SmartInternz Externships

Stream: Applied Data Science

Week: 3

Done By: Adith Sreeram A S

Reg No: 20BCD7134

Campus: VIT-AP

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

2) Load the dataset

In [2]:
```
df = pd.read_csv("housing.csv")
```

In [3]:
```
df.head()
```

Out[3]:

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

In [4]:
```
df.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
```

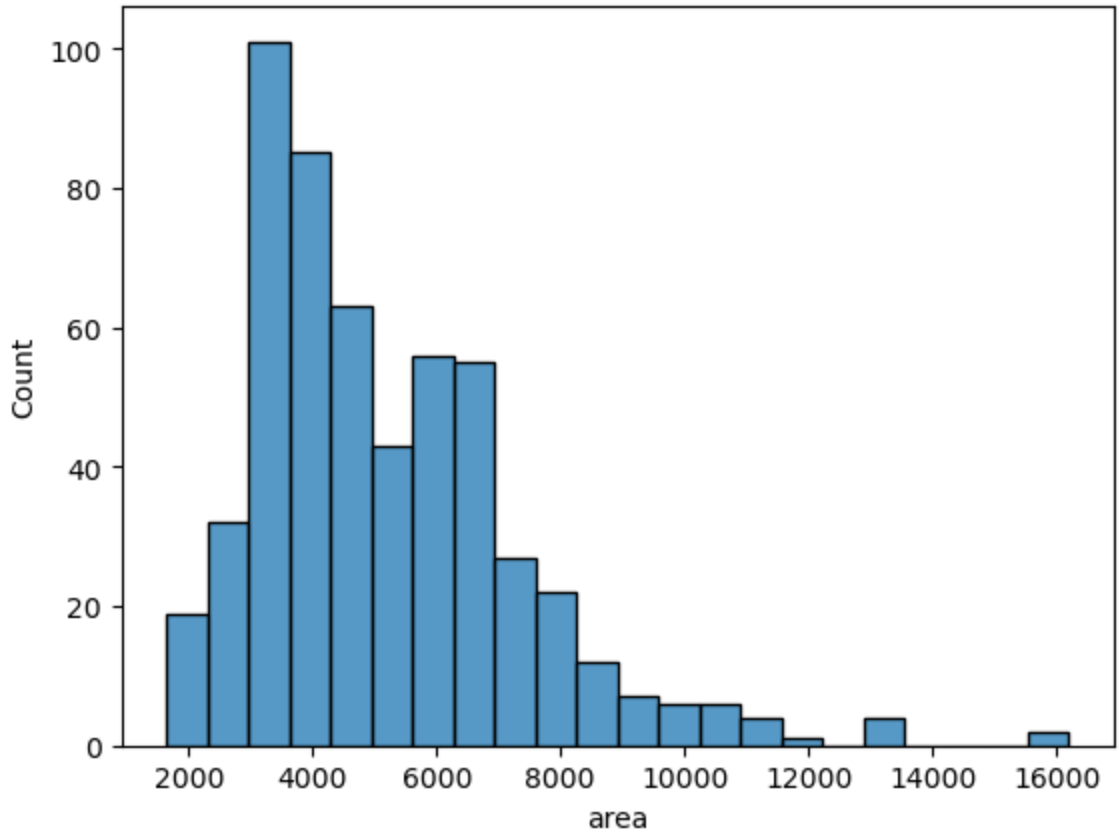
3. Perform Below Visualizations.

- Univariate Analysis
- Bi - Variate Analysis
- Multi - Variate Analysis

univariate analysis

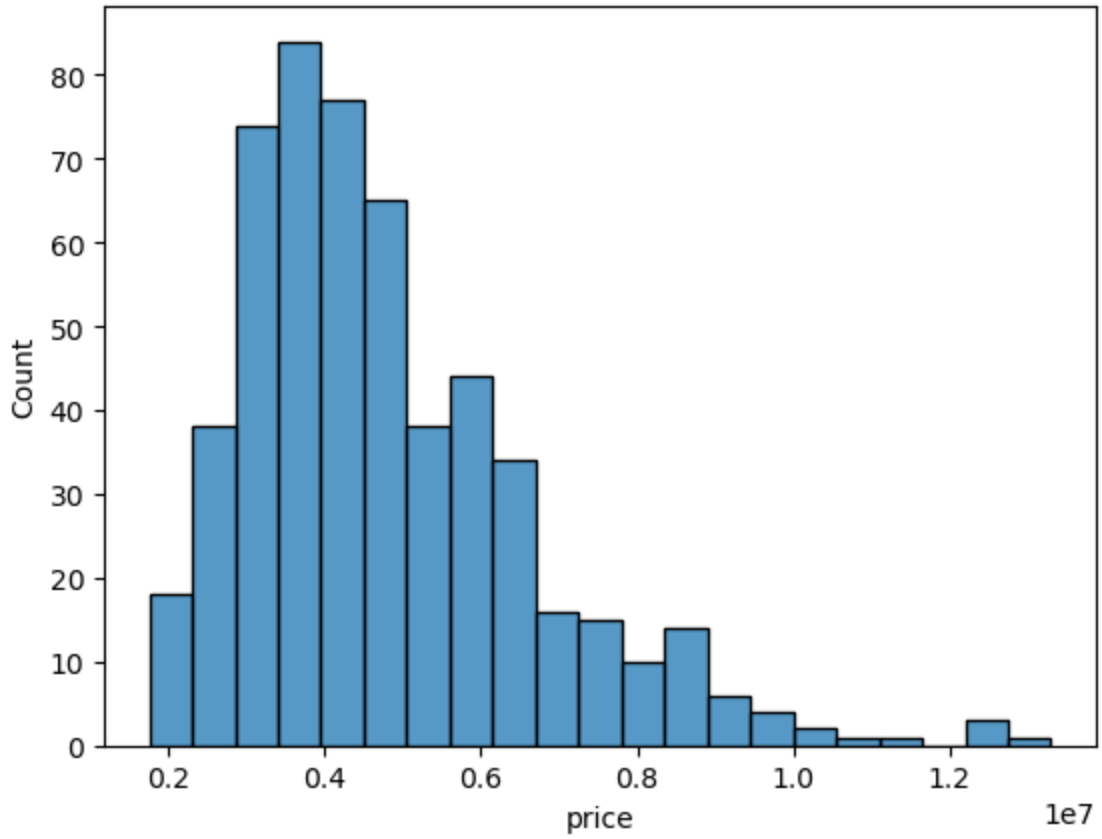
In [5]:
```
sns.histplot(df['area'])
```

Out[5]:
```
<Axes: xlabel='area', ylabel='Count'>
```



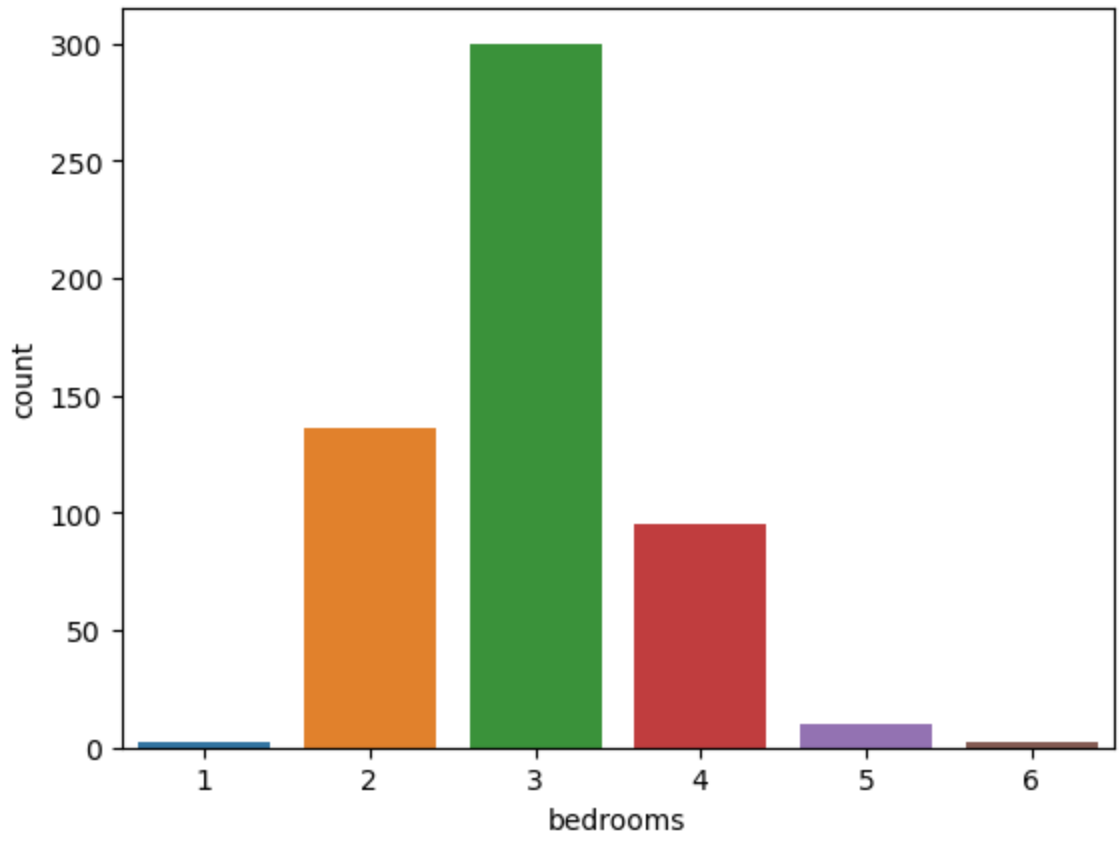
In [6]:
```
sns.histplot(df['price'])
```

Out[6]:
```
<Axes: xlabel='price', ylabel='Count'>
```



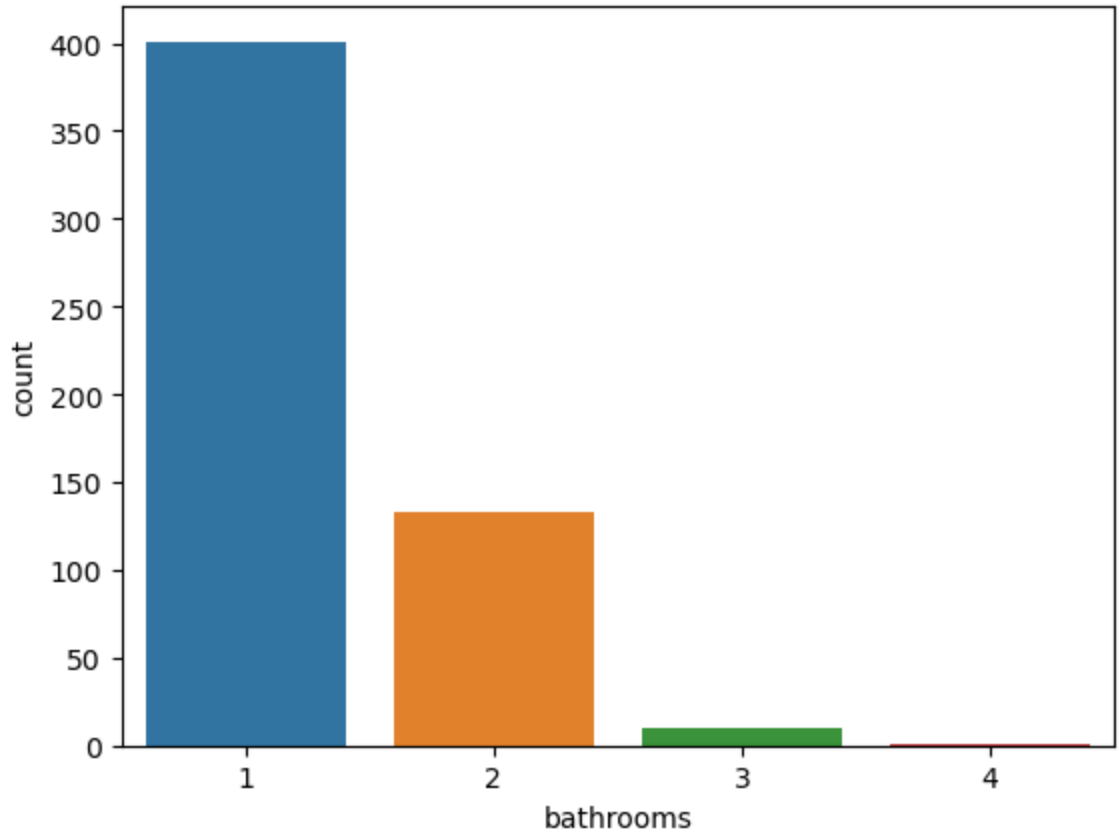
In [7]:
```
sns.countplot(x = df['bedrooms'])
```

Out[7]:
```
<Axes: xlabel='bedrooms', ylabel='count'>
```



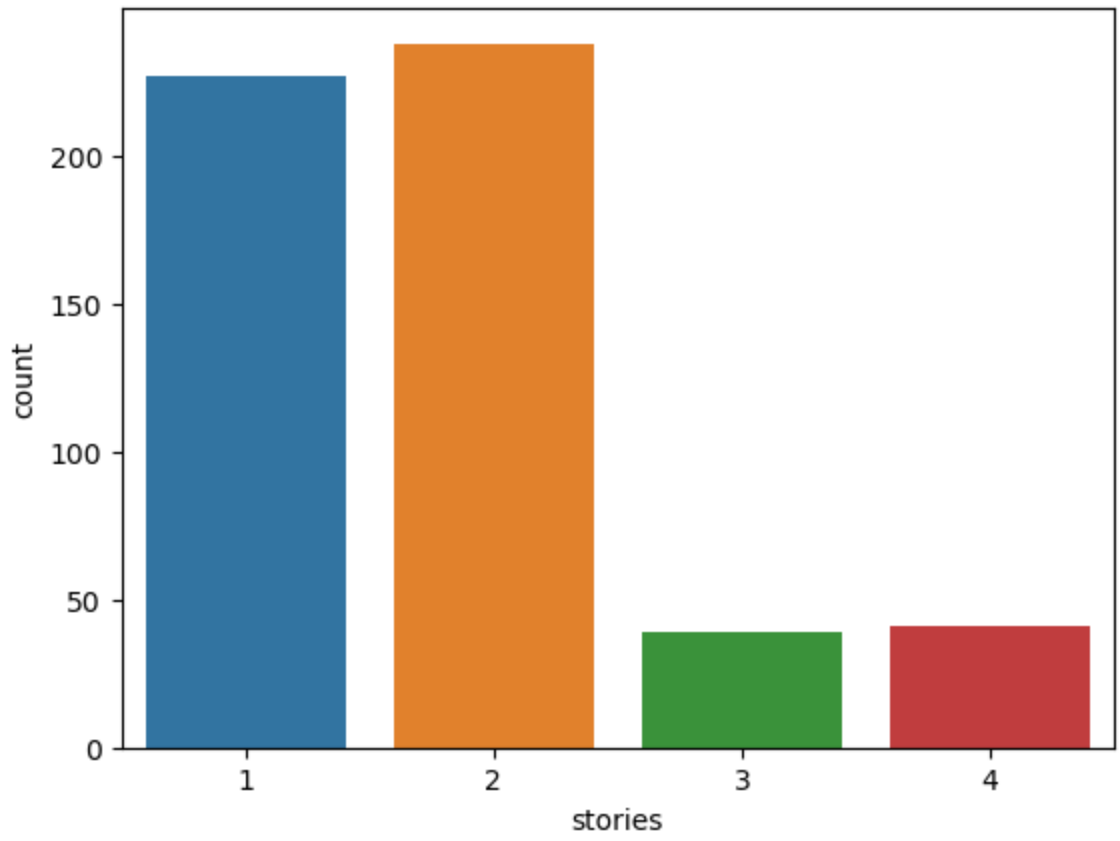
In [8]:
```
sns.countplot(x = df['bathrooms'])
```

Out[8]:
```
<Axes: xlabel='bathrooms', ylabel='count'>
```



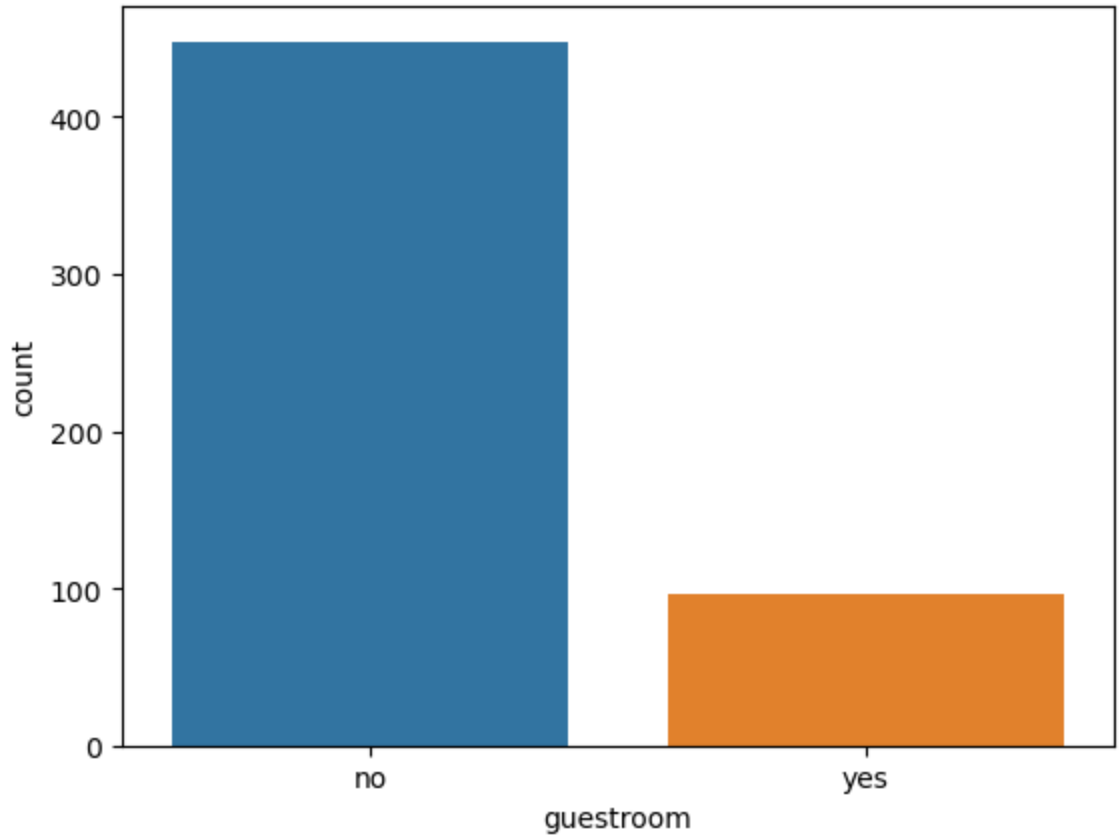
In [9]:
```
sns.countplot(x = df['stories'])
```

Out[9]:
```
<Axes: xlabel='stories', ylabel='count'>
```



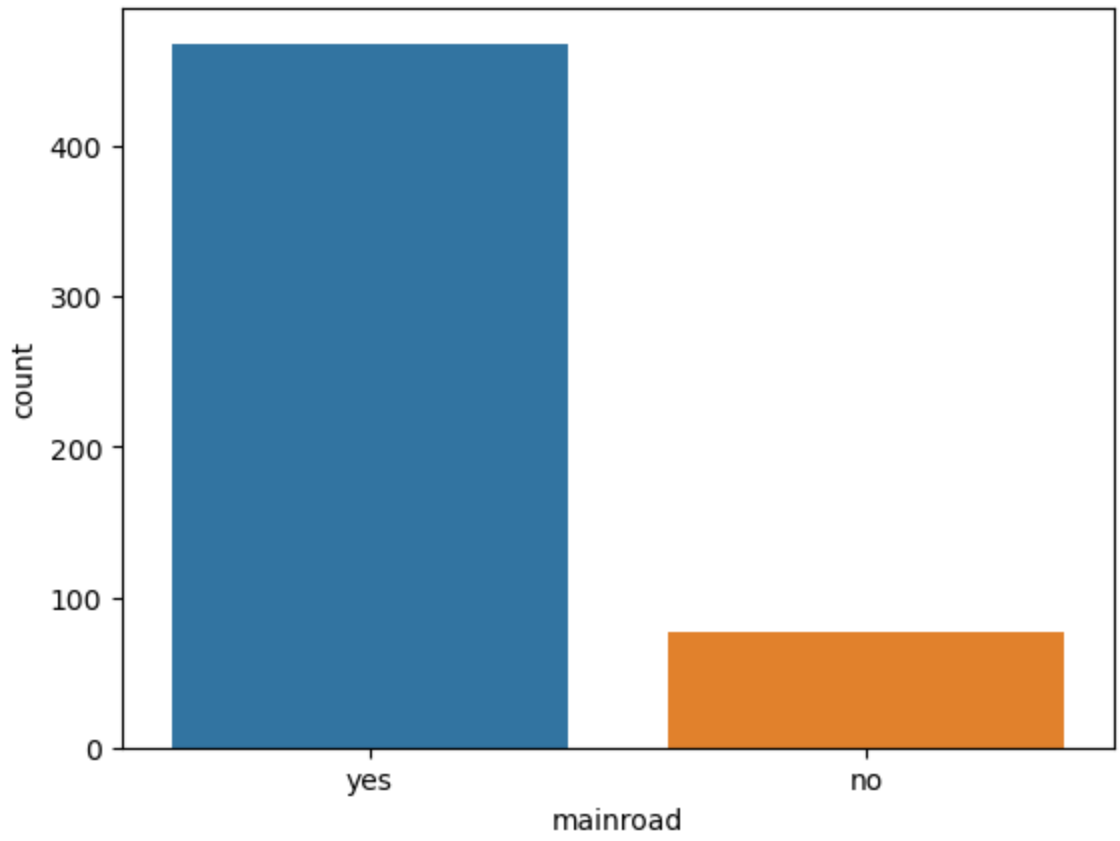
In [10]:
```
sns.countplot(x = df['guestroom'])
```

Out[10]:
```
<Axes: xlabel='guestroom', ylabel='count'>
```



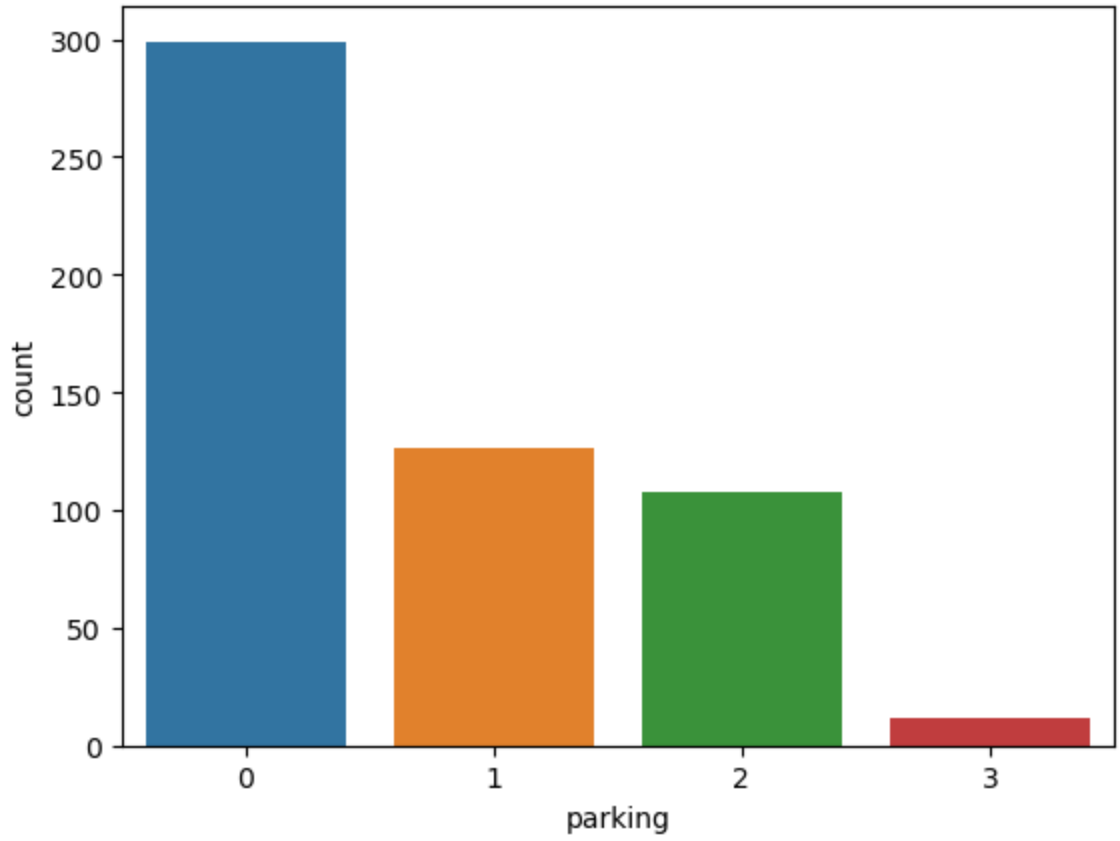
In [11]:
```
sns.countplot(x = df['mainroad'])
```

Out[11]:
```
<Axes: xlabel='mainroad', ylabel='count'>
```



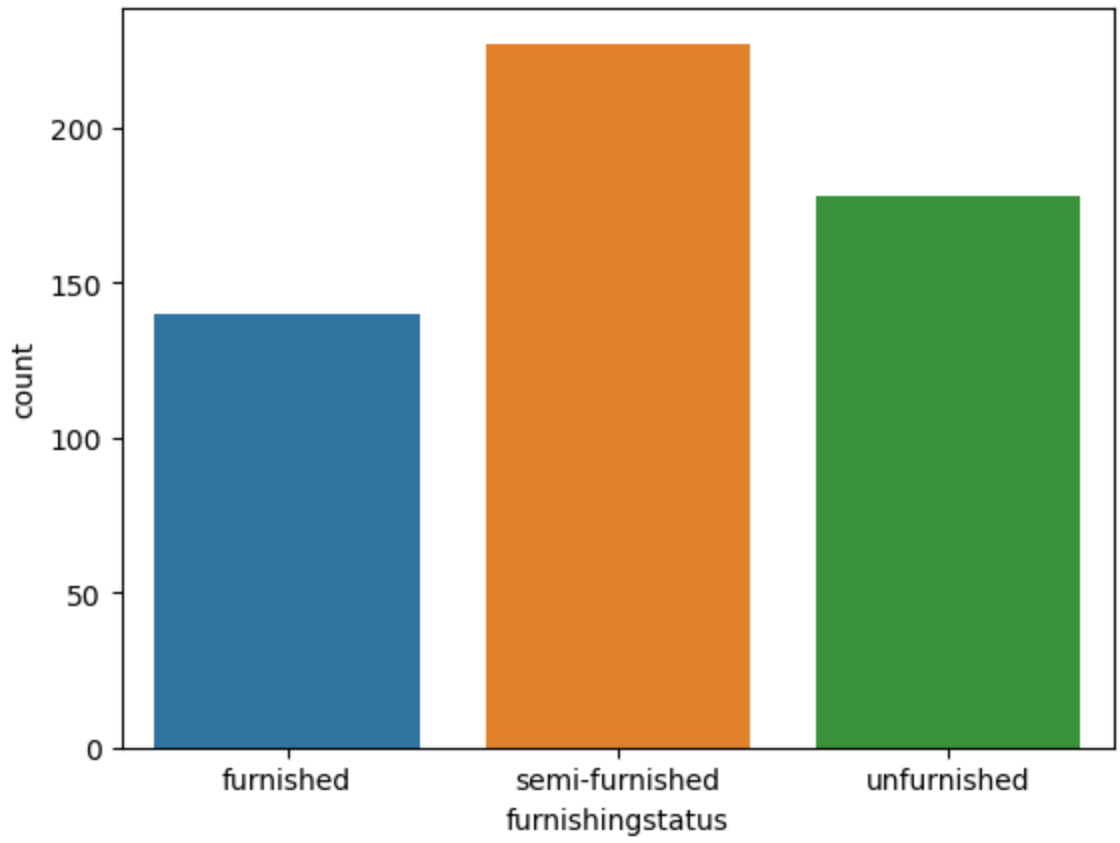
In [12]:
```
sns.countplot(x = df['parking'])
```

Out[12]:
```
<Axes: xlabel='parking', ylabel='count'>
```



In [13]:
```
sns.countplot(x = df['furnishingstatus'])
```

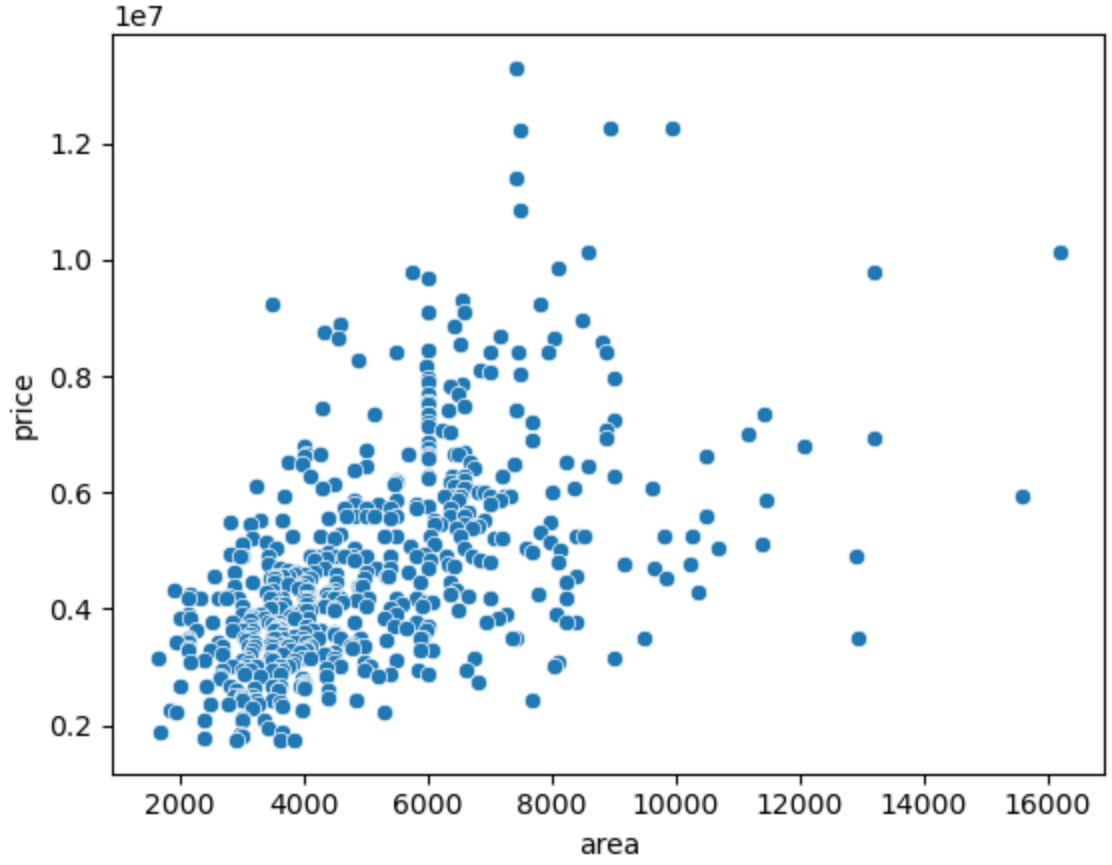
Out[13]:
```
<Axes: xlabel='furnishingstatus', ylabel='count'>
```



bivariate analysis

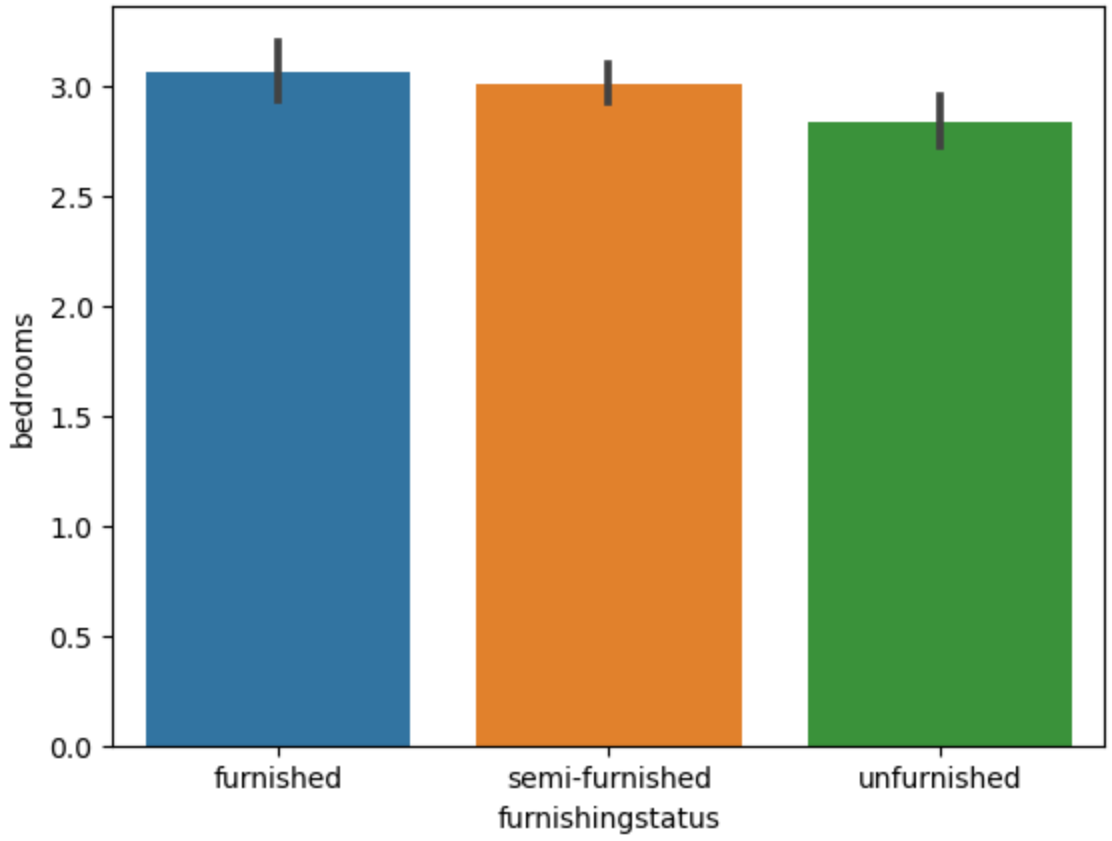
In [14]:
```
sns.scatterplot(data = df, x = 'area', y = 'price')
```

Out[14]:
```
<Axes: xlabel='area', ylabel='price'>
```



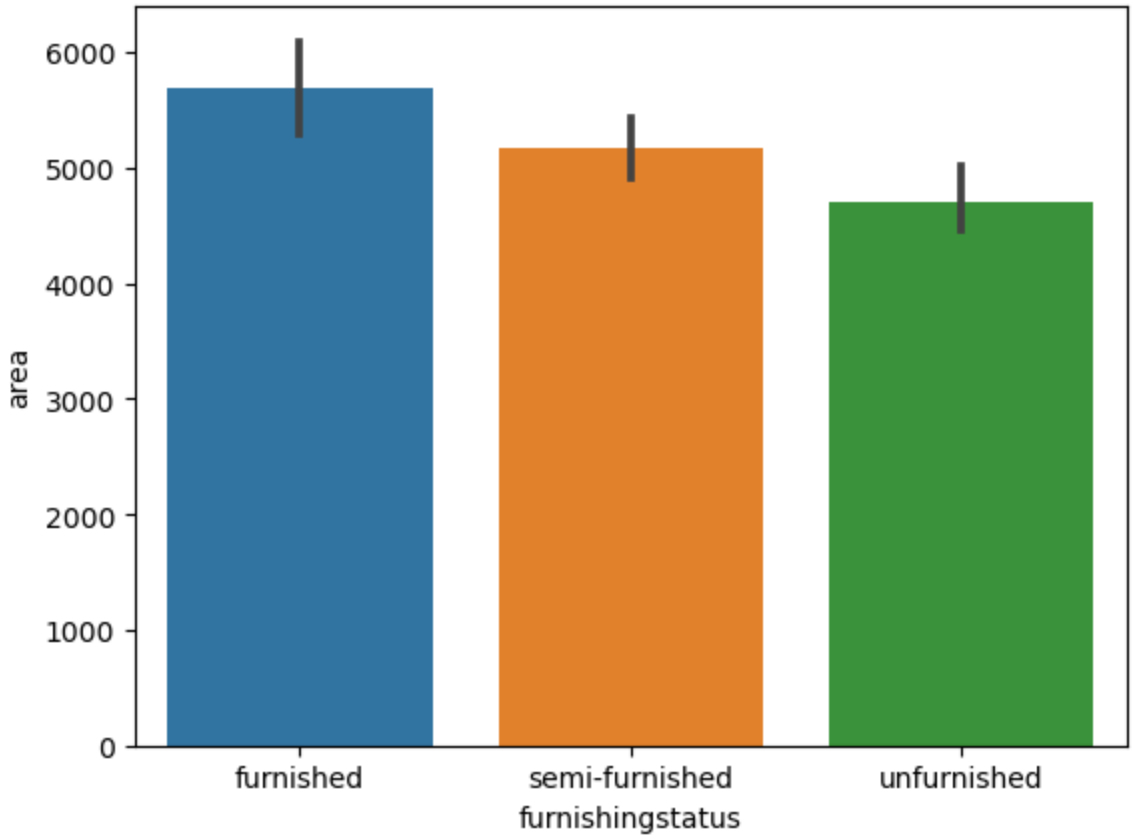
In [15]:
```
sns.barplot(data = df, x = 'furnishingstatus', y = 'bedrooms')
```

Out[15]:
```
<Axes: xlabel='furnishingstatus', ylabel='bedrooms'>
```



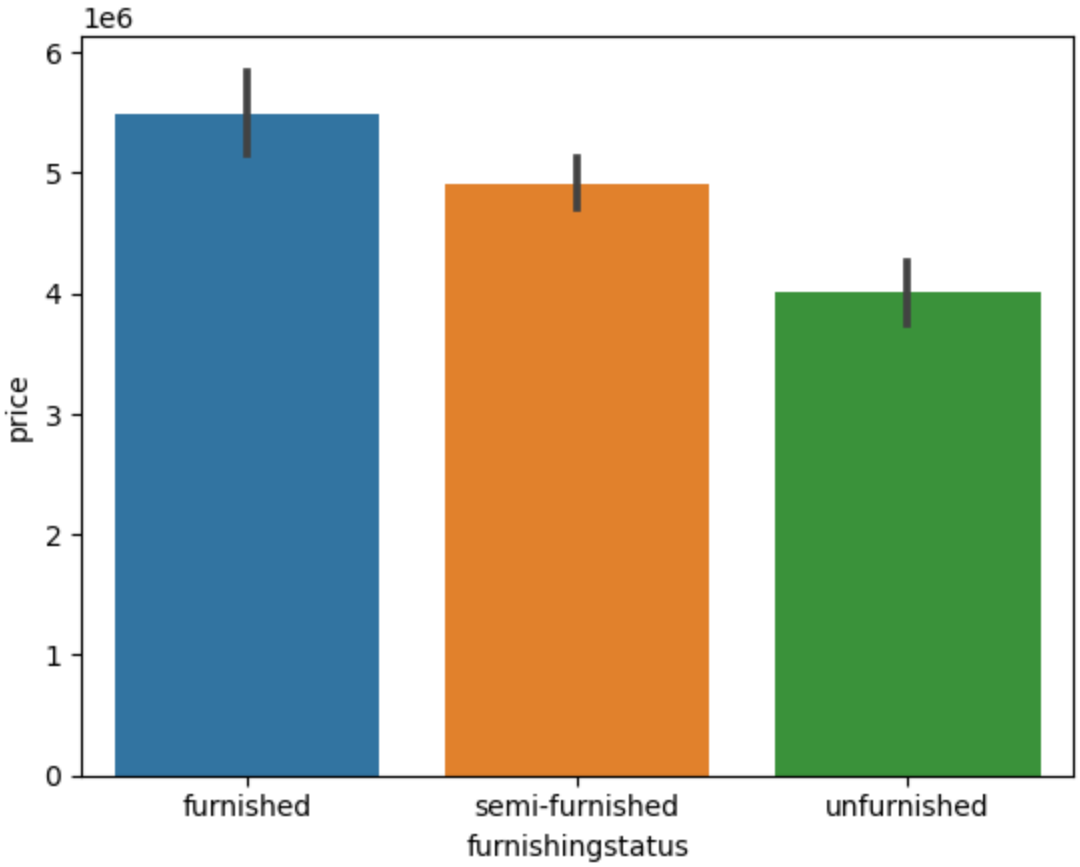
In [16]:
```
sns.barplot(data = df, x = 'furnishingstatus', y = 'area')
```

Out[16]:
```
<Axes: xlabel='furnishingstatus', ylabel='area'>
```



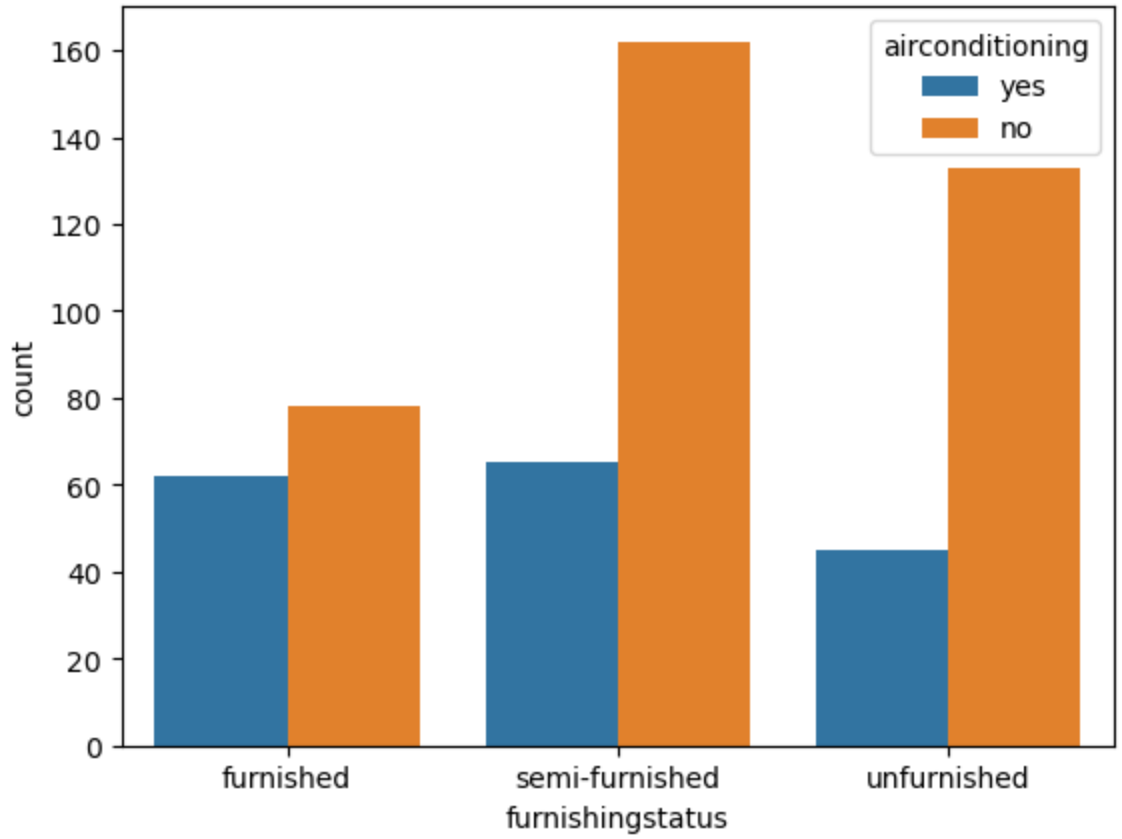
In [17]:
```
sns.barplot(data = df, x = 'furnishingstatus', y = 'price')
```

Out[17]:
```
<Axes: xlabel='furnishingstatus', ylabel='price'>
```



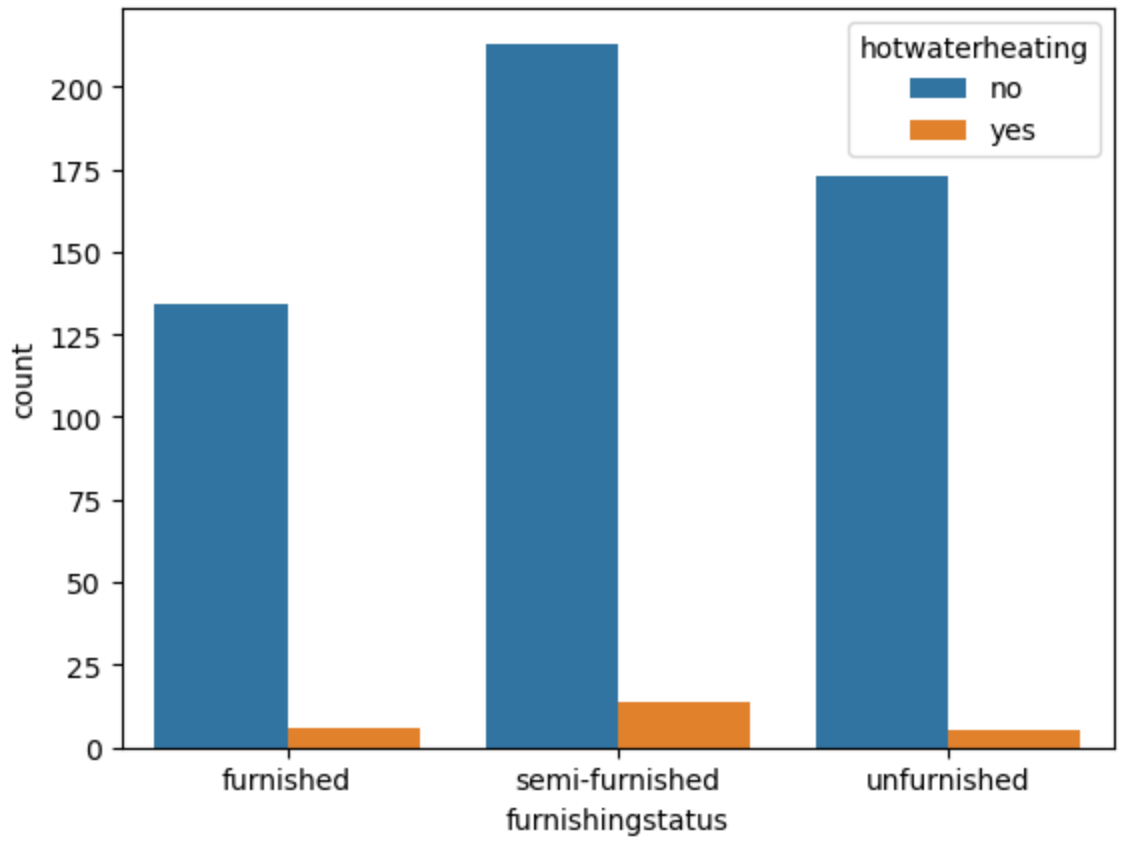
In [18]:
```
sns.countplot(x = df['furnishingstatus'], hue = df['airconditioning'])
```

Out[18]:
```
<Axes: xlabel='furnishingstatus', ylabel='count'>
```



In [19]:
```
sns.countplot(x = df['furnishingstatus'], hue = df['hotwaterheating'])
```

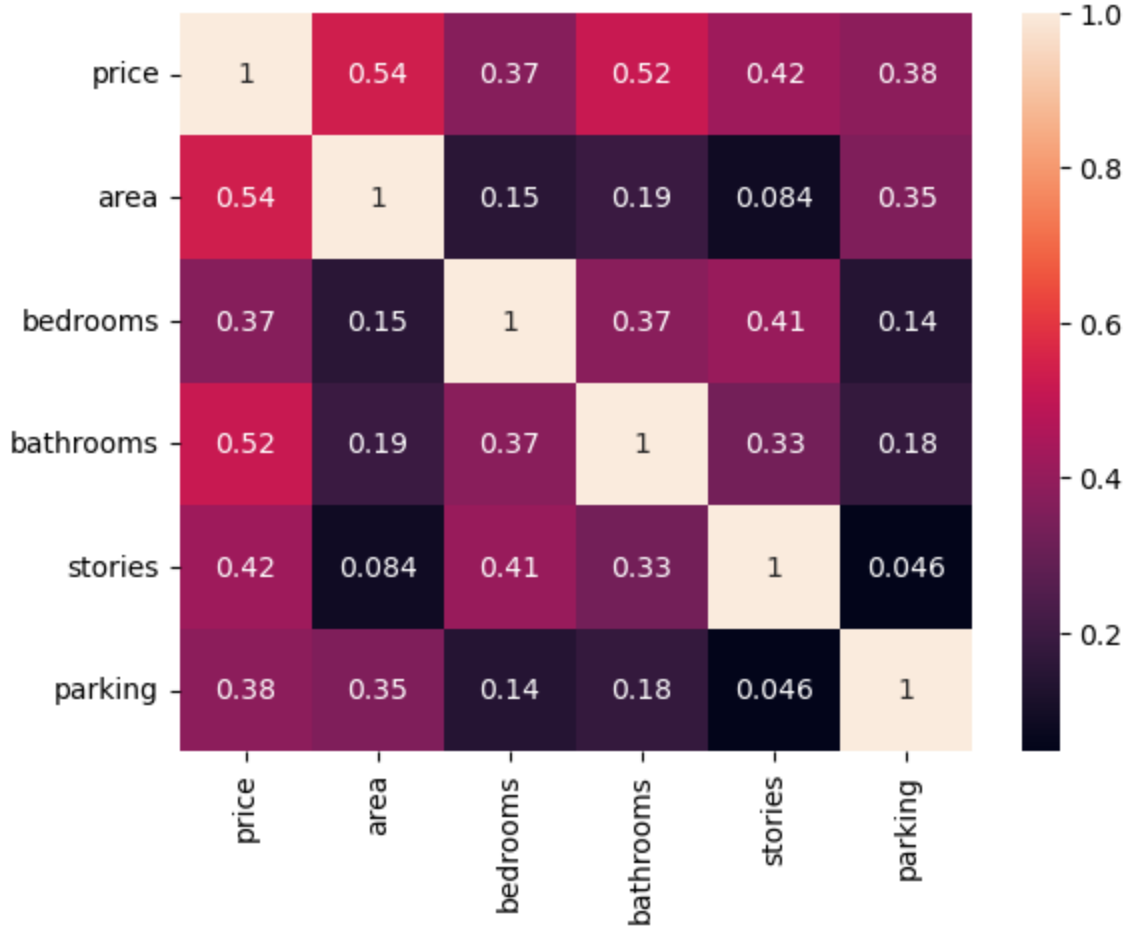
Out[19]:
```
<Axes: xlabel='furnishingstatus', ylabel='count'>
```



multivariate analysis

In [20]:
```
sns.heatmap(df.corr(numeric_only=True), annot = True)
```

Out[20]:
```
<Axes: >
```



4. Perform descriptive statistics on the dataset.

In [21]:
```
df.describe()
```

Out[21]:

	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

5. Handle the Missing values.

In [22]:
```
df.isnull().sum()
```

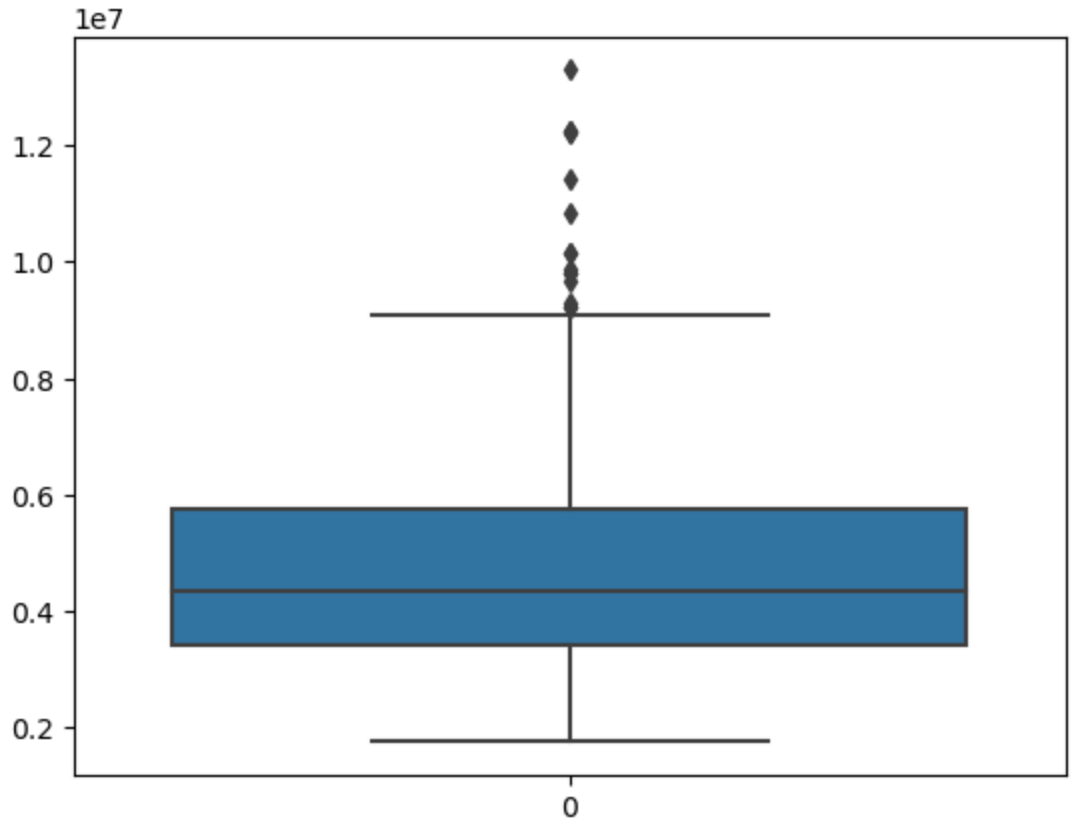
Out[22]:
```
price        0
area         0
bedrooms     0
bathrooms    0
stories      0
```

```
mainroad            0
guestroom           0
basement            0
hotwaterheating     0
airconditioning     0
parking             0
furnishingstatus    0
dtype: int64
```

6. Find the outliers and replace the outliers

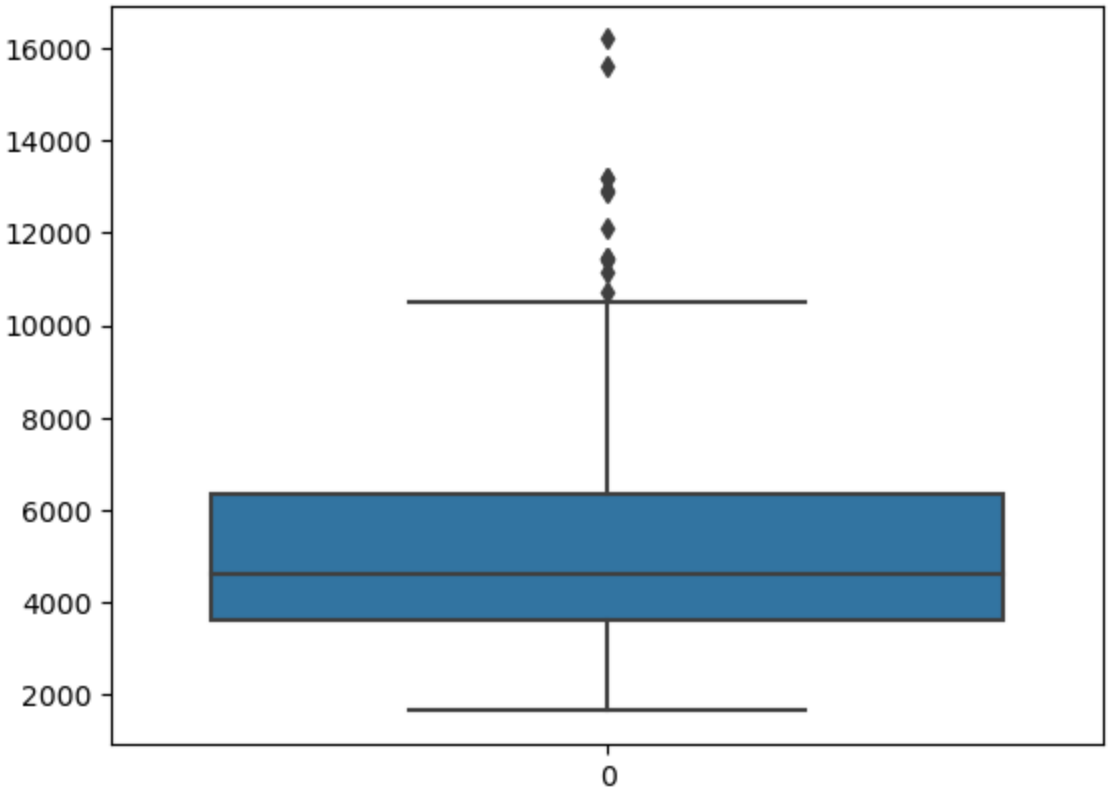
In [23]:
```
sns.boxplot(df['price'])
```

Out[23]:
```
<Axes: >
```



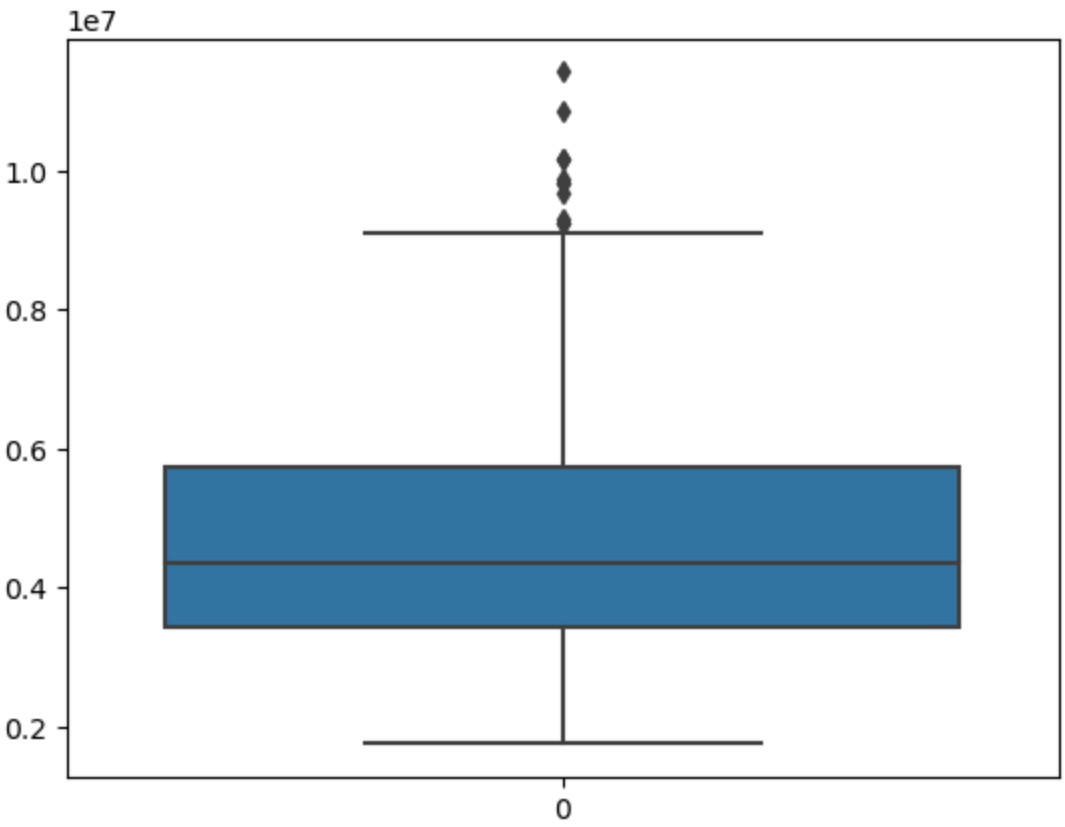
In [24]:
```
sns.boxplot(df['area'])
```

Out[24]:
```
<Axes: >
```



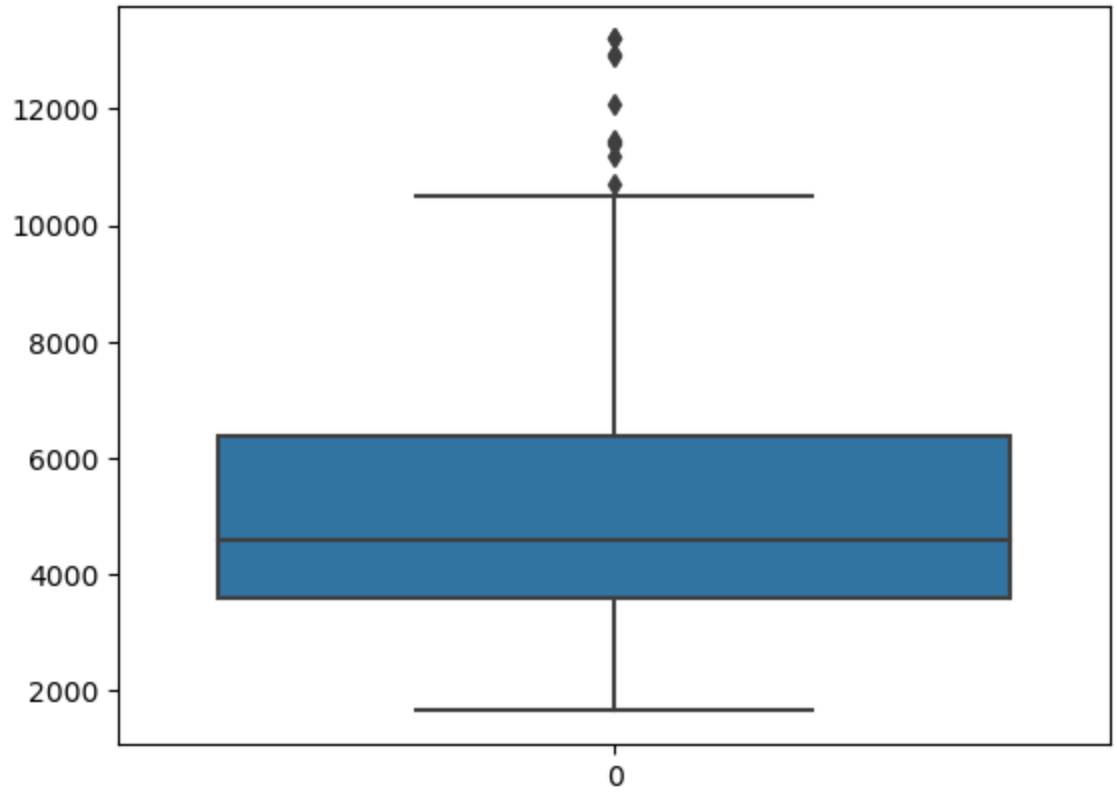
In [25]:
```
median_age = df['price'].median()
df["price"] = np.where(df["price"] >12000000, median_age, df['price'])
sns.boxplot(df['price'])
```

Out[25]:
```
<Axes: >
```



In [26]:
```
median_area = df['area'].median()
df["area"] = np.where(df["area"] > 14000, median_area, df['area'])
sns.boxplot(df['area'])
```

Out[26]:
```
<Axes: >
```



7. Check for Categorical columns and perform encoding.

In [27]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [28]:
```
encoding = pd.get_dummies(df, columns = ['mainroad', 'guestroom', 'basement','hotwaterhe
                                         'airconditioning', 'furnishingstatus'])
```

In [29]:
```
encoding.head()
```

Out[29]:

	price	area	bedrooms	bathrooms	stories	parking	mainroad_no	mainroad_yes	guestroom_no	guest
0	4340000.0	7420.0	4	2	3	2	0	1	1	
1	4340000.0	8960.0	4	4	4	3	0	1	1	
2	4340000.0	9960.0	3	2	2	2	0	1	1	
3	4340000.0	7500.0	4	2	2	3	0	1	1	
4	11410000.0	7420.0	4	1	2	2	0	1	0	

8. Split the data into dependent and independent variables

In [30]:
```
df.columns
```

Out[30]:
```
Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
       'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
       'parking', 'furnishingstatus'],
      dtype='object')
```

In [65]:
```
# independent variables
X = encoding.drop(['price'], axis = 1)
X.head()
```

Out[65]:

	area	bedrooms	bathrooms	stories	parking	mainroad_no	mainroad_yes	guestroom_no	guestroom_yes	b
0	7420.0	4	2	3	2	0	1	1	0	
1	8960.0	4	4	4	3	0	1	1	0	
2	9960.0	3	2	2	2	0	1	1	0	
3	7500.0	4	2	2	3	0	1	1	0	
4	7420.0	4	1	2	2	0	1	0	1	

In [66]:
```
# dependent variables
y = df[['price']]
y.head()
```

Out[66]:

	price
0	4340000.0
1	4340000.0
2	4340000.0
3	4340000.0
4	11410000.0

9. Scaling the independent variables

In [67]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_std = scaler.fit_transform(X)
```

In [68]:
```
x_std
```

Out[68]:
```
array([[ 1.11756482,  1.40341936,  1.42181174, ...,  1.70084013,
        -0.84488844, -0.6964292 ],
       [ 1.8623093 ,  1.40341936,  5.40580863, ...,  1.70084013,
        -0.84488844, -0.6964292 ],
       [ 2.34590961,  0.04727831,  1.42181174, ..., -0.58794474,
         1.18358821, -0.6964292 ],
       ...,
       [-0.72011635, -1.30886273, -0.57018671, ..., -0.58794474,
        -0.84488844,  1.43589615],
       [-1.06347257,  0.04727831, -0.57018671, ...,  1.70084013,
        -0.84488844, -0.6964292 ],
       [-0.60888828,  0.04727831, -0.57018671, ..., -0.58794474,
        -0.84488844,  1.43589615]])
```

10. Split the data into training and testing

In [69]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=0
```

11. Build the Model

In [70]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [71]:
```
lr = LinearRegression()
```

In [ ]:
```

```

12. Train the Model

In [72]:
```
lr.fit( X_train, y_train )
```

Out[72]:
```
LinearRegression()
```

In [73]:
```
print("Value of the coefficients: \n", lr.coef_)
print("-----------------------------------")
print("Value of the intercept: \n", lr.intercept_)
```

```
Value of the coefficients:
 [[ 2.60781675e+02  9.30932038e+04  8.20605321e+05  3.96961106e+05
   1.00796216e+05 -3.10469714e+05  3.10469714e+05 -3.32132424e+05
   3.32132424e+05 -1.72635846e+05  1.72635846e+05 -7.33905647e+05
   7.33905647e+05 -4.83024979e+05  4.83024979e+05  1.04827468e+05
   1.38634062e+05 -2.43461530e+05]]
-----------------------------------
Value of the intercept:
 [2156139.9017023]
```

13. Test the Model

In [74]:
```
Y_pred = lr.predict(X_test)
```

14. Measure the performance using Metrics.

In [81]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [82]:
```
print("MSE",mean_squared_error(y_test,Y_pred))
print("-----------------------------------")
print("RMSE",np.sqrt(mean_squared_error(y_test,Y_pred)))
print("-----------------------------------")
print("R-Square", r2_score(y_test,Y_pred))
```

```
MSE 1326521791171.129
-----------------------------------
RMSE 1151747.2774750236
-----------------------------------
R-Square 0.5718914765881087
```

In [ ]:
```

```