In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

In [8]:
```
df=pd.read_csv(r"C:\Users\Biswajeet Jena\Downloads\Documents\Csv\Bengaluru_House_Data.csv")
df
```

Out[8]:

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200	2.0	1.0	51.00
...	...	...	...	...	...	...	...	...	...
13315	Built-up Area	Ready To Move	Whitefield	5 Bedroom	ArsiaEx	3453	4.0	0.0	231.00
13316	Super built-up Area	Ready To Move	Richards Town	4 BHK	NaN	3600	5.0	NaN	400.00
13317	Built-up Area	Ready To Move	Raja Rajeshwari Nagar	2 BHK	Mahla T	1141	2.0	1.0	60.00
13318	Super built-up Area	18-Jun	Padmanabhanagar	4 BHK	SollyCl	4689	4.0	1.0	488.00
13319	Super built-up Area	Ready To Move	Doddathoguru	1 BHK	NaN	550	1.0	1.0	17.00

13320 rows × 9 columns

In [9]:
```
c=df.groupby('area_type')
```

In [10]:
```
for area_typ,area_typ_df in c:
    print(area_typ)
    print(area_typ_df)
```

```
Built-up  Area
           area_type   availability              location       size  \
2      Built-up  Area  Ready To Move            Uttarahalli      3 BHK
13     Built-up  Area  Ready To Move              Gottigere      2 BHK
20     Built-up  Area  Ready To Move                Kengeri      1 BHK
27     Built-up  Area         20-Dec             Whitefield      3 BHK
34     Built-up  Area  Ready To Move          Kasturi Nagar      3 BHK
...               ...           ...                    ...        ...
13280  Built-up  Area  Ready To Move               Sarjapur      3 BHK
13307  Built-up  Area  Ready To Move           Billekahalli      3 BHK
13308  Built-up  Area  Ready To Move      Bannerghatta Road      3 BHK
13315  Built-up  Area  Ready To Move             Whitefield  5 Bedroom
13317  Built-up  Area  Ready To Move  Raja Rajeshwari Nagar      2 BHK

       society total_sqft  bath  balcony  price
2          NaN       1440   2.0      3.0   62.0
13         NaN       1100   2.0      2.0   40.0
20         NaN        600   1.0      1.0   15.0
27         NaN       1610   3.0      2.0   81.0
34     Kantsce       1925   3.0      NaN  125.0
...        ...        ...   ...      ...    ...
13280      NaN       1425   3.0      2.0   57.0
13307      NaN       1805   3.0      3.0  134.0
13308  Baanise       1527   3.0      1.0  142.0
13315  ArsiaEx       3453   4.0      0.0  231.0
13317  Mahla T       1141   2.0      1.0   60.0

[2418 rows x 9 columns]
Carpet  Area
         area_type   availability           location   size  society  \
297    Carpet  Area  Ready To Move  Maruthi Sevanagar  2 BHK  SMikaay
340    Carpet  Area  Ready To Move   Muneshwara Nagar  2 BHK      NaN
352    Carpet  Area  Ready To Move       Sampigehalli  3 BHK  Puens G
594    Carpet  Area  Ready To Move       Sanjay nagar  2 BHK      NaN
622    Carpet  Area  Ready To Move  Bannerghatta Road  3 BHK      NaN
...             ...           ...                ...    ...      ...
13006  Carpet  Area  Ready To Move   Kereguddadahalli  3 BHK  Jaood G
13183  Carpet  Area  Ready To Move      Shivaji Nagar  2 BHK      NaN
13214  Carpet  Area  Ready To Move  Bannerghatta Road  3 BHK  Vemit S
13238  Carpet  Area         19-Jan       Hennur Bande  2 BHK      NaN
13305  Carpet  Area  Ready To Move           Hulimavu  1 BHK      NaN

      total_sqft  bath  balcony  price
297          950   2.0      2.0   47.0
```

```
340         1230   2.0      3.0   48.0
352         1592   3.0      2.0   75.0
594         1100   2.0      2.0   98.0
622         1445   3.0      2.0   95.0
...          ...   ...      ...    ...
13006       1280   3.0      1.0   42.0
13183        600   1.0      1.0   65.0
13214       1470   2.0      1.0   85.0
13238       1200   2.0      3.0   70.0
13305        500   1.0      3.0  220.0

[87 rows x 9 columns]
Plot  Area
        area_type   availability                  location       size  society  \
1      Plot  Area  Ready To Move       Chikka Tirupathi  4 Bedroom  Theanmp
9      Plot  Area  Ready To Move           Gandhi Bazar  6 Bedroom      NaN
11     Plot  Area  Ready To Move             Whitefield  4 Bedroom  Prrry M
14     Plot  Area  Ready To Move               Sarjapur  3 Bedroom  Skityer
22     Plot  Area  Ready To Move           Thanisandra  4 Bedroom  Soitya
...           ...           ...                    ...        ...      ...
13291  Plot  Area         18-Jan         Weavers Colony  1 Bedroom      NaN
13300  Plot  Area  Ready To Move          Hosakerehalli  5 Bedroom      NaN
13303  Plot  Area  Ready To Move         Vidyaranyapura  5 Bedroom      NaN
13306  Plot  Area  Ready To Move  Rajarajeshwari Nagara  4 Bedroom      NaN
13311  Plot  Area  Ready To Move       Ramamurthy Nagar  7 Bedroom      NaN

      total_sqft  bath  balcony  price
1           2600   5.0      3.0  120.0
9           1020   6.0      NaN  370.0
11          2785   5.0      3.0  295.0
14          2250   3.0      2.0  148.0
22          2800   5.0      2.0  380.0
...          ...   ...      ...    ...
13291        812   1.0      0.0   26.0
13300       1500   6.0      2.0  145.0
13303        774   5.0      3.0   70.0
13306       1200   5.0      NaN  325.0
13311       1500   9.0      2.0  250.0

[2025 rows x 9 columns]
Super built-up  Area
                 area_type   availability                  location   size  \
0      Super built-up  Area         19-Dec  Electronic City Phase II  2 BHK
3      Super built-up  Area  Ready To Move        Lingadheeranahalli  3 BHK
4      Super built-up  Area  Ready To Move                  Kothanur  2 BHK
5      Super built-up  Area  Ready To Move                Whitefield  2 BHK
6      Super built-up  Area         18-May          Old Airport Road  4 BHK
...                     ...           ...                       ...    ...
13313  Super built-up  Area  Ready To Move                Uttarahalli  3 BHK
13314  Super built-up  Area  Ready To Move         Green Glen Layout  3 BHK
13316  Super built-up  Area  Ready To Move             Richards Town  4 BHK
13318  Super built-up  Area         18-Jun           Padmanabhanagar  4 BHK
13319  Super built-up  Area  Ready To Move              Doddathoguru  1 BHK

       society total_sqft  bath  balcony   price
0      Coomee        1056   2.0      1.0   39.07
3      Soiewre       1521   3.0      1.0   95.00
4          NaN       1200   2.0      1.0   51.00
5      DuenaTa       1170   2.0      1.0   38.00
6      Jaades        2732   4.0      NaN  204.00
...       ...         ...   ...      ...     ...
13313  Aklia R       1345   2.0      1.0   57.00
13314  SoosePr       1715   3.0      3.0  112.00
13316      NaN       3600   5.0      NaN  400.00
13318  SollyCl       4689   4.0      1.0  488.00
13319      NaN        550   1.0      1.0   17.00

[8790 rows x 9 columns]
```

In [11]:
```
df.groupby('area_type')['area_type'].agg('count')
```

Out[11]:
```
area_type
Built-up  Area          2418
Carpet  Area              87
Plot  Area              2025
Super built-up  Area    8790
Name: area_type, dtype: int64
```

In [12]:
```
df1=df.drop(['area_type','availability','society','balcony'],axis='columns')
df1
```

Out[12]:

	location	size	total_sqft	bath	price
0	Electronic City Phase II	2 BHK	1056	2.0	39.07
1	Chikka Tirupathi	4 Bedroom	2600	5.0	120.00
2	Uttarahalli	3 BHK	1440	2.0	62.00
3	Lingadheeranahalli	3 BHK	1521	3.0	95.00
4	Kothanur	2 BHK	1200	2.0	51.00
...	...	...	...	...	...
13315	Whitefield	5 Bedroom	3453	4.0	231.00
13316	Richards Town	4 BHK	3600	5.0	400.00
13317	Raja Rajeshwari Nagar	2 BHK	1141	2.0	60.00
13318	Padmanabhanagar	4 BHK	4689	4.0	488.00
13319	Doddathoguru	1 BHK	550	1.0	17.00

13320 rows × 5 columns

In [13]:
```
df1.isna().any()
```

Out[13]:
```
location       True
size           True
total_sqft    False
bath           True
price         False
dtype: bool
```

In [14]:
```
df1.isnull().sum()
```

Out[14]:
```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

DATA CLEANING

In [15]:
```
df3=df1.dropna()
df3
```

Out[15]:

	location	size	total_sqft	bath	price
0	Electronic City Phase II	2 BHK	1056	2.0	39.07
1	Chikka Tirupathi	4 Bedroom	2600	5.0	120.00
2	Uttarahalli	3 BHK	1440	2.0	62.00
3	Lingadheeranahalli	3 BHK	1521	3.0	95.00
4	Kothanur	2 BHK	1200	2.0	51.00
...	...	...	...	...	...
13315	Whitefield	5 Bedroom	3453	4.0	231.00
13316	Richards Town	4 BHK	3600	5.0	400.00
13317	Raja Rajeshwari Nagar	2 BHK	1141	2.0	60.00
13318	Padmanabhanagar	4 BHK	4689	4.0	488.00
13319	Doddathoguru	1 BHK	550	1.0	17.00

13246 rows × 5 columns

In [16]:
```
df3.isnull().sum()
```

Out[16]:
```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

In [17]:
```
df3['size'].unique()
```

Out[17]:
```
array(['2 BHK', '4 Bedroom', '3 BHK', '4 BHK', '6 Bedroom', '3 Bedroom',
       '1 BHK', '1 RK', '1 Bedroom', '8 Bedroom', '2 Bedroom',
       '7 Bedroom', '5 BHK', '7 BHK', '6 BHK', '5 Bedroom', '11 BHK',
       '9 BHK', '9 Bedroom', '27 BHK', '10 Bedroom', '11 Bedroom',
       '10 BHK', '19 BHK', '16 BHK', '43 Bedroom', '14 BHK', '8 BHK',
       '12 Bedroom', '13 BHK', '18 Bedroom'], dtype=object)
```

FEATURE ENGINEERING

In [18]:
```
df3['bhk']=df3['size'].apply(lambda x: int(x.split(' ')[0]))
```

```
C:\Users\Biswajeet Jena\AppData\Local\Temp\ipykernel_20040\2989175054.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy
  df3['bhk']=df3['size'].apply(lambda x: int(x.split(' ')[0]))
```

In [19]:
```
df3.drop('size',axis='columns',inplace=True)
df3
```

```
C:\Users\Biswajeet Jena\AppData\Local\Temp\ipykernel_20040\1652136885.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy
  df3.drop('size',axis='columns',inplace=True)
```

Out[19]:

	location	total_sqft	bath	price	bhk
0	Electronic City Phase II	1056	2.0	39.07	2
1	Chikka Tirupathi	2600	5.0	120.00	4
2	Uttarahalli	1440	2.0	62.00	3
3	Lingadheeranahalli	1521	3.0	95.00	3
4	Kothanur	1200	2.0	51.00	2
...	...	...	...	...	...
13315	Whitefield	3453	4.0	231.00	5
13316	Richards Town	3600	5.0	400.00	4
13317	Raja Rajeshwari Nagar	1141	2.0	60.00	2
13318	Padmanabhanagar	4689	4.0	488.00	4
13319	Doddathoguru	550	1.0	17.00	1

13246 rows × 5 columns

In [20]:
```
df3['bhk'].unique()
```

Out[20]:
```
array([ 2,  4,  3,  6,  1,  8,  7,  5, 11,  9, 27, 10, 19, 16, 43, 14, 12,
       13, 18], dtype=int64)
```

In [21]:
```
df3[df3['bhk']>20]
```

Out[21]:

	location	total_sqft	bath	price	bhk
1718	2Electronic City Phase II	8000	27.0	230.0	27
4684	Munnekollal	2400	40.0	660.0	43

In [22]:
```
df3['total_sqft'].unique()
```

Out[22]:
```
array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

In [23]:
```
def is_float(x):
    try:
        float(x)
    except:
        return False
    return True
```

In [24]:
```
df3[~df3['total_sqft'].apply(is_float)].head(30)
```

Out[24]:

	location	total_sqft	bath	price	bhk
30	Yelahanka	2100 - 2850	4.0	186.000	4
122	Hebbal	3067 - 8156	4.0	477.000	4
137	8th Phase JP Nagar	1042 - 1105	2.0	54.005	2
165	Sarjapur	1145 - 1340	2.0	43.490	2
188	KR Puram	1015 - 1540	2.0	56.800	2
410	Kengeri	34.46Sq. Meter	1.0	18.500	1
549	Hennur Road	1195 - 1440	2.0	63.770	2
648	Arekere	4125Perch	9.0	265.000	9
661	Yelahanka	1120 - 1145	2.0	48.130	2
672	Bettahalsoor	3090 - 5002	4.0	445.000	4
772	Banashankari Stage VI	1160 - 1195	2.0	59.935	2
775	Basavanagara	1000Sq. Meter	2.0	93.000	1
850	Bannerghatta Road	1115 - 1130	2.0	58.935	2
872	Singapura Village	1100Sq. Yards	2.0	45.000	2
886	Chandapura	520 - 645	1.0	15.135	1
927	Thanisandra	1000 - 1285	2.0	43.415	2
959	Kammasandra	650 - 665	1.0	18.410	1
990	Sarjapur	633 - 666	1.0	17.535	1
1019	Marathi Layout	5.31Acres	1.0	110.000	1
1086	Narasapura	30Acres	2.0	29.500	2
1178	Yelahanka	1445 - 1455	3.0	65.255	3
1183	Magadi Road	884 - 1116	2.0	46.500	2
1187	Thanisandra	850 - 1093	2.0	36.435	2
1400	Chamrajpet	716Sq. Meter	9.0	296.000	9
1484	Hebbal	547.34 - 827.31	2.0	42.720	2
1542	Sarjapur Road	580 - 650	1.0	17.835	1
1614	Sarjapur Road	3425 - 3435	6.0	228.500	4
1643	Talaghattapura	1804 - 2273	3.0	120.000	3
1683	Old Madras Road	3630 - 3800	6.0	224.500	4
1694	JP Nagar	4000 - 5249	4.0	314.500	4

In [25]:
```
df3.head(33)
```

Out[25]:

	location	total_sqft	bath	price	bhk
0	Electronic City Phase II	1056	2.0	39.07	2
1	Chikka Tirupathi	2600	5.0	120.00	4
2	Uttarahalli	1440	2.0	62.00	3
3	Lingadheeranahalli	1521	3.0	95.00	3
4	Kothanur	1200	2.0	51.00	2
5	Whitefield	1170	2.0	38.00	2
6	Old Airport Road	2732	4.0	204.00	4
7	Rajaji Nagar	3300	4.0	600.00	4
8	Marathahalli	1310	3.0	63.25	3
9	Gandhi Bazar	1020	6.0	370.00	6
10	Whitefield	1800	2.0	70.00	3
11	Whitefield	2785	5.0	295.00	4
12	7th Phase JP Nagar	1000	2.0	38.00	2
13	Gottigere	1100	2.0	40.00	2
14	Sarjapur	2250	3.0	148.00	3
15	Mysore Road	1175	2.0	73.50	2
16	Bisuvanahalli	1180	3.0	48.00	3
17	Raja Rajeshwari Nagar	1540	3.0	60.00	3
18	Ramakrishnappa Layout	2770	4.0	290.00	3
19	Manayata Tech Park	1100	2.0	48.00	2
20	Kengeri	600	1.0	15.00	1
21	Binny Pete	1755	3.0	122.00	3
22	Thanisandra	2800	5.0	380.00	4
23	Bellandur	1767	3.0	103.00	3
24	Thanisandra	510	1.0	25.25	1
25	Mangammanapalya	1250	3.0	56.00	3
26	Electronic City	660	1.0	23.10	2
27	Whitefield	1610	3.0	81.00	3
28	Ramagondanahalli	1151	2.0	48.77	2
29	Electronic City	1025	2.0	47.00	3
30	Yelahanka	2100 - 2850	4.0	186.00	4
31	Bisuvanahalli	1075	2.0	35.00	3
32	Hebbal	1760	2.0	123.00	3

In [26]:
```
def convert_sqft_to_num(x):
    tokens = x.split('-')
    if len(tokens) == 2:
        return (float(tokens[0])+float(tokens[1]))/2
    try:
        return float(x)
    except:
        return None
```

In [27]:
```
convert_sqft_to_num('10')
```

Out[27]:
```
10.0
```

In [28]:
```
convert_sqft_to_num('2100 - 2850')
```

Out[28]:
```
2475.0
```

In [29]:
```
convert_sqft_to_num('34.46Sq. Meter')
```

In [30]:
```
df4 = df3.copy()
df4.total_sqft = df4.total_sqft.apply(convert_sqft_to_num)
df4.head()
```

Out[30]:

	location	total_sqft	bath	price	bhk
0	Electronic City Phase II	1056.0	2.0	39.07	2
1	Chikka Tirupathi	2600.0	5.0	120.00	4
2	Uttarahalli	1440.0	2.0	62.00	3
3	Lingadheeranahalli	1521.0	3.0	95.00	3
4	Kothanur	1200.0	2.0	51.00	2

In [31]:
```
df4.loc[30]
```

Out[31]:
```
location      Yelahanka
total_sqft       2475.0
bath                4.0
price             186.0
bhk                   4
Name: 30, dtype: object
```

In [32]:
```
df4.loc[410]
```

Out[32]:
```
location      Kengeri
total_sqft        NaN
bath              1.0
price            18.5
bhk                 1
Name: 410, dtype: object
```

In [33]:
```
df4['price_per_sqft']=(df4.price*100000)/(df4.total_sqft)
df4
```

Out[33]:

	location	total_sqft	bath	price	bhk	price_per_sqft
0	Electronic City Phase II	1056.0	2.0	39.07	2	3699.810606
1	Chikka Tirupathi	2600.0	5.0	120.00	4	4615.384615
2	Uttarahalli	1440.0	2.0	62.00	3	4305.555556
3	Lingadheeranahalli	1521.0	3.0	95.00	3	6245.890861
4	Kothanur	1200.0	2.0	51.00	2	4250.000000
...	...	...	...	...	...	...
13315	Whitefield	3453.0	4.0	231.00	5	6689.834926
13316	Richards Town	3600.0	5.0	400.00	4	11111.111111
13317	Raja Rajeshwari Nagar	1141.0	2.0	60.00	2	5258.545136
13318	Padmanabhanagar	4689.0	4.0	488.00	4	10407.336319
13319	Doddathoguru	550.0	1.0	17.00	1	3090.909091

13246 rows × 6 columns

In [34]:
```
df4.location.unique()
```

Out[34]:
```
array(['Electronic City Phase II', 'Chikka Tirupathi', 'Uttarahalli', ...,
       '12th cross srinivas nagar banshankari 3rd stage',
       'Havanur extension', 'Abshot Layout'], dtype=object)
```

In [35]:
```
len(df4.location.unique())
```

Out[35]:
```
1304
```

In [36]:
```
loc_dlt=df4.groupby('location')['location'].agg('count')
loc_dlt
```

Out[36]:
```
location
 Anekal                 1
 Banaswadi              1
 Basavangudi            1
 Bhoganhalli            1
 Devarabeesana Halli    6
                      ..
t.c palya               1
tc.palya                4
vinayakanagar           1
white field,kadugodi    1
whitefiled              1
Name: location, Length: 1304, dtype: int64
```

In [37]:
```
df4.groupby('location')['location'].agg('count').sort_values(ascending=False)
```

Out[37]:
```
location
Whitefield             534
Sarjapur  Road         392
Electronic City        302
Kanakpura Road         266
Thanisandra            233
                      ...
 Banaswadi               1
Kanakadasa Layout        1
Kanakapur main road      1
Kanakapura  Rod          1
whitefiled               1
Name: location, Length: 1304, dtype: int64
```

In [38]:
```
len(loc_dlt[loc_dlt<=10])
```

Out[38]:
```
1063
```

In [39]:
```
loc_less_than_10=loc_dlt[loc_dlt<=10]
loc_less_than_10
```

Out[39]:
```
location
 Anekal                 1
 Banaswadi              1
 Basavangudi            1
 Bhoganhalli            1
 Devarabeesana Halli    6
                       ..
t.c palya               1
tc.palya                4
vinayakanagar           1
white field,kadugodi    1
whitefiled              1
Name: location, Length: 1063, dtype: int64
```

DIMENSIONAL REDUXIBILITY

In [40]:
```
df4.location=df4.location.apply(lambda x: 'other' if x in loc_less_than_10 else x )
df4
```

Out[40]:

	location	total_sqft	bath	price	bhk	price_per_sqft
0	Electronic City Phase II	1056.0	2.0	39.07	2	3699.810606
1	Chikka Tirupathi	2600.0	5.0	120.00	4	4615.384615
2	Uttarahalli	1440.0	2.0	62.00	3	4305.555556
3	Lingadheeranahalli	1521.0	3.0	95.00	3	6245.890861
4	Kothanur	1200.0	2.0	51.00	2	4250.000000
...	...	...	...	...	...	...
13315	Whitefield	3453.0	4.0	231.00	5	6689.834926
13316	other	3600.0	5.0	400.00	4	11111.111111
13317	Raja Rajeshwari Nagar	1141.0	2.0	60.00	2	5258.545136
13318	Padmanabhanagar	4689.0	4.0	488.00	4	10407.336319
13319	Doddathoguru	550.0	1.0	17.00	1	3090.909091

13246 rows × 6 columns

In [41]:
```
len(df4.location.unique())
```

Out[41]:
```
242
```

OUTLIER REMOVAL

In [42]:
```
df4[df4.total_sqft/df4.bhk<300]
```

Out[42]:

	location	total_sqft	bath	price	bhk	price_per_sqft
9	other	1020.0	6.0	370.0	6	36274.509804
45	HSR Layout	600.0	9.0	200.0	8	33333.333333
58	Murugeshpalya	1407.0	4.0	150.0	6	10660.980810
68	other	1350.0	7.0	85.0	8	6296.296296
70	other	500.0	3.0	100.0	3	20000.000000
...	...	...	...	...	...	...
13277	other	1400.0	7.0	218.0	7	15571.428571
13279	other	1200.0	5.0	130.0	6	10833.333333
13281	Margondanahalli	1375.0	5.0	125.0	5	9090.909091
13303	Vidyaranyapura	774.0	5.0	70.0	5	9043.927649
13311	Ramamurthy Nagar	1500.0	9.0	250.0	7	16666.666667

744 rows × 6 columns

In [43]:
```
len(df4[df4.total_sqft/df4.bhk<300])
```

Out[43]:
```
744
```

In [44]:
```
df4.shape
```

Out[44]:
```
(13246, 6)
```

In [45]:
```
df5=df4[~(df4.total_sqft/df4.bhk<300)]
df5
```

Out[45]:

	location	total_sqft	bath	price	bhk	price_per_sqft
0	Electronic City Phase II	1056.0	2.0	39.07	2	3699.810606
1	Chikka Tirupathi	2600.0	5.0	120.00	4	4615.384615
2	Uttarahalli	1440.0	2.0	62.00	3	4305.555556
3	Lingadheeranahalli	1521.0	3.0	95.00	3	6245.890861
4	Kothanur	1200.0	2.0	51.00	2	4250.000000
...	...	...	...	...	...	...
13315	Whitefield	3453.0	4.0	231.00	5	6689.834926
13316	other	3600.0	5.0	400.00	4	11111.111111
13317	Raja Rajeshwari Nagar	1141.0	2.0	60.00	2	5258.545136
13318	Padmanabhanagar	4689.0	4.0	488.00	4	10407.336319
13319	Doddathoguru	550.0	1.0	17.00	1	3090.909091

12502 rows × 6 columns

In [46]:
```
df5.price_per_sqft.describe()
```

Out[46]:
```
count     12456.000000
mean       6308.502826
std        4168.127339
min         267.829813
25%        4210.526316
50%        5294.117647
75%        6916.666667
max      176470.588235
Name: price_per_sqft, dtype: float64
```

In [47]:
```
def remove_pps_outliers(df):
    df_out = pd.DataFrame()
    for key, subdf in df.groupby('location'):
        m = np.mean(subdf.price_per_sqft)
        st = np.std(subdf.price_per_sqft)
        reduced_df = subdf[(subdf.price_per_sqft>(m-st)) & (subdf.price_per_sqft<=(m+st))]
        df_out = pd.concat([df_out,reduced_df],ignore_index=True)
    return df_out
df6 = remove_pps_outliers(df5)
df6
```

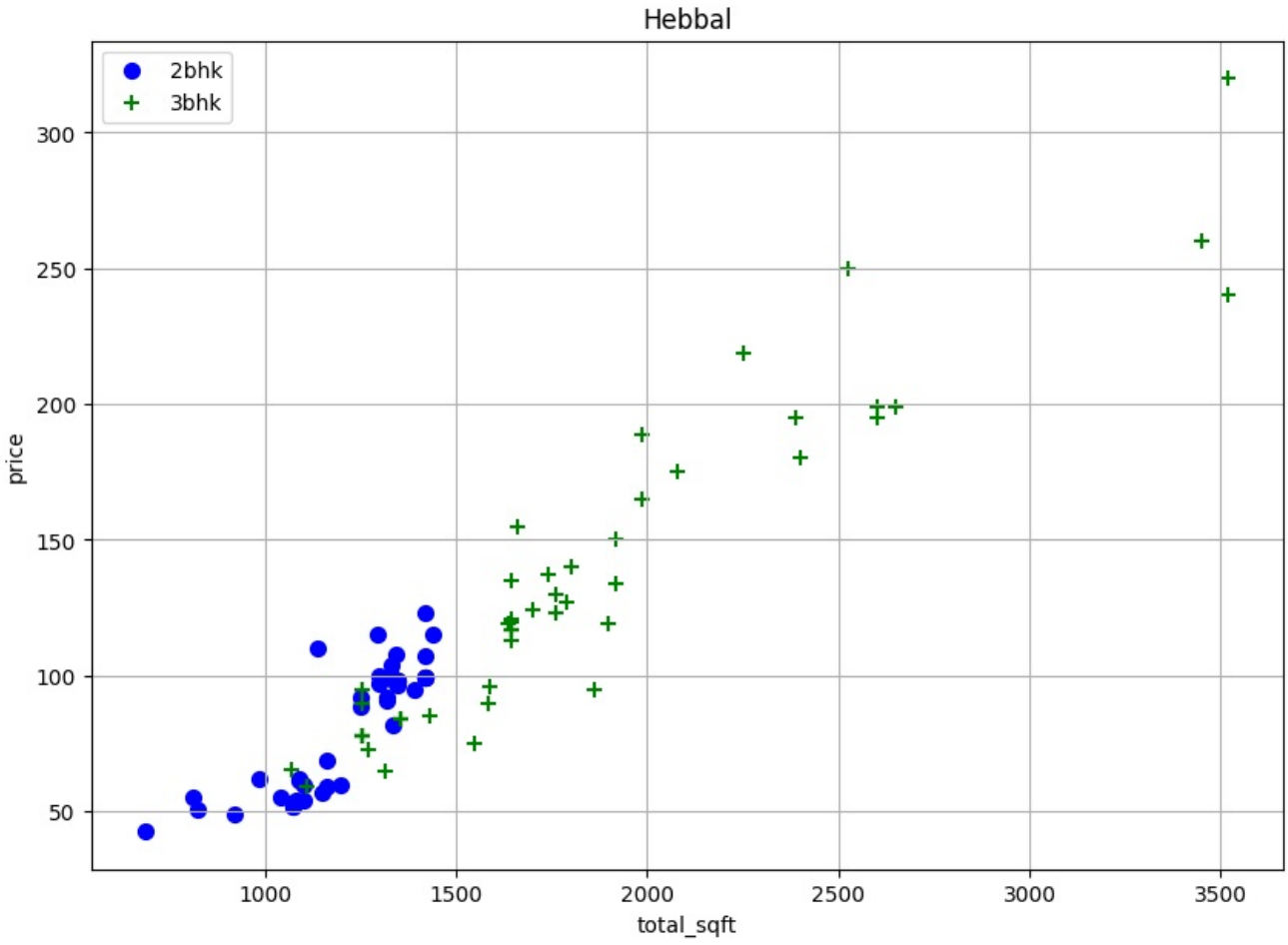
Out[47]:

	location	total_sqft	bath	price	bhk	price_per_sqft
0	Devarachikkanahalli	1250.0	2.0	44.00	3	3520.000000
1	Devarachikkanahalli	1250.0	2.0	40.00	2	3200.000000
2	Devarachikkanahalli	1200.0	2.0	83.00	2	6916.666667
3	Devarachikkanahalli	1170.0	2.0	40.00	2	3418.803419
4	Devarachikkanahalli	1425.0	2.0	65.00	3	4561.403509
...	...	...	...	...	...	...
10239	other	1353.0	2.0	110.00	2	8130.081301
10240	other	812.0	1.0	26.00	1	3201.970443
10241	other	1440.0	2.0	63.93	3	4439.583333
10242	other	1075.0	2.0	48.00	2	4465.116279
10243	other	3600.0	5.0	400.00	4	11111.111111

10244 rows × 6 columns

In [48]:
```
def scatter_plot(df,location):
    bhk2=df[(df.location==location) & (df.bhk==2)]
    bhk3=df[(df.location==location) & (df.bhk==3)]
    plt.figure(figsize=(10,7))
    plt.scatter(bhk2.total_sqft,bhk2.price,color='blue',label='2bhk',s=50)
    plt.scatter(bhk3.total_sqft,bhk3.price,marker='+',color='green',label='3bhk',s=50)
    plt.xlabel('total_sqft')
    plt.ylabel('price')
    plt.title(location)
    plt.grid()
    plt.legend()

scatter_plot(df6,"Hebbal")
```



In [49]:
```
def remove_bhk_outliers(df):
    exclude_indices = np.array([])
    for location, location_df in df.groupby('location'):
        bhk_stats = {}
        for bhk, bhk_df in location_df.groupby('bhk'):
            bhk_stats[bhk] = {
                'mean': np.mean(bhk_df.price_per_sqft),
                'std': np.std(bhk_df.price_per_sqft),
```

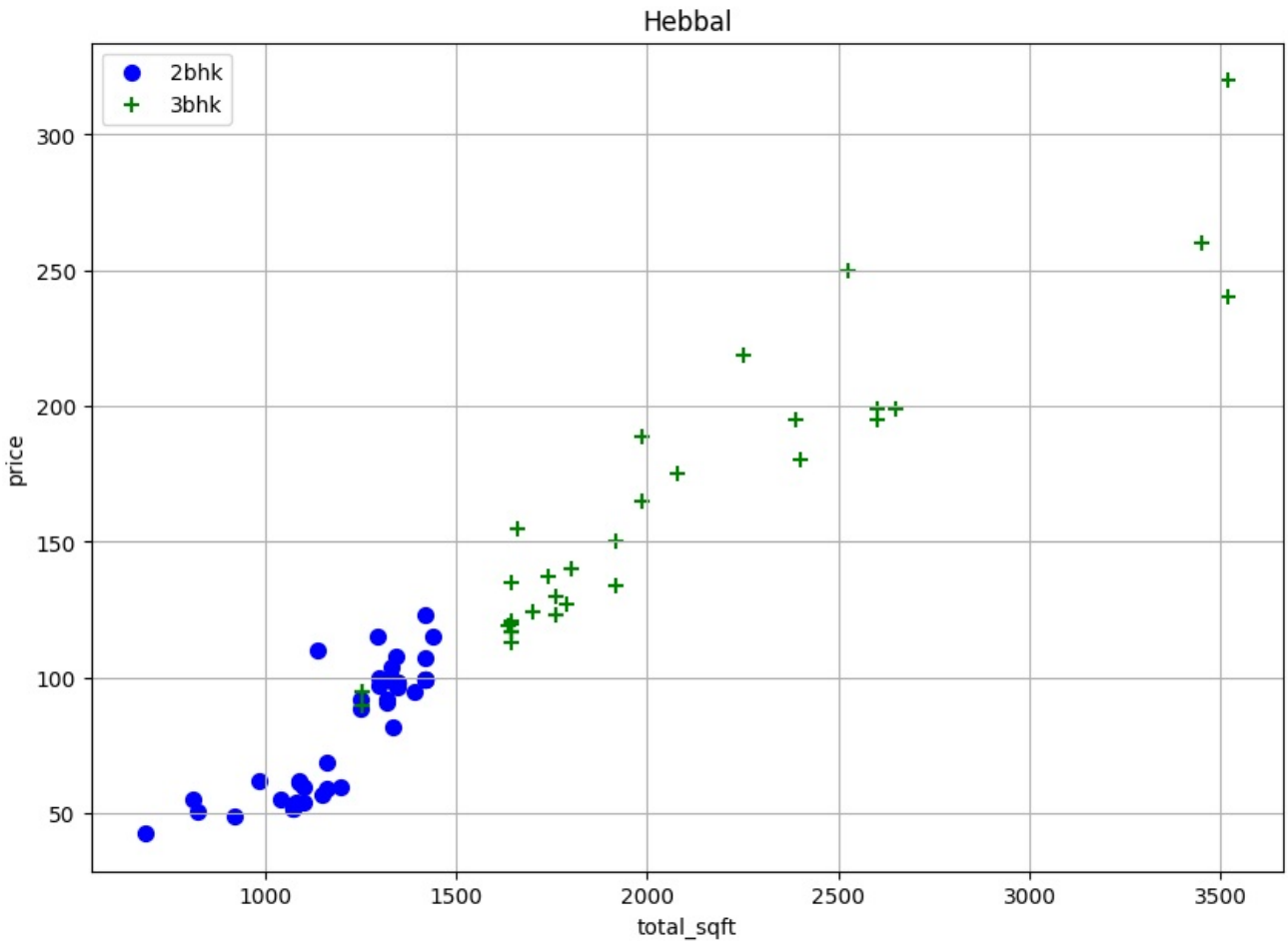
```
                'count': bhk_df.shape[0]
            }
        for bhk, bhk_df in location_df.groupby('bhk'):
            stats = bhk_stats.get(bhk-1)
            if stats and stats['count']>5:
                exclude_indices = np.append(exclude_indices, bhk_df[bhk_df.price_per_sqft<(stats['mean'])].index
    return df.drop(exclude_indices,axis='index')
df7 = remove_bhk_outliers(df6)
df7
```

Out[49]:

	location	total_sqft	bath	price	bhk	price_per_sqft
1	Devarachikkanahalli	1250.0	2.0	40.0	2	3200.000000
2	Devarachikkanahalli	1200.0	2.0	83.0	2	6916.666667
3	Devarachikkanahalli	1170.0	2.0	40.0	2	3418.803419
4	Devarachikkanahalli	1425.0	2.0	65.0	3	4561.403509
5	Devarachikkanahalli	947.0	2.0	43.0	2	4540.654699
...	...	...	...	...	...	...
10235	other	1200.0	2.0	70.0	2	5833.333333
10236	other	1800.0	1.0	200.0	1	11111.111111
10239	other	1353.0	2.0	110.0	2	8130.081301
10240	other	812.0	1.0	26.0	1	3201.970443
10243	other	3600.0	5.0	400.0	4	11111.111111

7342 rows × 6 columns

In [50]:
```
scatter_plot(df7,"Hebbal")
```



In [51]:
```
df7.location.unique()
```

Out[51]:
```
array([' Devarachikkanahalli', '1st Block Jayanagar',
       '1st Phase JP Nagar', '2nd Phase Judicial Layout',
       '2nd Stage Nagarbhavi', '5th Block Hbr Layout',
       '5th Phase JP Nagar', '6th Phase JP Nagar', '7th Phase JP Nagar',
       '8th Phase JP Nagar', '9th Phase JP Nagar', 'AECS Layout',
       'Abbigere', 'Akshaya Nagar', 'Ambalipura', 'Ambedkar Nagar',
       'Amruthahalli', 'Anandapura', 'Ananth Nagar', 'Anekal',
       'Anjanapura', 'Ardendale', 'Arekere', 'Attibele', 'BEML Layout',
       'BTM 2nd Stage', 'BTM Layout', 'Babusapalaya', 'Badavala Nagar',
       'Balagere', 'Banashankari', 'Banashankari Stage II',
       'Banashankari Stage III', 'Banashankari Stage V',
       'Banashankari Stage VI', 'Banaswadi', 'Banjara Layout',
       'Bannerghatta', 'Bannerghatta Road', 'Basavangudi',
       'Basaveshwara Nagar', 'Battarahalli', 'Begur', 'Begur Road',
       'Bellandur', 'Benson Town', 'Bharathi Nagar', 'Bhoganhalli',
       'Billekahalli', 'Binny Pete', 'Bisuvanahalli', 'Bommanahalli',
       'Bommasandra', 'Bommasandra Industrial Area', 'Bommenahalli',
       'Brookefield', 'Budigere', 'CV Raman Nagar', 'Chamrajpet',
       'Chandapura', 'Channasandra', 'Chikka Tirupathi', 'Chikkabanavar',
       'Chikkalasandra', 'Choodasandra', 'Cooke Town', 'Cox Town',
       'Cunningham Road', 'Dasanapura', 'Dasarahalli', 'Devanahalli',
       'Dodda Nekkundi', 'Doddaballapur', 'Doddakallasandra',
       'Doddathoguru', 'Domlur', 'Dommasandra', 'EPIP Zone',
       'Electronic City', 'Electronic City Phase II',
       'Electronics City Phase 1', 'Frazer Town', 'GM Palaya',
       'Garudachar Palya', 'Giri Nagar', 'Gollarapalya Hosahalli',
       'Gottigere', 'Green Glen Layout', 'Gubbalala', 'Gunjur',
       'HAL 2nd Stage', 'HBR Layout', 'HRBR Layout', 'HSR Layout',
       'Haralur Road', 'Harlur', 'Hebbal', 'Hebbal Kempapura',
       'Hegde Nagar', 'Hennur', 'Hennur Road', 'Hoodi', 'Horamavu Agara',
       'Horamavu Banaswadi', 'Hormavu', 'Hosa Road', 'Hosakerehalli',
       'Hoskote', 'Hosur Road', 'Hulimavu', 'ISRO Layout', 'ITPL',
       'Iblur Village', 'Indira Nagar', 'JP Nagar', 'Jakkur', 'Jalahalli',
       'Jalahalli East', 'Jigani', 'Judicial Layout', 'KR Puram',
       'Kadubeesanahalli', 'Kadugodi', 'Kaggadasapura', 'Kaggalipura',
       'Kaikondrahalli', 'Kalena Agrahara', 'Kalyan nagar', 'Kambipura',
       'Kammanahalli', 'Kammasandra', 'Kanakapura', 'Kanakpura Road',
       'Kannamangala', 'Karuna Nagar', 'Kasavanhalli', 'Kasturi Nagar',
       'Kathriguppe', 'Kaval Byrasandra', 'Kenchenahalli', 'Kengeri',
       'Kengeri Satellite Town', 'Kereguddadahalli', 'Kodichikkanahalli',
       'Kodigehaali', 'Kodigehalli', 'Kodihalli', 'Kogilu', 'Konanakunte',
       'Koramangala', 'Kothannur', 'Kothanur', 'Kudlu', 'Kudlu Gate',
       'Kumaraswami Layout', 'Kundalahalli', 'LB Shastri Nagar',
       'Laggere', 'Lakshminarayana Pura', 'Lingadheeranahalli',
       'Magadi Road', 'Mahadevpura', 'Mahalakshmi Layout', 'Mallasandra',
       'Malleshpalya', 'Malleshwaram', 'Marathahalli', 'Margondanahalli',
       'Marsur', 'Mico Layout', 'Munnekollal', 'Murugeshpalya',
       'Mysore Road', 'NGR Layout', 'NRI Layout', 'Nagarbhavi',
       'Nagasandra', 'Nagavara', 'Nagavarapalya', 'Narayanapura',
       'Neeladri Nagar', 'Nehru Nagar', 'OMBR Layout', 'Old Airport Road',
       'Old Madras Road', 'Padmanabhanagar', 'Pai Layout', 'Panathur',
       'Parappana Agrahara', 'Pattandur Agrahara', 'Poorna Pragna Layout',
       'Prithvi Layout', 'R.T. Nagar', 'Rachenahalli',
       'Raja Rajeshwari Nagar', 'Rajaji Nagar', 'Rajiv Nagar',
       'Ramagondanahalli', 'Ramamurthy Nagar', 'Rayasandra',
       'Sahakara Nagar', 'Sanjay nagar', 'Sarakki Nagar', 'Sarjapur',
       'Sarjapur  Road', 'Sarjapura - Attibele Road',
       'Sector 2 HSR Layout', 'Sector 7 HSR Layout', 'Seegehalli',
       'Shampura', 'Shivaji Nagar', 'Singasandra', 'Somasundara Palya',
       'Sompura', 'Sonnenahalli', 'Subramanyapura', 'Sultan Palaya',
       'TC Palaya', 'Talaghattapura', 'Thanisandra', 'Thigalarapalya',
       'Thubarahalli', 'Thyagaraja Nagar', 'Tindlu', 'Tumkur Road',
       'Ulsoor', 'Uttarahalli', 'Varthur', 'Varthur Road', 'Vasanthapura',
       'Vidyaranyapura', 'Vijayanagar', 'Vishveshwarya Layout',
       'Vishwapriya Layout', 'Vittasandra', 'Whitefield',
       'Yelachenahalli', 'Yelahanka', 'Yelahanka New Town', 'Yelenahalli',
       'Yeshwanthpur', 'other'], dtype=object)
```

In [52]:
```
len(df7.location.unique())
```

Out[52]:
```
242
```

We can also label the categorical data using pandas get dummies function

In [53]:
```
d=pd.get_dummies(df7.location)
d
```

Out[53]:

	Devarachikkanahalli	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	6th Phase JP Nagar	7th Phase JP Nagar	8th Phase JP Nagar	...	Vishveshwarya Layout	Vis
1	1	0	0	0	0	0	0	0	0	0	...	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	
3	1	0	0	0	0	0	0	0	0	0	...	0	
4	1	0	0	0	0	0	0	0	0	0	...	0	
5	1	0	0	0	0	0	0	0	0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
10235	0	0	0	0	0	0	0	0	0	0	...	0	
10236	0	0	0	0	0	0	0	0	0	0	...	0	
10239	0	0	0	0	0	0	0	0	0	0	...	0	
10240	0	0	0	0	0	0	0	0	0	0	...	0	
10243	0	0	0	0	0	0	0	0	0	0	...	0	

7342 rows × 242 columns

In [54]:
```
df8=pd.concat([df7,d],axis=1)
df8
```

Out[54]:

	location	total_sqft	bath	price	bhk	price_per_sqft	Devarachikkanahalli	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	...	Vishve
1	Devarachikkanahalli	1250.0	2.0	40.0	2	3200.000000	1	0	0	0	...	
2	Devarachikkanahalli	1200.0	2.0	83.0	2	6916.666667	1	0	0	0	...	
3	Devarachikkanahalli	1170.0	2.0	40.0	2	3418.803419	1	0	0	0	...	
4	Devarachikkanahalli	1425.0	2.0	65.0	3	4561.403509	1	0	0	0	...	
5	Devarachikkanahalli	947.0	2.0	43.0	2	4540.654699	1	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
10235	other	1200.0	2.0	70.0	2	5833.333333	0	0	0	0	...	
10236	other	1800.0	1.0	200.0	1	11111.111111	0	0	0	0	...	
10239	other	1353.0	2.0	110.0	2	8130.081301	0	0	0	0	...	
10240	other	812.0	1.0	26.0	1	3201.970443	0	0	0	0	...	
10243	other	3600.0	5.0	400.0	4	11111.111111	0	0	0	0	...	

7342 rows × 248 columns

In [55]:
```
x=df8.drop(['location','price_per_sqft','price','other'],axis=1)
x
```

Out[55]:

	total_sqft	bath	bhk	Devarachikkanahalli	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	...	Vijayanagar	Vishves
1	1250.0	2.0	2	1	0	0	0	0	0	0	...	0	
2	1200.0	2.0	2	1	0	0	0	0	0	0	...	0	
3	1170.0	2.0	2	1	0	0	0	0	0	0	...	0	
4	1425.0	2.0	3	1	0	0	0	0	0	0	...	0	
5	947.0	2.0	2	1	0	0	0	0	0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
10235	1200.0	2.0	2	0	0	0	0	0	0	0	...	0	
10236	1800.0	1.0	1	0	0	0	0	0	0	0	...	0	
10239	1353.0	2.0	2	0	0	0	0	0	0	0	...	0	
10240	812.0	1.0	1	0	0	0	0	0	0	0	...	0	
10243	3600.0	5.0	4	0	0	0	0	0	0	0	...	0	

7342 rows × 244 columns

In [56]:
```
y=df8.price
y
```

Out[56]:
```
1         40.0
2         83.0
3         40.0
4         65.0
5         43.0
         ...
10235     70.0
10236    200.0
10239    110.0
10240     26.0
10243    400.0
Name: price, Length: 7342, dtype: float64
```

In [57]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=10)
```

In [58]:
```
x_train
```

Out[58]:

	total_sqft	bath	bhk	Devarachikkanahalli	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	...	Vijayanagar	Vishves
1286	1075.0	2.0	3	0	0	0	0	0	0	0	...	0	
3465	1370.0	2.0	3	0	0	0	0	0	0	0	...	0	
7753	694.0	1.0	1	0	0	0	0	0	0	0	...	0	
2693	1225.0	2.0	2	0	0	0	0	0	0	0	...	0	
2158	1025.0	2.0	2	0	0	0	0	0	0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
4244	1200.0	2.0	2	0	0	0	0	0	0	0	...	0	
1603	1050.0	2.0	2	0	0	0	0	0	0	0	...	0	
5663	1140.0	2.0	2	0	0	0	0	0	0	0	...	0	
10141	1200.0	4.0	4	0	0	0	0	0	0	0	...	0	
1532	975.0	2.0	2	0	0	0	0	0	0	0	...	0	

5873 rows × 244 columns

In [59]:
```
len(x_train)
```

Out[59]:
```
5873
```

In [60]:
```
x_test
```

Out[60]:

	total_sqft	bath	bhk	Devarachikkanahalli	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	...	Vijayanagar	Vishves
8826	1170.0	2.0	2	0	0	0	0	0	0	0	...	0	
1895	1060.0	2.0	2	0	0	0	0	0	0	0	...	0	
4337	1020.0	2.0	2	0	0	0	0	0	0	0	...	0	
6438	1360.0	2.0	3	0	0	0	0	0	0	0	...	0	
4761	1047.0	2.0	2	0	0	0	0	0	0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
3040	1232.0	2.0	2	0	0	0	0	0	0	0	...	0	
9808	1200.0	2.0	3	0	0	0	0	0	0	0	...	0	
6335	2400.0	3.0	3	0	0	0	0	0	0	0	...	0	
3294	1400.0	2.0	2	0	0	0	0	0	0	0	...	0	
10044	1050.0	2.0	2	0	0	0	0	0	0	0	...	0	

1469 rows × 244 columns

In [61]:
```
len(x_test)
```

Out[61]: 1469

In [62]:
```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import Lasso
```

In [63]:
```
m=LinearRegression()
m.fit(x_train,y_train)
m.score(x_test,y_test)
```

Out[63]: 0.8058812738705599

In [64]:
```
m1=DecisionTreeRegressor()
m1.fit(x_train,y_train)
m1.score(x_test,y_test)
```

Out[64]: 0.7290327208319309

In [65]:
```
m2=Lasso()
m2.fit(x_train,y_train)
m2.score(x_test,y_test)
```

Out[65]: 0.6969657087739667

In [66]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import GridSearchCV
```

In [67]:
```
cv=ShuffleSplit(n_splits=5,test_size=0.2,random_state=0)
```

In [68]:
```
cvs=cross_val_score(LinearRegression(),x,y,cv=cv)
cvs
```

Out[68]: array([0.80448443, 0.82608426, 0.84371263, 0.84698018, 0.80724343])

In [69]:
```
cvs1=cross_val_score(DecisionTreeRegressor(),x,y,cv=cv)
cvs1
```

Out[69]: array([0.5956277 , 0.54745308, 0.74537936, 0.79903168, 0.73517185])

In [70]:
```
cvs2=cross_val_score(Lasso(),x,y,cv=cv)
cvs2
```

Out[70]: array([0.70729852, 0.68443602, 0.6841262 , 0.68136161, 0.66197207])

In [71]:
```
def find_best_algorithm_using_grid_searchcv(x,y):
    algos={
        'LinearRegression':{
            'model':LinearRegression(),
            'params':{
                'normalize':[True,False]
```

```
            }
        },
        'Lasso':{
            'model':Lasso(),
            'params':{
                'alpha':[1,2],
                'selection':['random','cyclic']
            }
        },
        'DecissionTreeRegressor':{
            'model':DecisionTreeRegressor(),
            'params':{
                'criterion':['mse','friedman mse'],
                'splitter':['best','random']
            }
        }

    }
    score=[]
    cv=ShuffleSplit(n_splits=5,test_size=0.2,random_state=10)
    for algo_name,config in algos.items():
        gsc=GridSearchCV(config['model'],config['params'],cv=cv)
        gsc.fit(x,y)
        score.append({
            'model':algo_name,
            'best_score':gsc.best_score_,
            'best_params':gsc.best_params_
        })
    return pd.DataFrame(score,columns=['model','best_score','best_params'])
```

In [74]:
```
find_best_algorithm_using_grid_searchcv(x,y)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[74], line 1
----> 1 find_best_algorithm_using_grid_searchcv(x,y)

Cell In[71], line 29, in find_best_algorithm_using_grid_searchcv(x, y)
     27 for algo_name,config in algos.items():
     28     gsc=GridSearchCV(config['model'],config['params'],cv=cv)
---> 29     gsc.fit(x,y)
     30     score.append({
     31         'model':algo_name,
     32         'best_score':gsc.best_score_,
     33         'best_params':gsc.best_params_
     34     })
     35 return pd.DataFrame(score,columns=['model','best_score','best_params'])

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_search.py:874, in BaseSearchCV.fit(self, X, y, groups, **fit_params)
    868     results = self._format_results(
    869         all_candidate_params, n_splits, all_out, all_more_results
    870     )
    872     return results
--> 874 self._run_search(evaluate_candidates)
    876 # multimetric is determined here because in the case of a callable
    877 # self.scoring the return type is only known after calling
    878 first_test_score = all_out[0]["test_scores"]

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_search.py:1388, in GridSearchCV._run_search(self, evaluate_candidates)
   1386 def _run_search(self, evaluate_candidates):
   1387     """Search all candidates in param_grid"""
-> 1388     evaluate_candidates(ParameterGrid(self.param_grid))

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_search.py:821, in BaseSearchCV.fit.<locals>.evaluate_candidates(candidate_params, cv, more_results)
    813 if self.verbose > 0:
    814     print(
    815         "Fitting {0} folds for each of {1} candidates,"
    816         " totalling {2} fits".format(
    817             n_splits, n_candidates, n_candidates * n_splits
    818         )
    819     )
--> 821 out = parallel(
    822     delayed( fit and score)(
    823         clone(base_estimator),
    824         X,
    825         y,
    826         train=train,
    827         test=test,
    828         parameters=parameters,
    829         split_progress=(split_idx, n_splits),
```

```
    830         candidate progress=(cand_idx, n_candidates),
    831         **fit_and_score_kwargs,
    832     )
    833     for (cand idx, parameters), (split idx, (train, test)) in product(
    834         enumerate(candidate_params), enumerate(cv.split(X, y, groups))
    835     )
    836 )
    838 if len(out) < 1:
    839     raise ValueError(
    840         "No fits were performed. "
    841         "Was the CV iterator empty? "
    842         "Were there no candidates?"
    843     )

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\parallel.py:63, in Parallel.__cal
l__(self, iterable)
     58 config = get_config()
     59 iterable_with_config = (
     60     (_with_config(delayed_func, config), args, kwargs)
     61     for delayed_func, args, kwargs in iterable
     62 )
---> 63 return super().__call__(iterable_with_config)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:1085, in Parallel.__call__(s
elf, iterable)
   1076 try:
   1077     # Only set self._iterating to True if at least a batch
   1078     # was dispatched. In particular this covers the edge
   (...)
   1082     # was very quick and its callback already dispatched all the
   1083     # remaining jobs.
   1084     self._iterating = False
-> 1085     if self.dispatch_one_batch(iterator):
   1086         self._iterating = self._original_iterator is not None
   1088     while self.dispatch_one_batch(iterator):

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:901, in Parallel.dispatch_on
e_batch(self, iterator)
    899     return False
    900 else:
--> 901     self._dispatch(tasks)
    902     return True

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:819, in Parallel._dispatch(s
elf, batch)
    817 with self._lock:
    818     job_idx = len(self._jobs)
--> 819     job = self._backend.apply_async(batch, callback=cb)
    820     # A job can complete so quickly than its callback is
    821     # called before we get here, causing self._jobs to
    822     # grow. To ensure correct results ordering, .insert is
    823     # used (rather than .append) in the following line
    824     self._jobs.insert(job_idx, job)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\_parallel_backends.py:208, in Sequential
Backend.apply_async(self, func, callback)
    206 def apply_async(self, func, callback=None):
    207     """Schedule a func to be run"""
--> 208     result = ImmediateResult(func)
    209     if callback:
    210         callback(result)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\_parallel_backends.py:597, in ImmediateR
esult.__init__(self, batch)
    594 def __init__(self, batch):
    595     # Don't delay the application, to avoid keeping the input
    596     # arguments in memory
--> 597     self.results = batch()

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:288, in BatchedCalls.__call_
_(self)
    284 def __call__(self):
    285     # Set the default nested backend to self._backend but do not set the
    286     # change the default number of processes to -1
    287     with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 288         return [func(*args, **kwargs)
    289                 for func, args, kwargs in self.items]

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:288, in <listcomp>(.0)
    284 def __call__(self):
    285     # Set the default nested backend to self._backend but do not set the
    286     # change the default number of processes to -1
    287     with parallel_backend(self._backend, n_jobs=self._n_jobs):
```

```
--> 288         return [func(*args, **kwargs)
    289                 for func, args, kwargs in self.items]

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\parallel.py:123, in _FuncWrapper.
__call__(self, *args, **kwargs)
    121     config = {}
    122 with config_context(**config):
--> 123     return self.function(*args, **kwargs)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_validation.py:674, in
_fit_and_score(estimator, X, y, scorer, train, test, verbose, parameters, fit_params, return_train_score, return
_parameters, return_n_test_samples, return_times, return_estimator, split_progress, candidate_progress, error_sc
ore)
    671     for k, v in parameters.items():
    672         cloned_parameters[k] = clone(v, safe=False)
--> 674     estimator = estimator.set_params(**cloned_parameters)
    676 start_time = time.time()
    678 X_train, y_train = _safe_split(estimator, X, y, train)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:205, in BaseEstimator.set_param
s(self, **params)
    203 if key not in valid_params:
    204     local_valid_params = self._get_param_names()
--> 205     raise ValueError(
    206         f"Invalid parameter {key!r} for estimator {self}. "
    207         f"Valid parameters are: {local_valid_params!r}."
    208     )
    210 if delim:
    211     nested_params[key][sub_key] = value

ValueError: Invalid parameter 'normalize' for estimator LinearRegression(). Valid parameters are: ['copy_X', 'fi
t_intercept', 'n_jobs', 'positive'].
```

In [73]:
```
x.columns
```

Out[73]:
```
Index(['total_sqft', 'bath', 'bhk', ' Devarachikkanahalli',
       '1st Block Jayanagar', '1st Phase JP Nagar',
       '2nd Phase Judicial Layout', '2nd Stage Nagarbhavi',
       '5th Block Hbr Layout', '5th Phase JP Nagar',
       ...
       'Vijayanagar', 'Vishveshwarya Layout', 'Vishwapriya Layout',
       'Vittasandra', 'Whitefield', 'Yelachenahalli', 'Yelahanka',
       'Yelahanka New Town', 'Yelenahalli', 'Yeshwanthpur'],
      dtype='object', length=244)
```

In [75]:
```
np.where(x.columns=='total_sqft')[0][0]
```

Out[75]:
```
0
```

In [76]:
```
np.where(x.columns=='2nd Phase Judicial Layout')[0][0]
```

Out[76]:
```
6
```

In [77]:
```
def pedict(sqft,bath,bhk,location):
    li=np.where(x.columns==location)[0][0]
    x1=np.zeros(len(x.columns))
    x1[0]=sqft
    x1[1]=bath
    x1[2]=bhk
    if li>=0:
        x1[li]=1
    return m.predict([x1])[0]
```

In [78]:
```
pedict(1000,2,2,'1st Phase JP Nagar')
```

```
C:\Users\Biswajeet Jena\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:439: UserWarni
ng: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[78]:
```
87.68842893233926
```

In [79]:
```
pedict(1000,2,3,'1st Phase JP Nagar')
```

```
C:\Users\Biswajeet Jena\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:439: UserWarni
ng: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[79]:
```
83.31175971221967
```

here 2bhk price is higher than 3 bhk price because as we can see both of bhk having same areas and it may the rooms are very small in 3 bhk that's why the 2bhk price is higher than 3 bhk price.

In [80]:
```
pedict(2000,4,4,'1st Phase JP Nagar')
```

```
C:\Users\Biswajeet Jena\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:439: UserWarni
ng: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[80]: 168.3936865609789

In [81]:
```
df8[(df8.price>160) & (df8.location =='Indira Nagar')]
```

Out[81]:

	location	total_sqft	bath	price	bhk	price_per_sqft	Devarachikkanahalli	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	...	Vishveshwarya Layout	Vi
3497	Indira Nagar	2800.0	3.0	330.0	3	11785.714286	0	0	0	0	...	0	
3498	Indira Nagar	1650.0	3.0	200.0	3	12121.212121	0	0	0	0	...	0	
3502	Indira Nagar	2070.0	3.0	225.0	3	10869.565217	0	0	0	0	...	0	
3504	Indira Nagar	1400.0	2.0	168.0	2	12000.000000	0	0	0	0	...	0	
3505	Indira Nagar	2400.0	4.0	525.0	4	21875.000000	0	0	0	0	...	0	
3508	Indira Nagar	1470.0	2.0	170.0	2	11564.625850	0	0	0	0	...	0	
3509	Indira Nagar	2700.0	3.0	324.0	3	12000.000000	0	0	0	0	...	0	
3511	Indira Nagar	3250.0	8.0	600.0	8	18461.538462	0	0	0	0	...	0	
3512	Indira Nagar	2601.0	4.0	312.0	3	11995.386390	0	0	0	0	...	0	
3514	Indira Nagar	1740.0	3.0	191.0	3	10977.011494	0	0	0	0	...	0	
3515	Indira Nagar	1650.0	3.0	220.0	3	13333.333333	0	0	0	0	...	0	
3518	Indira Nagar	1440.0	3.0	275.0	3	19097.222222	0	0	0	0	...	0	
3519	Indira Nagar	1475.0	2.0	171.0	2	11593.220339	0	0	0	0	...	0	
3520	Indira Nagar	3200.0	4.0	440.0	4	13750.000000	0	0	0	0	...	0	
3521	Indira Nagar	4000.0	4.0	700.0	4	17500.000000	0	0	0	0	...	0	
3523	Indira Nagar	2400.0	4.0	405.0	4	16875.000000	0	0	0	0	...	0	
3524	Indira Nagar	1800.0	5.0	350.0	5	19444.444444	0	0	0	0	...	0	
3527	Indira Nagar	1650.0	3.0	200.0	3	12121.212121	0	0	0	0	...	0	
3528	Indira Nagar	2800.0	4.0	365.0	4	13035.714286	0	0	0	0	...	0	
3532	Indira Nagar	2400.0	6.0	475.0	6	19791.666667	0	0	0	0	...	0	

20 rows × 248 columns

In [82]:
```
pedict(1000,2,2,'Indira Nagar')
```

```
C:\Users\Biswajeet Jena\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:439: UserWarni
ng: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[82]: 166.46974451388854

In [83]:
```
pedict(1400,2,2,'Indira Nagar')
```

```
C:\Users\Biswajeet Jena\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:439: UserWarni
ng: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[83]: 196.80784598012815

Save our model to a file using joblib

In [84]:
```
import joblib
```

In [85]:
```
joblib.dump(m,'reppp')
```

Out[85]:
```
['reppp']
```

Save our model to a file using pilckle

In [86]:
```
import pickle
```

In [87]:
```
with open('banglore_property_price_prediction','wb') as f:
    pickle.dump(m,f)
```

In [88]:
```
import json
columns={
    'columns_data':[col.lower() for col in x.columns]
}
with open('column.json','w') as f:
    f.write(json.dumps(columns))
```

In [ ]:

In [ ]: