House Price

Goal of the project

predict the price of a house by its features.if you are buyer or seller of the house but you do not know the exact price of the house , so machine learning algorithms can help to predict the price of the house just providing features of the target house.

About Dataset

This dataset provides comprehensive information for house price prediction, with 13 column names:

Price: The price of the house.
Area: The total area of the house in square feet.
Bedrooms: The number of bedrooms in the house.
Bathrooms: The number of bathrooms in the house.
Stories: The number of stories in the house.
Mainroad: Whether the house is connected to the main road (Yes/No).
Guestroom: Whether the house has a guest room (Yes/No).
Basement: Whether the house has a basement (Yes/No).
Hot water heating: Whether the house has a hot water heating system (Yes/No).
Airconditioning: Whether the house has an air conditioning system (Yes/No).
Parking: The number of parking spaces available within the house.
Prefarea: Whether the house is located in a preferred area (Yes/No).
Furnishing status: The furnishing status of the house (Fully Furnished, Semi-Furnished, Unfurnished).

In [ ]:
```
import numpy as np
```

In [ ]:
```
import pandas as pd
```

In [ ]:
```
df=pd.read_csv(r'C:\Users\hp\Desktop\Housing.csv')
```

In [211]:
```
df
```

Out[211]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwat
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	
...	...	...	...	...	...	...	...	...	
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	

545 rows × 13 columns

In [212]:
```
df.shape
```

Out[212]:
```
(545, 13)
```

In [213]:
```
df.sample(5)
```

Out[213]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwate
360	3710000	4040	2	1	1	yes	no	no	
524	2380000	3264	2	1	1	yes	no	no	
45	7560000	6000	3	2	3	yes	no	no	
104	6195000	5500	3	2	1	yes	yes	yes	
177	5243000	6050	3	1	1	yes	no	yes	

In [214]:
```
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
```

The column 'price' is numeric.
The column 'area(m2)' is numeric.
The column 'bedrooms' is numeric.
The column 'bathrooms' is numeric.
The column 'stories' is numeric.
The column 'mainroad' is catagorical
. The column 'guestroom' is catagorical.
The column 'basement' is catagorical.
The column 'hotwaterheating' is catagorical.
The column 'airconditioning' is catagorical.
The column 'parking' is numeric.
The column 'prefarea' is catagorical.
The column 'furnishingstatus' is catagorical.

In [215]:
```
df.describe()
```

Out[215]:

	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 [216]:
```
df.isnull().sum()
```

Out[216]:
```
price             0
area              0
bedrooms          0
bathrooms         0
stories           0
mainroad          0
guestroom         0
basement          0
hotwaterheating   0
airconditioning   0
parking           0
prefarea          0
furnishingstatus  0
dtype: int64
```

thers is no missing in this data set

In [217]:
```
df.duplicated().sum()
```

Out[217]:
```
0
```

there is no duplicate in this data set

In [218]:
```
df.corr()
```

Out[218]:

	price	area	bedrooms	bathrooms	stories	parking
price	1.000000	0.535997	0.366494	0.517545	0.420712	0.384394
area	0.535997	1.000000	0.151858	0.193820	0.083996	0.352980
bedrooms	0.366494	0.151858	1.000000	0.373930	0.408564	0.139270
bathrooms	0.517545	0.193820	0.373930	1.000000	0.326165	0.177496
stories	0.420712	0.083996	0.408564	0.326165	1.000000	0.045547
parking	0.384394	0.352980	0.139270	0.177496	0.045547	1.000000

correlation

price most strongly cor realtion with area
area most strongly cor realtion with price
bedrooms most strongly cor relation with stories
bathrooms most strongly cor realtion with price
stories most strongly cor realtion with parking
parkings most strongs cor realtion with price

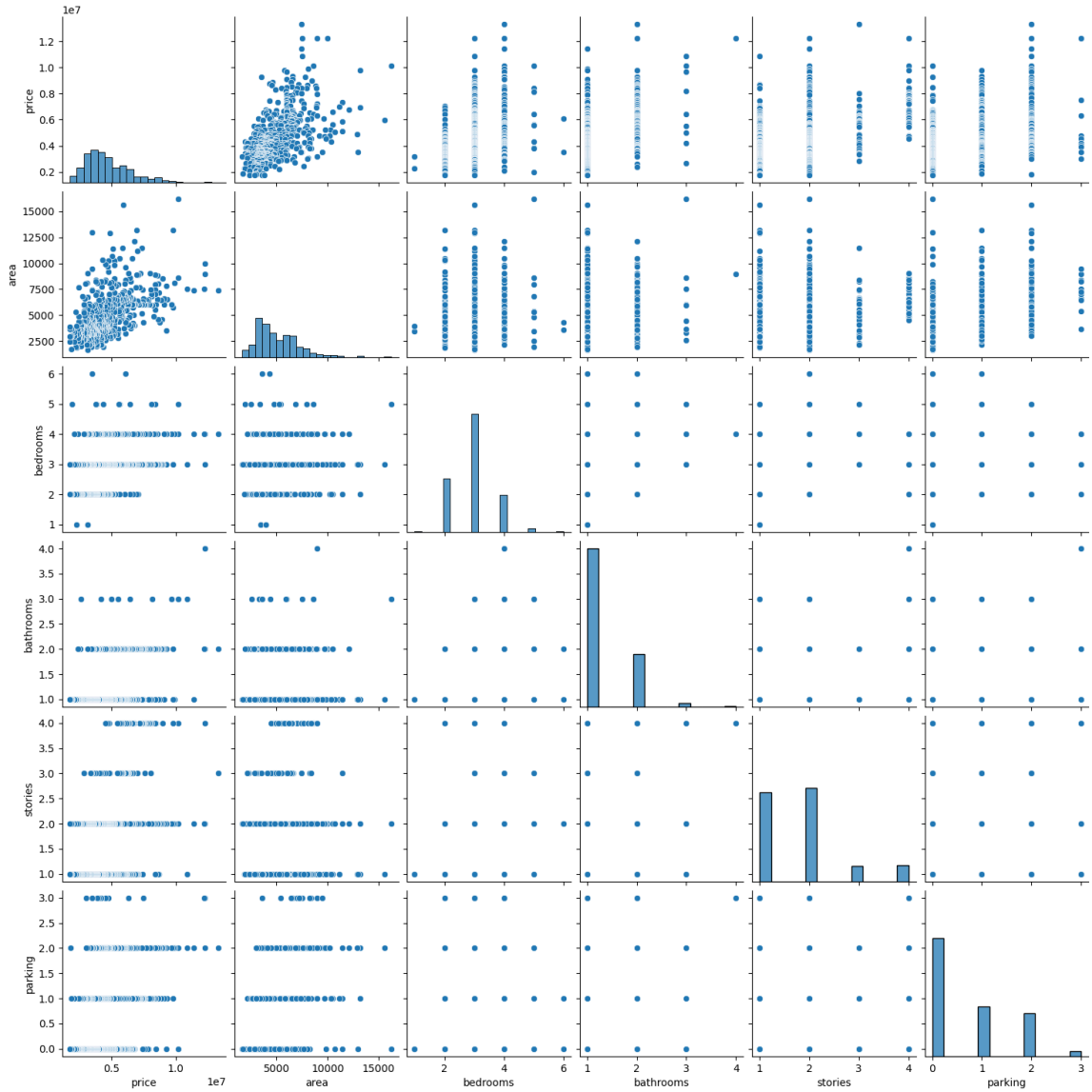
EDA

In [219]:
```
import matplotlib.pyplot as plt
```

In [220]:
```
import seaborn as sns
```

In [221]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [222]:
```
sns.pairplot(df)
plt.show()
```



correlation

In [266]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[266]: <AxesSubplot:>

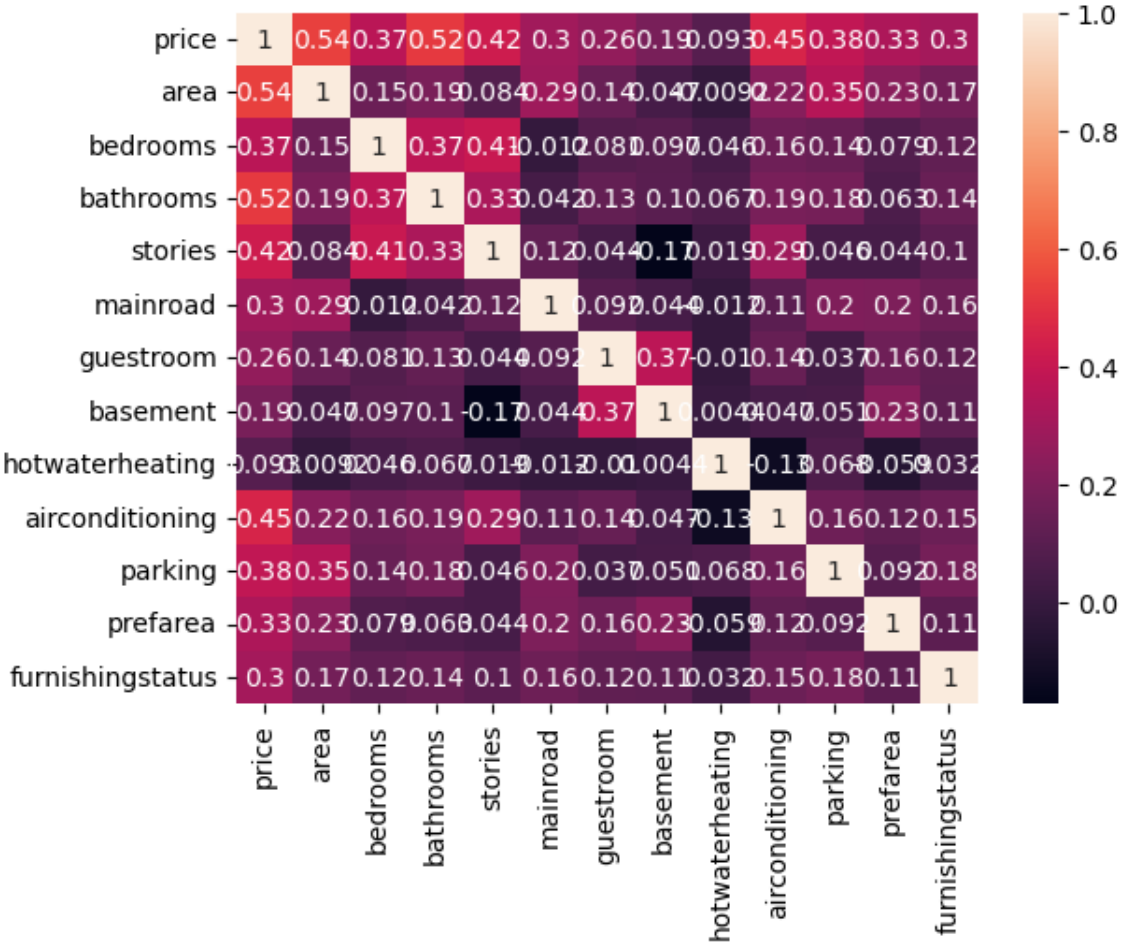
price most strongly cor realtion with area
area most strongly cor realtion with price
bedrooms most strongly cor relation with stories
bathrooms most strongly cor realtion with price
stories most strongly cor realtion with parking
parkings most strongs cor realtion with price

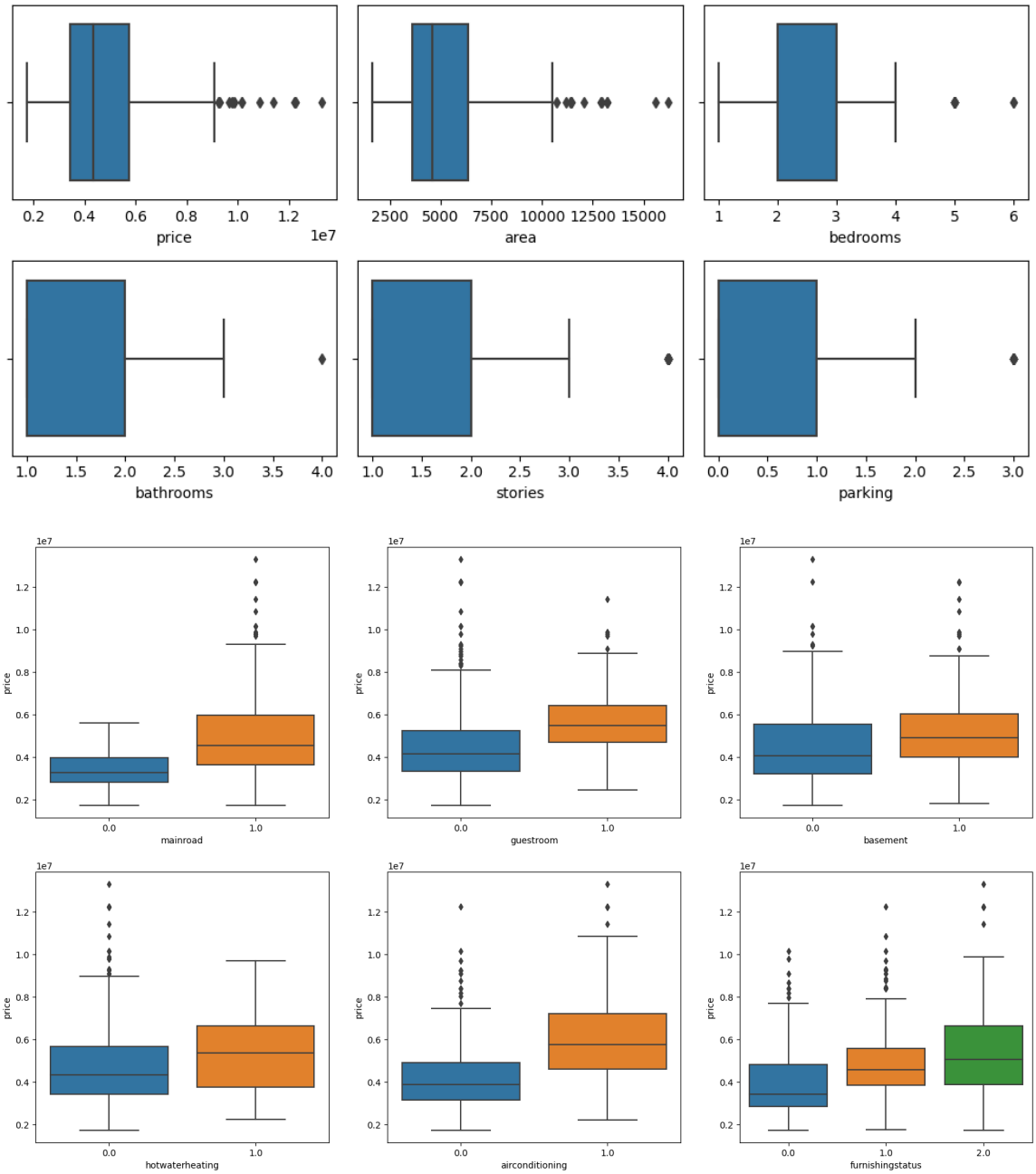
finding outlier in numerical columns in the data set ?

In [267]:
```
# Outlier Analysis
fig, axs = plt.subplots(2,3, figsize = (10,5))
plt1 = sns.boxplot(df['price'], ax = axs[0,0])
plt2 = sns.boxplot(df['area'], ax = axs[0,1])
plt3 = sns.boxplot(df['bedrooms'], ax = axs[0,2])
plt1 = sns.boxplot(df['bathrooms'], ax = axs[1,0])
plt2 = sns.boxplot(df['stories'], ax = axs[1,1])
plt3 = sns.boxplot(df['parking'], ax = axs[1,2])

plt.tight_layout()
```

finding outlier of catagorical columns with respect to house price column?

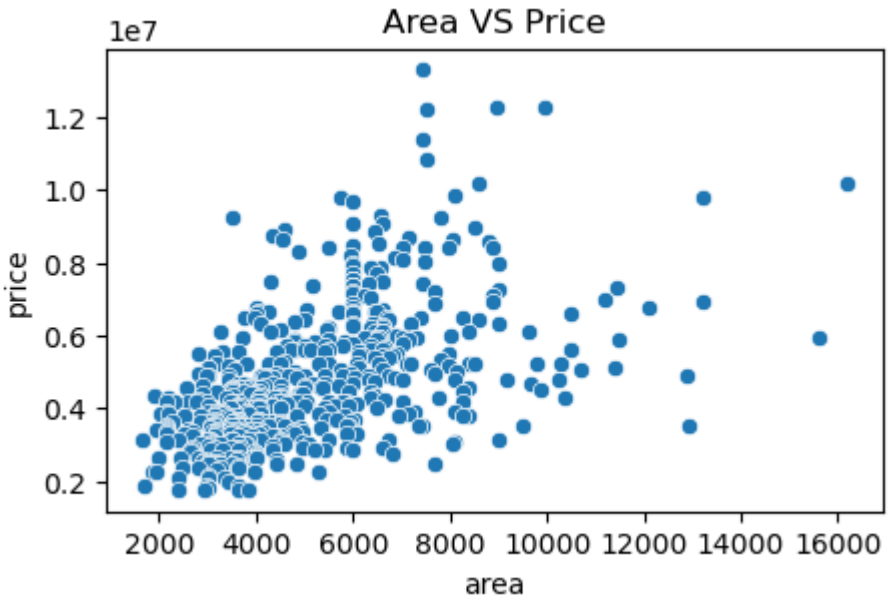
In [268]:
```
plt.figure(figsize=(20, 12))
plt.subplot(2,3,1)
sns.boxplot(x = 'mainroad', y = 'price', data = df)
plt.subplot(2,3,2)
sns.boxplot(x = 'guestroom', y = 'price', data = df)
plt.subplot(2,3,3)
sns.boxplot(x = 'basement', y = 'price', data = df)
plt.subplot(2,3,4)
sns.boxplot(x = 'hotwaterheating', y = 'price', data = df)
plt.subplot(2,3,5)
sns.boxplot(x = 'airconditioning', y = 'price', data = df)
plt.subplot(2,3,6)
sns.boxplot(x = 'furnishingstatus', y = 'price', data = df)
plt.show()
```





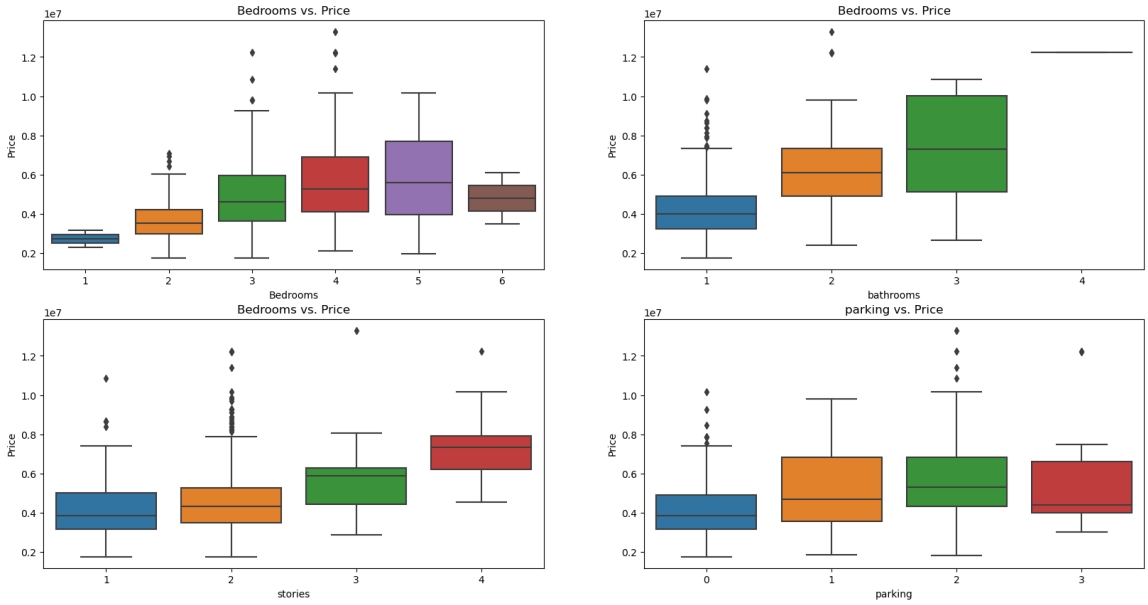
how does the area effect the price?

In [223]:
```
plt.figure(figsize=(5,3))
sns.scatterplot(x=df.area,y=df.price)
plt.title("Area VS Price")
plt.show()
```



****Price comparision with bedroom,bathroom,stories and parking**

In [224]:
```
fig=plt.subplots(2,2,figsize=(20,10))

plt.subplot(2,2,1)

sns.boxplot(x="bedrooms", y="price", data=df)
plt.xlabel("Bedrooms")
plt.ylabel("Price")
plt.title("Bedrooms vs. Price")

plt.subplot(2,2,2)
sns.boxplot(x="bathrooms", y="price", data=df)
plt.xlabel("bathrooms")
plt.ylabel("Price")
plt.title("Bedrooms vs. Price")

plt.subplot(2,2,3)

sns.boxplot(x="stories", y="price", data=df)
plt.xlabel("stories")
plt.ylabel("Price")
plt.title("Bedrooms vs. Price")

plt.subplot(2,2,4)

sns.boxplot(x="parking", y="price", data=df)
plt.xlabel("parking")
plt.ylabel("Price")
plt.title("parking vs. Price")

plt.show()
```



**furnishingstatus with the price

In [265]:
```
import plotly.express as px

fig=px.scatter(x=df['furnishingstatus'],y=df['price'],color=df['furnishingst
fig.show()
```

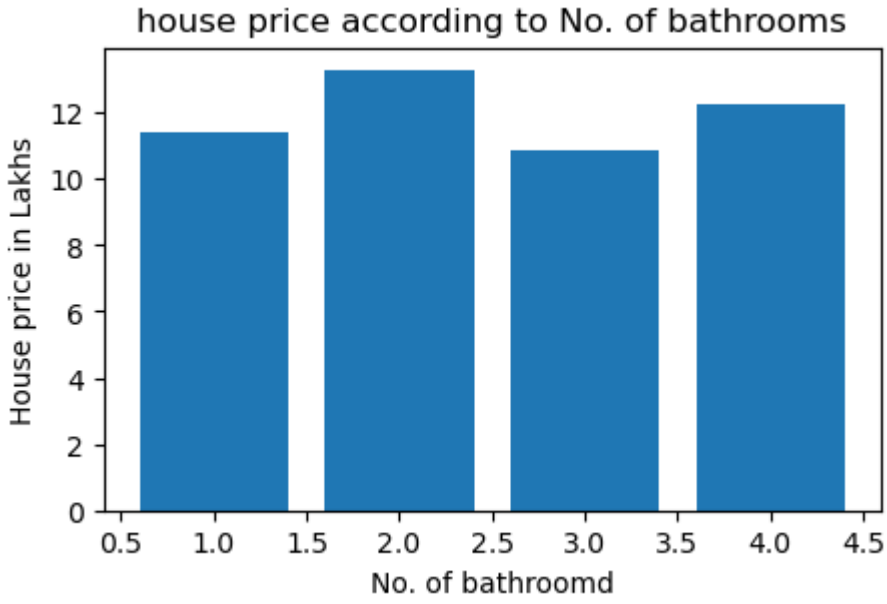
How does the number of bedrooms affect the sale price?

In [226]:
```
# price convert into lakhs
house_price=df['price']/1000000
```

In [227]:
```
plt.figure(figsize=(5,3))
plt.bar(df.bedrooms,house_price)
plt.xlabel("No. of bedrooms")
plt.ylabel("House price in Lakhs")
plt.title("house price according to No. of bedrooms")
plt.show()
```

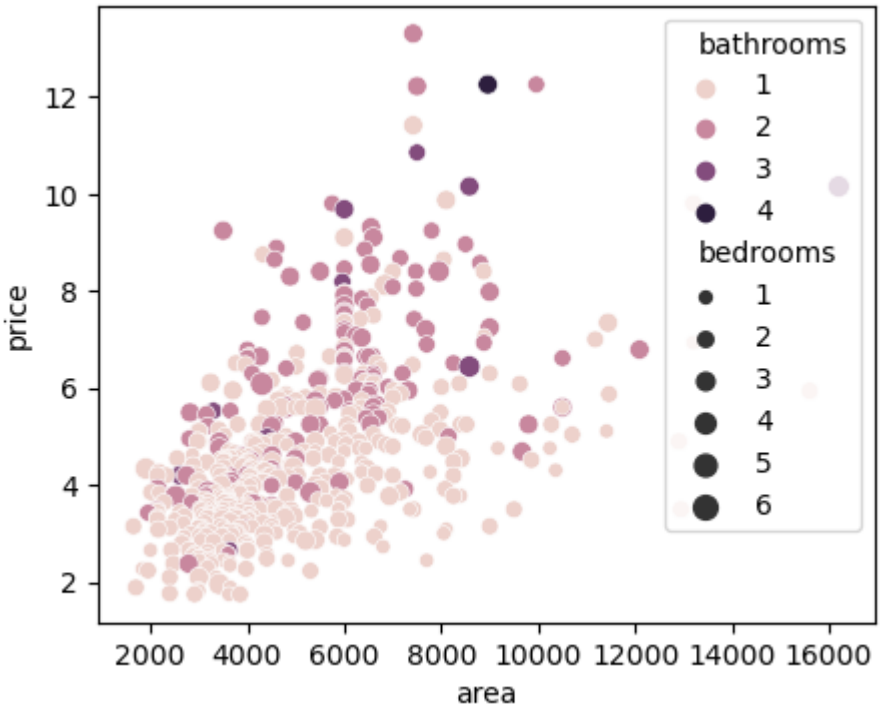
How does the number of bathrooms affect the sale price?

In [229]:
```
plt.figure(figsize=(5,3))
plt.bar(df.bathrooms,house_price)
plt.xlabel("No. of bathroomd")
plt.ylabel("House price in Lakhs")
plt.title("house price according to No. of bathrooms")
plt.show()
```



How does the number of bathrooms and bedrooms affect the sale price?

In [230]:
```
plt.figure(figsize=(5,4))
sns.scatterplot(x=df.area, y=house_price, hue=df.bathrooms, size=df.bedrooms
plt.show()
```



Data prepration Model

In [231]:
```
df.columns
```

Out[231]:
```
Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
       'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
       'parking', 'prefarea', 'furnishingstatus'],
      dtype='object')
```

In [232]:
```
df.head()
```

Out[232]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwater
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	

independance variable='area', 'bedrooms', 'bathrooms', 'stories', 'mainroad','guestroom', 'basement', 'hotwaterheating', 'airconditioning','parking', 'prefarea','furnishingstatus'
depandance variable= price

Applying One hot Encoding

In [233]:
```
from sklearn.preprocessing import OrdinalEncoder
```

In [234]:
```
oe=OrdinalEncoder(categories=[['no','yes']])
col=['mainroad','guestroom','basement','hotwaterheating','airconditioning','
for col_n in col:
  df[col_n]=oe.fit_transform(df[[col_n]])
```

In [235]:
```
df.head()
```

Out[235]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwater
0	13300000	7420	4	2	3	1.0	0.0	0.0	
1	12250000	8960	4	4	4	1.0	0.0	0.0	
2	12250000	9960	3	2	2	1.0	0.0	1.0	
3	12215000	7500	4	2	2	1.0	0.0	1.0	
4	11410000	7420	4	1	2	1.0	1.0	1.0	

In [236]:
```
oren=OrdinalEncoder(categories=[['unfurnished','semi-furnished','furnished']
df['furnishingstatus']=oren.fit_transform(df[['furnishingstatus']])
```

In [237]:
```
df.head()
```

Out[237]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwater
0	13300000	7420	4	2	3	1.0	0.0	0.0	
1	12250000	8960	4	4	4	1.0	0.0	0.0	
2	12250000	9960	3	2	2	1.0	0.0	1.0	
3	12215000	7500	4	2	2	1.0	0.0	1.0	
4	11410000	7420	4	1	2	1.0	1.0	1.0	

In [238]:
```
df.sample(10)
```

Out[238]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwat
163	5425000	6825	3	1	1	1.0	1.0	1.0	
160	5460000	6210	4	1	4	1.0	1.0	0.0	
312	4098500	3600	3	1	1	1.0	0.0	1.0	
394	3500000	3480	3	1	1	0.0	0.0	0.0	
442	3220000	2684	2	1	1	1.0	0.0	0.0	
8	9870000	8100	4	1	2	1.0	1.0	1.0	
247	4550000	8400	4	1	4	1.0	0.0	0.0	
224	4760000	10240	2	1	1	1.0	0.0	0.0	
222	4760000	9166	2	1	1	1.0	0.0	1.0	
194	5005000	8150	3	2	1	1.0	1.0	1.0	

In [239]:
```
x=df[['area', 'bedrooms', 'bathrooms', 'stories', 'mainroad','guestroom', 'b
y=df['price']/100000
x['area']=x['area']/1000
```

In [240]:
```
x
```

Out[240]:

	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating
0	7.42	4	2	3	1.0	0.0	0.0	0.0
1	8.96	4	4	4	1.0	0.0	0.0	0.0
2	9.96	3	2	2	1.0	0.0	1.0	0.0
3	7.50	4	2	2	1.0	0.0	1.0	0.0
4	7.42	4	1	2	1.0	1.0	1.0	0.0
...	...	...	...	...	...	...	...	...
540	3.00	2	1	1	1.0	0.0	1.0	0.0
541	2.40	3	1	1	0.0	0.0	0.0	0.0
542	3.62	2	1	1	1.0	0.0	0.0	0.0
543	2.91	3	1	1	0.0	0.0	0.0	0.0
544	3.85	3	1	2	1.0	0.0	0.0	0.0

545 rows × 12 columns

In [241]:
```
y
```

Out[241]:
```
0      133.0000
1      122.5000
2      122.5000
3      122.1500
4      114.1000
         ...
540     18.2000
541     17.6715
542     17.5000
543     17.5000
544     17.5000
Name: price, Length: 545, dtype: float64
```

Spiliting Data into training and test set

In [242]:
```
from sklearn.model_selection import train_test_split
```

In [243]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_stat
```

In [244]:
```
print(x_train)
print(x_test)
print(y_train)
print(y_test)
```

```
       area  bedrooms  bathrooms  stories  mainroad  guestroom  basement
\
454   4.500         3          1        2       1.0        0.0       0.0
392   3.990         3          1        2       1.0        0.0       0.0
231   4.320         3          1        1       1.0        0.0       0.0
271   1.905         5          1        2       0.0        0.0       1.0
250   3.510         3          1        3       1.0        0.0       0.0
..      ...       ...        ...      ...       ...        ...       ...
70    4.000         3          2        2       1.0        0.0       1.0
277  10.360         2          1        1       1.0        0.0       0.0
9     5.750         3          2        4       1.0        1.0       0.0
359   3.600         3          1        1       1.0        0.0       0.0
192   6.600         3          1        1       1.0        1.0       1.0

     hotwaterheating  airconditioning  parking  prefarea  furnishingstatus
454              0.0              1.0        0       0.0               0.0
392              0.0              0.0        0       0.0               1.0
231              0.0              0.0        0       1.0               1.0
271              0.0              0.0        0       0.0               1.0
250              0.0              0.0        0       0.0               1.0
..               ...              ...      ...       ...               ...
70               0.0              1.0        0       1.0               1.0
277              0.0              0.0        1       1.0               1.0
9                0.0              1.0        1       1.0               0.0
359              0.0              0.0        1       0.0               0.0
192              0.0              0.0        0       1.0               2.0

[381 rows x 12 columns]
      area  bedrooms  bathrooms  stories  mainroad  guestroom  basement  \
239   4.00         3          1        2       1.0        0.0       0.0
113   9.62         3          1        1       1.0        0.0       1.0
325   3.46         4          1        2       1.0        0.0       0.0
66   13.20         2          1        1       1.0        0.0       1.0
479   3.66         4          1        2       0.0        0.0       0.0
..     ...       ...        ...      ...       ...        ...       ...
477   4.96         2          1        1       1.0        0.0       0.0
505   4.00         3          1        2       1.0        0.0       0.0
347   3.35         3          1        2       1.0        0.0       0.0
224  10.24         2          1        1       1.0        0.0       0.0
38    6.00         3          1        4       1.0        1.0       0.0

     hotwaterheating  airconditioning  parking  prefarea  furnishingstatus
239              0.0              0.0        1       0.0               2.0
113              0.0              0.0        2       1.0               2.0
325              0.0              1.0        0       0.0               1.0
66               1.0              0.0        1       0.0               2.0
479              0.0              0.0        0       0.0               0.0
..               ...              ...      ...       ...               ...
477              0.0              0.0        0       0.0               0.0
505              0.0              1.0        0       0.0               0.0
347              0.0              0.0        0       0.0               0.0
224              0.0              1.0        2       1.0               0.0
38               0.0              1.0        2       0.0               0.0

[164 rows x 12 columns]
454    31.43
392    35.00
231    46.90
271    43.40
250    45.15
       ...
```

```
70     67.90
277    43.05
9      98.00
359    37.10
192    50.40
Name: price, Length: 381, dtype: float64
239    45.850
113    60.830
325    40.075
66     69.300
479    29.400
        ...
477    29.400
505    26.530
347    38.360
224    47.600
38     79.625
Name: price, Length: 164, dtype: float64
```

In [245]:
```
print("X_train:", x_train.shape)
print("X_test:", x_test.shape)
print("y_train:", y_train.shape)
print("y_test:", y_test.shape)
```

```
X_train: (381, 12)
X_test: (164, 12)
y_train: (381,)
y_test: (164,)
```

feature scaling

In [246]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [247]:
```
mnc=MinMaxScaler()
```

In [248]:
```
mnc.fit_transform(x_train)
```

Out[248]:
```
array([[0.19354839, 0.5       , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.1564952 , 0.5       , 0.        , ..., 0.        , 0.        ,
        0.5       ],
       [0.18047079, 0.5       , 0.        , ..., 0.        , 1.        ,
        0.5       ],
       ...,
       [0.28436501, 0.5       , 0.5       , ..., 0.33333333, 1.        ,
        0.        ],
       [0.12816042, 0.5       , 0.        , ..., 0.33333333, 0.        ,
        0.        ],
       [0.34612031, 0.5       , 0.        , ..., 0.        , 1.        ,
        1.        ]])
```

In [249]:
```
x_train_mnc=mnc.transform(x_train)
x_test_mnc=mnc.transform(x_test)
```

In [250]:
```
x_train_df=pd.DataFrame(x_train_mnc,columns=x_train.columns)
```

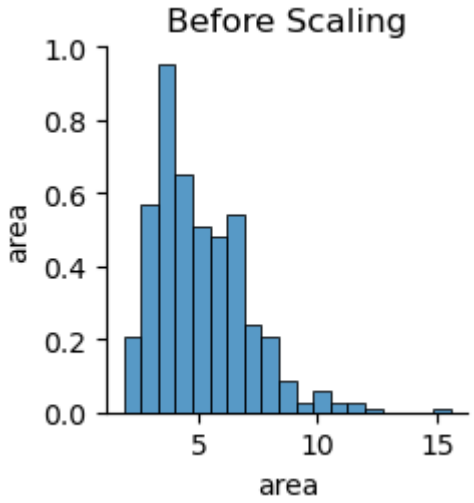
In [251]:
```
x_test_df=pd.DataFrame(x_test_mnc,columns=x_test.columns)
```

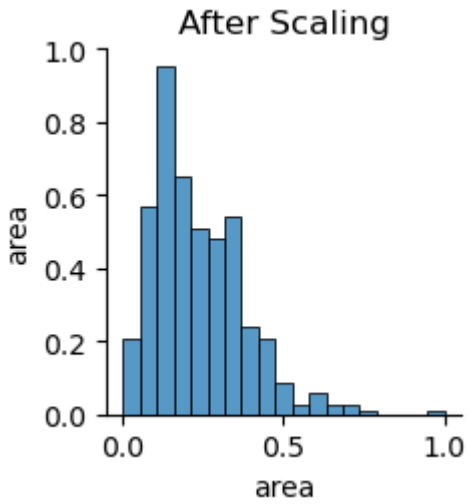
In [252]:
```
x_train_df.describe().round(2)
```

Out[252]:

	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheati
count	381.00	381.00	381.00	381.00	381.00	381.00	381.00	381.
mean	0.23	0.49	0.14	0.27	0.87	0.17	0.34	0.
std	0.14	0.18	0.24	0.29	0.34	0.38	0.47	0.
min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.
25%	0.13	0.25	0.00	0.00	1.00	0.00	0.00	0.
50%	0.20	0.50	0.00	0.33	1.00	0.00	0.00	0.
75%	0.33	0.50	0.50	0.33	1.00	0.00	1.00	0.
max	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.

In [253]:
```
# Before scaling
sns.pairplot(x_train[['area']])
plt.title("Before Scaling")
plt.show()

# After scaling
sns.pairplot(x_train_df[['area']])
plt.title("After Scaling")
plt.show()
```





** thier is no change after scaling

Applying Machine learning Model

In [254]:
```
from sklearn.linear_model import LinearRegression
```

In [255]:
```
lr=LinearRegression()
```

In [256]:
```
lr.fit(x_train,y_train)
```

Out[256]:
```
LinearRegression()
```

In [257]:
```
lr.coef_
```

Out[257]:
```
array([ 2.48857876,  1.34994406,  9.5058338 ,  4.18321569,  4.66890751,
        3.68497644,  3.59364424, 12.46653309,  8.97037026,  2.23301809,
        6.96754525,  2.30222653])
```

In [258]:
```
lr.intercept_
```

Out[258]:
```
-3.5331183360928904
```

predict the value of home and test

In [259]:
```
coef_df=pd.DataFrame(lr.coef_,x.columns,columns=['coefficient'])
```

In [260]:
```
coef_df
```

Out[260]:

	coefficient
area	2.488579
bedrooms	1.349944
bathrooms	9.505834
stories	4.183216
mainroad	4.668908
guestroom	3.684976
basement	3.593644
hotwaterheating	12.466533
airconditioning	8.970370
parking	2.233018
prefarea	6.967545
furnishingstatus	2.302227

In [261]:
```
# Make predictions
predictions = lr.predict(x_test)
```

In [262]:
```
predictions
```

Out[262]:
```
array([ 39.84967274,  62.44647723,  44.2909099 ,  73.27161486,
        28.84712135,  69.80921091,  32.92173526,  31.58447774,
        35.46517873,  82.71816411,  66.18193859,  37.51711149,
        37.53935403,  45.83853187,  39.66468549,  20.25300972,
        39.79784916,  36.1581415 ,  31.99944395,  46.76419071,
        58.24326873,  64.1769914 ,  46.77337712,  27.28847044,
        53.77896734,  57.07297174,  53.62417436,  54.38360521,
        56.17773482,  58.98735928,  33.00971795,  63.60929404,
        71.64452047,  29.81160306,  44.51574862,  51.52282505,
        49.70121011,  36.50908243,  29.36483251,  40.05062876,
        80.12393067,  49.75403389,  64.33427878,  36.14305536,
        39.06804952,  62.95100537,  45.47614942,  27.57858497,
        41.25556506,  65.34659521,  39.63785236,  69.78462055,
        26.08877822,  29.40798154,  35.74053204,  51.88052848,
        70.8525341 ,  40.49008067,  28.78583685,  44.07158208,
        60.04782606,  66.88667666,  33.20792848,  71.66946569,
        26.26313836,  50.3711751 ,  66.90131278,  26.06346969,
        38.27137257,  50.29135   ,  43.56153054,  71.80792622,
        51.96956719,  59.28376954,  40.6956174 ,  45.39855598,
        29.59980651,  75.19193726,  26.0519806 ,  36.7488136 ,
        43.56153054,  60.04926747,  51.03302538,  54.43102514,
        38.15751943,  41.1892861 ,  47.22065955,  50.50970105,
        39.09500694,  43.06251269,  32.68097203,  58.67054031,
        31.18760654,  36.09654694,  45.62963964, 103.86330666,
        29.66017992,  70.6870154 ,  43.69845489,  45.26625951,
        64.57898598,  32.9344218 ,  45.45378648,  34.11726706,
        73.26598023,  51.45430856,  40.73440292,  49.90832461,
        63.19251381,  26.53338189,  27.47904182,  20.4210987 ,
        26.48361032,  45.71428393,  30.56588608,  44.11681885,
        68.75497538,  26.58315347,  41.68622859,  84.50917513,
        24.17943499,  51.91889426,  26.35665022,  48.05677038,
        27.39519309,  32.80223594,  74.06378536,  51.08683427,
        51.72012042,  38.29962036,  45.59413825,  32.20792383,
        55.38905979,  26.58315347,  47.83025012,  85.08441812,
        45.16843661,  28.98732654,  43.33314626,  20.32155555,
        48.84850653,  50.12884874,  62.55335571,  47.67586098,
        64.68578421,  72.96868438,  47.24566352,  72.41429332,
        64.33668419,  36.93440768,  60.37737783,  31.97732919,
        26.83778911,  67.97360569,  59.19061286,  55.71541395,
        95.84411355,  76.85234397,  45.92048149,  29.86807743,
        41.98257183,  31.39462538,  63.41172495,  63.47717335])
```

In [263]:
```
y_test
```

Out[263]:
```
239    45.850
113    60.830
325    40.075
66     69.300
479    29.400
       ...
477    29.400
505    26.530
347    38.360
224    47.600
38     79.625
Name: price, Length: 164, dtype: float64
```

In [264]:
```
lr.score(x_test,y_test)
```

Out[264]:
```
0.723501522320035
```

In [ ]:
```

```