In [1]:
```
# import all libraries and filters

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

LOADING THE DATA :

In [3]:
```
# Reading the file :

housing_df = pd.read_csv('E:\Housing real estate DELHI.csv')
housing_df.head()
```

Out[3]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterhea
0	13300000	7420	4	2	3	yes	no	no	
1	12250000	8960	4	4	4	yes	no	no	
2	12250000	9960	3	2	2	yes	no	yes	
3	12215000	7500	4	2	2	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	

Shape of the dataframe:

In [4]:
```
housing_df.shape
```

Out[4]:
```
(545, 13)
```

info of dataframe:

In [5]:
```
housing_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   price             545 non-null    int64 
 1   area              545 non-null    int64 
 2   bedrooms          545 non-null    int64 
 3   bathrooms         545 non-null    int64 
 4   stories           545 non-null    int64 
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64 
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

describe the dataframe:

In [6]:

```
housing_df.describe (percentiles = [0.10,0.25, 0.50, 0.75, 0.90, 0.99])
```

Out[6]:

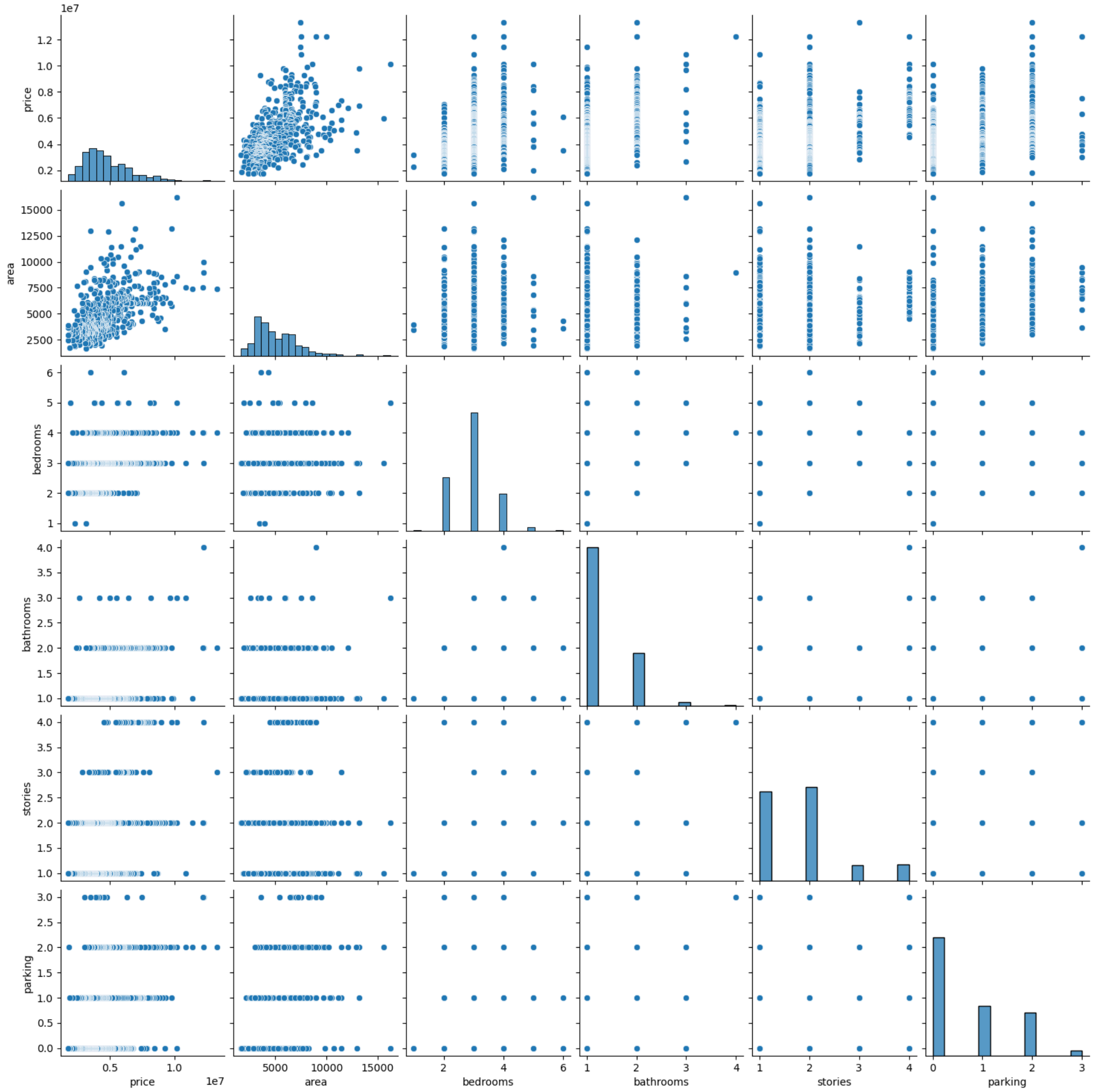
	price	area	bedrooms	bathrooms	stories	parking
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.000000
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.693578
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.861586
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.000000
10%	2.835000e+06	3000.000000	2.000000	1.000000	1.000000	0.000000
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.000000
90%	7.350000e+06	7980.000000	4.000000	2.000000	3.000000	2.000000
99%	1.054200e+07	12543.600000	5.000000	3.000000	4.000000	3.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

Visualizing the data :

In [7]:

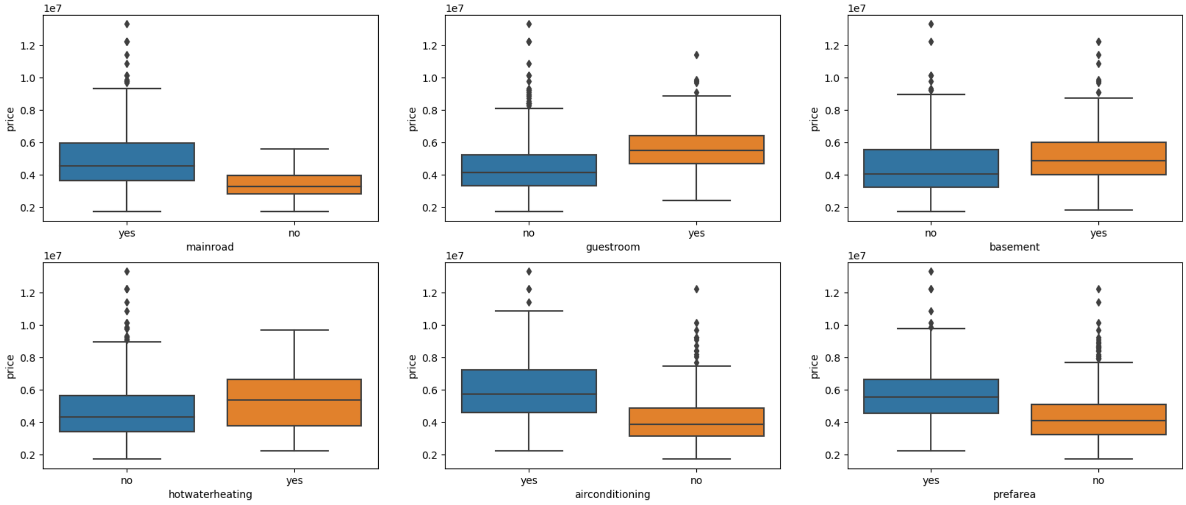
```
# a pair plot of numerical variable :

sns.pairplot(data = housing_df)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x275e71987c0>



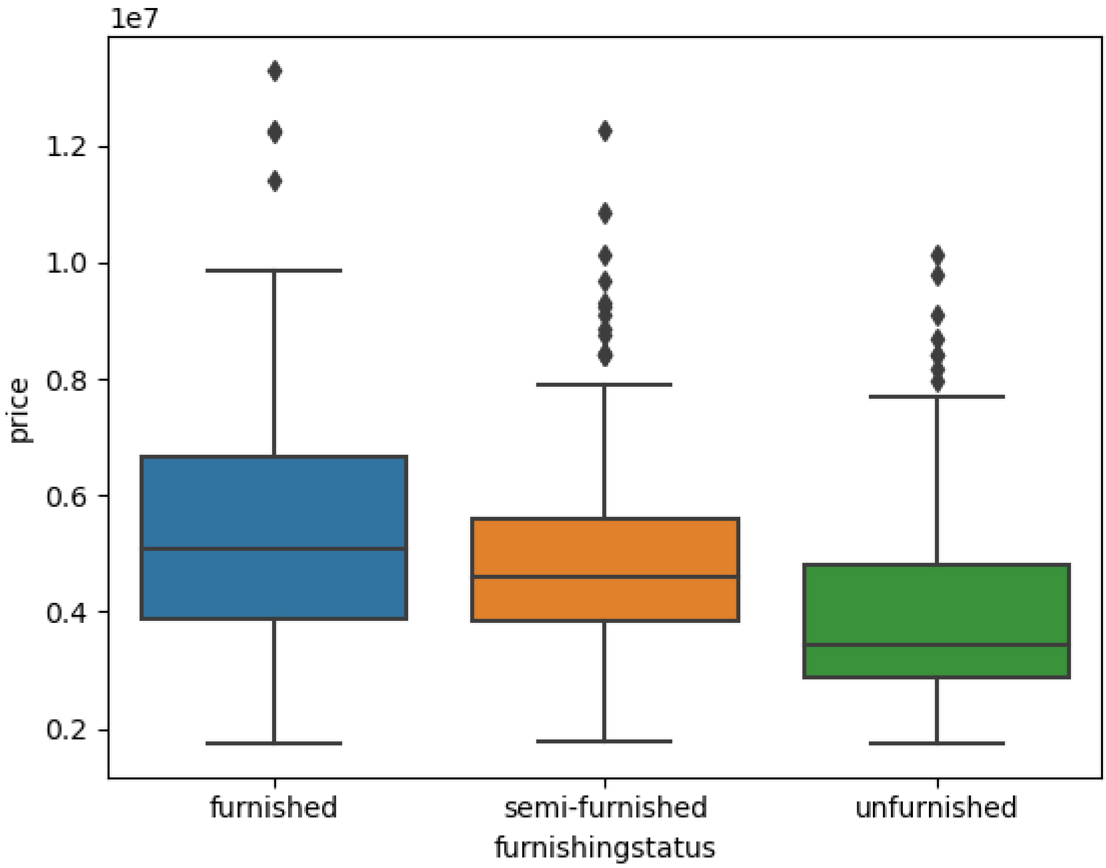
In [8]:
```
# Categorical varibale:

plt.figure(figsize = (20,8))

plt.subplot(2,3,1)
sns.boxplot(x='mainroad', y = 'price' , data = housing_df)

plt.subplot(2,3,2)
sns.boxplot(x = 'guestroom' , y = 'price' , data = housing_df)

plt.subplot(2,3,3)
sns.boxplot( x= 'basement' , y = 'price' , data = housing_df)

plt.subplot(2,3,4)
sns.boxplot(x= 'hotwaterheating', y = 'price' , data = housing_df)

plt.subplot(2,3,5)
sns.boxplot(x= 'airconditioning' , y = 'price', data = housing_df)

plt.subplot(2,3,6)
sns.boxplot( x= 'prefarea' , y = 'price' , data = housing_df)
```

Out[8]:
```
<AxesSubplot:xlabel='prefarea', ylabel='price'>
```



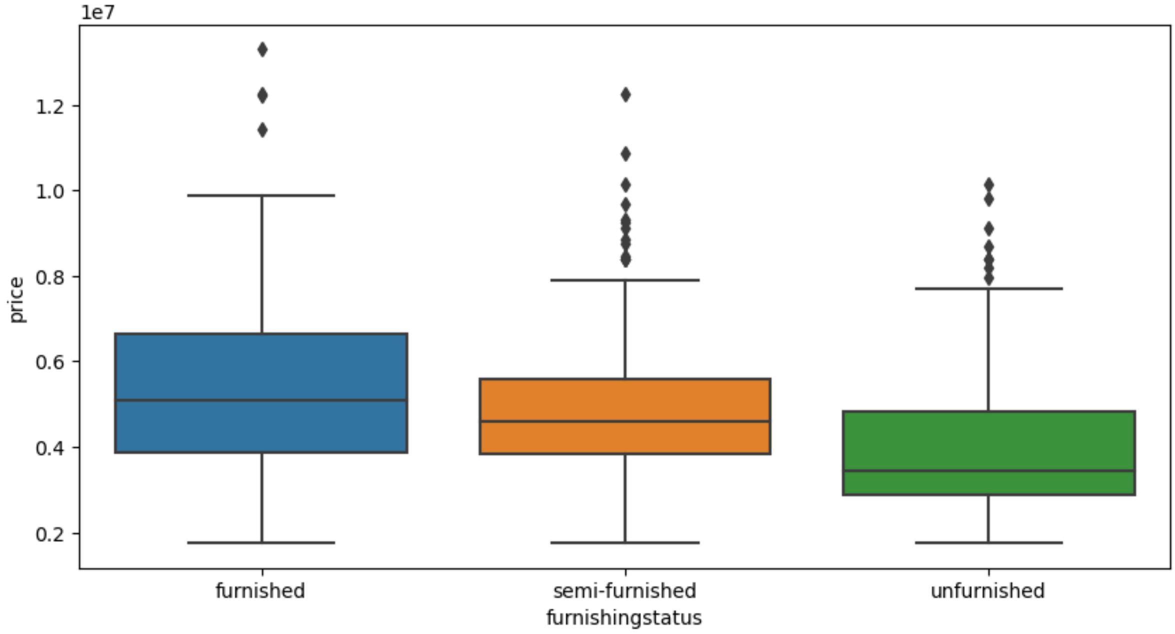
In [9]:
```
sns.boxplot(x = 'furnishingstatus', y = 'price' , data = housing_df)
```

Out[9]:
```
<AxesSubplot:xlabel='furnishingstatus', ylabel='price'>
```



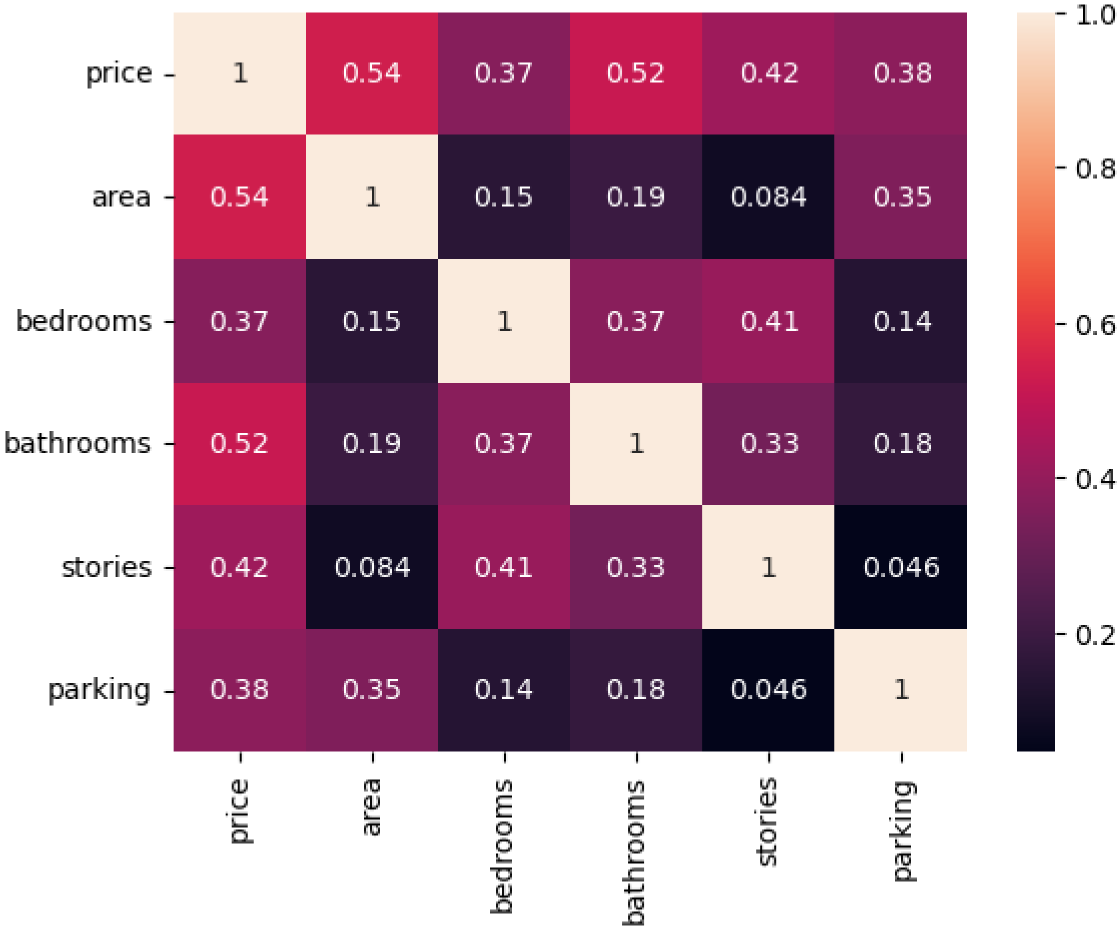
In [10]:
```
plt.figure(figsize=(10,5))
sns.boxplot(x = 'furnishingstatus', y = 'price', data = housing_df)
```

Out[10]:
```
<AxesSubplot:xlabel='furnishingstatus', ylabel='price'>
```



In [11]:
```
sns.heatmap(housing_df.corr(), annot= True)
```

Out[11]: <AxesSubplot:>



here we see high correlation between price , area and bathrooms.

Data Prepration

In [12]:
```
# converting yes to 1 and No to
```

```
variable_list =['mainroad','guestroom', 'basement', 'hotwaterheating', 'airconditi
def binary_map(x) :
    return x.map({'yes' :1 , 'no' : 0})

housing_df[variable_list] = housing_df[variable_list].apply(binary_map)
```

In [13]:
```
housing_df.head()
```

Out[13]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterhea
0	13300000	7420	4	2	3	1	0	0	
1	12250000	8960	4	4	4	1	0	0	
2	12250000	9960	3	2	2	1	0	1	
3	12215000	7500	4	2	2	1	0	1	
4	11410000	7420	4	1	2	1	1	1	

Dummy variables:

In [14]:
```
# Lets us create a dummy variable for furnishing status as 3 level values

status = pd.get_dummies(housing_df['furnishingstatus'], drop_first = True)
status.head()
```

Out[14]:

	semi-furnished	unfurnished
0	0	0
1	0	0
2	1	0
3	0	0
4	0	0

In [15]:
```
housing_df = pd.concat([housing_df, status], axis = 1)
```

In [16]:
```
housing_df.head()
```

Out[16]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterhea
0	13300000	7420	4	2	3	1	0	0	
1	12250000	8960	4	4	4	1	0	0	
2	12250000	9960	3	2	2	1	0	1	
3	12215000	7500	4	2	2	1	0	1	
4	11410000	7420	4	1	2	1	1	1	

In [17]:
```
housing_df.drop(['furnishingstatus'], axis = 1, inplace = True)
```

In [18]:
```
housing_df.head()
```

Out[18]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterhea
0	13300000	7420	4	2	3	1	0	0	
1	12250000	8960	4	4	4	1	0	0	
2	12250000	9960	3	2	2	1	0	1	
3	12215000	7500	4	2	2	1	0	1	
4	11410000	7420	4	1	2	1	1	1	

Splitting the data into Test Train Split

In [19]:

```
df_train, df_test = train_test_split(housing_df, train_size =0.7, test_size =0.3,
```

In [20]:

```
df_train.shape
```

Out[20]:

```
(381, 14)
```

Rescaling the features:

In [21]:

```
scaler = MinMaxScaler()

# applying the scaler only to below variable
num_var = ['price', 'area','bedrooms','bathrooms','stories','parking']

df_train[num_var] = scaler.fit_transform(df_train[num_var])
```

In [22]:

```
df_train.head()
```

Out[22]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotw
359	0.169697	0.155227	0.4	0.0	0.000000	1	0	0	
19	0.615152	0.403379	0.4	0.5	0.333333	1	0	0	
159	0.321212	0.115628	0.4	0.5	0.000000	1	1	1	
35	0.548133	0.454417	0.4	0.5	1.000000	1	0	0	
28	0.575758	0.538015	0.8	0.5	0.333333	1	0	1	

In [23]:

```
df_train.describe()
```

Out[23]:

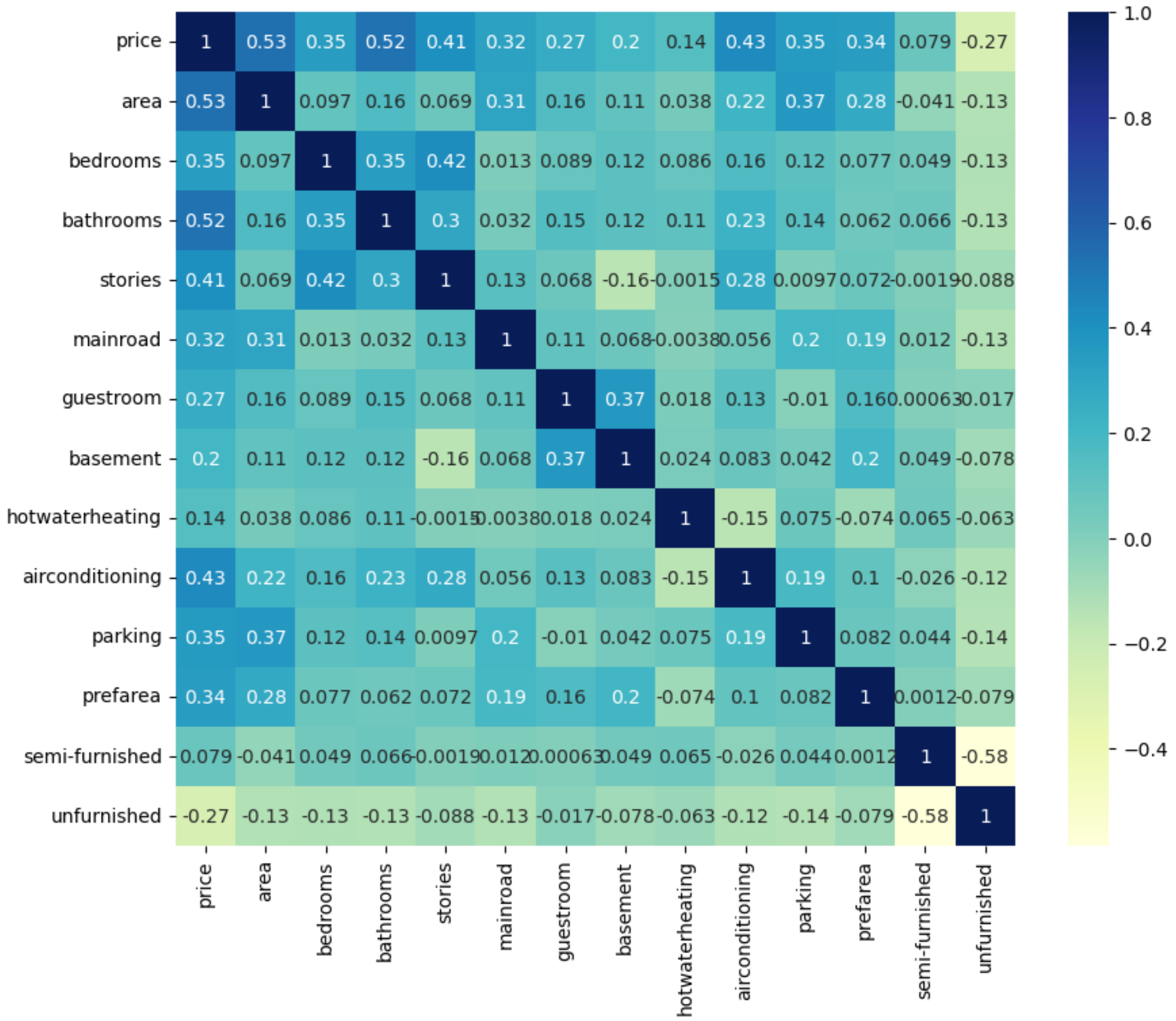
	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	base
count	381.000000	381.000000	381.000000	381.000000	381.000000	381.000000	381.000000	381.00
mean	0.260333	0.288710	0.386352	0.136483	0.268591	0.855643	0.170604	0.35
std	0.157607	0.181420	0.147336	0.237325	0.295001	0.351913	0.376657	0.47
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.151515	0.155227	0.200000	0.000000	0.000000	1.000000	0.000000	0.00
50%	0.221212	0.234424	0.400000	0.000000	0.333333	1.000000	0.000000	0.00
75%	0.345455	0.398099	0.400000	0.500000	0.333333	1.000000	0.000000	1.00
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00

All the values are in the range o and 1.

In [24]:

```
# Lets us check the correlation of train data :

plt.figure(figsize = (10,8))
sns.heatmap(df_train.corr(), annot = True, cmap= 'YlGnBu')
```

Out[24]: <AxesSubplot:>



we see high correlation between price and area , price and bathrooms, bedrooms and stories and many more.

Dividing X and Y for model building:

In [25]:
```
y_train = df_train.pop('price')
x_train = df_train
```

Building a linear model :

We will be using two methods : 1 . using statsmodels.api 2. using RFE

Method 1 : Using statsmodels.api

In [26]:
```
import statsmodels.api as sm
```

In [27]:
```
# area

x_train_sm = sm.add_constant(x_train[['area']])

lr_1 = sm.OLS(y_train, x_train_sm).fit()
```

In [28]:
```
lr_1.params
```

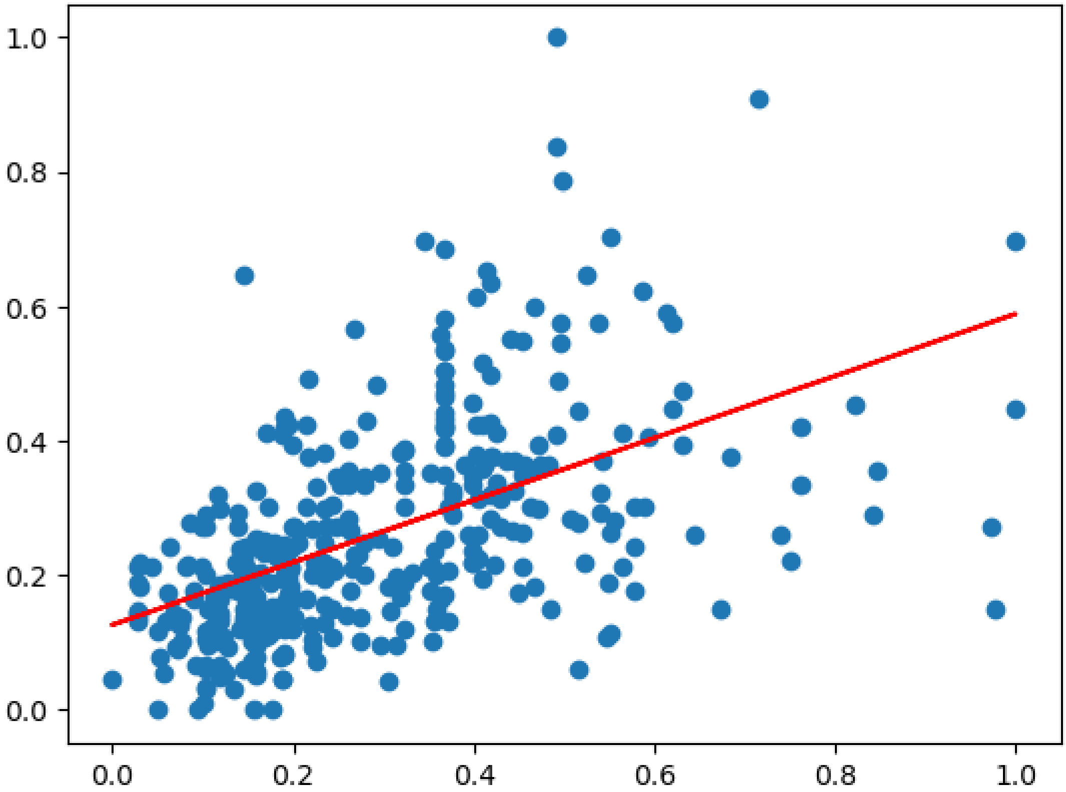
Out[28]:
```
const    0.126894
area     0.462192
dtype: float64
```

In [29]:
```
print(lr_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.283
Model:                            OLS   Adj. R-squared:                  0.281
Method:                 Least Squares   F-statistic:                     149.6
Date:                Tue, 04 Jul 2023   Prob (F-statistic):           3.15e-29
Time:                        12:47:42   Log-Likelihood:                 227.23
No. Observations:                 381   AIC:                            -450.5
Df Residuals:                     379   BIC:                            -442.6
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1269      0.013      9.853      0.000       0.102       0.152
area           0.4622      0.038     12.232      0.000       0.388       0.536
==============================================================================
Omnibus:                       67.313   Durbin-Watson:                   2.018
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              143.063
Skew:                           0.925   Prob(JB):                     8.59e-32
Kurtosis:                       5.365   Cond. No.                         5.99
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

Variable Area just explains 28% variance

In [30]:
```
plt.scatter(x_train_sm.iloc[:,1], y_train)
plt.plot(x_train_sm.iloc[:,1], 0.126894 + 0.462192*x_train_sm.iloc[:,1], 'r')
plt.show()
```



Through the line is passing through the data , we see that area could explain only 28% variance. so let us add another variable

In [31]:
```
# Area and bathjrooms

x_train_sm  = sm.add_constant(x_train[['area', 'bathrooms']])

lr_2 = sm.OLS(y_train, x_train_sm).fit()
```

In [32]:
```
lr_2.params
```

Out[32]:
```
const        0.104589
area         0.398396
bathrooms    0.298374
dtype: float64
```

In [33]:
```
print(lr_2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.480
Model:                            OLS   Adj. R-squared:                  0.477
Method:                 Least Squares   F-statistic:                     174.1
Date:                Tue, 04 Jul 2023   Prob (F-statistic):           2.51e-54
Time:                        12:47:43   Log-Likelihood:                 288.24
No. Observations:                 381   AIC:                            -570.5
Df Residuals:                     378   BIC:                            -558.6
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1046      0.011      9.384      0.000       0.083       0.127
area           0.3984      0.033     12.192      0.000       0.334       0.463
bathrooms      0.2984      0.025     11.945      0.000       0.249       0.347
==============================================================================
Omnibus:                       62.839   Durbin-Watson:                   2.157
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              168.790
Skew:                           0.784   Prob(JB):                     2.23e-37
Kurtosis:                       5.859   Cond. No.                         6.17
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

Adjusted R-squared increased from 28.1% to 47.7% . let us add one more variable and check:

In [34]:
```
# Area , bathrooms, and bedrooms

x_trains_sm = sm.add_constant(x_train[['area', 'bedrooms', 'bathrooms']])

lr_3 = sm.OLS(y_train, x_train_sm).fit()
```

In [35]:
```
lr_3.params
```

Out[35]:
```
const        0.104589
area         0.398396
bathrooms    0.298374
dtype: float64
```

In [36]:
```
print(lr_3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.480
Model:                            OLS   Adj. R-squared:                  0.477
Method:                 Least Squares   F-statistic:                     174.1
Date:                Tue, 04 Jul 2023   Prob (F-statistic):           2.51e-54
Time:                        12:47:43   Log-Likelihood:                 288.24
No. Observations:                 381   AIC:                            -570.5
Df Residuals:                     378   BIC:                            -558.6
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1046      0.011      9.384      0.000       0.083       0.127
area           0.3984      0.033     12.192      0.000       0.334       0.463
bathrooms      0.2984      0.025     11.945      0.000       0.249       0.347
==============================================================================
Omnibus:                       62.839   Durbin-Watson:                   2.157
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              168.790
Skew:                           0.784   Prob(JB):                     2.23e-37
Kurtosis:                       5.859   Cond. No.                         6.17
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

Lets us do the other way - Let us build the model by adding all the variables to the model and drop those which are insignificant:

In [37]:
```
x_train.columns
```

Out[37]:
```
Index(['area', 'bedrooms', 'bathrooms', 'stories', 'mainroad', 'guestroom',
       'basement', 'hotwaterheating', 'airconditioning', 'parking', 'prefarea',
       'semi-furnished', 'unfurnished'],
      dtype='object')
```

In [38]:
```
x_train_sm = sm.add_constant(x_train)
lr_4 = sm.OLS(y_train, x_train_sm).fit()
```

In [39]:
```
lr_4.params
```

Out[39]:
```
const              0.020033
area               0.234664
bedrooms           0.046735
bathrooms          0.190823
stories            0.108516
mainroad           0.050441
guestroom          0.030428
basement           0.021595
hotwaterheating    0.084863
airconditioning    0.066881
parking            0.060735
prefarea           0.059428
semi-furnished     0.000921
unfurnished       -0.031006
dtype: float64
```

In [40]:
```
print(lr_4.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.681
Model:                            OLS   Adj. R-squared:                  0.670
Method:                 Least Squares   F-statistic:                     60.40
Date:                Tue, 04 Jul 2023   Prob (F-statistic):           8.83e-83
Time:                        12:47:43   Log-Likelihood:                 381.79
No. Observations:                 381   AIC:                            -735.6
Df Residuals:                     367   BIC:                            -680.4
Df Model:                          13
Covariance Type:            nonrobust
================================================================================
=
                      coef    std err          t      P>|t|      [0.025      0.97
5]
--------------------------------------------------------------------------------
-
const               0.0200      0.021      0.955      0.340      -0.021       0.06
1
area                0.2347      0.030      7.795      0.000       0.175       0.29
4
bedrooms            0.0467      0.037      1.267      0.206      -0.026       0.11
9
bathrooms           0.1908      0.022      8.679      0.000       0.148       0.23
4
stories             0.1085      0.019      5.661      0.000       0.071       0.14
6
mainroad            0.0504      0.014      3.520      0.000       0.022       0.07
9
guestroom           0.0304      0.014      2.233      0.026       0.004       0.05
7
basement            0.0216      0.011      1.943      0.053      -0.000       0.04
3
hotwaterheating     0.0849      0.022      3.934      0.000       0.042       0.12
7
airconditioning     0.0669      0.011      5.899      0.000       0.045       0.08
9
parking             0.0607      0.018      3.365      0.001       0.025       0.09
6
prefarea            0.0594      0.012      5.040      0.000       0.036       0.08
3
semi-furnished      0.0009      0.012      0.078      0.938      -0.022       0.02
4
unfurnished        -0.0310      0.013     -2.440      0.015      -0.056      -0.00
6
==============================================================================
Omnibus:                       93.687   Durbin-Watson:                   2.093
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              304.917
Skew:                           1.091   Prob(JB):                     6.14e-67
Kurtosis:                       6.801   Cond. No.                         14.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

We see that , certain variables have p-values > 0.05. Before dropping any variables, lets us check VIF as well :

VIF:

In [41]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [42]:
```
vif = pd.DataFrame()
vif["Features"] = x_train.columns
vif["VIF"] = [variance_inflation_factor(x_train.values, i) for i in range (x_train
vif["VIF"] = round(vif["VIF"], 2)
vif = vif.sort_values(by = 'VIF', ascending = False)
vif
```

Out[42]:

	Features	VIF
1	bedrooms	7.33
4	mainroad	6.02
0	area	4.67
3	stories	2.70
11	semi-furnished	2.19
9	parking	2.12
6	basement	2.02
12	unfurnished	1.82
8	airconditioning	1.77
2	bathrooms	1.67
10	prefarea	1.51
5	guestroom	1.47
7	hotwaterheating	1.14

Lets us drop variable semi-furnished as p-value of semi-furnished is 0.938

In [43]:
```
# Dropping the variable semi- furnished

x = x_train.drop('semi-furnished', axis = 1)
```

In [44]:
```
x.columns
```

Out[44]:
```
Index(['area', 'bedrooms', 'bathrooms', 'stories', 'mainroad', 'guestroom',
       'basement', 'hotwaterheating', 'airconditioning', 'parking', 'prefarea',
       'unfurnished'],
      dtype='object')
```

In [45]:
```
x_sm = sm.add_constant(x)
lr_5 = sm.OLS(y_train, x_sm).fit()
```

In [46]:
```
print(lr_5.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.681
Model:                            OLS   Adj. R-squared:                  0.671
Method:                 Least Squares   F-statistic:                     65.61
Date:                Tue, 04 Jul 2023   Prob (F-statistic):           1.07e-83
Time:                        12:47:43   Log-Likelihood:                 381.79
No. Observations:                 381   AIC:                            -737.6
Df Residuals:                     368   BIC:                            -686.3
Df Model:                          12
Covariance Type:            nonrobust
==================================================================================
=
                      coef    std err          t      P>|t|      [0.025      0.97
5]
----------------------------------------------------------------------------------
-
const               0.0207      0.019      1.098      0.273      -0.016       0.05
8
area                0.2344      0.030      7.845      0.000       0.176       0.29
3
bedrooms            0.0467      0.037      1.268      0.206      -0.026       0.11
9
bathrooms           0.1909      0.022      8.697      0.000       0.148       0.23
4
stories             0.1085      0.019      5.669      0.000       0.071       0.14
6
mainroad            0.0504      0.014      3.524      0.000       0.022       0.07
9
guestroom           0.0304      0.014      2.238      0.026       0.004       0.05
7
basement            0.0216      0.011      1.946      0.052      -0.000       0.04
3
hotwaterheating     0.0849      0.022      3.941      0.000       0.043       0.12
7
airconditioning     0.0668      0.011      5.923      0.000       0.045       0.08
9
parking             0.0608      0.018      3.372      0.001       0.025       0.09
6
prefarea            0.0594      0.012      5.046      0.000       0.036       0.08
3
unfurnished        -0.0316      0.010     -3.096      0.002      -0.052      -0.01
2
==============================================================================
Omnibus:                       93.538   Durbin-Watson:                   2.092
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              303.844
Skew:                           1.090   Prob(JB):                     1.05e-66
Kurtosis:                       6.794   Cond. No.                         14.1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

Now Bedrooms and Basement looks insignificant . lets us check VIF

In [47]:
```
vif = pd. DataFrame()
vif["Fetaures"] = x.columns
vif["VIF"] = [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = 'VIF', ascending = False)
vif
```

Out[47]:

	Fetaures	VIF
1	bedrooms	6.59
4	mainroad	5.68
0	area	4.67
3	stories	2.69
9	parking	2.12
6	basement	2.01
8	airconditioning	1.77
2	bathrooms	1.67
10	prefarea	1.51
5	guestroom	1.47
11	unfurnished	1.40
7	hotwaterheating	1.14

In [48]:

```
# Lets us drop bedrooms

x = x.drop('bedrooms', axis = 1)
```

In [49]:

```
x_sm = sm.add_constant(x)
lr_6 = sm.OLS(y_train, x_sm).fit()
```

In [50]:

```
print(lr_6.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.680
Model:                            OLS   Adj. R-squared:                  0.671
Method:                 Least Squares   F-statistic:                     71.31
Date:                Tue, 04 Jul 2023   Prob (F-statistic):           2.73e-84
Time:                        12:47:44   Log-Likelihood:                 380.96
No. Observations:                 381   AIC:                            -737.9
Df Residuals:                     369   BIC:                            -690.6
Df Model:                          11
Covariance Type:            nonrobust
==================================================================================
=
                      coef    std err          t      P>|t|      [0.025      0.97
5]
----------------------------------------------------------------------------------
-
const               0.0357      0.015      2.421      0.016       0.007       0.06
5
area                0.2347      0.030      7.851      0.000       0.176       0.29
4
bathrooms           0.1965      0.022      9.132      0.000       0.154       0.23
9
stories             0.1178      0.018      6.654      0.000       0.083       0.15
3
mainroad            0.0488      0.014      3.423      0.001       0.021       0.07
7
guestroom           0.0301      0.014      2.211      0.028       0.003       0.05
7
basement            0.0239      0.011      2.183      0.030       0.002       0.04
5
hotwaterheating     0.0864      0.022      4.014      0.000       0.044       0.12
9
airconditioning     0.0665      0.011      5.895      0.000       0.044       0.08
9
parking             0.0629      0.018      3.501      0.001       0.028       0.09
8
prefarea            0.0596      0.012      5.061      0.000       0.036       0.08
3
unfurnished        -0.0323      0.010     -3.169      0.002      -0.052      -0.01
2
==============================================================================
Omnibus:                       97.661   Durbin-Watson:                   2.097
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              325.388
Skew:                           1.130   Prob(JB):                     2.20e-71
Kurtosis:                       6.923   Cond. No.                         10.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

P-values of all variables looks fine . let us check VIF.

In [51]:

```
vif= pd.DataFrame()
vif['Features'] = x.columns
vif['VIF']= [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]
vif['VIF'] = round(vif['VIF'],2)
vif = vif.sort_values(by = 'VIF', ascending = False)
vif
```

Out[51]:

	Features	VIF
3	mainroad	4.79
0	area	4.55
2	stories	2.23
8	parking	2.10
5	basement	1.87
7	airconditioning	1.76
1	bathrooms	1.61
9	prefarea	1.50
4	guestroom	1.46
10	unfurnished	1.33
6	hotwaterheating	1.12

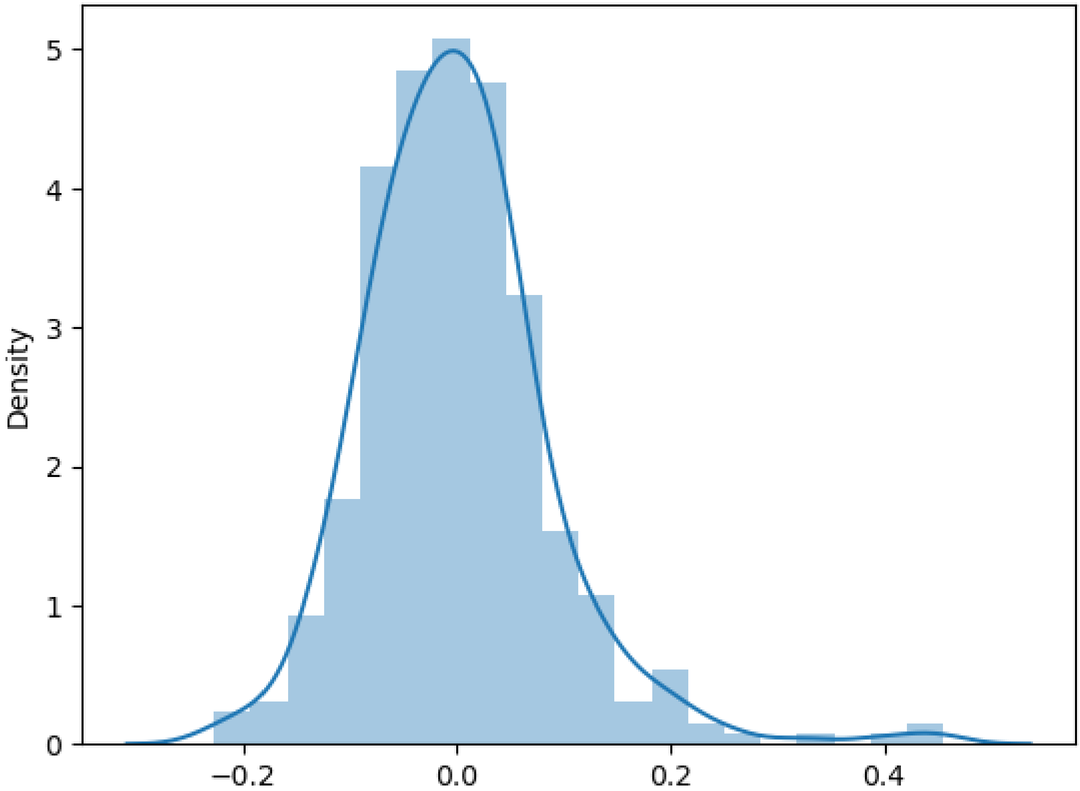
Residual Analysis Of Train Data

In [52]:
```
y_train_pred = lr_6.predict(x_sm)
```

In [53]:
```
residual = y_train- y_train_pred
```

In [54]:
```
sns.distplot(residual, bins = 20)
```

Out[54]: <AxesSubplot:ylabel='Density'>



Error terms are normally distributed.

Making Prediction Using The Final Model:

In [55]:
```
# these variables we scalled in Train data ... so let us scale the same variables

num_var =['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']


df_test[num_var]= scaler.transform(df_test[num_var])
```

In [56]:
```
df_test.describe()
```

Out[56]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basei
count	164.000000	164.000000	164.000000	164.000000	164.000000	164.000000	164.000000	164.00
mean	0.263176	0.298548	0.408537	0.158537	0.268293	0.865854	0.195122	0.34
std	0.172077	0.211922	0.147537	0.281081	0.276007	0.341853	0.397508	0.47
min	0.006061	-0.016367	0.200000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.142424	0.148011	0.400000	0.000000	0.000000	1.000000	0.000000	0.00
50%	0.226061	0.259724	0.400000	0.000000	0.333333	1.000000	0.000000	0.00
75%	0.346970	0.397439	0.400000	0.500000	0.333333	1.000000	0.000000	1.00
max	0.909091	1.263992	0.800000	1.500000	1.000000	1.000000	1.000000	1.00

In [57]:
```
y_test = df_test.pop('price')
x_test = df_test
```

In [58]:
```
x_test_sm = sm.add_constant(x_test)
```

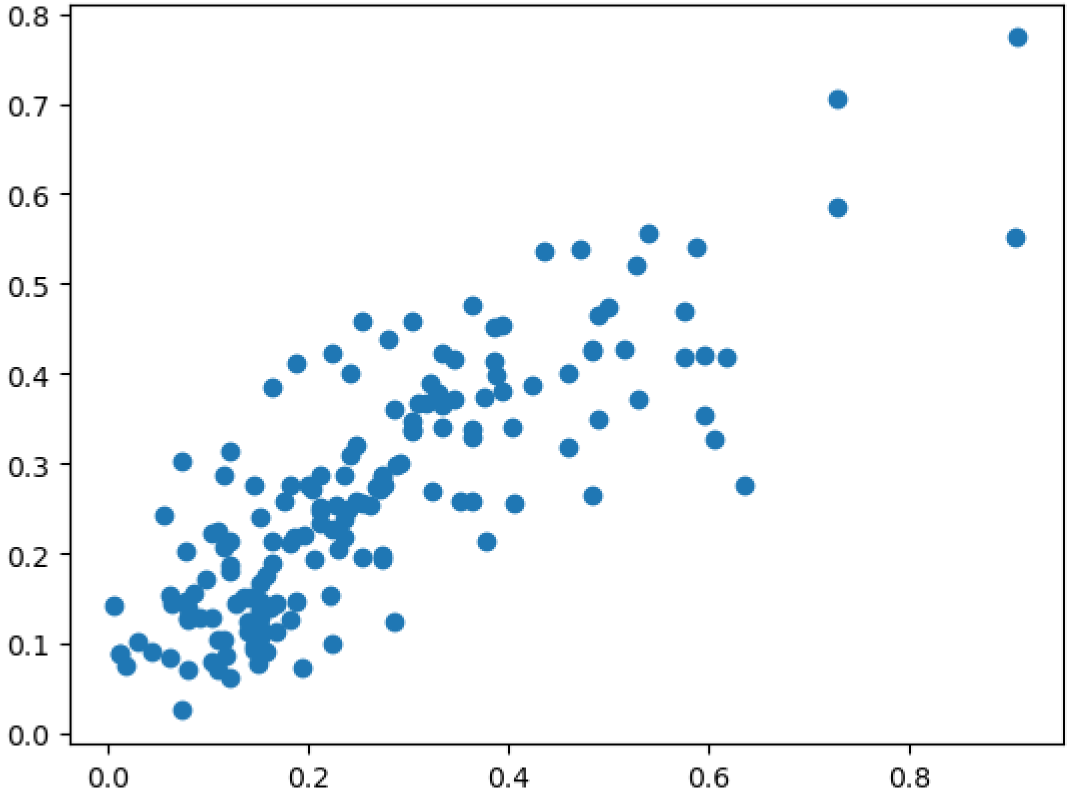
In [59]:
```
x_test_sm = x_test_sm.drop(['semi-furnished', 'bedrooms'], axis = 1)
```

In [60]:
```
y_pred = lr_6.predict(x_test_sm)
```

Model Evaluation :

In [61]:
```
plt.scatter(y_test, y_pred)
```

Out[61]:
```
<matplotlib.collections.PathCollection at 0x275ef1d8430>
```



In [62]:
```
lr_6.summary()
```

Out[62]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.680
Model:	OLS	**Adj. R-squared:**	0.671
Method:	Least Squares	**F-statistic:**	71.31
Date:	Tue, 04 Jul 2023	**Prob (F-statistic):**	2.73e-84
Time:	12:47:44	**Log-Likelihood:**	380.96
No. Observations:	381	**AIC:**	-737.9
Df Residuals:	369	**BIC:**	-690.6
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0357	0.015	2.421	0.016	0.007	0.065
area	0.2347	0.030	7.851	0.000	0.176	0.294
bathrooms	0.1965	0.022	9.132	0.000	0.154	0.239
stories	0.1178	0.018	6.654	0.000	0.083	0.153
mainroad	0.0488	0.014	3.423	0.001	0.021	0.077
guestroom	0.0301	0.014	2.211	0.028	0.003	0.057
basement	0.0239	0.011	2.183	0.030	0.002	0.045
hotwaterheating	0.0864	0.022	4.014	0.000	0.044	0.129
airconditioning	0.0665	0.011	5.895	0.000	0.044	0.089
parking	0.0629	0.018	3.501	0.001	0.028	0.098
prefarea	0.0596	0.012	5.061	0.000	0.036	0.083
unfurnished	-0.0323	0.010	-3.169	0.002	-0.052	-0.012

Omnibus:	97.661	**Durbin-Watson:**	2.097
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	325.388
Skew:	1.130	**Prob(JB):**	2.20e-71
Kurtosis:	6.923	**Cond. No.**	10.6

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Method 2 : Using RFE:

Splitting The Data Into Train Split

In [63]:
```
df_train, df_test = train_test_split(housing_df, train_size = 0.7, test_size = 0.3
```

In [64]:
```
df_train.shape
```
Out[64]:
```
(381, 14)
```

In [65]:
```
df_test.shape
```
Out[65]:
```
(164, 14)
```

Scaling Of The Data:

In [66]:
```
var_list =['price','area','bedrooms','bathrooms','stories','parking']
scaler = MinMaxScaler()

df_train[var_list] = scaler.fit_transform(df_train[var_list])
```

In [67]:
```
df_train.describe()
```
Out[67]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basei
count	381.000000	381.000000	381.000000	381.000000	381.000000	381.000000	381.000000	381.00
mean	0.260333	0.288710	0.386352	0.136483	0.268591	0.855643	0.170604	0.35
std	0.157607	0.181420	0.147336	0.237325	0.295001	0.351913	0.376657	0.47
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.151515	0.155227	0.200000	0.000000	0.000000	1.000000	0.000000	0.00
50%	0.221212	0.234424	0.400000	0.000000	0.333333	1.000000	0.000000	0.00
75%	0.345455	0.398099	0.400000	0.500000	0.333333	1.000000	0.000000	1.00
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00

Dividing X and Y Model Building:

In [68]:
```
y_train = df_train.pop('price')
x_train = df_train
```

RFE

In [69]:
```
from sklearn.feature_selection import RFE
```

In [70]:
```
from sklearn.linear_model import LinearRegression
```

In [71]:
```
lm = LinearRegression()
lm.fit(x_train, y_train)


rfe = RFE(lm,n_features_to_select=10)
rfe = rfe.fit(x_train, y_train)
```

In [72]:
```
list(zip(x_train.columns,rfe.support_, rfe.ranking_))
```

Out[72]:
```
[('area', True, 1),
 ('bedrooms', True, 1),
 ('bathrooms', True, 1),
 ('stories', True, 1),
 ('mainroad', True, 1),
 ('guestroom', True, 1),
 ('basement', False, 3),
 ('hotwaterheating', True, 1),
 ('airconditioning', True, 1),
 ('parking', True, 1),
 ('prefarea', True, 1),
 ('semi-furnished', False, 4),
 ('unfurnished', False, 2)]
```

In [73]:
```
support_col = x_train.columns[rfe.support_]
support_col
```

Out[73]:
```
Index(['area', 'bedrooms', 'bathrooms', 'stories', 'mainroad', 'guestroom',
       'hotwaterheating', 'airconditioning', 'parking', 'prefarea'],
      dtype='object')
```

In [74]:
```
discarded_col = x_train.columns[~rfe.support_]
discarded_col
```

Out[74]:
```
Index(['basement', 'semi-furnished', 'unfurnished'], dtype='object')
```

Building The Model Using Supported Columns:

In [75]:
```
x_train_rfe = x_train[support_col]
```

In [76]:
```
x_train_rfe_sm = sm.add_constant(x_train_rfe)
```

In [77]:
```
lr_rfe = sm.OLS(y_train, x_train_rfe_sm).fit()
```

In [78]:
```
lr_rfe.params
```

Out[78]:
```
const              0.002721
area               0.236257
bedrooms           0.066102
bathrooms          0.198169
stories            0.097722
mainroad           0.055649
guestroom          0.038136
hotwaterheating    0.089673
airconditioning    0.071079
parking            0.063739
prefarea           0.064326
dtype: float64
```

In [ ]:

In [79]:
```
print(lr_rfe.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.669
Model:                            OLS   Adj. R-squared:                  0.660
Method:                 Least Squares   F-statistic:                     74.89
Date:                Tue, 04 Jul 2023   Prob (F-statistic):           1.28e-82
Time:                        12:47:46   Log-Likelihood:                 374.65
No. Observations:                 381   AIC:                            -727.3
Df Residuals:                     370   BIC:                            -683.9
Df Model:                          10
Covariance Type:            nonrobust
==================================================================================
=
                       coef    std err          t      P>|t|      [0.025      0.97
5]
----------------------------------------------------------------------------------
-
const                0.0027      0.018      0.151      0.880      -0.033       0.03
8
area                 0.2363      0.030      7.787      0.000       0.177       0.29
6
bedrooms             0.0661      0.037      1.794      0.074      -0.006       0.13
9
bathrooms            0.1982      0.022      8.927      0.000       0.155       0.24
2
stories              0.0977      0.019      5.251      0.000       0.061       0.13
4
mainroad             0.0556      0.014      3.848      0.000       0.027       0.08
4
guestroom            0.0381      0.013      2.934      0.004       0.013       0.06
4
hotwaterheating      0.0897      0.022      4.104      0.000       0.047       0.13
3
airconditioning      0.0711      0.011      6.235      0.000       0.049       0.09
3
parking              0.0637      0.018      3.488      0.001       0.028       0.10
0
prefarea             0.0643      0.012      5.445      0.000       0.041       0.08
8
==============================================================================
Omnibus:                       86.105   Durbin-Watson:                   2.098
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              286.069
Skew:                           0.992   Prob(JB):                     7.60e-63
Kurtosis:                       6.753   Cond. No.                         13.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

Variable bedrooms is significant

In [80]:
```
x_train_rfe_1 = x_train_rfe.drop(['bedrooms'], axis = 1)
```

In [81]:
```
x_train_rfe_new = sm.add_constant(x_train_rfe_1)
```

In [82]:
```
lr_rfe_1 = sm.OLS(y_train, x_train_rfe_new).fit()
```

In [83]:
```
print(lr_rfe_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.666
Model:                            OLS   Adj. R-squared:                  0.658
Method:                 Least Squares   F-statistic:                     82.37
Date:                Tue, 04 Jul 2023   Prob (F-statistic):           6.67e-83
Time:                        12:47:46   Log-Likelihood:                 373.00
No. Observations:                 381   AIC:                            -726.0
Df Residuals:                     371   BIC:                            -686.6
Df Model:                           9
Covariance Type:            nonrobust
==================================================================================
=
                      coef    std err          t      P>|t|      [0.025      0.97
5]
----------------------------------------------------------------------------------
-
const               0.0242      0.013      1.794      0.074      -0.002       0.05
1
area                0.2367      0.030      7.779      0.000       0.177       0.29
7
bathrooms           0.2070      0.022      9.537      0.000       0.164       0.25
0
stories             0.1096      0.017      6.280      0.000       0.075       0.14
4
mainroad            0.0536      0.014      3.710      0.000       0.025       0.08
2
guestroom           0.0390      0.013      2.991      0.003       0.013       0.06
5
hotwaterheating     0.0921      0.022      4.213      0.000       0.049       0.13
5
airconditioning     0.0710      0.011      6.212      0.000       0.049       0.09
4
parking             0.0669      0.018      3.665      0.000       0.031       0.10
3
prefarea            0.0653      0.012      5.513      0.000       0.042       0.08
9
==============================================================================
Omnibus:                       91.542   Durbin-Watson:                   2.107
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              315.402
Skew:                           1.044   Prob(JB):                     3.25e-69
Kurtosis:                       6.938   Cond. No.                         10.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

All the P -values looks significant. let us check VIF

In [84]:

```
vif = pd.DataFrame()
vif['Feature']= x_train_rfe_1.columns
vif['VIF'] = [variance_inflation_factor(x_train_rfe_1.values,i) for i in range (x_
vif = vif.sort_values(by = 'VIF', ascending = False)
vif
```

Out[84]:

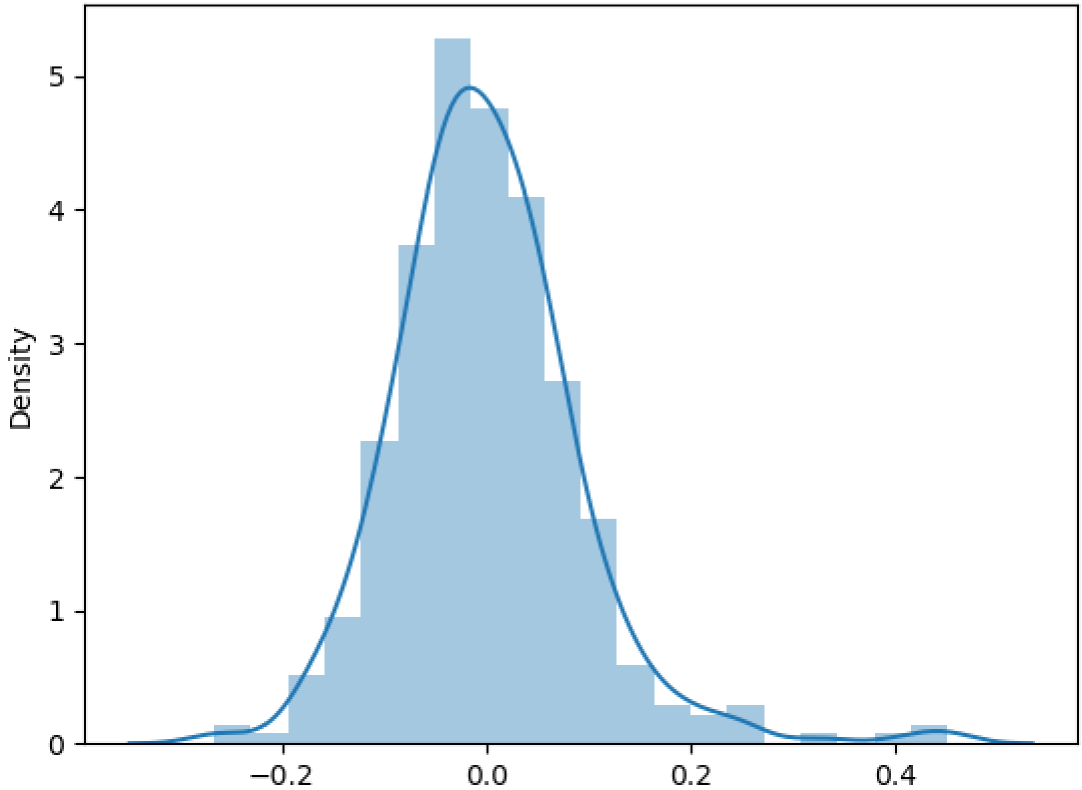
	Feature	VIF
0	area	4.516773
3	mainroad	4.263472
2	stories	2.120356
7	parking	2.096114
6	airconditioning	1.748100
1	bathrooms	1.578669
8	prefarea	1.466057
4	guestroom	1.300287
5	hotwaterheating	1.121364

Residual Analysis

In [85]:
```
y_train_pred = lr_rfe_1.predict(x_train_rfe_new)

res = y_train - y_train_pred
```

In [86]:
```
sns.distplot(res, bins= 20)
```

Out[86]: <AxesSubplot:ylabel='Density'>



Error terms are normally distributed.

Making Predictions Using The Final Model

In [87]:
```
var_list =['price', 'area','bedrooms','bathrooms','stories','parking']

df_test[var_list] = scaler.transform(df_test[var_list])
```

In [88]:
```
y_test = df_test.pop('price')
x_test = df_test
```

In [89]:
```
col = x_train_rfe_1.columns
```

In [90]:
```
x_test_new = x_test[col]
```

In [91]:
```
x_test_new.columns
```

Out[91]:
```
Index(['area', 'bathrooms', 'stories', 'mainroad', 'guestroom',
       'hotwaterheating', 'airconditioning', 'parking', 'prefarea'],
      dtype='object')
```

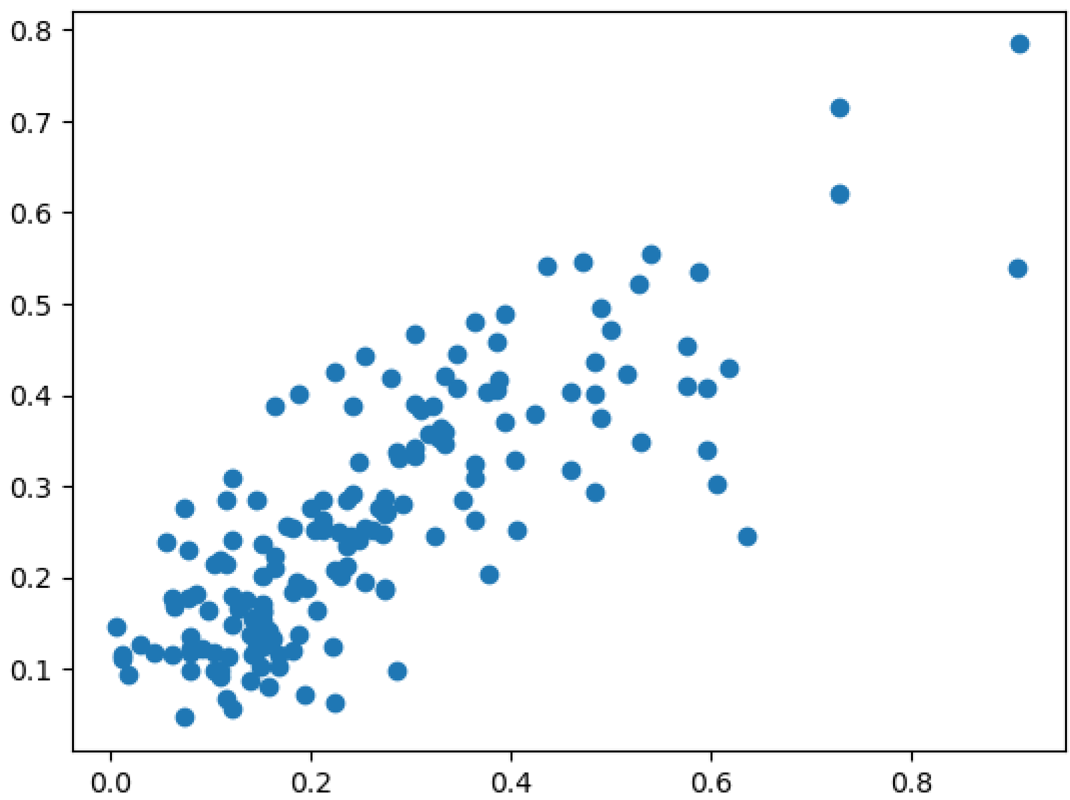
In [92]:
```
x_test_rfe = sm.add_constant(x_test_new)
```

In [93]:
```
y_pred = lr_rfe_1.predict(x_test_rfe)
```

Model Evaluation

In [94]:
```
plt.scatter(y_test, y_pred)
```

Out[94]:
```
<matplotlib.collections.PathCollection at 0x275efc005e0>
```



In [95]:
```
print(lr_rfe_1.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  price   R-squared:                       0.666
Model:                            OLS   Adj. R-squared:                  0.658
Method:                 Least Squares   F-statistic:                     82.37
Date:                Tue, 04 Jul 2023   Prob (F-statistic):           6.67e-83
Time:                        12:47:47   Log-Likelihood:                 373.00
No. Observations:                 381   AIC:                            -726.0
Df Residuals:                     371   BIC:                            -686.6
Df Model:                           9                                         
Covariance Type:            nonrobust                                         
==================================================================================
=
                      coef    std err          t      P>|t|      [0.025      0.97
5]
----------------------------------------------------------------------------------
-
const               0.0242      0.013      1.794      0.074      -0.002       0.05
1
area                0.2367      0.030      7.779      0.000       0.177       0.29
7
bathrooms           0.2070      0.022      9.537      0.000       0.164       0.25
0
stories             0.1096      0.017      6.280      0.000       0.075       0.14
4
mainroad            0.0536      0.014      3.710      0.000       0.025       0.08
2
guestroom           0.0390      0.013      2.991      0.003       0.013       0.06
5
hotwaterheating     0.0921      0.022      4.213      0.000       0.049       0.13
5
airconditioning     0.0710      0.011      6.212      0.000       0.049       0.09
4
parking             0.0669      0.018      3.665      0.000       0.031       0.10
3
prefarea            0.0653      0.012      5.513      0.000       0.042       0.08
9
==============================================================================
Omnibus:                       91.542   Durbin-Watson:                   2.107
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              315.402
Skew:                           1.044   Prob(JB):                     3.25e-69
Kurtosis:                       6.938   Cond. No.                         10.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: