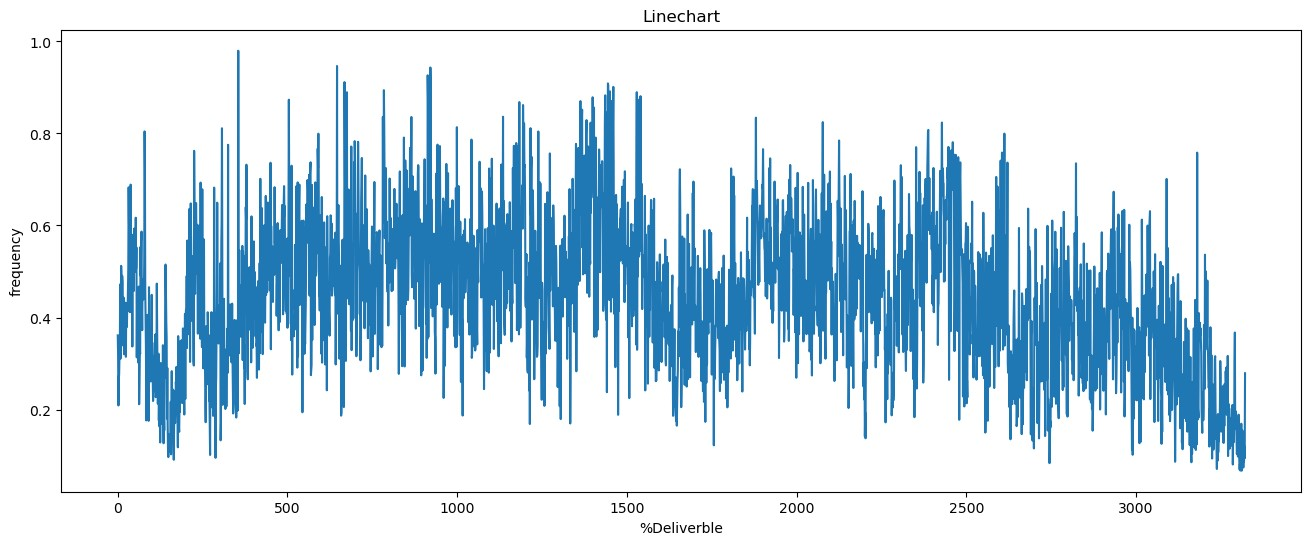
df['%Deliverble']**.**plot(figsize**=**(16,6))

plt**.**title('Linechart')

plt**.**xlabel('%Deliverble')

plt**.**ylabel('frequency')

plt**.**show()



x**=**df**.**drop(['Trades','Symbol','Series','Date','Open'],axis**=**1)

x

| Prev Close | **High** | **Low** | **Last** | **Close** | **VWAP** | **Volume** | **Turnover** | **Deliverable Volume** | **%Deliverble** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 440.00 | 1050.00 | 770.00 | 959.0 | 962.90 | 984.72 | 27294366 | 2.687719e+15 | 9859619 | 0.3612 |
| **1** | 962.90 | 990.00 | 874.00 | 885.0 | 893.90 | 941.38 | 4581338 | 4.312765e+14 | 1453278 | 0.3172 |
| **2** | 893.90 | 914.75 | 841.00 | 887.0 | 884.20 | 888.09 | 5124121 | 4.550658e+14 | 1069678 | 0.2088 |
| **3** | 884.20 | 958.00 | 890.00 | 929.0 | 921.55 | 929.17 | 4609762 | 4.283257e+14 | 1260913 | 0.2735 |
| **4** | 921.55 | 995.00 | 922.00 | 980.0 | 969.30 | 965.65 | 2977470 | 2.875200e+14 | 816123 | 0.2741 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **3317** | 725.35 | 739.65 | 728.90 | 729.2 | 730.75 | 733.25 | 9390549 | 6.885658e+14 | 838079 | 0.0892 |
| **3318** | 730.75 | 757.50 | 727.35 | 748.6 | 749.15 | 747.67 | 20573107 | 1.538191e+15 | 1779639 | 0.0865 |
| **3319** | 749.15 | 760.00 | 741.10 | 743.4 | 746.25 | 751.02 | 11156977 | 8.379106e+14 | 1342353 | 0.1203 |
| **3320** | 746.25 | 765.85 | 743.40 | 746.4 | 746.75 | 753.06 | 13851910 | 1.043139e+15 | 1304895 | 0.0942 |
| **3321** | 746.75 | 759.45 | 724.50 | 726.4 | 730.05 | 743.35 | 12600934 | 9.366911e+14 | 3514692 | 0.2789 |

3322 rows × 10 columns

y**=**df['Open']

y

0 770.00

1 984.00

2 909.00

3 890.00

4 939.75

...

3317 733.00

3318 735.00

3319 755.00

3320 753.20

3321 739.00

Name: Open, Length: 3322, dtype: float64

**from** sklearn.model\_selection **import** train\_test\_split

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,random\_state**=**0)

print(x\_train**.**shape)

print(x\_test**.**shape)

print(y\_train**.**shape)

print(y\_test**.**shape)

(2491, 10)

(831, 10)

(2491,)

(831,)

**from** sklearn.linear\_model **import** LinearRegression

regressor**=**LinearRegression()

regressor**.**fit(x\_train,y\_train)

LinearRegression()

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.  
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

print(regressor**.**coef\_)

[ 1.21048807e-01 9.02188308e-01 8.57417613e-01 -7.96874614e-02

-5.03159563e-01 -2.98514478e-01 5.85816992e-07 -8.03445272e-15

-6.69329092e-07 2.60132535e+00]

print(regressor**.**intercept\_)

-1.208988299101975

predicted**=**regressor**.**predict(x\_test)

predicted

Out[26]:

array([ 115.3791438 , 152.3900684 , 345.51876009, 237.86884907,

430.44935639, 160.63754617, 895.75500097, 361.92738707,

269.28518046, 151.19094501, 567.17857245, 356.30481078,

310.58438244, 615.37441103, 286.74954365, 272.58486197,

311.48521807, 1279.26351561, 152.49532814, 260.15813173,

697.37165272, 257.64047288, 373.62492704, 745.46066827,

358.27220066, 745.79214934, 305.14787887, 277.13142591,

154.537318 , 320.47753349, 184.58399994, 546.88486203,

386.25977976, 557.88368236, 1060.67323261, 486.05554471,

223.17363672, 303.56484198, 238.57961563, 151.98183235,

344.2021991 , 387.85799134, 372.85187594, 122.86193727,

297.32664787, 385.59856199, 397.81143041, 553.46397099,

329.87748619, 135.27437007, 535.53986147, 139.03614046,

352.94596117, 135.60175634, 724.4437263 , 793.80540934,

313.90643571, 125.6536819 , 299.5341966 , 372.43654102,

162.31028443, 314.15104497, 314.51310949, 592.56059923,

126.81426608, 112.87485634, 146.29591397, 343.44900051,

514.9326673 , 355.69817009, 397.58826567, 824.2

print(x\_test)

Prev Close High Low Last Close VWAP Volume \

1182 116.25 117.50 114.20 115.50 116.60 115.82 373291

1396 150.00 153.90 149.50 151.10 151.85 151.15 1328772

1773 346.90 349.45 340.90 344.25 345.15 344.53 2928237

2057 235.70 243.20 235.75 242.90 241.95 239.16 4606790

2520 428.25 435.50 424.50 428.15 428.85 429.36 2734628

... ... ... ... ... ... ... ...

346 405.30 427.80 402.00 409.90 410.55 416.57 634433

995 128.85 131.00 124.40 127.25 127.30 128.12 535841

2481 388.15 389.25 379.05 387.75 388.85 386.39 5100915

2808 383.05 386.00 377.50 380.80 379.70 381.00 1825681

1679 289.10 292.90 286.40 289.40 289.30 288.43 5686505

Turnover Deliverable Volume %Deliverble

1182 4.323421e+12 135600 0.3633

1396 2.008477e+13 1055534 0.7944

1773 1.008855e+14 971955 0.3319

2057 1.101764e+14 3124785 0.6783

2520 1.174129e+14 1248502 0.4566

... ... ... ...

346 2.642873e+13 165133 0.2603

995 6.865426e+12 179657 0.3353

2481 1.970928e+14 3562863 0.6985

2808 6.955919e+13 745258 0.4082

1679 1.640133e+14 3547109 0.6238

[831 rows x 10 columns]

predicted**.**shape

Out[28]:

(831,)

dframe**=**pd**.**DataFrame(y\_test,predicted)

dframe

Out[29]:

|  | **Open** |
| --- | --- |
| **115.379144** | NaN |
| **152.390068** | NaN |
| **345.518760** | NaN |
| **237.868849** | NaN |
| **430.449356** | NaN |
| **...** | ... |
| **415.627825** | NaN |
| **127.810173** | NaN |
| **380.898601** | NaN |
| **383.023589** | NaN |
| **290.137455** | NaN |

831 rows × 1 columns

dfr**=**pd**.**DataFrame({'Actual Price':y\_test,'Predicted Price':predicted})

dfr

Out[30]:

|  | **Actual Price** | **Predicted Price** |
| --- | --- | --- |
| **1182** | 117.50 | 115.379144 |
| **1396** | 150.05 | 152.390068 |
| **1773** | 348.00 | 345.518760 |
| **2057** | 236.45 | 237.868849 |
| **2520** | 428.00 | 430.449356 |
| **...** | ... | ... |
| **346** | 402.00 | 415.627825 |
| **995** | 127.35 | 127.810173 |
| **2481** | 388.15 | 380.898601 |
| **2808** | 383.00 | 383.023589 |
| **1679** | 289.10 | 290.137455 |

831 rows × 2 columns

dfr**.**head(60)

Out[31]:

|  | **Actual Price** | **Predicted Price** |
| --- | --- | --- |
| **1182** | 117.50 | 115.379144 |
| **1396** | 150.05 | 152.390068 |
| **1773** | 348.00 | 345.518760 |
| **2057** | 236.45 | 237.868849 |
| **2520** | 428.00 | 430.449356 |
| **1359** | 160.80 | 160.637546 |
| **118** | 897.80 | 895.755001 |
| **3029** | 362.30 | 361.927387 |
| **3063** | 273.50 | 269.285180 |
| **751** | 151.00 | 151.190945 |
| **501** | 566.05 | 567.178572 |
| **3147** | 356.90 | 356.304811 |
| **1939** | 313.00 | 310.584382 |
| **361** | 607.40 | 615.374411 |
| **1751** | 287.90 | 286.749544 |
| **2158** | 274.00 | 272.584862 |
| **316** | 313.00 | 311.485218 |
| **22** | 1263.35 | 1279.263516 |
| **1501** | 150.50 | 152.495328 |
| **1997** | 261.30 | 260.158132 |
| **3300** | 698.00 | 697.371653 |
| **2225** | 260.00 | 257.640473 |
| **2963** | 372.25 | 373.624927 |
| **576** | 736.40 | 745.460668 |
| **3031** | 358.70 | 358.272201 |
| **581** | 750.10 | 745.792149 |
| **3082** | 306.00 | 305.147879 |
| **1715** | 275.80 | 277.131426 |
| **942** | 155.10 | 154.537318 |
| **2692** | 317.00 | 320.477533 |
| **2101** | 184.00 | 184.584000 |
| **480** | 545.00 | 546.884862 |
| **2411** | 386.95 | 386.259780 |
| **496** | 559.60 | 557.883682 |
| **33** | 1045.60 | 1060.673233 |
| **3241** | 485.40 | 486.055545 |
| **2140** | 222.00 | 223.173637 |
| **2271** | 303.50 | 303.564842 |
| **3049** | 222.30 | 238.579616 |
| **1299** | 150.30 | 151.981832 |
| **306** | 343.80 | 344.202199 |
| **2996** | 389.15 | 387.857991 |
| **3208** | 373.00 | 372.851876 |
| **1237** | 123.45 | 122.861937 |
| **2263** | 296.75 | 297.326648 |
| **2482** | 386.40 | 385.598562 |
| **351** | 405.00 | 397.811430 |
| **405** | 562.90 | 553.463971 |
| **3038** | 334.70 | 329.877486 |
| **1087** | 134.00 | 135.274370 |
| **3252** | 536.80 | 535.539861 |
| **826** | 138.00 | 139.036140 |
| **3169** | 352.00 | 352.945961 |
| **1262** | 136.25 | 135.601756 |
| **3310** | 750.00 | 724.443726 |
| **685** | 784.90 | 793.805409 |
| **2698** | 317.35 | 313.906436 |
| **1423** | 126.00 | 125.653682 |
| **1721** | 295.60 | 299.534197 |
| **3022** | 374.00 | 372.436541 |

**from** sklearn.metrics **import** confusion\_matrix,accuracy\_score

train\_accuracy**=**regressor**.**score(x\_train,y\_train)

print('train\_accuracy:',train\_accuracy)

test\_accuracy**=**regressor**.**score(x\_test,y\_test)

print('test\_accuracy:',test\_accuracy)

train\_accuracy: 0.999155403718174

test\_accuracy: 0.9992925939370841

**import** math

**from** sklearn **import** metrics

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: 2.9548362038826643

Mean Squared Error: 30.054759300206765

Root Mean Squared Error: 5.482222113359397

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| graph**=**dfr**.**head(40)  graph  Out[34]:   |  | **Actual Price** | **Predicted Price** | | --- | --- | --- | | **1182** | 117.50 | 115.379144 | | **1396** | 150.05 | 152.390068 | | **1773** | 348.00 | 345.518760 | | **2057** | 236.45 | 237.868849 | | **2520** | 428.00 | 430.449356 | | **1359** | 160.80 | 160.637546 | | **118** | 897.80 | 895.755001 | | **3029** | 362.30 | 361.927387 | | **3063** | 273.50 | 269.285180 | | **751** | 151.00 | 151.190945 | | **501** | 566.05 | 567.178572 | | **3147** | 356.90 | 356.304811 | | **1939** | 313.00 | 310.584382 | | **361** | 607.40 | 615.374411 | | **1751** | 287.90 | 286.749544 | | **2158** | 274.00 | 272.584862 | | **316** | 313.00 | 311.485218 | | **22** | 1263.35 | 1279.263516 | | **1501** | 150.50 | 152.495328 | | **1997** | 261.30 | 260.158132 | | **3300** | 698.00 | 697.371653 | | **2225** | 260.00 | 257.640473 | | **2963** | 372.25 | 373.624927 | | **576** | 736.40 | 745.460668 | | **3031** | 358.70 | 358.272201 | | **581** | 750.10 | 745.792149 | | **3082** | 306.00 | 305.147879 | | **1715** | 275.80 | 277.131426 | | **942** | 155.10 | 154.537318 | | **2692** | 317.00 | 320.477533 | | **2101** | 184.00 | 184.584000 | | **480** | 545.00 | 546.884862 | | **2411** | 386.95 | 386.259780 | | **496** | 559.60 | 557.883682 | | **33** | 1045.60 | 1060.673233 | | **3241** | 485.40 | 486.055545 | | **2140** | 222.00 | 223.173637 | | **2271** | 303.50 | 303.564842 | | **3049** | 222.30 | 238.579616 | | **1299** | 150.30 | 151.981832 |   graph**.**plot(kind**=**'bar')  plt**.**title('BarGraph')  plt**.**xlabel('Actual&prediceted price')  plt**.**ylabel('frequency')  plt**.**show()    **from** sklearn.model\_selection **import** train\_test\_split  x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.90,random\_state**=**100)  print(x\_train**.**shape)  print(x\_test**.**shape)  print(y\_train**.**shape)  print(y\_test**.**shape)  (332, 10)  (2990, 10)  (332,)  (2990,)  **from** sklearn.ensemble **import** RandomForestRegressor  regressor **=** RandomForestRegressor(n\_estimators**=**100,random\_state**=**0)  regressor**.**fit(x\_train,y\_train)  Out[37]:  RandomForestRegressor(random\_state=0)  **In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**  predicted**=**regressor**.**predict(x\_test)  predicted  Out[38]:  array([ 604.71 , 301.025 , 182.862 , ..., 1062.355 , 397.5765,  113.6385])  **from** sklearn.metrics **import** confusion\_matrix,accuracy\_score  train\_accuracy**=**regressor**.**score(x\_train,y\_train)  print('train\_accuracy:',train\_accuracy)  R\_test\_accuracy**=**regressor**.**score(x\_test,y\_test)  print('test\_accuracy:',R\_test\_accuracy)  train\_accuracy: 0.9991042556766557  test\_accuracy: 0.994877679891376  **import** matplotlib.pyplot **as** plt  linear\_regression\_accuracy **=** 0.999155403718174  random\_forest\_accuracy **=**0.994877679891376  accuracy\_scores **=** [linear\_regression\_accuracy, random\_forest\_accuracy]  model\_names **=** ['Linear Regression', 'Random Forest Regression']  plt**.**bar(model\_names, accuracy\_scores)  plt**.**xlabel('Regression Models')  plt**.**ylabel('Test Accuracy')  plt**.**title('Comparison of Test Accuracy: Linear Regression vs Random Forest Regression')  plt**.**show() |  |  |
|  |  |  |
|  |  |  |
| **C:\Users\GPT BANTWAL\Pictures\download (4).png** |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  | Mean Absolute Error: 2.9548362038826643 |
|  |  | Mean Squared Error: 30.054759300206765 |
|  |  | Root Mean Squared Error: 5.482222113359397 |
|  |  | **import** math |
|  |  | **from** sklearn **import** metrics |
|  |  | 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: 2.9548362038826643 |
|  |  | Mean Squared Error: 30.054759300206765 |
|  |  | Root Mean Squared Error: 5.482222113359397 |
|  |  | **import** math |
|  |  | **from** sklearn **import** metrics |
|  |  | 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: 2.9548362038826643 |
|  |  | Mean Squared Error: 30.054759300206765 |
|  |  | Root Mean Squared Error: 5.482222113359397 |
|  |  | **import** math |
|  |  | **from** sklearn **import** metrics |
|  |  | 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))) |
| **2140** | 222.00 | Mean Absolute Error: 2.9548362038826643 |
| **2271** | 303.50 | Mean Squared Error: 30.054759300206765 |
| **3049** | 222.30 | Root Mean Squared Error: 5.482222113359397 |
| **1299** | 150.30 | **import** math |
| **306** | 343.80 | **from** sklearn **import** metrics |
| **2996** | 389.15 | print('Mean Absolute Error:',metrics**.**mean\_absolute\_error(y\_test,predicted)) |
| **3208** | 373.00 | print('Mean Squared Error:',metrics**.**mean\_squared\_error(y\_test,predicted)) |
| **1237** | 123.45 | print('Root Mean Squared Error:',math**.**sqrt(metrics**.**mean\_squared\_error(y\_test,predicted))) |
| **2263** | 296.75 | Mean Absolute Error: 2.9548362038826643 |
| **2482** | 386.40 | Mean Squared Error: 30.054759300206765 |
| **351** | 405.00 | Root Mean Squared Error: 5.482222113359397 |
| **405** | 562.90 | **import** math |
| **3038** | 334.70 | **from** sklearn **import** metrics |
| **1087** | 134.00 | print('Mean Absolute Error:',metrics**.**mean\_absolute\_error(y\_test,predicted)) |
| **3252** | 536.80 | print('Mean Squared Error:',metrics**.**mean\_squared\_error(y\_test,predicted)) |
| **826** | 138.00 | print('Root Mean Squared Error:',math**.**sqrt(metrics**.**mean\_squared\_error(y\_test,predicted))) |
| **3169** | 352.00 | Mean Absolute Error: 2.9548362038826643 |
| **1262** | 136.25 | Mean Squared Error: 30.054759300206765 |
| **3310** | 750.00 | Root Mean Squared Error: 5.482222113359397 |
| **685** | 784.90 | **import** math |
| **2698** | 317.35 | **from** sklearn **import** metrics |
| **1423** | 126.00 | print('Mean Absolute Error:',metrics**.**mean\_absolute\_error(y\_test,predicted)) |
| **1721** | 295.60 | print('Mean Squared Error:',metrics**.**mean\_squared\_error(y\_test,predicted)) |
| **3022** |  |  |

