```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

dataset = pd.read_csv("Bengaluru_House_Data.csv")

dataset.head()
```

```
              area_type   availability                  location
size  \
0  Super built-up  Area         19-Dec  Electronic City Phase II
2 BHK
1            Plot  Area  Ready To Move          Chikka Tirupathi  4
Bedroom
2        Built-up  Area  Ready To Move               Uttarahalli
3 BHK
3  Super built-up  Area  Ready To Move        Lingadheeranahalli
3 BHK
4  Super built-up  Area  Ready To Move                  Kothanur
2 BHK

   society total_sqft  bath  balcony   price
0  Coomee        1056   2.0      1.0   39.07
1  Theanmp       2600   5.0      3.0  120.00
2      NaN       1440   2.0      3.0   62.00
3  Soiewre       1521   3.0      1.0   95.00
4      NaN       1200   2.0      1.0   51.00
```

```
dataset.tail()
```

```
                  area_type   availability                  location
size  \
13315        Built-up  Area  Ready To Move             Whitefield  5
Bedroom
13316  Super built-up  Area  Ready To Move          Richards Town
4 BHK
13317        Built-up  Area  Ready To Move  Raja Rajeshwari Nagar
2 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
13315  ArsiaEx       3453   4.0      0.0  231.0
13316      NaN       3600   5.0      NaN  400.0
13317  Mahla T       1141   2.0      1.0   60.0
13318  SollyCl       4689   4.0      1.0  488.0
13319      NaN        550   1.0      1.0   17.0
```

DATA CLEANING

```
* counting each values in the column

dataset['availability'].value_counts()

availability
Ready To Move    10581
18-Dec             307
18-May             295
18-Apr             271
18-Aug             200
                 ...
15-Aug               1
17-Jan               1
16-Nov               1
16-Jan               1
14-Jul               1
Name: count, Length: 81, dtype: int64

dataset['area_type'].value_counts()

area_type
Super built-up  Area    8790
Built-up  Area          2418
Plot  Area              2025
Carpet  Area              87
Name: count, dtype: int64

* Drop some unwanted columns

dataset.drop(['area_type','availability','society','balcony'],axis=1,i
nplace=True)

dataset.head()

                    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

dataset.isnull().sum()

location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

```
* Checking for NaN values

print(dataset.isnull().sum())

location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

If we want to fill the null value we can use the *datsaet.fillna()* method if we want to fill with the specific values such as mean, median the we can use *dataset.fillna(dataset.mean())* or *dataset.fillna(dataset.meian())* and other specific values such as 0 then *dataset.fillna(0)* and using forward and backward filling *dataset.ffill()* for forward filling and *dataset.bfill()* for backward filling or else we can simply drop the Null rows using *dataset.dropna()*

```
## remove the null value

df2 = dataset.dropna()

df2.isnull().sum()

location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64

df2.head(10)

                   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
5                Whitefield      2 BHK       1170   2.0   38.00
6          Old Airport Road      4 BHK       2732   4.0  204.00
7              Rajaji Nagar      4 BHK       3300   4.0  600.00
8              Marathahalli      3 BHK       1310   3.0   63.25
9              Gandhi Bazar  6 Bedroom       1020   6.0  370.00

## make size column more convenient to read by creating new column
name BHK

##Lambda functions are often used in situations where a small, short-
lived function is needed,
```

```
# and defining a full function using the def keyword might be overly
verbose.
# They are commonly used with functions like map, filter, and apply in
Python.

df2['BHK']=df2['size'].apply(lambda x: x.split(' ')[0]).astype(int)
```

```
C:\Users\rsibr\AppData\Local\Temp\ipykernel_11148\4030866578.py:7:
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#
returning-a-view-versus-a-copy
  df2['BHK']=df2['size'].apply(lambda x: x.split(' ')[0]).astype(int)
```

```
df2.head()
```

	location	size	total_sqft	bath	price	BHK
0	Electronic City Phase II	2 BHK	1056	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600	5.0	120.00	4
2	Uttarahalli	3 BHK	1440	2.0	62.00	3
3	Lingadheeranahalli	3 BHK	1521	3.0	95.00	3
4	Kothanur	2 BHK	1200	2.0	51.00	2

```
## Find the unique values
df2['location'].unique()
```

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

```
## Access the dataset which has the BHK value greater than 20
df2[df2.BHK>20]
```

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

```
df2['total_sqft'].unique() ## There are ranges in the total_sqft
columns
```

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

```
def is_float(X):
    try:
```

```
        float(X)
        #float(df2.iloc[1,2])
    except:
        return False
        #float(df2.iloc[30,2])
    return True



df2[~df2['total_sqft'].apply(is_float)].head(10)
```

	location	size	total_sqft	bath	price	BHK
30	Yelahanka	4 BHK	2100 - 2850	4.0	186.000	4
122	Hebbal	4 BHK	3067 - 8156	4.0	477.000	4
137	8th Phase JP Nagar	2 BHK	1042 - 1105	2.0	54.005	2
165	Sarjapur	2 BHK	1145 - 1340	2.0	43.490	2
188	KR Puram	2 BHK	1015 - 1540	2.0	56.800	2
410	Kengeri	1 BHK	34.46Sq. Meter	1.0	18.500	1
549	Hennur Road	2 BHK	1195 - 1440	2.0	63.770	2
648	Arekere	9 Bedroom	4125Perch	9.0	265.000	9
661	Yelahanka	2 BHK	1120 - 1145	2.0	48.130	2
672	Bettahalsoor	4 Bedroom	3090 - 5002	4.0	445.000	4

```
## Handle the ranges and NaN values

def convert_range_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

df2['total_sqft'] = df2['total_sqft'].apply(convert_range_to_num)
```

```
C:\Users\rsibr\AppData\Local\Temp\ipykernel_11148\597009050.py:12:
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#
returning-a-view-versus-a-copy
  df2['total_sqft'] = df2['total_sqft'].apply(convert_range_to_num)
```

```
df2.isnull().sum()
```

```
location       0
size           0
total_sqft    46
```

```
bath            0
price           0
BHK             0
dtype: int64
```

```
## drop NaN values
df3=df2.dropna()

df3.isnull().sum()
```

```
location      0
size          0
total_sqft    0
bath          0
price         0
BHK           0
dtype: int64
```

FEATURE ENGINEERING

Feature engineering is the process of creating new features or modifying existing ones in a dataset to improve the performance of a machine learning model. It involves transforming raw data into a format that better represents the underlying problem and enhances the model's ability to make accurate predictions.

```
df3.head()
```

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

```
## Creating new column name price_per_sqft to store the price

df3['price_per_sqft'] = (df3['price'] * 100000 )/ df3['total_sqft']
```

```
C:\Users\rsibr\AppData\Local\Temp\ipykernel_11148\1744370180.py:3:
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#
returning-a-view-versus-a-copy
  df3['price_per_sqft'] = (df3['price'] * 100000 )/ df3['total_sqft']
```

```
## Location column is actully a categorical column so if we have so
many category it will be a problem

print(f"Total categories in the location column :
{len(df3.location.unique())}")
```

```
Total categories in the location column : 1298
```

```
## Get the counts of location rows per location

df3.location = df3.location.apply(lambda x: x.strip())
location_stats = df3.groupby('location')
['location'].agg('count').sort_values(ascending = False)
location_stats
```

```
C:\Users\rsibr\AppData\Local\Temp\ipykernel_11148\2767593622.py:3:
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#
returning-a-view-versus-a-copy
  df3.location = df3.location.apply(lambda x: x.strip())

location
Whitefield               533
Sarjapur  Road           392
Electronic City          304
Kanakpura Road           264
Thanisandra              235
                        ...
1 Giri Nagar               1
Kanakapura Road,           1
Kanakapura main  Road      1
Kannur                     1
whitefiled                 1
Name: location, Length: 1287, dtype: int64
```

I am going to create a one location call other location and assign all the locations which has less than 10 rows per location

```
len(location_stats[location_stats<10])
#location_less_than_10_data = df3.location
```

```
1033
```

```
location_with_lessthan_10_raws = location_stats[location_stats<10]

df3.location = df3.location.apply(lambda x: 'other' if x in
location_with_lessthan_10_raws else x)
```

```
C:\Users\rsibr\AppData\Local\Temp\ipykernel_11148\1741390667.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#
returning-a-view-versus-a-copy
  df3.location = df3.location.apply(lambda x: 'other' if x in
location_with_lessthan_10_raws else x)

df3

                        location       size  total_sqft  bath   price
BHK  \
0      Electronic City Phase II      2 BHK      1056.0   2.0   39.07
2
1              Chikka Tirupathi  4 Bedroom      2600.0   5.0  120.00
4
2                   Uttarahalli      3 BHK      1440.0   2.0   62.00
3
3            Lingadheeranahalli      3 BHK      1521.0   3.0   95.00
3
4                      Kothanur      2 BHK      1200.0   2.0   51.00
2
...                         ...        ...         ...   ...     ...
...
13315                Whitefield  5 Bedroom      3453.0   4.0  231.00
5
13316                     other      4 BHK      3600.0   5.0  400.00
4
13317     Raja Rajeshwari Nagar      2 BHK      1141.0   2.0   60.00
2
13318           Padmanabhanagar      4 BHK      4689.0   4.0  488.00
4
13319              Doddathoguru      1 BHK       550.0   1.0   17.00
1

       price_per_sqft
0         3699.810606
1         4615.384615
2         4305.555556
3         6245.890861
4         4250.000000
...               ...
13315     6689.834926
13316    11111.111111
13317     5258.545136
13318    10407.336319
13319     3090.909091
```

```
[13200 rows x 7 columns]
```

```
print(f"After assign location to others in the location column, unique category is : {len(df3.location.unique())}")
```

```
After assign location to others in the location column, unique category is : 255
```

OUT-LIER DETECTION AND REMOVE

Outlier detection and removal refer to the process of identifying and handling data points that deviate significantly from the majority of the data in a dataset. Outliers are observations that are unusually distant from other data points and can distort the analysis or modeling process.

```
df3.head()
```

```
                  location       size  total_sqft  bath   price  BHK
\
0  Electronic City Phase II      2 BHK      1056.0   2.0   39.07    2

1          Chikka Tirupathi  4 Bedroom      2600.0   5.0  120.00    4

2               Uttarahalli      3 BHK      1440.0   2.0   62.00    3

3        Lingadheeranahalli      3 BHK      1521.0   3.0   95.00    3

4                  Kothanur      2 BHK      1200.0   2.0   51.00    2


   price_per_sqft
0     3699.810606
1     4615.384615
2     4305.555556
3     6245.890861
4     4250.000000
```

```
df3.drop('size',axis=1,inplace=True)
```

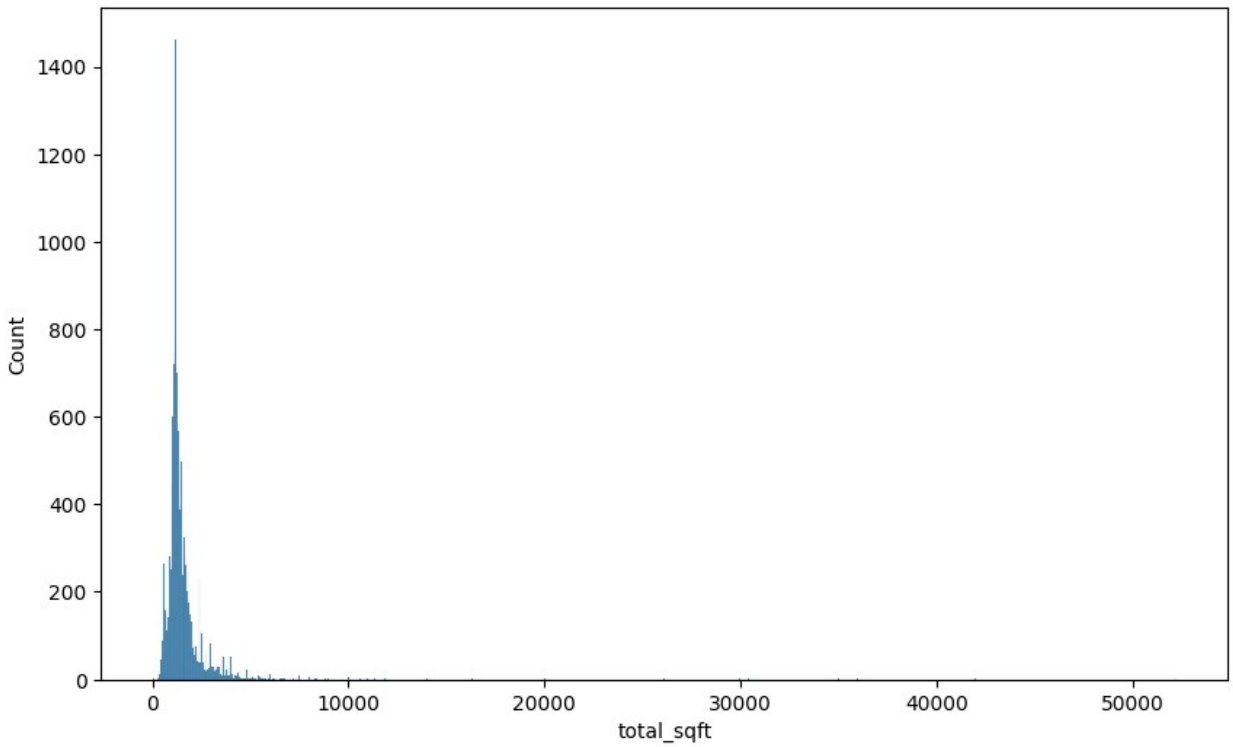
```
C:\Users\rsibr\AppData\Local\Temp\ipykernel_11148\2308154834.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df3.drop('size',axis=1,inplace=True)
```

```
df3.head()
```

```
                   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
```

```
## Visualize the Outlier

import seaborn as sns
plt.figure(figsize=(10,6))
sns.histplot(df3.total_sqft)
```

```
<Axes: xlabel='total_sqft', ylabel='Count'>
```




```
df3.shape
```

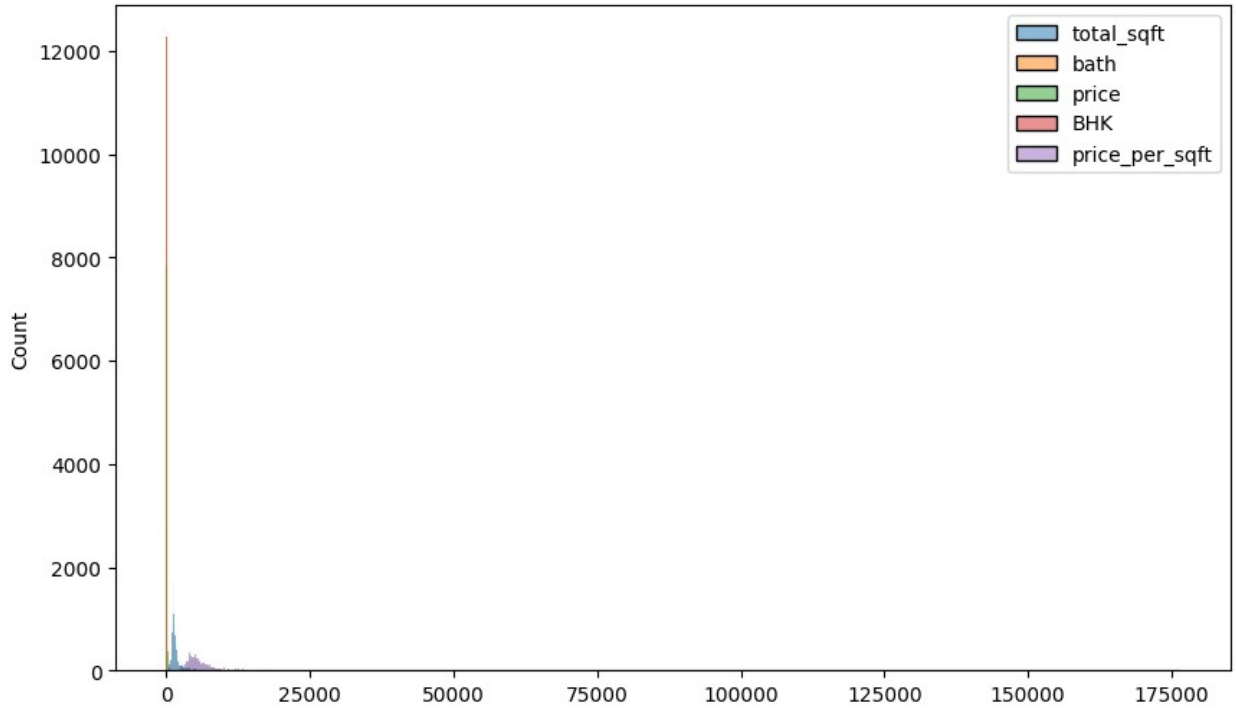
```
(13200, 6)
```

```
## Remove the rows which has the total_sqft for one BHK is lessthan
300(a threshold)

df4 = df3[df3['total_sqft']/df3['BHK'] > 300]
df4.shape
```

```
(12274, 6)
```

```
plt.figure(figsize=(10,6))
sns.histplot(df4)
```

```
<Axes: ylabel='Count'>
```




```
## when we consider the price_per_sqft

df4.price_per_sqft.describe()
```

```
count     12274.000000
mean       6211.880230
std        4053.214807
min         267.829813
25%        4200.000000
50%        5263.157895
75%        6825.474875
max      176470.588235
Name: price_per_sqft, dtype: float64
```

```
## It is imposible to have a price min price Rs. 267 so we have to remove those using sd method

mean = np.mean(df4.price_per_sqft)
std = np.std(df4.price_per_sqft)
mean - std
```

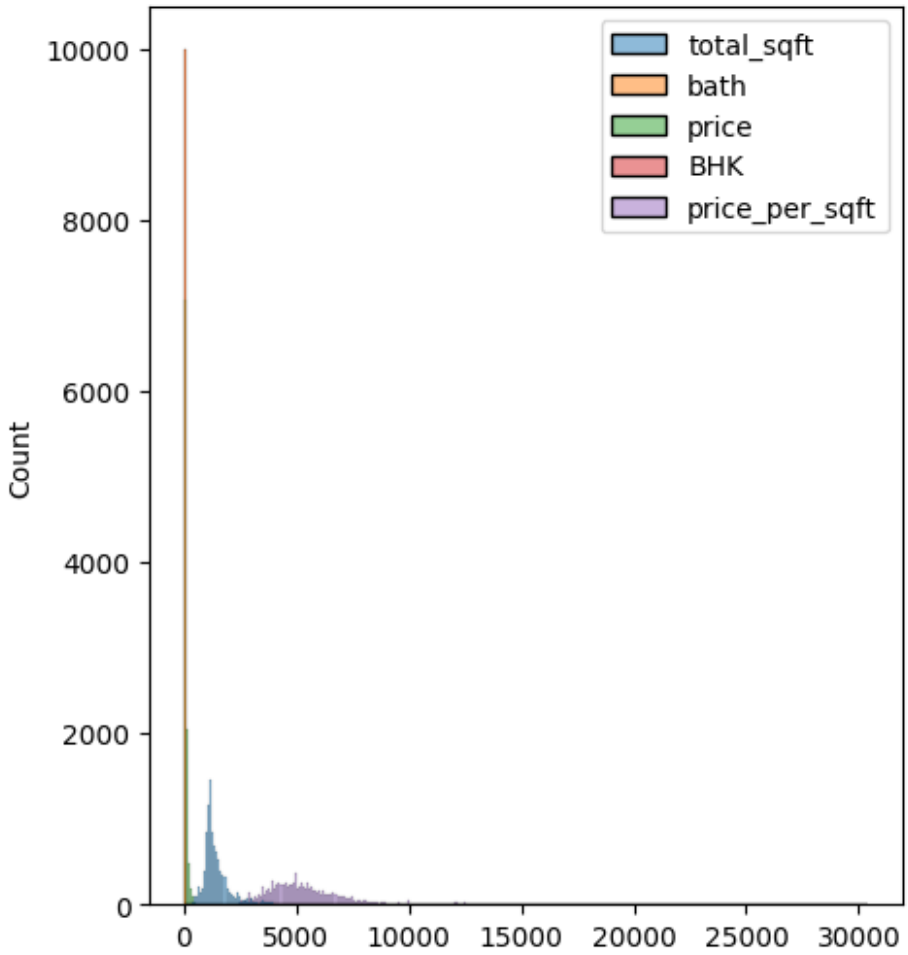
```
2158.8305408000597
```

```
def remove_outl(dataset):
    df_out = pd.DataFrame()
    for key,subdf in dataset.groupby('location'):
        mean = np.mean(subdf.price_per_sqft)
        std = np.std(subdf.price_per_sqft)
        reduced_df = subdf[(subdf.price_per_sqft> (mean - std)) &
(subdf.price_per_sqft < (mean + std))]
        df_out = pd.concat([df_out,reduced_df],ignore_index = True)
    return df_out

df5 = remove_outl(df4)

df5.shape
```

```
(9996, 6)
```

```
plt.figure(figsize=(5,6))
sns.histplot(df5)
```

```
<Axes: ylabel='Count'>
```




```
df5.head(10)
```

	location	total_sqft	bath	price	BHK	price_per_sqft
0	1st Block Jayanagar	2850.0	4.0	428.0	4	15017.543860
1	1st Block Jayanagar	1630.0	3.0	194.0	3	11901.840491
2	1st Block Jayanagar	1875.0	2.0	235.0	3	12533.333333
3	1st Block Jayanagar	1200.0	2.0	130.0	3	10833.333333
4	1st Block Jayanagar	1235.0	2.0	148.0	2	11983.805668
5	1st Block Jayanagar	2750.0	4.0	413.0	4	15018.181818
6	1st Block Jayanagar	2450.0	4.0	368.0	4	15020.408163
7	1st Block Koramangala	1415.0	2.0	110.0	2	7773.851590
8	1st Block Koramangala	860.0	2.0	65.5	2	7616.279070
9	1st Block Koramangala	3000.0	3.0	300.0	4	10000.000000

```
df5.bath.unique()
```

```
array([ 4.,  3.,  2.,  5.,  8.,  1.,  6., 14.,  7.,  9., 12., 16.,
13.])
```

```
## checking bathroom(bath) with bedroom(BHK) in genral it is not
possible to have a bathroom > bedroom so we can find that and remove
```

```
df6 = df5[~(df5.bath > df5.BHK + 1)]
df6.head()
```

	location	total_sqft	bath	price	BHK	price_per_sqft
0	1st Block Jayanagar	2850.0	4.0	428.0	4	15017.543860
1	1st Block Jayanagar	1630.0	3.0	194.0	3	11901.840491
2	1st Block Jayanagar	1875.0	2.0	235.0	3	12533.333333
3	1st Block Jayanagar	1200.0	2.0	130.0	3	10833.333333
4	1st Block Jayanagar	1235.0	2.0	148.0	2	11983.805668

```
df5.shape
```

```
(9996, 6)
```

```
df6.shape
```

```
(9905, 6)
```

```
df6.drop('price_per_sqft',axis=1,inplace=True)
```

```
C:\Users\rsibr\AppData\Local\Temp\ipykernel_11148\296776384.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#
returning-a-view-versus-a-copy
  df6.drop('price_per_sqft',axis=1,inplace=True)
```

```
df6.head()
```

	location	total_sqft	bath	price	BHK
0	1st Block Jayanagar	2850.0	4.0	428.0	4
1	1st Block Jayanagar	1630.0	3.0	194.0	3
2	1st Block Jayanagar	1875.0	2.0	235.0	3
3	1st Block Jayanagar	1200.0	2.0	130.0	3
4	1st Block Jayanagar	1235.0	2.0	148.0	2

```
df6.location.unique()
```

```
array(['1st Block Jayanagar', '1st Block Koramangala',
       '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 1st Stage', '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',
'Basapura',
       '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', 'Dairy Circle', 'Dasanapura',
       'Dasarahalli', 'Devanahalli', 'Devarachikkanahalli',
       'Dodda Nekkundi', 'Doddaballapur', 'Doddakallasandra',
       'Doddathoguru', 'Dodsworth Layout', 'Domlur', 'Dommasandra',
       'EPIP Zone', 'Electronic City', 'Electronic City Phase II',
       'Electronics City Phase 1', 'Frazer Town', 'GM Palaya',
       'Ganga Nagar', 'Garudachar Palya', 'Giri Nagar',
       'Gollarapalya Hosahalli', 'Gottigere', 'Green Glen Layout',
       'Gubbalala', 'Gunjur', 'Gunjur Palya', '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', 'Kalkere', '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', 'Nagadevanahalli',
       'Naganathapura', 'Nagappa Reddy 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',
       'Sadashiva Nagar', 'Sahakara Nagar', 'Sanjay nagar',
       'Sarakki Nagar', 'Sarjapur', 'Sarjapur  Road',
       'Sarjapura - Attibele Road', 'Sector 1 HSR Layout',
       '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)
```

```
## Convert Gategorical DATA to numerical DATA using LabelEncoder()
method

from sklearn import preprocessing as prp

loc = prp.LabelEncoder()

loc.fit(df6.location.unique())

df6['location'] = loc.transform(df6.location)
```

```
C:\Users\rsibr\AppData\Local\Temp\ipykernel_11148\140464787.py:9:
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#
returning-a-view-versus-a-copy
  df6['location'] = loc.transform(df6.location)
```

```
df6
```

	location	total_sqft	bath	price	BHK
0	0	2850.0	4.0	428.00	4
1	0	1630.0	3.0	194.00	3
2	0	1875.0	2.0	235.00	3
3	0	1200.0	2.0	130.00	3
4	0	1235.0	2.0	148.00	2
...	...	...	...	...	...
9991	254	1353.0	2.0	110.00	2
9992	254	812.0	1.0	26.00	1
9993	254	1440.0	2.0	63.93	3
9994	254	1075.0	2.0	48.00	2
9995	254	3600.0	5.0	400.00	4

```
[9905 rows x 5 columns]
```

FEATURE SELECTION

```
X = df6.drop('price',axis=1)
y = df6['price']

from sklearn.model_selection import train_test_split as tts

X_train,X_test,y_train,y_test = tts(X,y,test_size = 0.2, random_state
= 5)
```

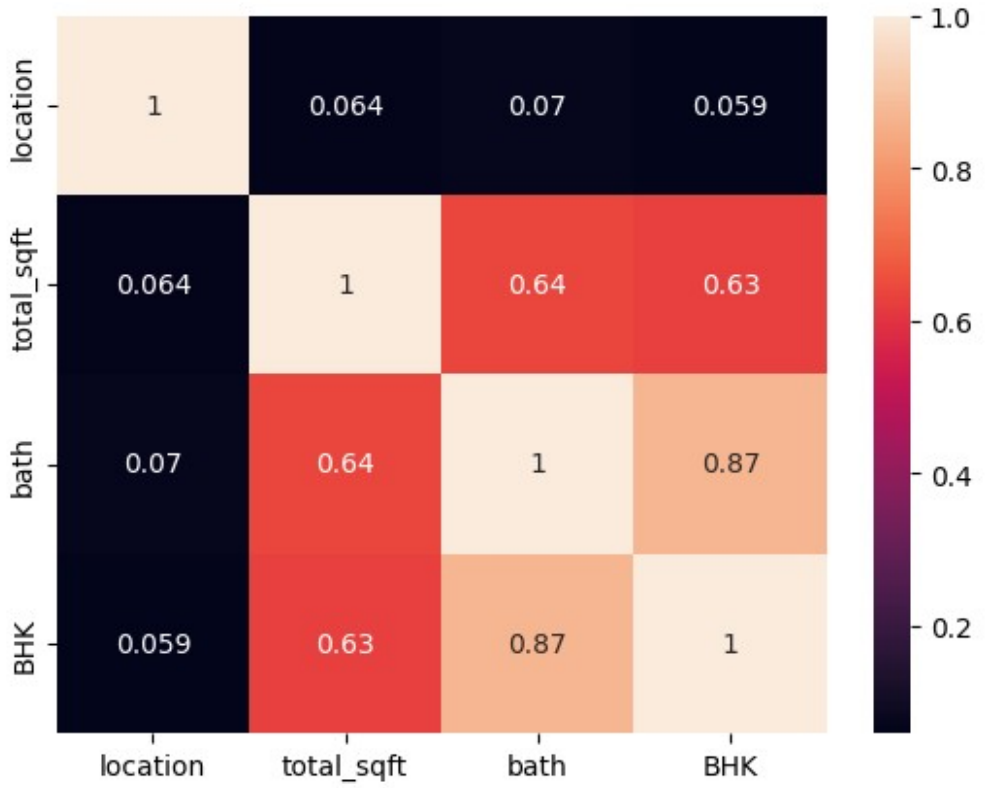
CORRELATION

Note : In this data set we don't want to find the Correlation between features because here there are only 4 features *(WE ONLY DO CHECK THE CORRELATION FOR X_train, but if we want to remove then we should remove the respective X_test row also)* . corr() is useful when you have more columns(features) data

```
## calculate the correlation
corr_mat = X_train.corr()

corr_mat
```

	location	total_sqft	bath	BHK
location	1.000000	0.063973	0.070146	0.058648
total_sqft	0.063973	1.000000	0.639161	0.625529
bath	0.070146	0.639161	1.000000	0.868982
BHK	0.058648	0.625529	0.868982	1.000000

```
## Visualize the correlation using heatmap
sns.heatmap(corr_mat,annot=True)#,cmap=plt.cm.CMRmap_r)
plt.show()
```



MODEL TRAINING

```
* 1.Linear Regression Model

from sklearn.linear_model import LinearRegression

lin_model = LinearRegression()
lin_model.fit(X_train,y_train)

LinearRegression()

predict = lin_model.predict(X_test)

print(predict[0:5])
print(y_test[0:5])

lin_model.score(X_test,y_test)

[ 55.05439694 125.37934033  89.35605717  53.8910169   76.6676293 ]
8792     46.79
```

```
5776    131.00
2216     48.75
3273     68.59
7264     55.00
Name: price, dtype: float64

0.6419718092586191
```