

```
import numpy as np
import pandas as pd
from google.colab import drive
drive.mount('/content/drive')
%cd /content/drive/MyDrive
data = pd.read_csv("ML Project/Bengaluru_House_Data.csv")
```

➞ Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.r /content/drive/MyDrive

```
data.head()
```

➞

	area_type	availability	location	size	society	total_sqft	bath	bal
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	

```
data.shape
```

➞ (13320, 9)

```
data.info()
```

➞

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 9 columns):
#   Column          Non-Null Count  Dtype
---  -
0   area_type       13320 non-null  object
1   availability     13320 non-null  object
2   location        13319 non-null  object
3   size            13304 non-null  object
4   society         7818 non-null   object
5   total_sqft      13320 non-null  object
6   bath            13247 non-null  float64
7   balcony         12711 non-null  float64
8   price           13320 non-null  float64
dtypes: float64(3), object(6)
memory usage: 936.7+ KB
```

```
for column in data.columns:
    print(data[column].value_counts())
    print(""*20)
```

➞

```
area_type
Super built-up Area    8790
```

```

Built-up Area      2418
Plot Area          2025
Carpet Area        87
Name: count, dtype: int64
*****

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

location
Whitefield          540
Sarjapur Road       399
Electronic City     302
Kanakpura Road      273
Thanisandra         234
...
3rd Stage Raja Rajeshwari Nagar  1
Chuchangatta Colony              1
Electronic City Phase 1,         1
Chikbasavanapura                 1
Abshot Layout                    1
Name: count, Length: 1305, dtype: int64
*****

size
2 BHK      5199
3 BHK      4310
4 Bedroom   826
4 BHK       591
3 Bedroom   547
1 BHK       538
2 Bedroom   329
5 Bedroom   297
6 Bedroom   191
1 Bedroom   105
8 Bedroom    84
7 Bedroom    83
5 BHK        59
9 Bedroom    46
6 BHK        30
7 BHK        17
1 RK         13
10 Bedroom   12
9 BHK         8
8 BHK         5
11 BHK        2
10 BHK        2

```

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



	0
area_type	0
availability	0
location	1
size	16
society	5502
total_sqft	0
bath	73
balcony	609
price	0

dtype: int64

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

```
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
```

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



	count
location	
Whitefield	540
Sarjapur Road	399
Electronic City	302
Kanakpura Road	273
Thanisandra	234
...	...
3rd Stage Raja Rajeshwari Nagar	1
Chuchangatta Colony	1
Electronic City Phase 1,	1
Chikbasavanapura	1
Abshot Layout	1

1305 rows × 1 columns

dtype: int64

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

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

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

```
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        13320 non-null  object  
1   size            13320 non-null  object  
2   total_sqft      13320 non-null  object  
3   bath            13320 non-null  float64 
4   price           13320 non-null  float64 
dtypes: float64(2), object(3)
memory usage: 520.4+ KB
```

```
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
```

```
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
	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 sq ft
```

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

```
data['price_per_sqft']
```



	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

13320 rows × 1 columns

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()
```



	count
location	
Whitefield	540
Sarjapur Road	399
Electronic City	302
Kanakpura Road	273
Thanisandra	234
...	...
Mango Garden Layout	1
Milk Colony	1
Basnashankari,6th stage,	1
Near ullas theater	1
N R Layout	1

1306 rows × 1 columns

dtype: int64

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

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



	count
location	
1st Block Koramangala	10
Dairy Circle	10
Nagadevanahalli	10
Sadashiva Nagar	10
Naganathapura	10
...	...
Xavier Layout	1
Ramanagara Channapatna	1
Maheswari Nagar	1
Hsr layout sector3	1
Thyagraj Nagar	1

1054 rows × 1 columns

dtype: int64

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




	count
location	
other	2886
Whitefield	541
Sarjapur Road	399
Electronic City	304
Kanakpura Road	273
...	...
Tindlu	11
Marsur	11
2nd Phase Judicial Layout	11
Thyagaraja Nagar	11
HAL 2nd Stage	11

242 rows × 1 columns

dtype: int64

#Outlier detection and removal

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()



0

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()
```



	price_per_sqft
count	12530.000000
mean	6303.979357
std	4162.237981
min	267.829813
25%	4210.526316
50%	5294.117647
75%	6916.666667
max	176470.588235

dtype: float64

```
import numpy as np
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 < (stats['me
return df.drop(exclude_indices, axis='index')

```

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

```
data.shape
```

➡ (7361, 7)

```
data
```

		location	size	total_sqft	bath	price	bhk	price_per_sqft
0		1st Block Jayanagar	4 BHK	2850.0	4.0	428.0	4	15017.543860
1		1st Block Jayanagar	3 BHK	1630.0	3.0	194.0	3	11901.840491
2		1st Block Jayanagar	3 BHK	1875.0	2.0	235.0	3	12533.333333
3		1st Block Jayanagar	3 BHK	1200.0	2.0	130.0	3	10833.333333
4		1st Block Jayanagar	2 BHK	1235.0	2.0	148.0	2	11983.805668
...	
10292		other	2 BHK	1200.0	2.0	70.0	2	5833.333333
10293		other	1 BHK	1800.0	1.0	200.0	1	11111.111111
10296		other	2 BHK	1353.0	2.0	110.0	2	8130.081301
10297		other	1 Bedroom	812.0	1.0	26.0	1	3201.970443
10300		other	4 BHK	3600.0	5.0	400.0	4	11111.111111

7361 rows × 7 columns

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

```
#Cleaned Data
```

```
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,Lasso,Ridge
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
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)
```



```
(5888, 4)
(1473, 4)
```

```
#Applying Linear Regression
```

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

```
scaler=StandardScaler()
```

```
lr = LinearRegression()
```

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

```
y_pred_lr=pipe.predict(X_test)
```

```
r2_score(y_test,y_pred_lr)
```

```
0.8233783132019383
```

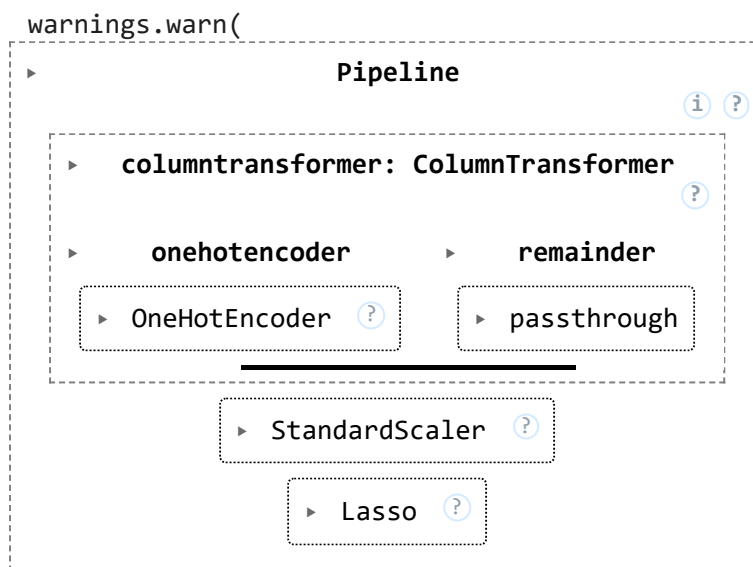
```
#Applying Lasso
```

```
lasso=Lasso()
```

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

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

```
/usr/local/lib/python3.11/dist-packages/sklearn/compose/_column_transformer.py:1667:
The format of the columns of the 'remainder' transformer in ColumnTransformer.transfo
At the moment the remainder columns are stored as indices (of type int). With the sam
To use the new behavior now and suppress this warning, use ColumnTransformer(force_in
```



```
y_pred_lasso=pipe.predict(X_test)
```

```
r2_score(y_test,y_pred_lasso)
```

```
0.8128285650772719
```

```
#Applying Ridge
```

```
ridge=Ridge()
```

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

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