

```
In [5]: import numpy as np
import pandas as pd
```

```
In [84]: df=pd.read_csv("Bengaluru_House_Data.csv")
```

```
In [85]: df
```

Out[85]:

	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	SollyCI	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 [86]: df.head()
```

Out[86]:

	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

```
In [87]: df.shape
```

Out[87]: (13320, 9)

```
In [88]: df.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
```

```
In [89]: df.isnull().sum()
```

```
Out[89]: area_type      0
availability  0
location      1
size          16
society       5502
total_sqft    0
bath          73
balcony       609
price         0
dtype: int64
```

```
In [90]: df.drop(columns=["area_type", "availability", "society", "balcony"], inplace=True)
```

```
In [91]: df
```

```
Out[91]:
```

	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 [92]: df.describe()
```

Out[92]:

	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

In [93]: `df.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
```

In [94]: `df["location"].value_counts()`

Out[94]:

```
Whitefield          540
Sarjapur Road      399
Electronic City     302
Kanakpura Road     273
Thanisandra        234
...
Bapuji Layout       1
1st Stage Radha Krishna Layout  1
BEML Layout 5th stage  1
singapura paradise  1
Abshot Layout       1
Name: location, Length: 1305, dtype: int64
```

In [95]: `df["location"]=df["location"].fillna("Sarjapur Road")`

In []:

In [96]: `df["size"]=df["size"].fillna("2 BHK")`

In [97]: `df["bath"]=df["bath"].fillna(df["bath"].median())`

In [98]: `df.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
```

```
In [99]: df["bhk"]=df["size"].str.split().str.get(0).astype(int)
```

```
In [102]: df[df["bhk"] > 20]
```

```
Out[102]:
```

	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

```
In [103]: df["total_sqft"].unique()
```

```
Out[103]: array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

```
In [104]: 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
```

```
In [105]: df["total_sqft"]=df["total_sqft"].apply(convertRange)
```

```
In [106]: df.head()
```

```
Out[106]:
```

	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

```
In [107]: df["price_per_sqft"]=df["price"]*100000/df["total_sqft"]
```

```
In [108]: df["price_per_sqft"]
```

```
Out[108]: 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
```

```
In [109... df.describe()
```

```
Out[109]:
```

	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

```
In [110... df["location"] = df["location"].apply(lambda x: x.strip())
location_count=df["location"].value_counts()
```

```
In [111... location_count_less_10=location_count[location_count<=10]
```

```
In [112... location_count_less_10
```

```
Out[112]: Dairy Circle      10
          Nagappa Reddy Layout  10
          Basapura      10
          1st Block Koramangala  10
          Sector 1 HSR Layout  10
          ..
          Bapuji Layout      1
          1st Stage Radha Krishna Layout  1
          BEML Layout 5th stage  1
          singapura paradise  1
          Abshot Layout      1
          Name: location, Length: 1053, dtype: int64
```

```
In [113... df["location"]=df["location"].apply(lambda x: "other" if x in location_count_less_10 else
```

```
In [114... df["location"].value_counts()
```

```

Out[114]: other                2885
          Whitefield           541
          Sarjapur Road        400
          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

```

outlier detection

```
In [115... df.describe()
```

```

Out[115]:
```

	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

```
In [117... (df["total_sqft"]/df["bhk"]).describe()
```

```

Out[117]: 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

```

```
In [119... df=df[df["total_sqft"]/df["bhk"]>=300]
```

```
In [120... df.describe()
```

Out[120]:

	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

In [121...

df.shape

Out[121]: (12530, 7)

In [122...

df.price_per_sqft.describe()

Out[122]:

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

In [123...

```
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
df=remove_outliers_sqft(df)
df.describe()
```

Out[123]:

	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

In [128...

```
def bhk_outlier_remove(df):
    exclude_indices = np.array([])
```

Loading [MathJax]/extensions/Safe.js

```

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
return df.drop(exclude_indices,axis=0)

```

```
In [129... df=bhk_outlier_remove(df)
```

```
In [131... df.shape
```

```
Out[131]: (7360, 7)
```

```
In [ ]: df.drop(columns=["size", "price_per_sqft"], inplace=True)
```

```
In [136... df
```

```
Out[136]:
```

	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
...
10292	other	1200.0	2.0	70.0	2
10293	other	1800.0	1.0	200.0	1
10296	other	1353.0	2.0	110.0	2
10297	other	812.0	1.0	26.0	1
10300	other	3600.0	5.0	400.0	4

7360 rows × 5 columns

```
In [137... df.to_csv("cleaned_data.csv")
```

```
In [138... x=df.drop(columns=["price"])
y=df["price"]
```

```
In [140... x.head(2)
```

```
Out[140]:
```

	location	total_sqft	bath	bhk
0	1st Block Jayanagar	2850.0	4.0	4
1	1st Block Jayanagar	1630.0	3.0	3

```
In [141... y.head(2)
```



```
Out[141]: 0      428.0
          1      194.0
          Name: price, dtype: float64
```

```
In [151... 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
```

```
In [152... X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
In [153... X_train.shape
```

```
Out[153]: (5888, 4)
```

```
In [154... X_test.shape
```

```
Out[154]: (1472, 4)
```

```
In [155... # applying linear regression
```

```
In [160... column_trans=make_column_transformer((OneHotEncoder(sparse=False), ["location"]), remainder=
```

```
In [163... scaler=StandardScaler())
```

```
In [164... lr=LinearRegression()
```

```
In [165... pipe=make_pipeline(column_trans, scaler, lr)
```

```
In [166... pipe.fit(X_train, y_train)
```

```
Out[166]: Pipeline(steps=[('columntransformer',
                           ColumnTransformer(remainder='passthrough',
                                                transformers=[('onehotencoder',
                                                                OneHotEncoder(sparse=False),
                                                                ['location'])])),
                           ('standardscaler', StandardScaler()),
                           ('linearregression', LinearRegression())])
```

```
In [167... y_pred_lr=pipe.predict(X_test)
```

```
In [168... r2_score(y_test, y_pred_lr)
```

```
Out[168]: 0.8296165353105762
```

```
In [169... # applying lasso regression
```

```
In [170... lasso= Lasso()
```

```
In [171... pipe=make_pipeline(column_trans, scaler, lasso)
```

```
In [172... pipe.fit(X_train, y_train)
```

```
Out[172]: Pipeline(steps=[('columntransformer',
                             ColumnTransformer(remainder='passthrough',
                                                  transformers=[('onehotencoder',
                                                                OneHotEncoder(sparse=False),
                                                                ['location'])])),
                             ('standardscaler', StandardScaler()), ('lasso', Lasso())])
```

```
In [175... y_pred_lasso=pipe.predict(X_test)
```

```
In [176... r2_score(y_test,y_pred_lasso)
```

```
Out[176]: 0.8199181874762704
```

```
In [177... # applying ridge regression
```

```
In [178... ridge=Ridge()
```

```
In [179... pipe=make_pipeline(column_trans, scaler, ridge)
```

```
In [180... pipe.fit(X_train,y_train)
```

```
Out[180]: Pipeline(steps=[('columntransformer',
                             ColumnTransformer(remainder='passthrough',
                                                  transformers=[('onehotencoder',
                                                                OneHotEncoder(sparse=False),
                                                                ['location'])])),
                             ('standardscaler', StandardScaler()), ('ridge', Ridge())])
```

```
In [181... y_pred_ridge=pipe.predict(X_test)
```

```
In [182... r2_score(y_test,y_pred_ridge)
```

```
Out[182]: 0.8296651410179635
```

```
In [183... print("no_regularization", r2_score(y_test,y_pred_lr))
print("Lasso", r2_score(y_test,y_pred_lasso))
print("Ridge", r2_score(y_test,y_pred_ridge))

no_regularization 0.8296165353105762
Lasso 0.8199181874762704
Ridge 0.8296651410179635
```

```
In [184... import pickle
```

```
In [186... pickle.dump(pipe, open("RidgeModel.pkl", "wb"))
```

```
In [ ]:
```