Import libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
file=r"https://drive.google.com/uc?export=download&id=1xxDtrZKfuWQfl-6KA9XEd_eatitNPnk
```

In [3]:
```
data=pd.read_csv(file)
```

In [4]:
```
data
```

Out[4]:

	bath	balcony	price	total_sqft_int	bhk	price_per_sqft	area_typeSuper built-up Area	area_typeBuilt-up Area	area_
0	3.0	2.0	150.00	1672.0	3	8971.291866	1	0	
1	3.0	3.0	149.00	1750.0	3	8514.285714	0	1	
2	3.0	2.0	150.00	1750.0	3	8571.428571	1	0	
3	2.0	2.0	40.00	1250.0	2	3200.000000	1	0	
4	2.0	2.0	83.00	1200.0	2	6916.666667	0	0	
...	...	...	...	...	...	...	...	...	
7115	3.0	2.0	325.00	2900.0	3	11206.896552	1	0	
7116	3.0	1.0	84.83	1780.0	3	4765.730337	1	0	
7117	2.0	1.0	48.00	880.0	2	5454.545455	0	0	
7118	2.0	1.0	55.00	1000.0	2	5500.000000	0	0	
7119	2.0	1.0	78.00	1400.0	3	5571.428571	0	0	

7120 rows × 108 columns

In [5]:
```
data.shape
```

Out[5]:
```
(7120, 108)
```

In [17]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7120 entries, 0 to 7119
Columns: 108 entries, bath to location_Tumkur Road
dtypes: float64(5), int64(103)
memory usage: 5.9 MB
```

In [7]:
```
data.describe()
```

Out[7]:

	bath	balcony	price	total_sqft_int	bhk	price_per_sqft	area_typeSup built-up Ar
count	7120.000000	7120.000000	7120.000000	7120.000000	7120.000000	7120.000000	7120.0000
mean	2.390871	1.572759	96.454991	1479.729806	2.465169	5923.806855	0.7507
std	0.876822	0.770583	116.185034	913.779769	0.841520	2556.650935	0.4326
min	1.000000	0.000000	10.000000	350.000000	1.000000	1250.000000	0.0000
25%	2.000000	1.000000	49.230000	1100.000000	2.000000	4416.761042	1.0000
50%	2.000000	2.000000	69.000000	1255.000000	2.000000	5417.855613	1.0000
75%	3.000000	2.000000	104.000000	1640.250000	3.000000	6618.285651	1.0000
max	9.000000	3.000000	2912.000000	30400.000000	9.000000	35000.000000	1.0000

8 rows × 108 columns

In [15]:
```
x=data.drop("price",axis=1)
y=data["price"]

print("x=",x.shape)
print("y=",y.shape)
```

```
x= (7120, 107)
y= (7120,)
```

In [12]:
```
x
```

Out[12]:

	bath	balcony	total_sqft_int	bhk	price_per_sqft	area_typeSuper built-up Area	area_typeBuilt-up Area	area_typePlo Area
0	3.0	2.0	1672.0	3	8971.291866	1	0	(
1	3.0	3.0	1750.0	3	8514.285714	0	1	(
2	3.0	2.0	1750.0	3	8571.428571	1	0	(
3	2.0	2.0	1250.0	2	3200.000000	1	0	(
4	2.0	2.0	1200.0	2	6916.666667	0	0	
...	...	...	...	...	...	...	...	...
7115	3.0	2.0	2900.0	3	11206.896552	1	0	(
7116	3.0	1.0	1780.0	3	4765.730337	1	0	(
7117	2.0	1.0	880.0	2	5454.545455	0	0	
7118	2.0	1.0	1000.0	2	5500.000000	0	0	
7119	2.0	1.0	1400.0	3	5571.428571	0	0	

7120 rows × 107 columns

In [29]:
```
y
```

Out[29]:
```
0       150.00
1       149.00
2       150.00
3        40.00
4        83.00
         ...
7115    325.00
7116     84.83
7117     48.00
7118     55.00
7119     78.00
Name: price, Length: 7120, dtype: float64
```

In [16]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=50)

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= (5696, 107)
shape of x_test= (1424, 107)
shape of y_train= (5696,)
shape of y_test= (1424,)
```

Feature Scaling

In [19]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(x_train)
x_train=sc.transform(x_train)
x_test=sc.transform(x_test)
```

Model Training

In [20]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()

lr.fit(x_train,y_train)
```

Out[20]:
```
▾ LinearRegression
LinearRegression()
```

In [21]:
```
lr.intercept_
```

Out[21]:
```
97.3928493679775
```

In [22]:
```
lr.coef_
```

Out[22]:
```
array([ 4.96061905e-01, -8.01167316e-02,  7.91448451e+01, -9.63563893e+00,
        6.31774024e+01, -6.38257260e-01, -1.89944708e+00, -3.64060273e+00,
       -1.05956057e+00, -3.21472072e+00, -2.37479962e+00,  3.99282257e-01,
       -1.12268530e+00,  1.83354966e+00,  2.08184750e-01, -1.38264110e+00,
       -2.52761801e-01,  2.19953858e+00, -1.45469773e+00,  1.72049601e+00,
       -2.59544613e+00, -1.49092982e+00, -8.44547973e-01, -1.55485155e-01,
        1.32187532e+00,  1.57479927e+00, -4.40840839e+00, -8.69561591e-01,
       -6.21292963e-01,  1.59096552e+00, -3.80877975e-01, -3.82820503e-01,
        5.06715542e-01, -1.47192510e+00, -9.49308926e-02, -1.37196356e+00,
        4.98683546e-01,  6.70430961e-01,  3.50567969e+00,  3.20433473e-01,
        4.49176409e-01,  2.93870267e-01, -2.95545050e-01,  9.77906086e-01,
       -1.44652346e+00, -2.74103341e+00,  1.74967671e-01, -3.12111119e-02,
       -6.88327783e-01, -4.53956244e-01, -1.06037264e+00,  4.19889376e-01,
        8.71356372e-01, -6.05935185e-01, -9.16041555e-01,  1.20575562e+00,
        1.64117212e+00, -2.90682890e-01, -2.10292520e+00, -1.01863026e+00,
        1.01599381e+00, -5.86881647e-01, -1.59941062e-01,  1.30620457e-01,
       -1.37150848e+00,  1.40082640e+00, -9.02536373e-01, -8.66411734e-01,
       -7.43043759e-01,  2.25770476e-01, -7.60550377e-01, -2.51450532e-01,
       -1.67519033e+00, -1.21425400e+00,  1.73372186e-01,  1.13645124e+00,
       -1.12049968e+00,  8.76676097e-02, -4.50701489e-01, -1.76821663e+00,
        4.68969289e-01,  1.24765297e-01,  1.97159968e-01,  3.06018253e+00,
        2.04643516e+00, -9.09384643e-01, -5.09697046e-01,  2.29569548e+00,
        2.48487104e-01, -2.83325568e-02,  3.07343609e-01,  1.20927933e+00,
        7.57167742e-01, -1.09846608e+00,  1.11653028e-01, -7.75067461e-01,
        1.48921624e+00,  3.04151583e-01,  9.73005208e-01,  4.83455267e-01,
       -6.17053737e-01, -1.65044519e+00, -5.45177345e-01,  6.64870224e-01,
       -1.62132143e+00,  3.08498958e-01, -5.70026857e-01])
```

Predict value of home and test

In [24]:
```
x_test[0,:]
```

Out[24]:
```
array([-1.60278311, -2.03620318, -0.69501761, -1.75532587, -0.60222255,
        0.57140362, -0.46397877, -0.26089606, -1.96893685, -0.18373025,
       -0.1694577 , -0.1466997 , -0.12454231, -0.12160223, -0.12812824,
       -0.12526719, -0.12010681, -0.12526719, -0.11394031, -0.10053942,
       -0.1107353 , -0.1082724 , -0.11551113,  9.30756516, -0.09218776,
       -0.08409599, -0.08923672, -0.08823182, -0.09315135, -0.08923672,
       -0.07975227, -0.0830308 , -0.07749158, -0.08085949, -0.07633675,
       -0.07862985, -0.07749158, -0.07862985, -0.07749158, -0.07862985,
       -0.07862985, -0.07862985, -0.07633675, -0.07749158, -0.07028523,
       -0.07633675, -0.0677166 , -0.0739743 , -0.0751646 , -0.06639573,
       -0.06901264, -0.06901264, -0.0677166 , -0.06901264, -0.0677166 ,
       -0.06639573, -0.05785186, -0.06504853, -0.06639573, -0.06367332,
       -0.06504853, -0.06901264, -0.06504853, -0.06083125, -0.06083125,
       -0.06226825, -0.06639573, -0.06226825, -0.06083125, -0.05935999,
       -0.06226825, -0.05471275, -0.05935999, -0.06083125, -0.06083125,
       -0.05785186, -0.06226825, -0.05785186, -0.05785186, -0.05785186,
       -0.05935999, -0.06226825, -0.05630391, -0.06083125, -0.05630391,
       -0.06083125, -0.06083125, -0.05630391, -0.06083125, -0.05630391,
       -0.05630391, -0.0496379 , -0.05785186, -0.06083125, -0.05935999,
       -0.05630391, -0.05307449, -0.05471275, -0.05471275, -0.05307449,
       -0.05935999, -0.05785186, -0.05630391, -0.05471275, -0.0496379 ,
       -0.05471275, -0.05471275])
```

In [26]:
```
lr.predict([x_test[0,:]])
```

Out[26]:
```
array([24.52230994])
```

In [33]:
```
predict=lr.predict(x_test)
```

In [34]:
```
predict
```

Out[34]:
```
array([ 24.52230994, 125.54854193,  73.39336593, ...,  62.22853087,
        36.29815877,  74.08736658])
```

In [29]:
```
y_test
```

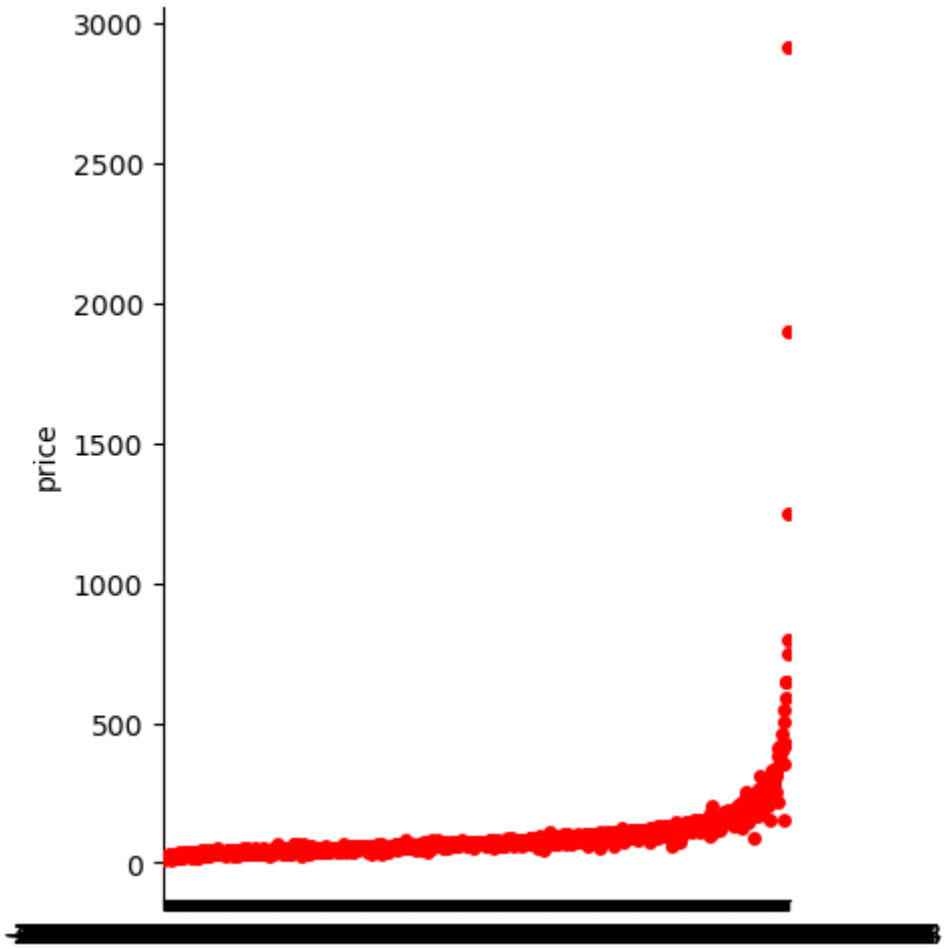
Out[29]:
```
3813     36.29
2623    124.00
5318     67.00
6292    155.00
2354    130.00
         ...
768     170.00
6591     52.00
4368     74.00
3090     44.80
5017     84.00
Name: price, Length: 1424, dtype: float64
```

In [32]:
```
lr.score(x_test,y_test)
```

Out[32]:
```
0.8038858047649013
```

In [44]:
```
sns.catplot(x=predict,y=y_test,color="r")
```

Out[44]:
```
<seaborn.axisgrid.FacetGrid at 0x1a5a8c425d0>
```



In [ ]: