```
import pandas as pd
import numpy as np


from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score


data = pd.read_csv('Bengaluru_House_Data.csv')


data.drop(columns=['area_type', 'availability', 'society', 'balcony'], inplace=True)


data.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

```
data.isna().sum()
```

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

```
data.describe()
```

	bath	price
count	13247.000000	13320.000000
mean	2.692610	112.565627
std	1.341458	148.971674
min	1.000000	8.000000
25%	2.000000	50.000000
50%	2.000000	72.000000
75%	3.000000	120.000000
max	40.000000	3600.000000

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   location    13319 non-null  object 
 1   size        13304 non-null  object 
 2   total_sqft  13320 non-null  object 
 3   bath        13247 non-null  float64
 4   price       13320 non-null  float64
dtypes: float64(2), object(3)
memory usage: 520.4+ KB
```

```
# Replacing misssing values
```

```
data['location']=data['location'].fillna('Sarjapur Road')
```

```
data['size']=data['size'].fillna('2 BHK')
```

```
data['bath']=data['bath'].fillna(data['bath'].median())
```

```
data['bhk']=data['size'].str.split().str.get(0).astype(int)
```

```
data[data.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

```
data['total_sqft'].unique()
```

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

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

```
data['total_sqft']=data['total_sqft'].apply(convertRange)
```

```
data.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

```
data['price_per_sqft'] = data['price'] *100000/ data['total_sqft']
```

```
data['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
Name: price_per_sqft, Length: 13320, dtype: float64
```

```
data.describe()
```

	total_sqft	bath	price	bhk	price_per_sqft
count	13274.000000	13320.000000	13320.000000	13320.000000	1.327400e+04
mean	1559.626694	2.688814	112.565627	2.802778	7.907501e+03
std	1238.405258	1.338754	148.971674	1.294496	1.064296e+05
min	1.000000	1.000000	8.000000	1.000000	2.678298e+02
25%	1100.000000	2.000000	50.000000	2.000000	4.266865e+03
50%	1276.000000	2.000000	72.000000	3.000000	5.434306e+03
75%	1680.000000	3.000000	120.000000	3.000000	7.311746e+03
max	52272.000000	40.000000	3600.000000	43.000000	1.200000e+07

```
data['location'].value_counts()
```

```
Whitefield                          540
Sarjapur  Road                      399
Electronic City                     302
Kanakpura Road                      273
Thanisandra                         234
                                   ...
1st Stage Radha Krishna Layout        1
BEML Layout 5th stage                 1
singapura paradise                    1
Uvce Layout                           1
Abshot Layout                         1
Name: location, Length: 1306, dtype: int64
```

```
data['location'] = data['location'].apply(lambda x: x.strip())
location_count= data['location'].value_counts()
```

```
location_count
```

```
Whitefield                            541
Sarjapur  Road                        399
Electronic City                       304
Kanakpura Road                        273
Thanisandra                           237
                                     ...
1Channasandra                           1
Hosahalli                               1
Vijayabank bank layout                  1
near Ramanashree California resort      1
Abshot Layout                           1
Name: location, Length: 1295, dtype: int64
```

```
location_count_less_10 = location_count[location_count<=10]
location_count_less_10
```

```
BTM 1st Stage                          10
Nagadevanahalli                        10
Basapura                               10
Sector 1 HSR Layout                    10
Dairy Circle                           10
                                       ..
1Channasandra                           1
Hosahalli                               1
Vijayabank bank layout                  1
near Ramanashree California resort      1
Abshot Layout                           1
Name: location, Length: 1054, dtype: int64
```

```
data['location']=data['location'].apply(lambda x: 'other' if x in location_count_less_10
```

```
data['location'].value_counts()
```

```
other                 2886
Whitefield             541
Sarjapur  Road         399
Electronic City        304
Kanakpura Road         273
                      ...
Nehru Nagar             11
Banjara Layout          11
LB Shastri Nagar        11
Pattandur Agrahara      11
Narayanapura            11
Name: location, Length: 242, dtype: int64
```

```
data.describe()
```

	total_sqft	bath	price	bhk	price_per_sqft
count	13274.000000	13320.000000	13320.000000	13320.000000	1.327400e+04
mean	1559.626694	2.688814	112.565627	2.802778	7.907501e+03
std	1238.405258	1.338754	148.971674	1.294496	1.064296e+05
min	1.000000	1.000000	8.000000	1.000000	2.678298e+02
25%	1100.000000	2.000000	50.000000	2.000000	4.266865e+03
50%	1276.000000	2.000000	72.000000	3.000000	5.434306e+03
75%	1680.000000	3.000000	120.000000	3.000000	7.311746e+03
max	52272.000000	40.000000	3600.000000	43.000000	1.200000e+07

```
(data['total_sqft']/data['bhk']).describe()
```

```
count    13274.000000
mean       575.074878
std        388.205175
min          0.250000
```

```
25%        473.333333
50%        552.500000
75%        625.000000
max      26136.000000
dtype: float64
```

```
data = data[((data['total_sqft']/data['bhk']) >= 300)]
data.describe()
```

	total_sqft	bath	price	bhk	price_per_sqft
count	12530.000000	12530.000000	12530.000000	12530.000000	12530.000000
mean	1594.564544	2.559537	111.382401	2.650838	6303.979357
std	1261.271296	1.077938	152.077329	0.976678	4162.237981
min	300.000000	1.000000	8.440000	1.000000	267.829813
25%	1116.000000	2.000000	49.000000	2.000000	4210.526316
50%	1300.000000	2.000000	70.000000	3.000000	5294.117647
75%	1700.000000	3.000000	115.000000	3.000000	6916.666667
max	52272.000000	16.000000	3600.000000	16.000000	176470.588235

```
data.shape
```

```
(12530, 7)
```

```
data.price_per_sqft.describe()
```

```
count     12530.000000
mean       6303.979357
std        4162.237981
min         267.829813
25%        4210.526316
50%        5294.117647
75%        6916.666667
max      176470.588235
Name: price_per_sqft, dtype: float64
```

```
def remove_outliers_sqft(df):

    df_output = pd.DataFrame()

    for key, subdf in df.groupby('location'):

        m = np.mean(subdf.price_per_sqft)

        st = np.std(subdf.price_per_sqft)

        gen_df = subdf[(subdf.price_per_sqft > (m-st)) & (subdf.price_per_sqft <= (m+st)

        df_output = pd.concat([df_output,gen_df],ignore_index =True)

    return df_output

data = remove_outliers_sqft(data)

data.describe()
```

	total_sqft	bath	price	bhk	price_per_sqft
count	10301.000000	10301.000000	10301.000000	10301.000000	10301.000000
mean	1508.440608	2.471702	91.286372	2.574896	5659.062876
std	880.694214	0.979449	86.342786	0.897649	2265.774749
min	300.000000	1.000000	10.000000	1.000000	1250.000000
25%	1110.000000	2.000000	49.000000	2.000000	4244.897959
50%	1286.000000	2.000000	67.000000	2.000000	5175.600739
75%	1650.000000	3.000000	100.000000	3.000000	6428.571429
max	30400.000000	16.000000	2200.000000	16.000000	24509.803922

```
def bhk_outlier_remover(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 < (st
          return df.drop(exclude_indices, axis='index')
```

```
data=bhk_outlier_remover(data)
```

```
data.shape
```

```
(10300, 7)
```

```
data
```

	location	size	total_sqft	bath	price	bhk	price_per_sqft
0	1st Block Jayanagar	4 BHK	2850.0	4.0	428.00	4	15017.543860
1	1st Block Jayanagar	3 BHK	1630.0	3.0	194.00	3	11901.840491
2	1st Block Jayanagar	3 BHK	1875.0	2.0	235.00	3	12533.333333
3	1st Block Jayanagar	3 BHK	1200.0	2.0	130.00	3	10833.333333
4	1st Block Jayanagar	2 BHK	1235.0	2.0	148.00	2	11983.805668
...	...	...	...	...	...	...	...
10296	other	2 BHK	1353.0	2.0	110.00	2	8130.081301
10297	other	1 Bedroom	812.0	1.0	26.00	1	3201.970443
10298	other	3 BHK	1440.0	2.0	63.93	3	4439.583333

```
data.drop(columns=['size','price_per_sqft'], inplace=True)
```

```
data.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

```
data.to_csv("Cleaned_data.csv")
```

```
X=data.drop(columns=['price'])
y=data['price']
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.compose import make_column_transformer
from sklearn.metrics import r2_score
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.2, random_state=0)
```

```
print(X_train.shape)
print(X_test.shape)
```

```
(8240, 4)
(2060, 4)
```

```
column_trans = make_column_transformer((OneHotEncoder(sparse=False), ['location']),
                                       remainder='passthrough')
```

```
scaler = StandardScaler()
```

```
lr = LinearRegression()
```

```
from sklearn.pipeline import make_pipeline

pipe = make_pipeline(column_trans, scaler, lr)
```

```
pipe.fit(X_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: Futu
  warnings.warn(
```