Assessment 3

Pallav Nag

20BCE2023

In [1]:
```
from sklearn import metrics
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.metrics import confusion_matrix,accuracy_score
dataset=pd.DataFrame(pd.read_csv('Housing (1).csv'))
```

In [2]:
```
dataset.head()
```

Out[2]:

	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	

In [3]:
```
dataset.tail()
```

Out[3]:

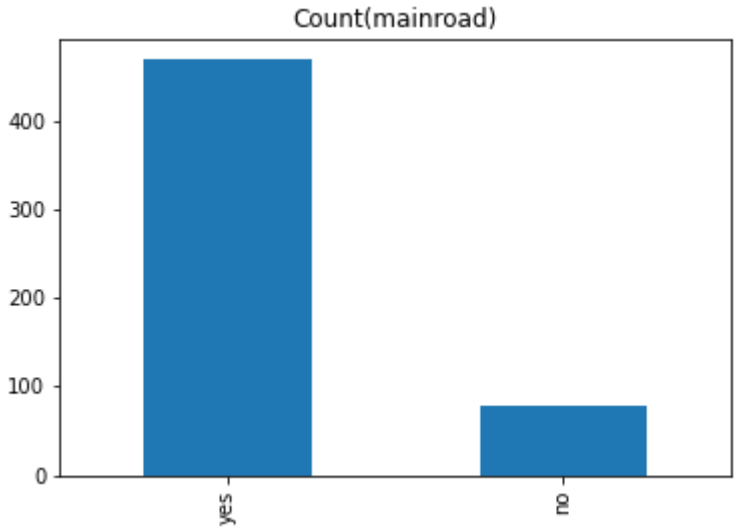
	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterhe
540	1820000	3000	2	1	1	yes	no	yes	
541	1767150	2400	3	1	1	no	no	no	
542	1750000	3620	2	1	1	yes	no	no	
543	1750000	2910	3	1	1	no	no	no	
544	1750000	3850	3	1	2	yes	no	no	

In [4]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 12 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  furnishingstatus  545 non-null    object
dtypes: int64(6), object(6)
memory usage: 51.2+ KB
```

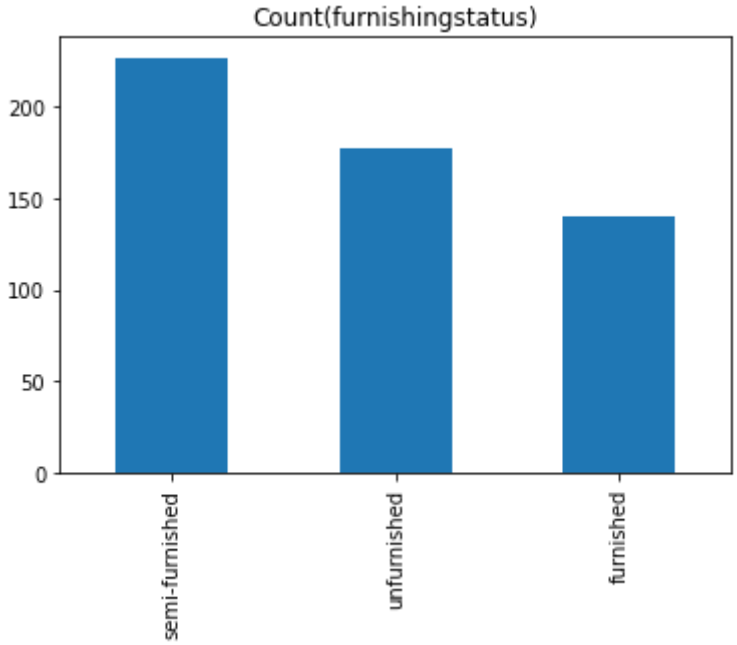
In [5]:
```
#Data Visualization
#Univariate Analysis
##BAR PLOT FOR MAINROAD
targ_cnt=dataset.mainroad.value_counts()
print('mainroad ',targ_cnt[1])
print('yes ',targ_cnt[0])
print('Survival Rate: ',round(targ_cnt[1]/(targ_cnt[0]+targ_cnt[1]),2),':1')
targ_cnt.plot(kind='bar',title='Count(mainroad)')
plt.show()
```

```
mainroad  77
yes  468
Survival Rate:  0.14 :1
```



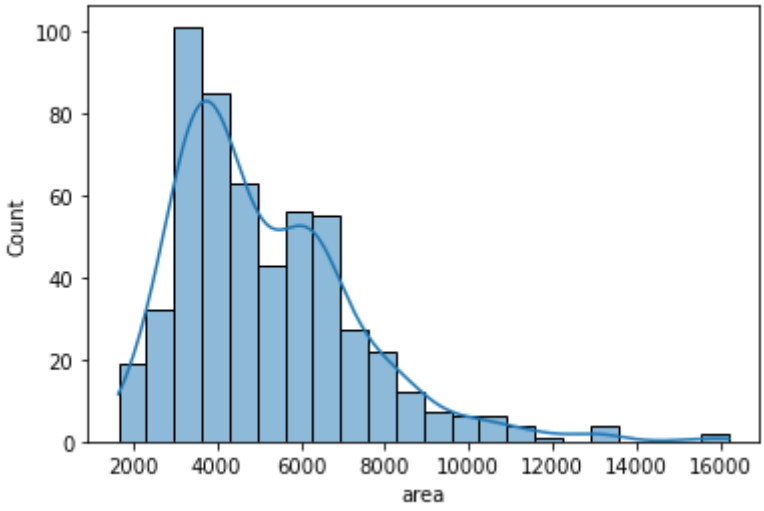
In [6]:
```
##BAR PLOT FOR FURNISHING STATUS
targ_cnt=dataset.furnishingstatus.value_counts()
print('SEMIFURNISHED ',targ_cnt[0])
print('UNFURNISHED',targ_cnt[1])
print('FURNISHED ',targ_cnt[2])
print('Survival Rate: ',round(targ_cnt[1]/(targ_cnt[0]+targ_cnt[1]),2),':1')
targ_cnt.plot(kind='bar',title='Count(furnishingstatus)')
plt.show()
```

```
SEMIFURNISHED  227
UNFURNISHED 178
FURNISHED  140
Survival Rate:  0.44 :1
```



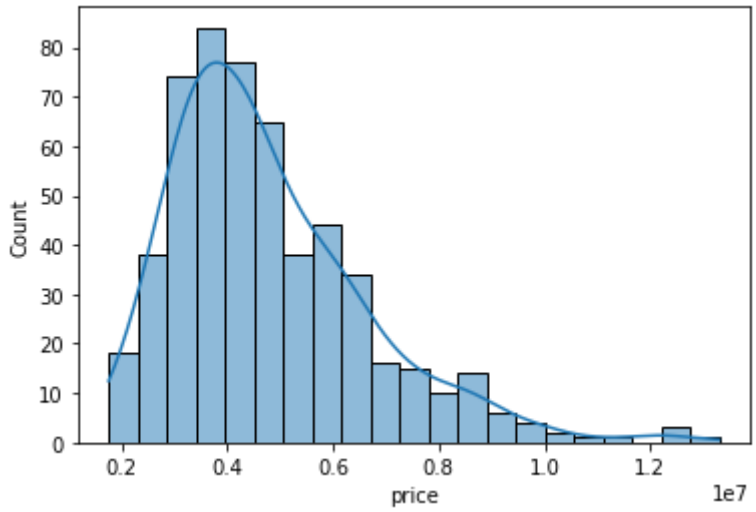
In [7]:
```
#histogram for AREA
sns.histplot(dataset.area,kde=True)
```

Out[7]:
```
<AxesSubplot:xlabel='area', ylabel='Count'>
```



In [8]:
```
#histogram for PRICE
sns.histplot(dataset.price,kde=True)
```

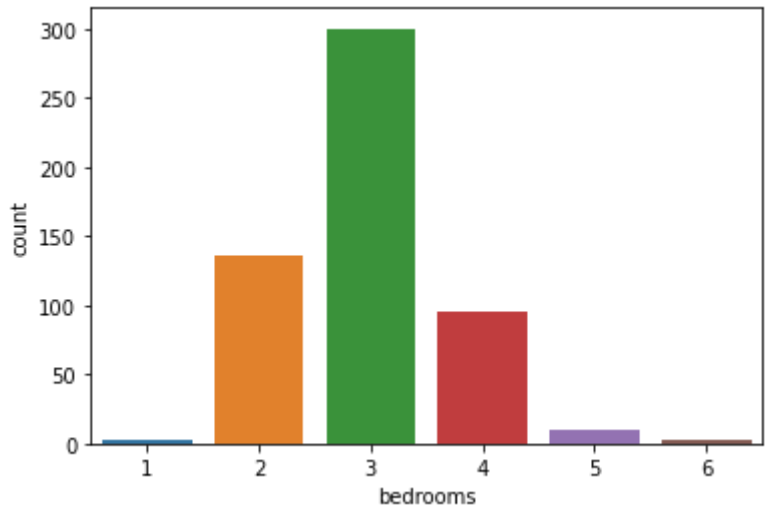
Out[8]:
```
<AxesSubplot:xlabel='price', ylabel='Count'>
```



In [9]:
```
#countplot for various attributes
sns.countplot(dataset.bedrooms)
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
```

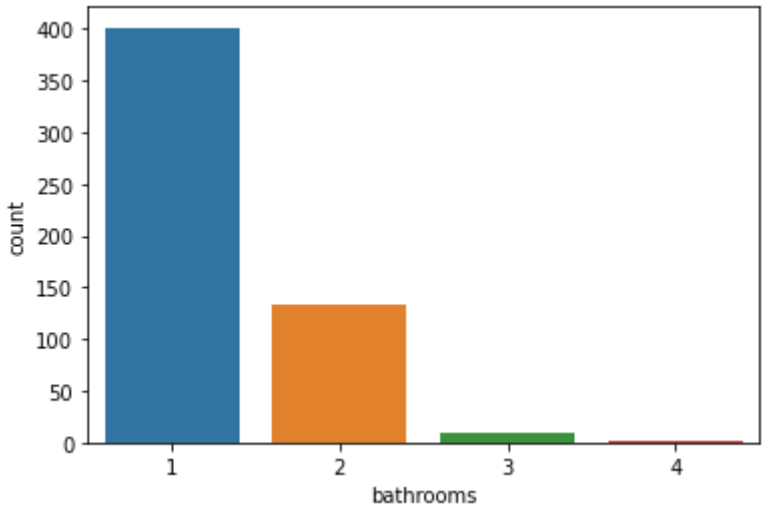
Out[9]:
```
<AxesSubplot:xlabel='bedrooms', ylabel='count'>
```



In [10]:
```
sns.countplot(dataset.bathrooms)
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
```

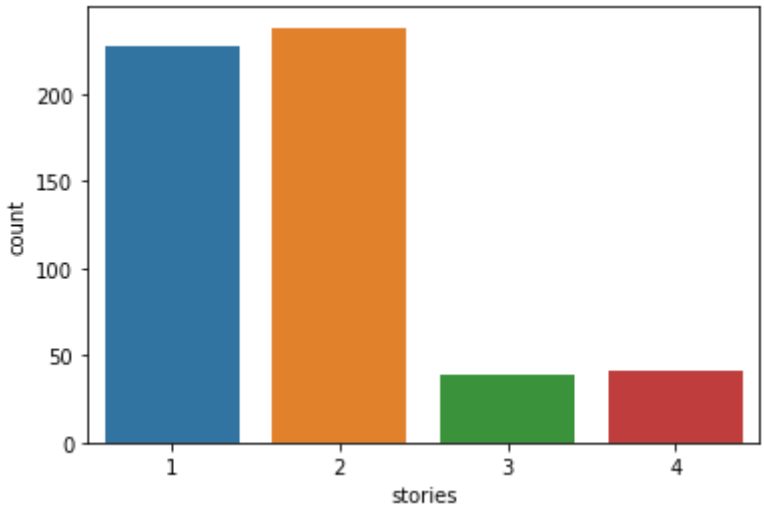
Out[10]:
```
<AxesSubplot:xlabel='bathrooms', ylabel='count'>
```



In [11]:
```
sns.countplot(dataset.stories)
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
```

Out[11]:
```
<AxesSubplot:xlabel='stories', ylabel='count'>
```



In [12]:
```
#Bivariate Analysis
#finding correlation among numerical data
dataset[['price','area']].corr()
```

Out[12]:

	price	area
price	1.000000	0.535997
area	0.535997	1.000000

In [13]:
```
dataset[['bedrooms','bathrooms','stories','parking']].corr()
```

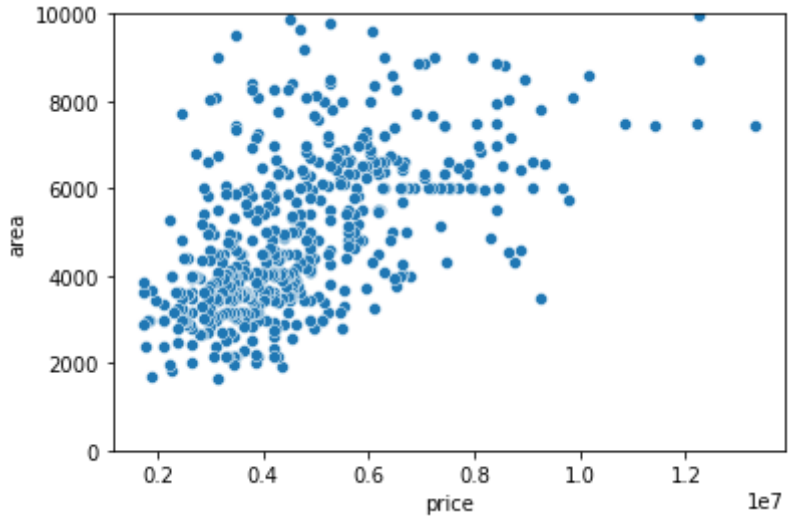
Out[13]:

	bedrooms	bathrooms	stories	parking
bedrooms	1.000000	0.373930	0.408564	0.139270
bathrooms	0.373930	1.000000	0.326165	0.177496
stories	0.408564	0.326165	1.000000	0.045547
parking	0.139270	0.177496	0.045547	1.000000

In [14]:
```
#scatterplot between price and area
sns.scatterplot(dataset.price, dataset.area)
plt.ylim(0,10000)
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variables as keyword args: x, y. From version 0.12, the only valid positi
onal argument will be `data`, and passing other arguments without an explicit keyw
ord will result in an error or misinterpretation.
  warnings.warn(
```

Out[14]:
```
(0.0, 10000.0)
```



In [15]:
```
#bi-variate analysis between one categorical and other numerical data
#mainroad
dataset.groupby(by="mainroad").agg("mean")[['price','area','bedrooms','bathrooms',
```

Out[15]:

mainroad	price	area	bedrooms	bathrooms	stories	parking
no	3.398905e+06	3606.441558	2.987013	1.233766	1.545455	0.259740
yes	4.991777e+06	5404.591880	2.961538	1.294872	1.848291	0.764957

In [16]:
```
#guestroom
dataset.groupby(by="guestroom").agg("mean")[['price','area','bedrooms','bathrooms'
```

Out[16]:

guestroom	price	area	bedrooms	bathrooms	stories	parking
no	4.544546e+06	5009.000000	2.937500	1.256696	1.787946	0.678571
yes	5.792897e+06	5804.257732	3.092784	1.422680	1.886598	0.762887

In [17]:
```
#basement
dataset.groupby(by="basement").agg("mean")[['price','area','bedrooms','bathrooms',
```

Out[17]:

basement	price	area	bedrooms	bathrooms	stories	parking
no	4.509966e+06	5075.025424	2.912429	1.248588	1.915254	0.661017
yes	5.242615e+06	5290.502618	3.062827	1.356021	1.602094	0.753927

In [18]:
```
#hotwaterheating
dataset.groupby(by="hotwaterheating").agg("mean")[['price','area','bedrooms','bathr
```

Out[18]:

hotwaterheating	price	area	bedrooms	bathrooms	stories	parking
no	4.728593e+06	5154.928846	2.957692	1.278846	1.801923	0.680769
yes	5.559960e+06	5059.280000	3.120000	1.440000	1.880000	0.960000

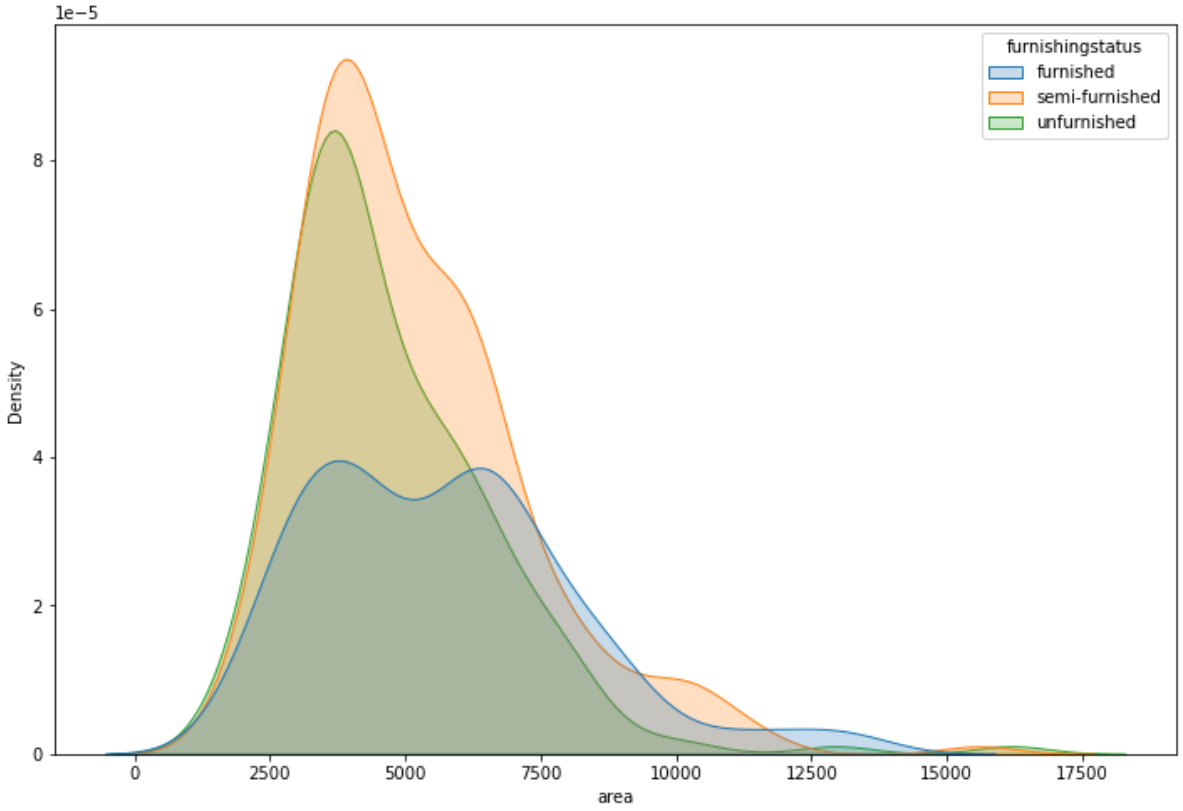
In [19]:
```
#airconditioning
dataset.groupby(by="airconditioning").agg("mean")[['price','area','bedrooms','bathr
```

Out[19]:

airconditioning	price	area	bedrooms	bathrooms	stories	parking
no	4.191940e+06	4823.109920	2.884718	1.222520	1.632708	0.600536
yes	6.013221e+06	5860.610465	3.139535	1.424419	2.180233	0.895349

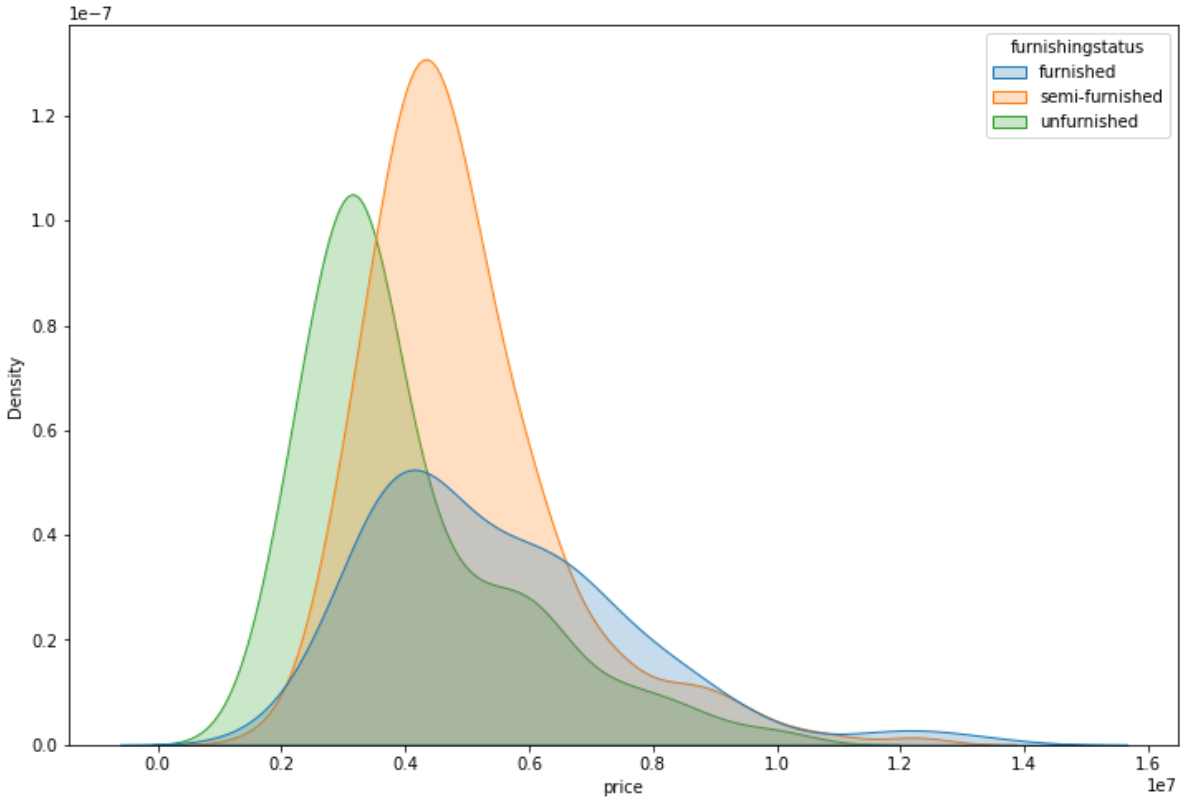
In [20]:
```
#kdeplot visualization for all numerical attributes with target attribute
plt.figure(figsize=(12,8))
sns.kdeplot(data=dataset, x= 'area',hue='furnishingstatus',fill=True)
```

Out[20]:
```
<AxesSubplot:xlabel='area', ylabel='Density'>
```



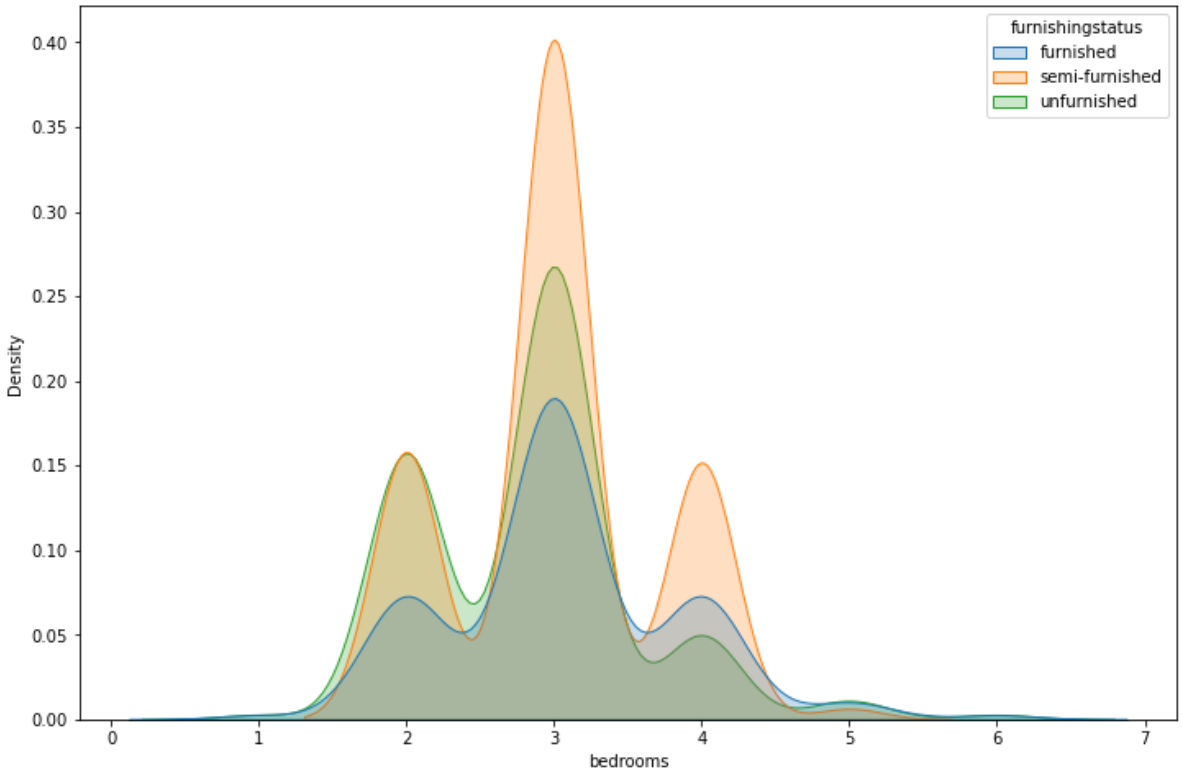
In [21]:
```
plt.figure(figsize=(12,8))
sns.kdeplot(data=dataset, x= 'price',hue='furnishingstatus',fill=True)
```

Out[21]:
```
<AxesSubplot:xlabel='price', ylabel='Density'>
```



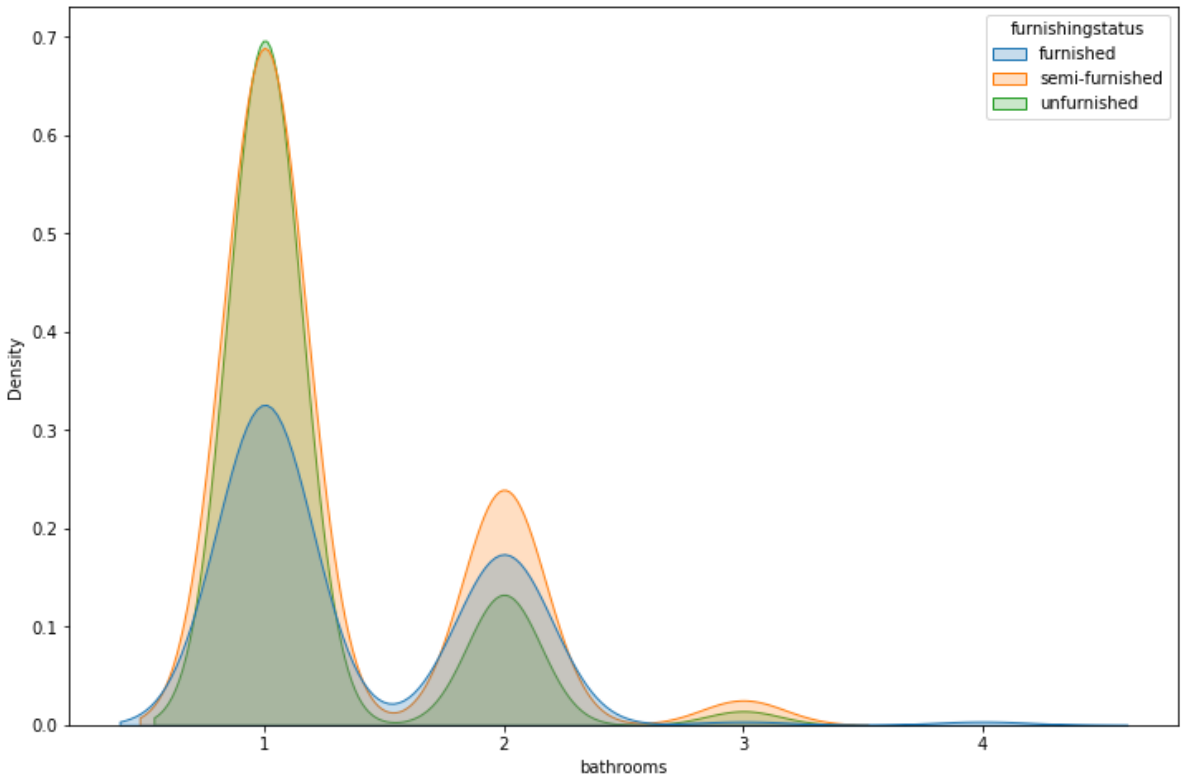
In [22]:
```
plt.figure(figsize=(12,8))
sns.kdeplot(data=dataset, x= 'bedrooms',hue='furnishingstatus',fill=True)
```

Out[22]:
```
<AxesSubplot:xlabel='bedrooms', ylabel='Density'>
```



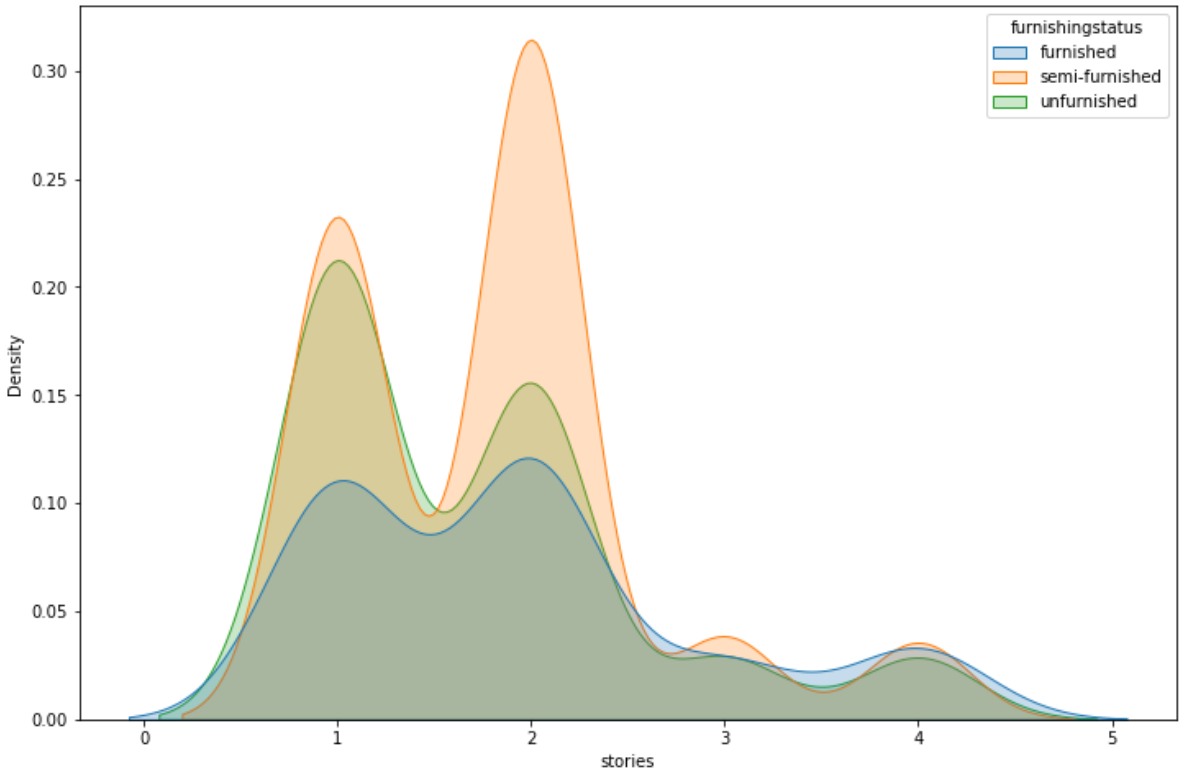
In [23]:
```
plt.figure(figsize=(12,8))
sns.kdeplot(data=dataset, x= 'bathrooms',hue='furnishingstatus',fill=True)
```

Out[23]:
```
<AxesSubplot:xlabel='bathrooms', ylabel='Density'>
```



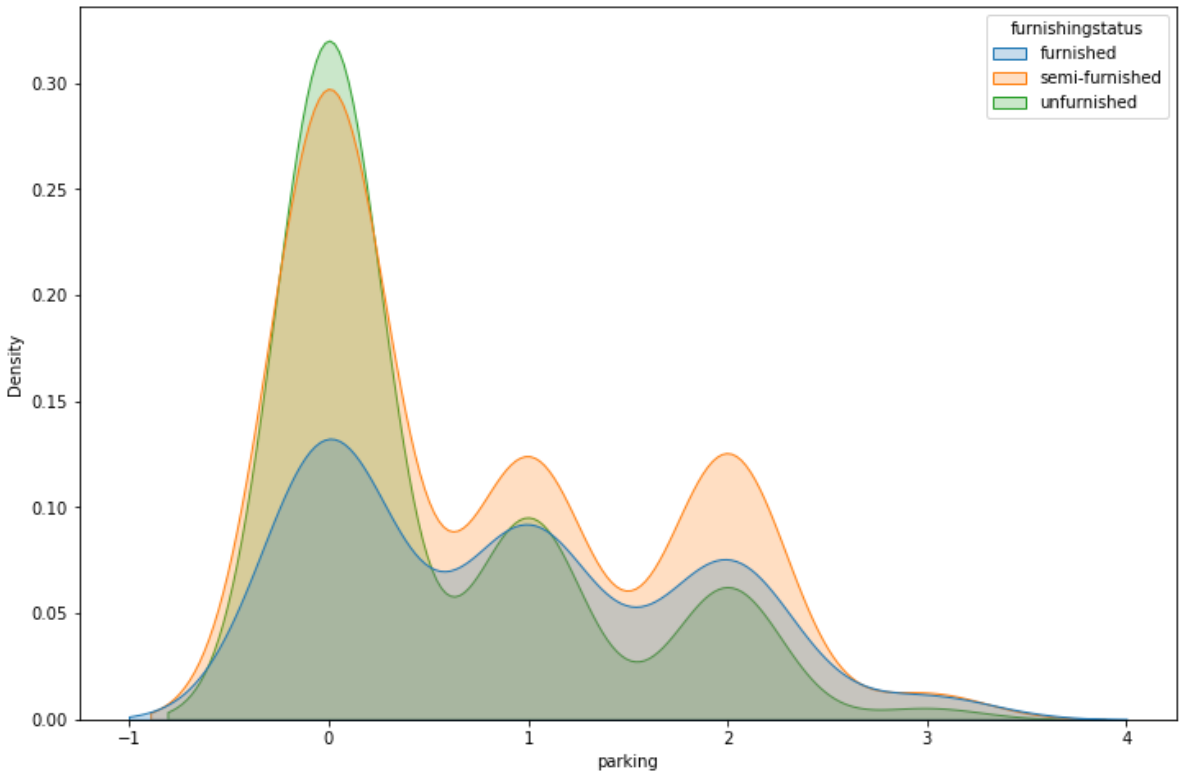
In [24]:
```
plt.figure(figsize=(12,8))
sns.kdeplot(data=dataset, x= 'stories',hue='furnishingstatus',fill=True)
```

Out[24]:
```
<AxesSubplot:xlabel='stories', ylabel='Density'>
```



In [25]:
```
plt.figure(figsize=(12,8))
sns.kdeplot(data=dataset, x= 'parking',hue='furnishingstatus',fill=True)
```

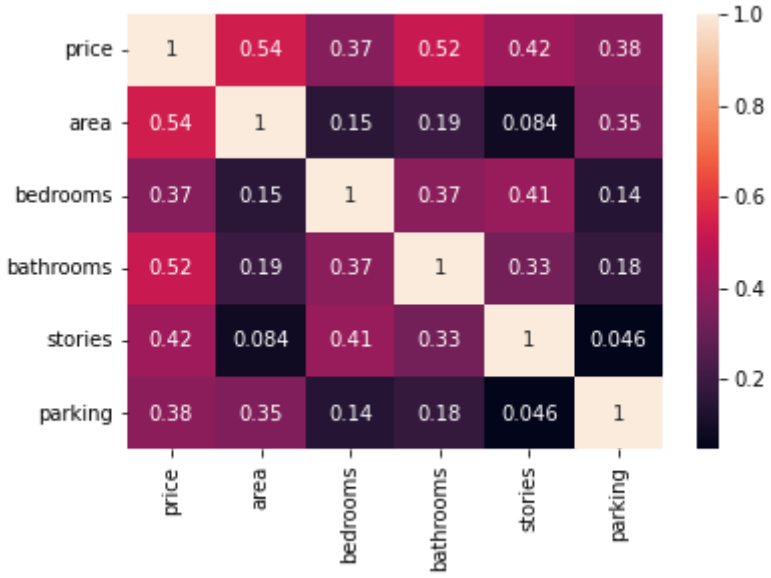
Out[25]:
```
<AxesSubplot:xlabel='parking', ylabel='Density'>
```



In [26]:
```
#MULTIVARIATE
#pairplot
#sns.pairplot(data=dataset[['price','area','bedrooms','bathrooms','stories','parki
```

In [27]:
```
#heatmap
sns.heatmap(dataset.corr(), annot=True)
```

Out[27]:
```
<AxesSubplot:>
```



In [28]:
```
#DESCRIBE
df=dataset
df.describe()
```

Out[28]:

	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
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
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

In [29]:
```
pd.get_dummies(df)
```

Out[29]:

	price	area	bedrooms	bathrooms	stories	parking	mainroad_no	mainroad_yes	guestr
0	13300000	7420	4	2	3	2	0	1	
1	12250000	8960	4	4	4	3	0	1	
2	12250000	9960	3	2	2	2	0	1	
3	12215000	7500	4	2	2	3	0	1	
4	11410000	7420	4	1	2	2	0	1	
...	...	...	...	...	...	...	...	...	
540	1820000	3000	2	1	1	2	0	1	
541	1767150	2400	3	1	1	0	1	0	
542	1750000	3620	2	1	1	0	0	1	
543	1750000	2910	3	1	1	0	1	0	
544	1750000	3850	3	1	2	0	0	1	

545 rows × 19 columns

In [30]:
```
np.mean(df)
```

```
D:\anaconda\lib\site-packages\numpy\core\fromnumeric.py:3438: FutureWarning: In a
future version, DataFrame.mean(axis=None) will return a scalar mean over the entir
e DataFrame. To retain the old behavior, use 'frame.mean(axis=0)' or just 'frame.m
ean()'
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
D:\anaconda\lib\site-packages\numpy\core\fromnumeric.py:3438: FutureWarning: Dropp
ing of nuisance columns in DataFrame reductions (with 'numeric_only=None') is depr
ecated; in a future version this will raise TypeError.  Select only valid columns
before calling the reduction.
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
```

Out[30]:
```
price        4.766729e+06
area         5.150541e+03
bedrooms     2.965138e+00
bathrooms    1.286239e+00
stories      1.805505e+00
parking      6.935780e-01
dtype: float64
```

In [31]:
```
df.median()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_4064\530051474.py:1: FutureWarning: Dropp
ing of nuisance columns in DataFrame reductions (with 'numeric_only=None') is depr
ecated; in a future version this will raise TypeError.  Select only valid columns
before calling the reduction.
  df.median()
```

Out[31]:
```
price        4340000.0
area            4600.0
bedrooms           3.0
bathrooms          1.0
stories            2.0
parking            0.0
dtype: float64
```

In [32]:
```
df.mode()
```

Out[32]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterhea
0	3500000	6000.0	3.0	1.0	2.0	yes	no	no	
1	4200000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [33]:

```
missing_values = df.isnull().sum()
print("Missing values:\n", missing_values)
print("\n No Missing Values \n")
```

```
Missing values:
 price               0
area                0
bedrooms            0
bathrooms           0
stories             0
mainroad            0
guestroom           0
basement            0
hotwaterheating     0
airconditioning     0
parking             0
furnishingstatus    0
dtype: int64

 No Missing Values
```

In [34]:

```
# Find outliers
def find_outliers(column):
    q1 = np.percentile(column, 25)
    q3 = np.percentile(column, 75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    outliers = column[(column < lower_bound) | (column > upper_bound)]
    return outliers

# Replace outliers
def replace_outliers(column):
    q1 = np.percentile(column, 25)
    q3 = np.percentile(column, 75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    column = np.where((column < lower_bound) | (column > upper_bound), column.medi
    return column

# Specify the columns with numerical variables for outlier detection and replacemer
numerical_columns = ['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking

# Find and replace outliers for each numerical column
for column in numerical_columns:
    outliers = find_outliers(df[column])
    print("Outliers in", column, ":\n", outliers)

    df[column] = replace_outliers(df[column])

# Check if outliers have been replaced
for column in numerical_columns:
    outliers = find_outliers(df[column])
    print("Outliers in", column, "after replacement:\n", outliers)
```

```
# Continue with the processed data (with replaced outliers)
processed_df = df
```

```
Outliers in price :
 0     13300000
1     12250000
2     12250000
3     12215000
4     11410000
5     10850000
6     10150000
7     10150000
8      9870000
9      9800000
10     9800000
11     9681000
12     9310000
13     9240000
14     9240000
Name: price, dtype: int64
Outliers in area :
 7      16200
10     13200
56     11440
64     11175
66     13200
69     12090
125    15600
129    11460
186    11410
191    10700
211    12900
403    12944
Name: area, dtype: int64
Outliers in bedrooms :
 7      5
28     5
34     5
89     5
112    6
143    5
152    5
271    5
340    5
356    5
395    6
536    5
Name: bedrooms, dtype: int64
Outliers in bathrooms :
 1    4
Name: bathrooms, dtype: int64
Outliers in stories :
 1      4
6      4
9      4
17     4
26     4
30     4
31     4
35     4
37     4
38     4
39     4
41     4
42     4
43     4
44     4
```

```
46     4
47     4
50     4
51     4
52     4
53     4
57     4
58     4
59     4
71     4
72     4
73     4
83     4
92     4
94     4
102    4
105    4
124    4
131    4
135    4
140    4
145    4
160    4
220    4
226    4
247    4
Name: stories, dtype: int64
Outliers in parking :
 1      3
3      3
47     3
93     3
225    3
247    3
299    3
304    3
323    3
331    3
401    3
472    3
Name: parking, dtype: int64
Outliers in price after replacement:
 15    9100000.0
16    9100000.0
17    8960000.0
18    8890000.0
19    8855000.0
Name: price, dtype: float64
Outliers in area after replacement:
 82     10500.0
142    10500.0
146    10500.0
171    10269.0
224    10240.0
277    10360.0
Name: area, dtype: float64
Outliers in bedrooms after replacement:
 Series([], Name: bedrooms, dtype: float64)
Outliers in bathrooms after replacement:
 Series([], Name: bathrooms, dtype: float64)
Outliers in stories after replacement:
 Series([], Name: stories, dtype: float64)
Outliers in parking after replacement:
 Series([], Name: parking, dtype: float64)
```

In [35]:

```
# Check for categorical columns
categorical_cols = processed_df.select_dtypes(include=['object']).columns.tolist()

# Perform encoding using one-hot encoding
encoded_df = pd.get_dummies(processed_df, columns=categorical_cols)

# Print the encoded DataFrame
print(encoded_df)
```

```
         price    area  bedrooms  bathrooms  stories  parking  mainroad_no  \
0    4340000.0  7420.0       4.0        2.0      3.0      2.0            0
1    4340000.0  8960.0       4.0        1.0      2.0      0.0            0
2    4340000.0  9960.0       3.0        2.0      2.0      2.0            0
3    4340000.0  7500.0       4.0        2.0      2.0      0.0            0
4    4340000.0  7420.0       4.0        1.0      2.0      2.0            0
..         ...     ...       ...        ...      ...      ...          ...
540  1820000.0  3000.0       2.0        1.0      1.0      2.0            0
541  1767150.0  2400.0       3.0        1.0      1.0      0.0            1
542  1750000.0  3620.0       2.0        1.0      1.0      0.0            0
543  1750000.0  2910.0       3.0        1.0      1.0      0.0            1
544  1750000.0  3850.0       3.0        1.0      2.0      0.0            0

     mainroad_yes  guestroom_no  guestroom_yes  basement_no  basement_yes  \
0               1             1              0            1             0
1               1             1              0            1             0
2               1             1              0            0             1
3               1             1              0            0             1
4               1             0              1            0             1
..            ...           ...            ...          ...           ...
540             1             1              0            0             1
541             0             1              0            1             0
542             1             1              0            1             0
543             0             1              0            1             0
544             1             1              0            1             0

     hotwaterheating_no  hotwaterheating_yes  airconditioning_no  \
0                     1                    0                   0
1                     1                    0                   0
2                     1                    0                   1
3                     1                    0                   0
4                     1                    0                   0
..                  ...                  ...                 ...
540                   1                    0                   1
541                   1                    0                   1
542                   1                    0                   1
543                   1                    0                   1
544                   1                    0                   1

     airconditioning_yes  furnishingstatus_furnished  \
0                      1                           1
1                      1                           1
2                      0                           0
3                      1                           1
4                      1                           1
..                   ...                         ...
540                    0                           0
541                    0                           0
542                    0                           0
543                    0                           1
544                    0                           0

     furnishingstatus_semi-furnished  furnishingstatus_unfurnished
0                                  0                             0
1                                  0                             0
2                                  1                             0
3                                  0                             0
4                                  0                             0
..                               ...                           ...
540                                0                             1
541                                1                             0
542                                0                             1
543                                0                             0
544                                0                             1
```

```
[545 rows x 19 columns]
```

In [36]:

```python
# Splitting the data into dependent (target) variable and independent (feature) var
X = encoded_df.drop('price', axis=1)  # Independent variables
y = encoded_df['price']  # Dependent variable

# Printing the independent and dependent variable dataframes
print("Independent Variables (Features):\n", X)
print("\nDependent Variable (Price):\n", y)
```

```
Independent Variables (Features):
        area  bedrooms  bathrooms  stories  parking  mainroad_no  mainroad_yes  \
0    7420.0       4.0        2.0      3.0      2.0            0             1
1    8960.0       4.0        1.0      2.0      0.0            0             1
2    9960.0       3.0        2.0      2.0      2.0            0             1
3    7500.0       4.0        2.0      2.0      0.0            0             1
4    7420.0       4.0        1.0      2.0      2.0            0             1
..      ...       ...        ...      ...      ...          ...           ...
540  3000.0       2.0        1.0      1.0      2.0            0             1
541  2400.0       3.0        1.0      1.0      0.0            1             0
542  3620.0       2.0        1.0      1.0      0.0            0             1
543  2910.0       3.0        1.0      1.0      0.0            1             0
544  3850.0       3.0        1.0      2.0      0.0            0             1

     guestroom_no  guestroom_yes  basement_no  basement_yes  \
0               1              0            1             0
1               1              0            1             0
2               1              0            0             1
3               1              0            0             1
4               0              1            0             1
..            ...            ...          ...           ...
540             1              0            0             1
541             1              0            1             0
542             1              0            1             0
543             1              0            1             0
544             1              0            1             0

     hotwaterheating_no  hotwaterheating_yes  airconditioning_no  \
0                     1                    0                   0
1                     1                    0                   0
2                     1                    0                   1
3                     1                    0                   0
4                     1                    0                   0
..                  ...                  ...                 ...
540                   1                    0                   1
541                   1                    0                   1
542                   1                    0                   1
543                   1                    0                   1
544                   1                    0                   1

     airconditioning_yes  furnishingstatus_furnished  \
0                      1                           1
1                      1                           1
2                      0                           0
3                      1                           1
4                      1                           1
..                   ...                         ...
540                    0                           0
541                    0                           0
542                    0                           0
543                    0                           1
544                    0                           0

     furnishingstatus_semi-furnished  furnishingstatus_unfurnished
0                                  0                             0
1                                  0                             0
2                                  1                             0
3                                  0                             0
4                                  0                             0
..                               ...                           ...
540                                0                             1
541                                1                             0
542                                0                             1
543                                0                             0
```

```
544                                0                                1

[545 rows x 18 columns]

Dependent Variable (Price):
 0      4340000.0
1      4340000.0
2      4340000.0
3      4340000.0
4      4340000.0
         ...
540    1820000.0
541    1767150.0
542    1750000.0
543    1750000.0
544    1750000.0
Name: price, Length: 545, dtype: float64
```

In [37]:

```python
# Select the independent variables columns
independent_vars = df[['area', 'bedrooms', 'bathrooms', 'stories', 'mainroad', 'gue
                       'basement', 'hotwaterheating', 'airconditioning', 'parking',

# Convert categorical variables to one-hot encoding
categorical_vars = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'aircon
encoder = OneHotEncoder(sparse=False, drop='first')
encoded_categorical = pd.DataFrame(encoder.fit_transform(independent_vars[categori

# Concatenate encoded categorical variables with the remaining numeric variables
numeric_vars = independent_vars.drop(columns=categorical_vars)
scaled_df = pd.concat([numeric_vars, encoded_categorical], axis=1)

# Initialize the StandardScaler
scaler = StandardScaler()

# Scale the independent variables
scaled_vars = scaler.fit_transform(scaled_df)

# Convert the scaled variables array back to a DataFrame
scaled_df = pd.DataFrame(scaled_vars, columns=scaled_df.columns)

# Print the scaled data
print(scaled_df)

scaled_df['price']=df['price']
```

```
         area  bedrooms  bathrooms   stories   parking         0         1  \
0    1.334706  1.647621   1.472436  2.213845  1.729065  0.405623 -0.465315
1    2.174467  1.647621  -0.574701  0.567807 -0.790562  0.405623 -0.465315
2    2.719766  0.125666   1.472436  0.567807  1.729065  0.405623 -0.465315
3    1.378330  1.647621   1.472436  0.567807 -0.790562  0.405623 -0.465315
4    1.334706  1.647621  -0.574701  0.567807  1.729065  0.405623  2.149083
..        ...       ...        ...       ...       ...       ...       ...
540 -1.075516 -1.396289  -0.574701 -1.078230  1.729065  0.405623 -0.465315
541 -1.402696  0.125666  -0.574701 -1.078230 -0.790562 -2.465344 -0.465315
542 -0.737431 -1.396289  -0.574701 -1.078230 -0.790562  0.405623 -0.465315
543 -1.124593  0.125666  -0.574701 -1.078230 -0.790562 -2.465344 -0.465315
544 -0.612012  0.125666  -0.574701  0.567807 -0.790562  0.405623 -0.465315

            2         3         4         5         6
0   -0.734539 -0.219265  1.472618 -0.844888 -0.696429
1   -0.734539 -0.219265  1.472618 -0.844888 -0.696429
2    1.361397 -0.219265 -0.679063  1.183588 -0.696429
3    1.361397 -0.219265  1.472618 -0.844888 -0.696429
4    1.361397 -0.219265  1.472618 -0.844888 -0.696429
..        ...       ...       ...       ...       ...
540  1.361397 -0.219265 -0.679063 -0.844888  1.435896
541 -0.734539 -0.219265 -0.679063  1.183588 -0.696429
542 -0.734539 -0.219265 -0.679063 -0.844888  1.435896
543 -0.734539 -0.219265 -0.679063 -0.844888 -0.696429
544 -0.734539 -0.219265 -0.679063 -0.844888  1.435896

[545 rows x 12 columns]
D:\anaconda\lib\site-packages\sklearn\utils\validation.py:1688: FutureWarning: Fea
ture names only support names that are all strings. Got feature names with dtypes:
['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
D:\anaconda\lib\site-packages\sklearn\utils\validation.py:1688: FutureWarning: Fea
ture names only support names that are all strings. Got feature names with dtypes:
['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

In [38]:
```
# Split the data into dependent and independent variables
X = scaled_df.drop('price', axis=1)  # Independent variables (features)
y = scaled_df['price']  # Dependent variable (target)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_st

# Print the shapes of the training and testing sets
print("Training set shape:", X_train.shape, y_train.shape)
print("Testing set shape:", X_test.shape, y_test.shape)
```

```
Training set shape: (381, 12) (381,)
Testing set shape: (164, 12) (164,)
```

In [39]:
```
model = LinearRegression()
```

In [40]:
```
scaled_df.head()
```

Out[40]:

	area	bedrooms	bathrooms	stories	parking	0	1	2	3
0	1.334706	1.647621	1.472436	2.213845	1.729065	0.405623	-0.465315	-0.734539	-0.219265
1	2.174467	1.647621	-0.574701	0.567807	-0.790562	0.405623	-0.465315	-0.734539	-0.219265
2	2.719766	0.125666	1.472436	0.567807	1.729065	0.405623	-0.465315	1.361397	-0.219265
3	1.378330	1.647621	1.472436	0.567807	-0.790562	0.405623	-0.465315	1.361397	-0.219265
4	1.334706	1.647621	-0.574701	0.567807	1.729065	0.405623	2.149083	1.361397	-0.219265

In [41]:
```
# Step 12: Train the Model
model.fit(X_train, y_train)
```

```
D:\anaconda\lib\site-packages\sklearn\utils\validation.py:1688: FutureWarning: Fea
ture names only support names that are all strings. Got feature names with dtypes:
['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

Out[41]:
```
LinearRegression()
```

In [42]:
```
# Step 13: Test the Model
y_pred = model.predict(X_test)
```

```
D:\anaconda\lib\site-packages\sklearn\utils\validation.py:1688: FutureWarning: Fea
ture names only support names that are all strings. Got feature names with dtypes:
['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

In [43]:
```
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 1505585445043.8713
```

In [44]:
```
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)
print("Mean Absolute Error (MAE):", mae)
print("Root Mean Squared Error (RMSE):", rmse)
print("R-squared (R2) Score:", r2)
```

```
Mean Squared Error (MSE): 1505585445043.8713
Mean Absolute Error (MAE): 944620.122197702
Root Mean Squared Error (RMSE): 1227023.0010247857
R-squared (R2) Score: 0.4704364397336236
```

In [45]:
```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor()


model.fit(X_train, y_train)


y_pred = model.predict(X_test)


mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
```

```
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)
print("Mean Absolute Error (MAE):", mae)
print("Root Mean Squared Error (RMSE):", rmse)
print("R-squared (R2) Score:", r2)
```

```
D:\anaconda\lib\site-packages\sklearn\utils\validation.py:1688: FutureWarning: Fea
ture names only support names that are all strings. Got feature names with dtypes:
['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
D:\anaconda\lib\site-packages\sklearn\utils\validation.py:1688: FutureWarning: Fea
ture names only support names that are all strings. Got feature names with dtypes:
['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
Mean Squared Error (MSE): 1396326934549.9744
Mean Absolute Error (MAE): 903560.6701219515
Root Mean Squared Error (RMSE): 1181662.7837712306
R-squared (R2) Score: 0.5088662253010999
```

In [ ]: