In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
data = pd.read_csv("Housing.csv")
```

In [3]:
```
data.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	

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

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

In [5]:
```
data.isna().sum()
```

Out[5]:
```
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
```

In [6]:
```
data.describe()
```

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
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 [7]:
```
data.head()
```

Out[7]:

	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 [8]:
```
data["mainroad"].value_counts()
```

Out[8]:
```
mainroad
yes    468
no      77
Name: count, dtype: int64
```

In [9]:
```
Main_road = pd.get_dummies(data["mainroad"] , drop_first = True)
```

In [10]:
```
Main_road = Main_road.astype(int)
```

In [11]:
```
data = pd.concat([Main_road , data] , axis = 1)
```

In [12]:
```
data = data.rename(columns = {"yes" : "main_road"})
```

In [13]:
```
guest_room = pd.get_dummies(data["guestroom"] , drop_first = True)
guest_room = guest_room.astype(int)
data = pd.concat([guest_room , data] , axis = 1)
```

In [14]:
```
data = data.rename(columns = {"yes" : "guest_room"})
```

In [15]:
```
base_ment = pd.get_dummies(data["basement"] , drop_first = True)
base_ment = base_ment.astype(int)
data = pd.concat([base_ment , data] , axis = 1)
data = data.rename(columns = {"yes" : "base_ment"})
```

In [16]:
```
hot_water = pd.get_dummies(data["hotwaterheating"] , drop_first = True)
hot_water = hot_water.astype(int)
```

```
data = pd.concat([hot_water , data] , axis = 1)
data = data.rename(columns = {"yes" : "hot_water"})
```

In [17]:
```
air_condition = pd.get_dummies(data["airconditioning"] , drop_first = True)
air_condition = air_condition.astype(int)
data = pd.concat([air_condition , data] , axis = 1)
data = data.rename(columns = {"yes" : "air_condition"})
```

In [18]:
```
df = data.drop(["mainroad" , "guestroom" ,"basement" , "hotwaterheating" , "airconc
```

In [19]:
```
df.head()
```

Out[19]:

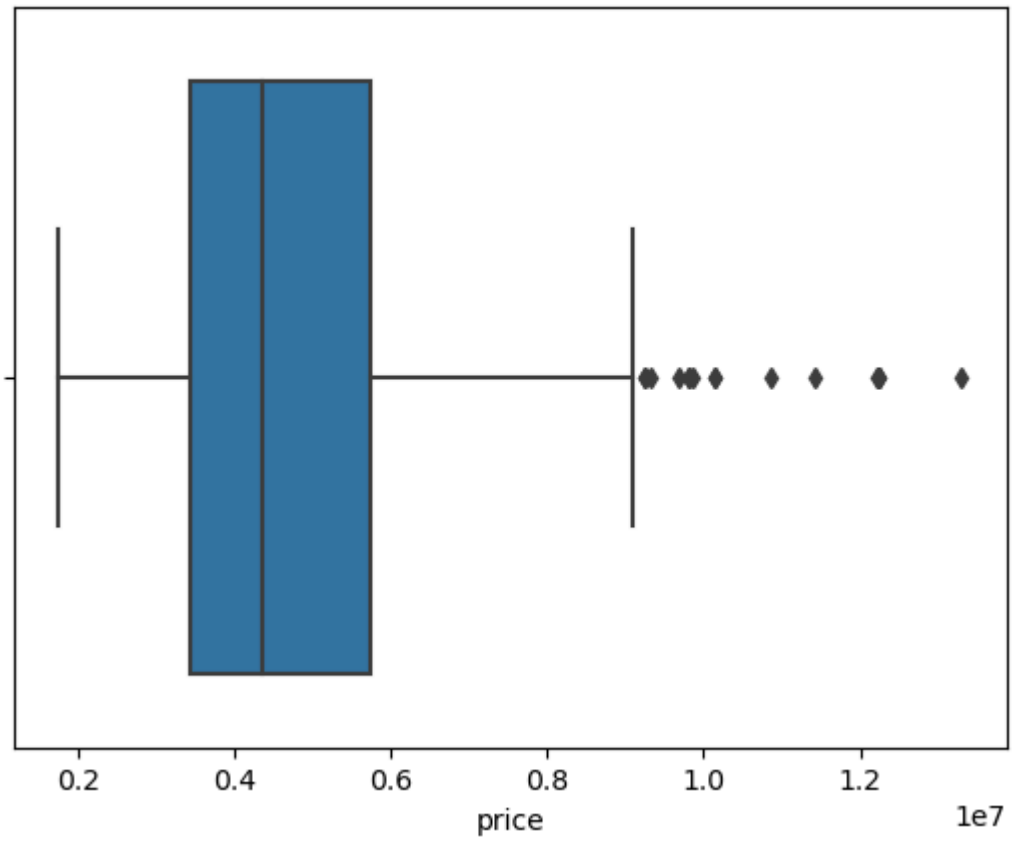
	air_condition	hot_water	base_ment	guest_room	main_road	price	area	bedrooms	bath
0	1	0	0	0	1	13300000	7420	4	
1	1	0	0	0	1	12250000	8960	4	
2	0	0	1	0	1	12250000	9960	3	
3	1	0	1	0	1	12215000	7500	4	
4	1	0	1	1	1	11410000	7420	4	

In [20]:
```
sns.scatterplot(data = df , x = "bedrooms" , y =  "price" , color = "red")
plt.xlabel("bed rooms")
plt.ylabel("price of the house")
plt.title("price of the house besed on the bedrooms")
```

Out[20]:
```
Text(0.5, 1.0, 'price of the house besed on the bedrooms')
```

In [21]:
```
sns.boxplot(x = df["price"])
```

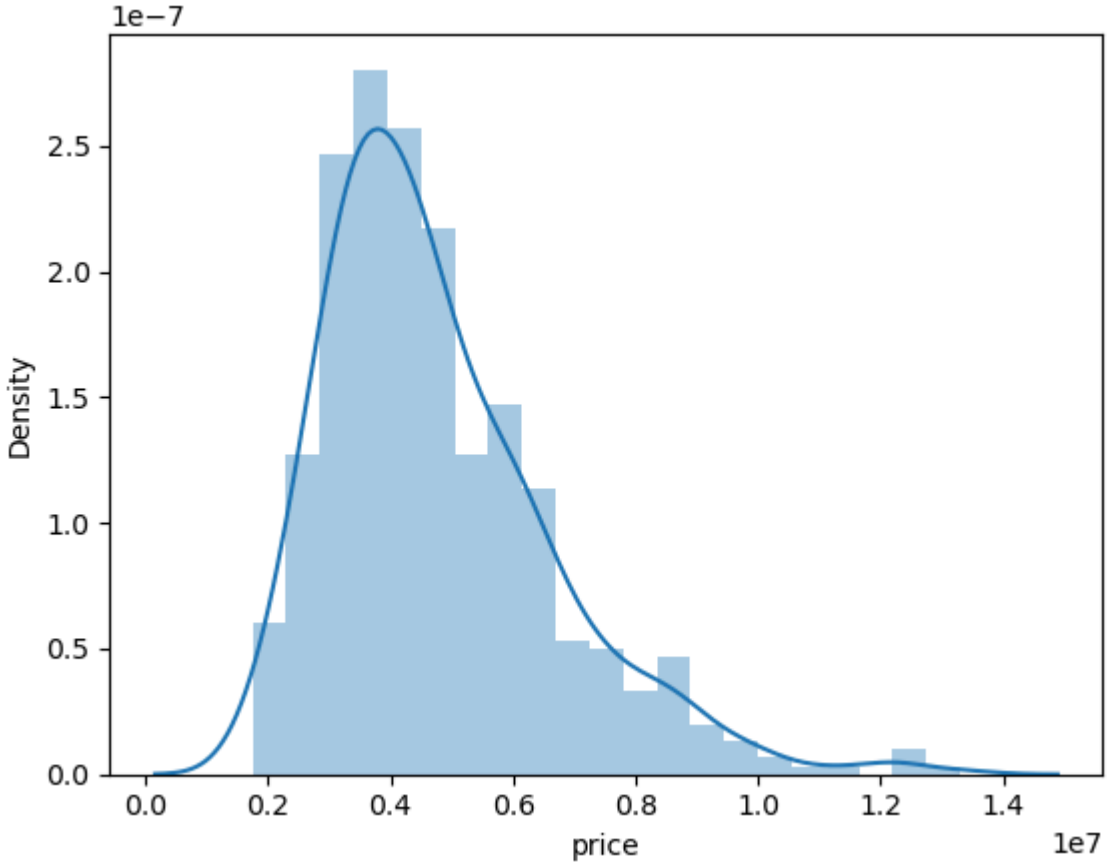
Out[21]:
```
<Axes: xlabel='price'>
```



In [22]:
```
sns.distplot(df["price"])
```

```
C:\Users\godde\AppData\Local\Temp\ipykernel_20732\50337492.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["price"])
```

Out[22]:
```
<Axes: xlabel='price', ylabel='Density'>
```



In [23]:
```
Q1 = df["price"].quantile(0.25)
Q2 = df["price"].quantile(0.50)
Q3 = df["price"].quantile(0.75)
print("Q1 = ", Q1)
print("Q2 = ", Q2)
print("Q3 = " , Q3)
```

```
Q1 =  3430000.0
Q2 =  4340000.0
Q3 =  5740000.0
```

In [24]:
```
IQR = Q3-Q1
```

In [25]:
```
IQR
```

Out[25]:
```
2310000.0
```

In [26]:
```
upper_level = Q3+1.5*IQR
lower_level = Q1-1.5*IQR
print("upper_level = " , upper_level)
print("lower_level = " , lower_level)
```

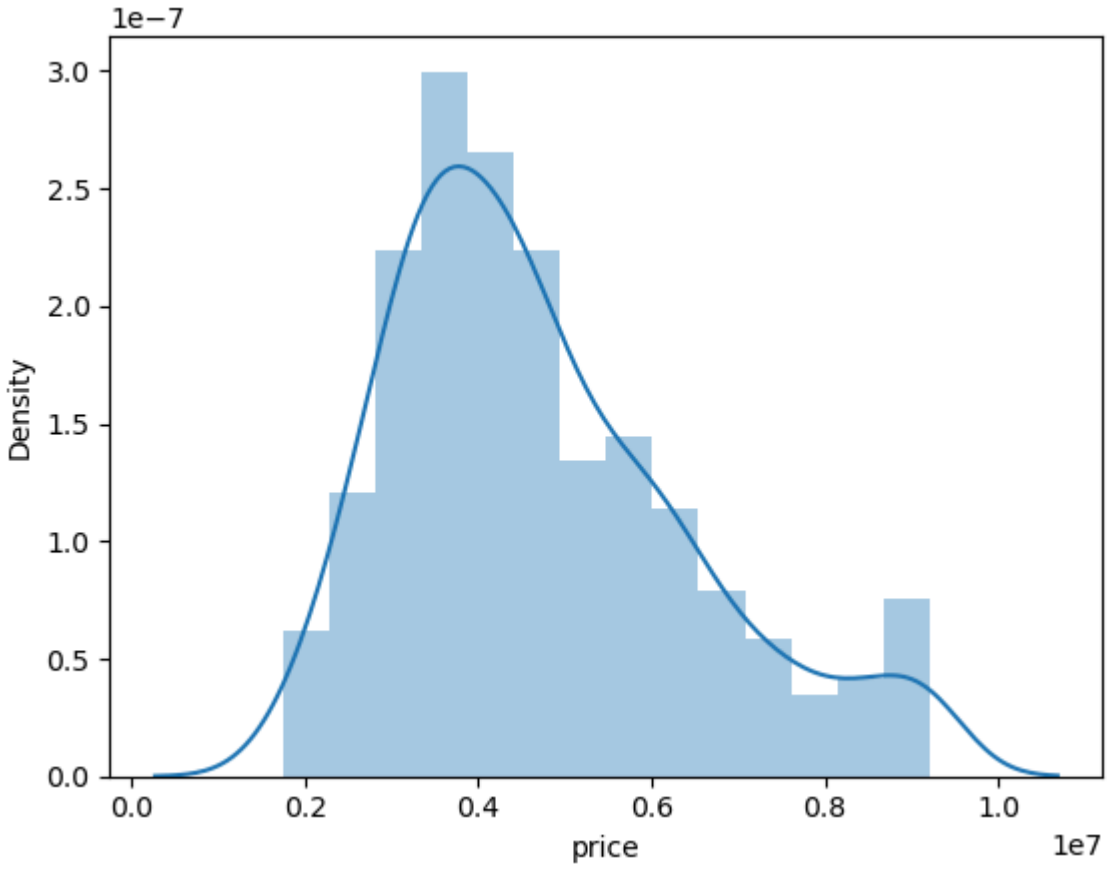
```
upper_level =  9205000.0
lower_level =  -35000.0
```

In [27]:
```
df["price"] = np.where(df["price"] > upper_level , upper_level,
                       np.where(df["price"] < lower_level , lower_level , df["price"
```

In [28]:
```
sns.distplot(df["price"])
```

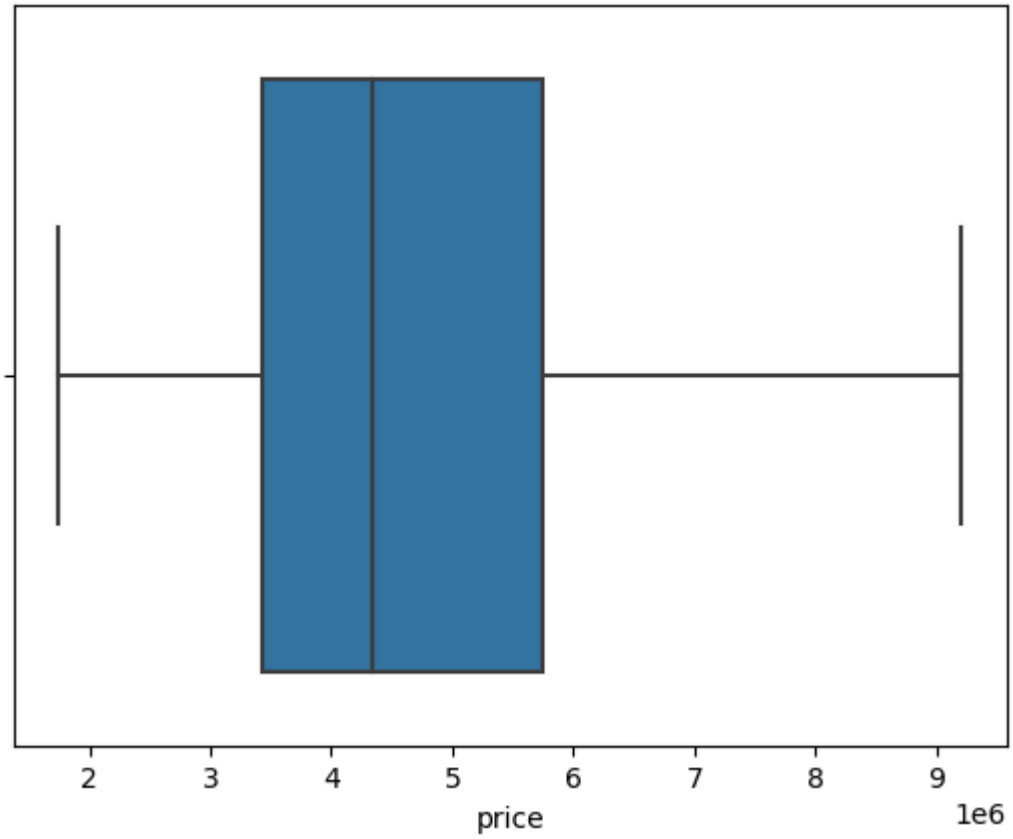
```
C:\Users\godde\AppData\Local\Temp\ipykernel_20732\50337492.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["price"])
```

Out[28]: <Axes: xlabel='price', ylabel='Density'>



In [29]:
```
sns.boxplot(x = df["price"])
```

Out[29]: <Axes: xlabel='price'>



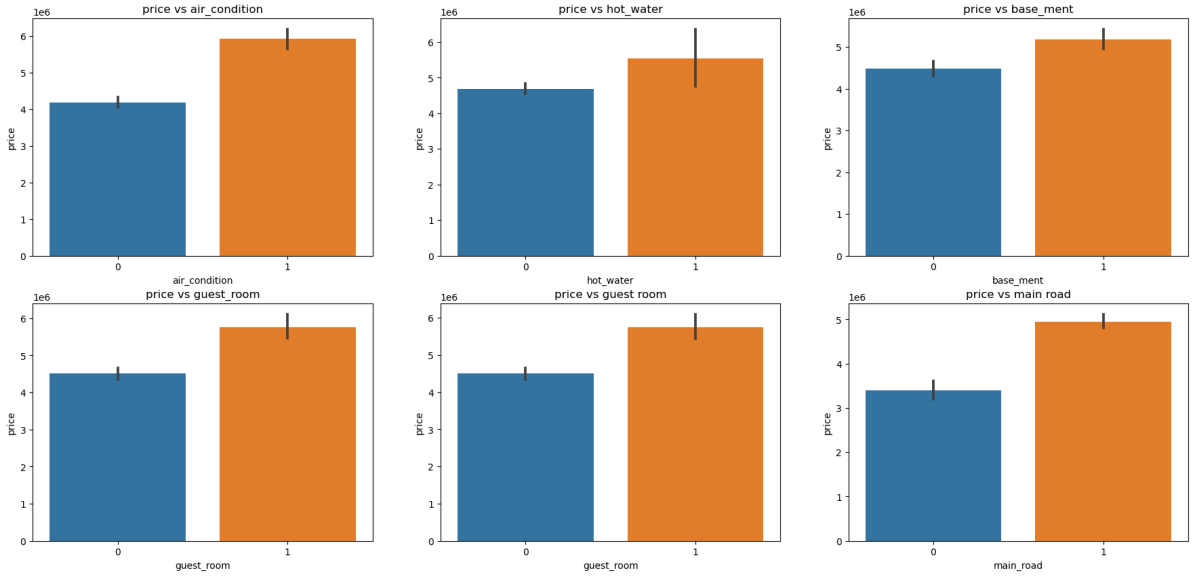
In [30]:
```
df.head()
```

Out[30]:

	air_condition	hot_water	base_ment	guest_room	main_road	price	area	bedrooms	bath
0	1	0	0	0	1	9205000.0	7420	4	
1	1	0	0	0	1	9205000.0	8960	4	
2	0	0	1	0	1	9205000.0	9960	3	
3	1	0	1	0	1	9205000.0	7500	4	
4	1	0	1	1	1	9205000.0	7420	4	

In [31]:
```
plt.figure(figsize = (22,10))
plt.subplot(2,3,1)
sns.barplot(data = df , x = "air_condition" , y = "price")
plt.title("price vs air_condition")
plt.subplot(2,3,2)
sns.barplot(data = df , x = "hot_water" , y = "price")
plt.title("price vs hot_water")
plt.subplot(2,3,3)
sns.barplot(data = df , x = "base_ment" , y = "price")
plt.title("price vs base_ment")
plt.subplot(2,3,4)
sns.barplot(data = df , x = "guest_room" , y = "price")
plt.title("price vs guest_room")
plt.subplot(2,3,5)
sns.barplot(data = df , x = "guest_room" , y = "price")
plt.title("price vs guest room")
plt.subplot(2,3,6)
sns.barplot(data = df , x = "main_road" , y = "price")
plt.title("price vs main road")
```

Out[31]:
```
Text(0.5, 1.0, 'price vs main road')
```



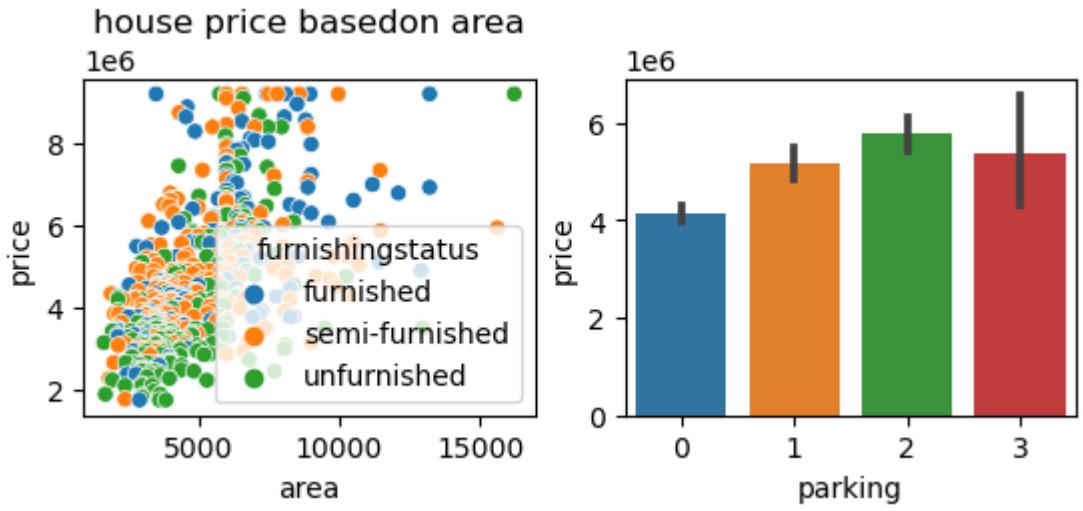
In [32]:
```
df.head()
```

Out[32]:

	air_condition	hot_water	base_ment	guest_room	main_road	price	area	bedrooms	bath
0	1	0	0	0	1	9205000.0	7420	4	
1	1	0	0	0	1	9205000.0	8960	4	
2	0	0	1	0	1	9205000.0	9960	3	
3	1	0	1	0	1	9205000.0	7500	4	
4	1	0	1	1	1	9205000.0	7420	4	

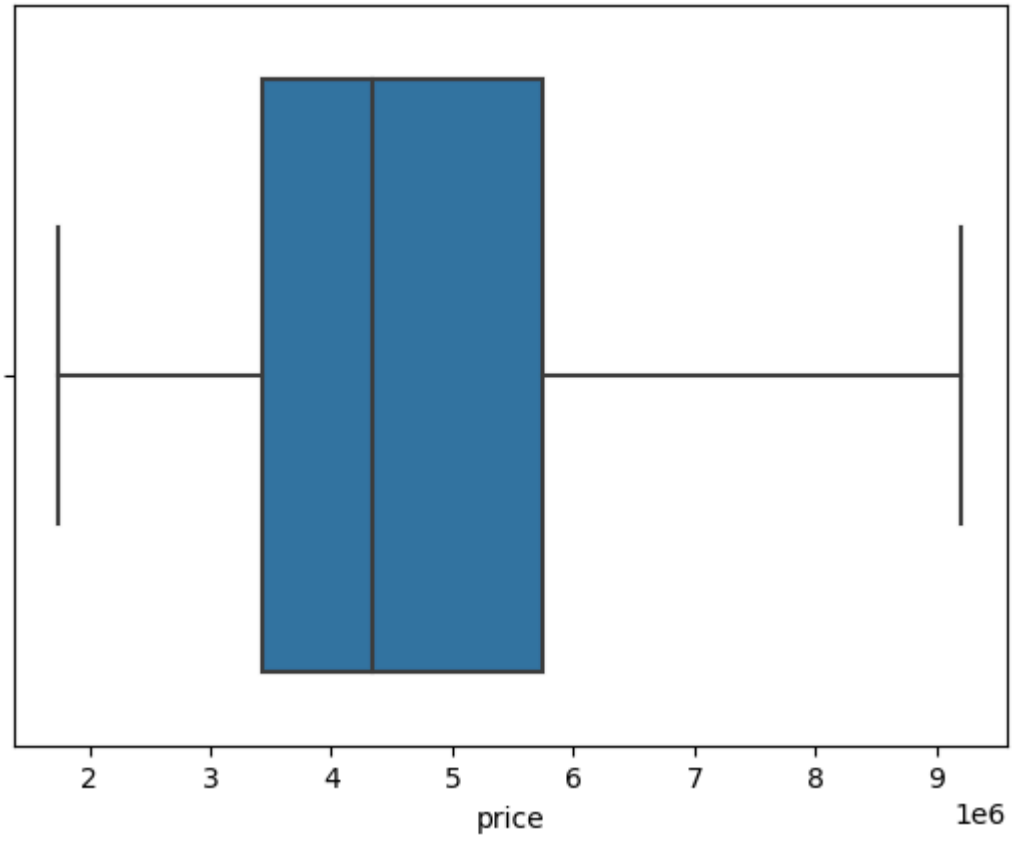
In [33]:
```
plt.subplot(2,2,1)
sns.scatterplot(data = df , x = "area" , y = "price" , hue = "furnishingstatus")
plt.title("house price basedon area")
plt.subplot(2,2,2)
sns.barplot(data = df , x = "parking" , y = "price"  )
```

Out[33]:
```
<Axes: xlabel='parking', ylabel='price'>
```



In [34]:
```
sns.boxplot(x = df["price"])
```

Out[34]:
```
<Axes: xlabel='price'>
```



In [35]:
```
data.head()
```

Out[35]:

	air_condition	hot_water	base_ment	guest_room	main_road	price	area	bedrooms	bath
0	1	0	0	0	1	13300000	7420	4	
1	1	0	0	0	1	12250000	8960	4	
2	0	0	1	0	1	12250000	9960	3	
3	1	0	1	0	1	12215000	7500	4	
4	1	0	1	1	1	11410000	7420	4	

In [36]:
```
df = df.drop("furnishingstatus" ,axis = 1)
```

In [37]:
```
df
```

Out[37]:

	air_condition	hot_water	base_ment	guest_room	main_road	price	area	bedrooms	ba
0	1	0	0	0	1	9205000.0	7420	4	
1	1	0	0	0	1	9205000.0	8960	4	
2	0	0	1	0	1	9205000.0	9960	3	
3	1	0	1	0	1	9205000.0	7500	4	
4	1	0	1	1	1	9205000.0	7420	4	
...	...	...	...	...	...	...	...	...	
540	0	0	1	0	1	1820000.0	3000	2	
541	0	0	0	0	0	1767150.0	2400	3	
542	0	0	0	0	1	1750000.0	3620	2	
543	0	0	0	0	0	1750000.0	2910	3	
544	0	0	0	0	1	1750000.0	3850	3	

545 rows × 11 columns

In [38]:
```
x = df.drop("price" , axis = 1)
```

In [39]:
```
y = df[["price"]]
```

In [40]:
```
x
```

Out[40]:

	air_condition	hot_water	base_ment	guest_room	main_road	area	bedrooms	bathrooms	s
0	1	0	0	0	1	7420	4	2	
1	1	0	0	0	1	8960	4	4	
2	0	0	1	0	1	9960	3	2	
3	1	0	1	0	1	7500	4	2	
4	1	0	1	1	1	7420	4	1	
...	...	...	...	...	...	...	...	...	
540	0	0	1	0	1	3000	2	1	
541	0	0	0	0	0	2400	3	1	
542	0	0	0	0	1	3620	2	1	
543	0	0	0	0	0	2910	3	1	
544	0	0	0	0	1	3850	3	1	

545 rows × 10 columns

In [41]:
```
y
```

Out[41]:

	price
0	9205000.0
1	9205000.0
2	9205000.0
3	9205000.0
4	9205000.0
...	...
540	1820000.0
541	1767150.0
542	1750000.0
543	1750000.0
544	1750000.0

545 rows × 1 columns

In [42]:
```
scalar = StandardScaler()
```

In [43]:
```
x = scalar.fit_transform(x)
```

In [44]:
```
x_train , x_test , y_train , y_test = train_test_split(x , y , test_size = 0.2 , ra
```

In [45]:
```
x_train
```

Out[45]:
```
array([[ 1.4726183 , -0.2192645 , -0.73453933, ..., -0.57018671,
         0.22441013, -0.80574124],
       [-0.67906259, -0.2192645 ,  1.3613975 , ...,  1.42181174,
         0.22441013, -0.80574124],
       [-0.67906259, -0.2192645 , -0.73453933, ..., -0.57018671,
        -0.92939666, -0.80574124],
       ...,
       [-0.67906259,  4.5607017 ,  1.3613975 , ..., -0.57018671,
         0.22441013,  0.35597563],
       [ 1.4726183 , -0.2192645 , -0.73453933, ..., -0.57018671,
        -0.92939666,  1.51769249],
       [-0.67906259, -0.2192645 ,  1.3613975 , ..., -0.57018671,
         0.22441013, -0.80574124]])
```

In [46]:
```
x_test
```

Out[46]:
```
array([[ 1.4726183 , -0.2192645 , -0.73453933, ...,  1.42181174,
         1.37821692,  1.51769249],
       [ 1.4726183 , -0.2192645 ,  1.3613975 , ..., -0.57018671,
        -0.92939666, -0.80574124],
       [ 1.4726183 , -0.2192645 ,  1.3613975 , ..., -0.57018671,
         0.22441013, -0.80574124],
       ...,
       [ 1.4726183 , -0.2192645 ,  1.3613975 , ..., -0.57018671,
         1.37821692,  1.51769249],
       [-0.67906259, -0.2192645 , -0.73453933, ..., -0.57018671,
        -0.92939666,  0.35597563],
       [ 1.4726183 , -0.2192645 ,  1.3613975 , ...,  1.42181174,
        -0.92939666,  1.51769249]])
```

In [47]:
```
y_train
```

Out[47]:

	price
505	2653000.0
238	4613000.0
246	4550000.0
2	9205000.0
307	4165000.0
...	...
153	5530000.0
528	2275000.0
74	6650000.0
176	5250000.0
338	3885000.0

436 rows × 1 columns

In [48]:
```
y_test
```

Out[48]:

	price
0	9205000.0
118	5950000.0
200	4900000.0
323	4025000.0
71	6755000.0
...	...
179	5215000.0
125	5943000.0
319	4060000.0
183	5145000.0
180	5215000.0

109 rows × 1 columns

In [49]:
```
model = DecisionTreeRegressor()
```

In [50]:
```
model
```

Out[50]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [51]:
```
model.fit(x_train , y_train)
```

Out[51]:
```
DecisionTreeRegressor
DecisionTreeRegressor()
```

In [52]:
```
y_prediction = model.predict(x_test)
```

In [53]:
```
y_prediction[71]
```

Out[53]:
```
7210000.0
```

In [54]:
```
y_test[:10]
```

Out[54]:

	price
0	9205000.0
118	5950000.0
200	4900000.0
323	4025000.0
71	6755000.0
198	4935000.0
446	3150000.0
533	2100000.0
494	2730000.0
536	1960000.0

In [55]:
```
error = mean_squared_error(y_test , y_prediction)
```

In [56]:
```
error
```

Out[56]:
```
2516478792002.956
```

In [57]:
```
error_2 = mean_squared_error(y_prediction , y_test)
```

In [58]:
```
error_2
```

Out[58]:
```
2516478792002.956
```

random forest

In [59]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [60]:
```
rfr = RandomForestRegressor(1000)
```

In [61]:
```
rfr
```

Out[61]:
```
RandomForestRegressor
RandomForestRegressor(n_estimators=1000)
```

In [62]:
```
rfr.fit(x_train , y_train)
```

```
C:\Users\godde\anaconda3\Lib\site-packages\sklearn\base.py:1151: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

Out[62]:
```
RandomForestRegressor
RandomForestRegressor(n_estimators=1000)
```

In [63]:
```
prediction = rfr.predict(x_test)
```

In [64]:
```
prediction[:10]
```

Out[64]:
```
array([8395531.66666667, 5887441.        , 3906136.5       ,
       4185863.5       , 6707848.        , 5155118.5       ,
       4935154.        , 3722481.81666667, 4178216.        ,
       3413746.        ])
```

In [65]:
```
y_test
```

Out[65]:

	price
0	9205000.0
118	5950000.0
200	4900000.0
323	4025000.0
71	6755000.0
...	...
179	5215000.0
125	5943000.0
319	4060000.0
183	5145000.0
180	5215000.0

109 rows × 1 columns

In [66]:
```
error_3 = mean_squared_error(y_test , prediction)
```

In [67]:
```
error_3
```

Out[67]:
```
1111613325196.62
```

In [ ]:
```

```

support vector machine

In [73]:
```
from sklearn.svm import SVR
```

In [91]:
```
model = SVR(kernel = "poly")
```

In [92]:
```
model
```

Out[92]:
```
SVR
SVR(kernel='poly')
```

In [93]:
```
model.fit(x_train , y_train)
```

```
C:\Users\godde\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected. Pleas
e change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[93]:
```
SVR
SVR(kernel='poly')
```

In [94]:
```
y_prediction_2 = model.predict(x_test)
```

In [95]:
```
y_prediction[:10]
```

Out[95]:
```
array([9205000., 4900000., 4007500., 3290000., 6580000., 5075000.,
       4620000., 4550000., 4900000., 3234000.])
```

In [96]:
```
y_test
```

Out[96]:

	price
0	9205000.0
118	5950000.0
200	4900000.0
323	4025000.0
71	6755000.0
...	...
179	5215000.0
125	5943000.0
319	4060000.0
183	5145000.0
180	5215000.0

109 rows × 1 columns

In [97]:
```
error_4 = mean_squared_error(y_test , y_prediction)
```

In [98]:
```
error_4
```

Out[98]:
```
2516478792002.956
```

In [ ]:
```

```

In [ ]:

In [ ]:

In [ ]: