```
import pandas as pd
import numpy as np
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files Banglore H…g Prices.csv

- **Banglore Housing Prices.csv**(text/csv) - 432109 bytes, last modified: 3/21/2023 - 100% done

```
Saving Banglore Housing Prices.csv to Banglore Housing Prices.csv
```

```
import io
```

```
home=pd.read_csv(io.BytesIO(uploaded['Banglore Housing Prices.csv']))
```

```
backup=home.copy()
# house=backup.copy()
home.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
```

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

```
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
```

Filling the missing values in the location,size and bath columns

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

```
home['size'].value_counts()
home['size']=home['size'].fillna("2 BHK")
```

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

```
home.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
```

Converting the values in size column to numerical values

```
home['bhk']=home['size'].str.split(" ").str.get(0)
```

```
home['bhk']=home['bhk'].astype(int)
```

Converting the range values of total_sqft column

```
home['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
```

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

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

```
home.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

```
home['location'].value_counts()
home.shape
```

```
    (13320, 7)
```

Replacing the location of count less than 10 with the value 'other'

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

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

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

```
    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
```

```
home.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

```
home.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   location        13320 non-null  object 
 1   size            13320 non-null  object 
 2   total_sqft      13274 non-null  float64
 3   bath            13320 non-null  float64
 4   price           13320 non-null  float64
 5   bhk             13320 non-null  int64  
 6   price_per_sqft  13274 non-null  float64
dtypes: float64(4), int64(1), object(2)
memory usage: 728.6+ KB
```

Removing the sqft less than 300

```
(home['total_sqft']/home['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
```

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

```
home['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
```

Removing the OUTLIERS

```
def remove_outliers_price_per_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
home=remove_outliers_price_per_sqft(home)
```

```
def outlier_bhk(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.values)

    return df.drop(exclude_indices,axis='index')
```

```
home=outlier_bhk(home)
home.shape
```

```
(7360, 7)
```

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

```
home.reset_index(drop=True)
```

	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
...	...	...	...	...	...
7355	other	1200.0	2.0	70.0	2
7356	other	1800.0	1.0	200.0	1
7357	other	1353.0	2.0	110.0	2
7358	other	812.0	1.0	26.0	1
7359	other	3600.0	5.0	400.0	4

7360 rows × 5 columns

```
home.shape
```

```
(7360, 5)
```

Data Cleaning Completed

Now Making the Linear regression model

```
home.to_csv("Cleaned_Home.csv")
```

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,accuracy_score
from sklearn.metrics import r2_score
from sklearn.preprocessing import OneHotEncoder,StandardScaler
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
```

```
x=home.drop(columns=['price'])
y=home['price']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

```
ohe=OneHotEncoder()
ohe.fit(x[['location']])
```

```
▾ OneHotEncoder
OneHotEncoder()
```

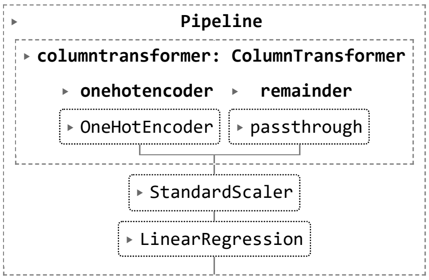
```
column_transformer=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['location']),remainder='passthrough')
```

```
scaler=StandardScaler(with_mean=False)
```

```
lr=LinearRegression()
```

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

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



▸ Pipeline

▸ columntransformer: ColumnTransformer

▸ onehotencoder ▸ remainder

▸ OneHotEncoder ▸ passthrough

▸ StandardScaler

▸ LinearRegression

```
y_pred=pipe.predict(x_test)
```

```
r2_score(y_test,y_pred)
```

```
0.8296447778761105
```

```
mean_squared_error(y_test,y_pred,squared=False)
```

```
37.3730840921021
```

```
scores=[]
scores1=[]
for i in range (1000):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=i)
    lr=LinearRegression()
    pipe=make_pipeline(column_transformer,lr)
    pipe.fit(x_train,y_train)
    y_pred=pipe.predict(x_test)
    scores1.append(mean_squared_error(y_test,y_pred,squared=False))
    scores.append(r2_score(y_test,y_pred))
```

```
scores[np.argmax(scores)]
```

```
0.9056206177221361
```

```
scores1[np.argmin(scores1)]
```

```
26.465182372679852
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=np.argmin(scores1))
lr = LinearRegression()
pipe = make_pipeline(column_transformer, lr)
pipe.fit(x_train, y_train)
y_pred = pipe.predict(x_test)
mean_squared_error(y_test, y_pred, squared=False)
```

```
26.465182372679852
```

PREDICTIONS

```
pipe.predict(pd.DataFrame(columns=['location','total_sqft','bath','bhk'],data=np.array(['Kengeri',3000,4,4]).reshape(1,4)))
```

```
array([211.96129363])
```

+ Code + Text

✓ 0s completed at 8:41 PM ● ✕