ADS Assignment-3

June 3, 2023

Name: Akash R Reg no: 20BCE1501

1 Problem Statement: House Price Prediction

1.1 Description:

House price prediction is a common problem in the real estate industry and involves predicting the selling price of a house based on various features and attributes. The problem is typically approached as a regression problem, where the target variable is the price of the house, and the features are various attributes of the house.

The features used in house price prediction can include both quantitative and categorical variables, such as the number of bedrooms, house area, bedrooms, furnished, nearness to main road, and various amenities such as a garage and other factors that may influence the value of the property.

Accurate predictions can help agents and appraisers price homes correctly, while homeowners can use the predictions to set a reasonable asking price for their properties. Accurate house price prediction can also be useful for buyers who are looking to make informed decisions about purchasing a property and obtaining a fair price for their investment.

1.2 Attribute Information:

Name - Description 1- Price-Prices of the houses 2- Area- Area of the houses 3- Bedrooms- No of house bedrooms 4- Bathrooms- No of bathrooms 5- Stories- No of house stories 6- Main Road-Weather connected to Main road 7- Guestroom-Weather has a guest room 8- Basement-Weather has a basement 9- Hot water heating- Weather has a hot water heater 10-Airconditioning-Weather has a air conditioner 11-Parking- No of house parking 12-Furnishing Status-Furnishing status of house

1.3 Building a Regression Model

Download the dataset: Dataset

Load the dataset into the tool.

Perform Below Visualizations.

Univariate Analysis

Bi-Variate Analysis

Multi-Variate Analysis

```
</li>
<li>Perform descriptive statistics on the dataset.</li>
<li>Check for Missing values and deal with them.</li>
<li>Find the outliers and replace them outliers</li>
<li>Check for Categorical columns and perform encoding.</li>
<li>Split the data into dependent and independent variables.</li>
<li>Scale the independent variables</li>
<li>Split the data into training and testing</li>
<li>Build the Model</li>
<li>Train the Model</li>
<li>Test the Model</li>
<li>Measure the performance using Metrics.</li>
```

1.3.1 Importing necessary libraries

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

1.3.2 1. Download the dataset: Dataset

Housing.csv downloaded.

1.3.3 2. Load the dataset into the tool

[2]:
```
data = pd.read_csv('Housing.csv')
data.head()
```

[2]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	furnishingstatus
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	furnished
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	furnished
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	semi-furnished
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	furnished
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	furnished

[3]:
```
data.isnull().any()
```

[3]:
```
price             False
area              False
bedrooms          False
```

```
bathrooms           False
stories             False
mainroad            False
guestroom           False
basement            False
hotwaterheating     False
airconditioning     False
parking             False
furnishingstatus    False
dtype: bool
```

[4]:
```
data.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
```

1.3.4 3. Perform Below Visualizations.

Univariate Analysis

Bi-Variate Analysis

Multi-Variate Analysis

Univariate Analysis

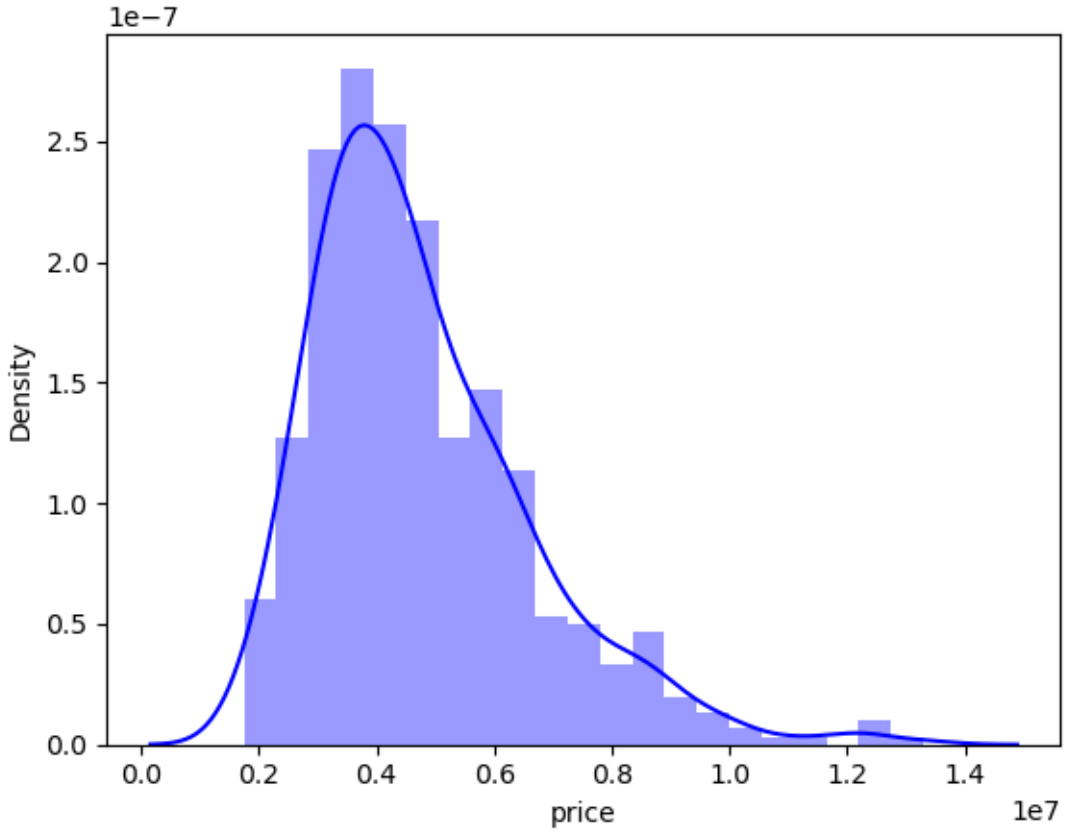
Distribution plot

[5]:
```
sns.distplot(data['price'], color = 'b')
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_10415/972929096.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(data['price'], color = 'b')
```

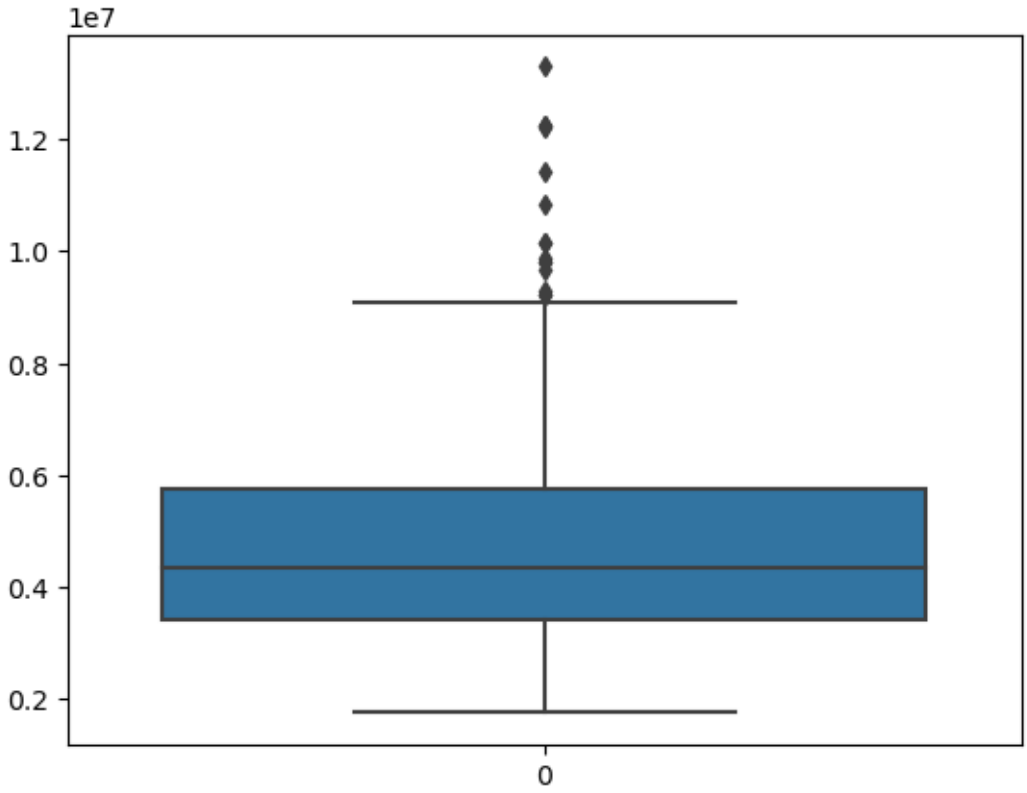
[5]: <Axes: xlabel='price', ylabel='Density'>



Box plot

[6]:
```
sns.boxplot(data['price'])
```

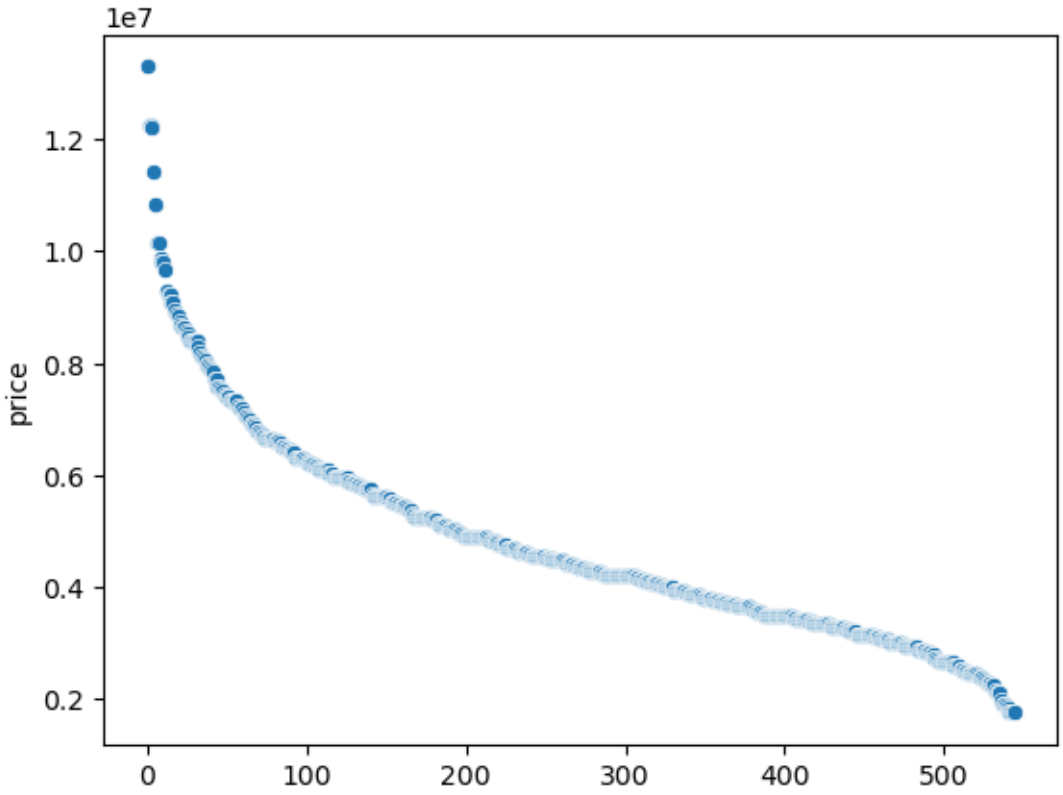
[6]: <Axes: >



Scatter plot

[7]:
```
sns.scatterplot(data['price'])
```

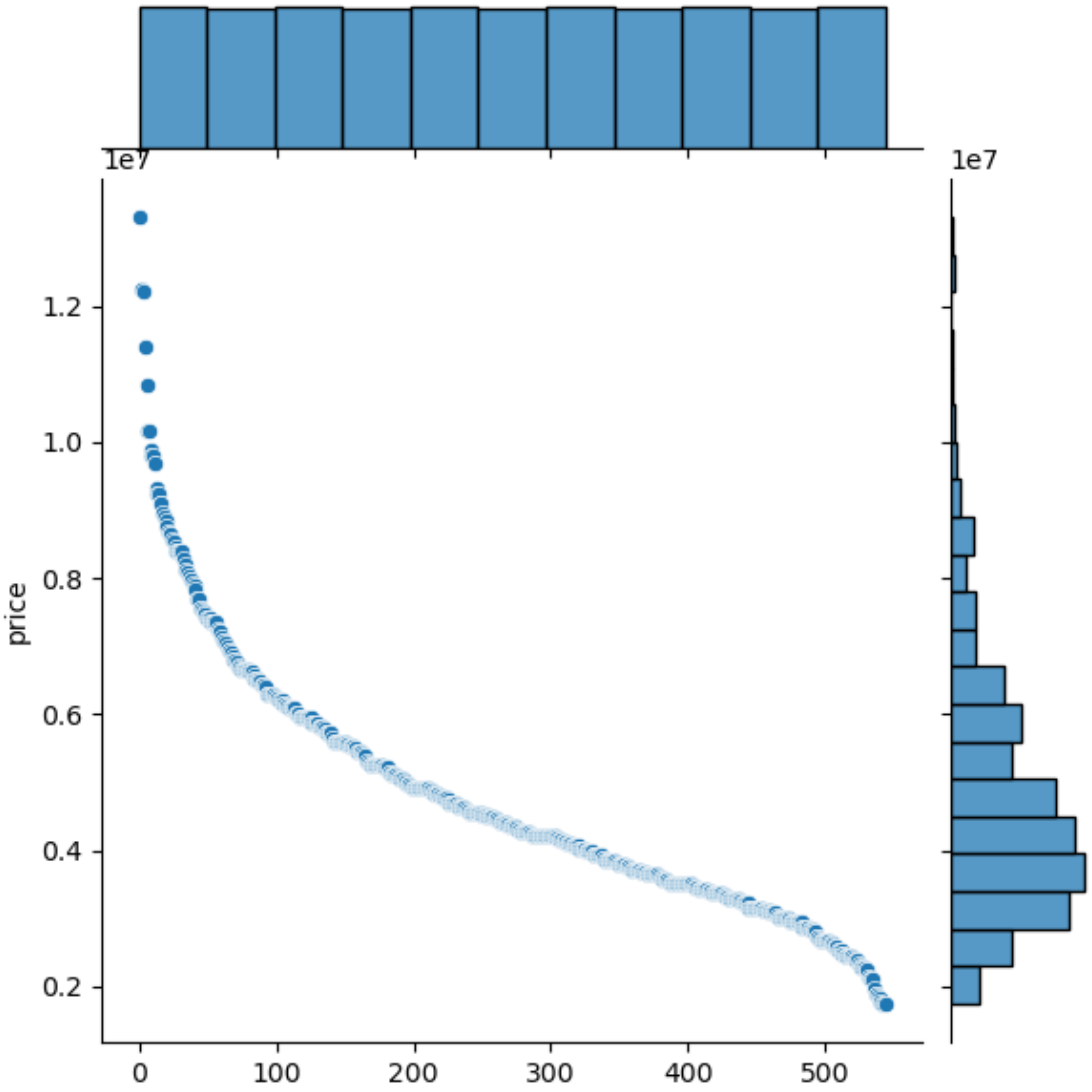
[7]:
```
<Axes: ylabel='price'>
```



Joint plot

[8]:
```
sns.jointplot(data['price'])
```

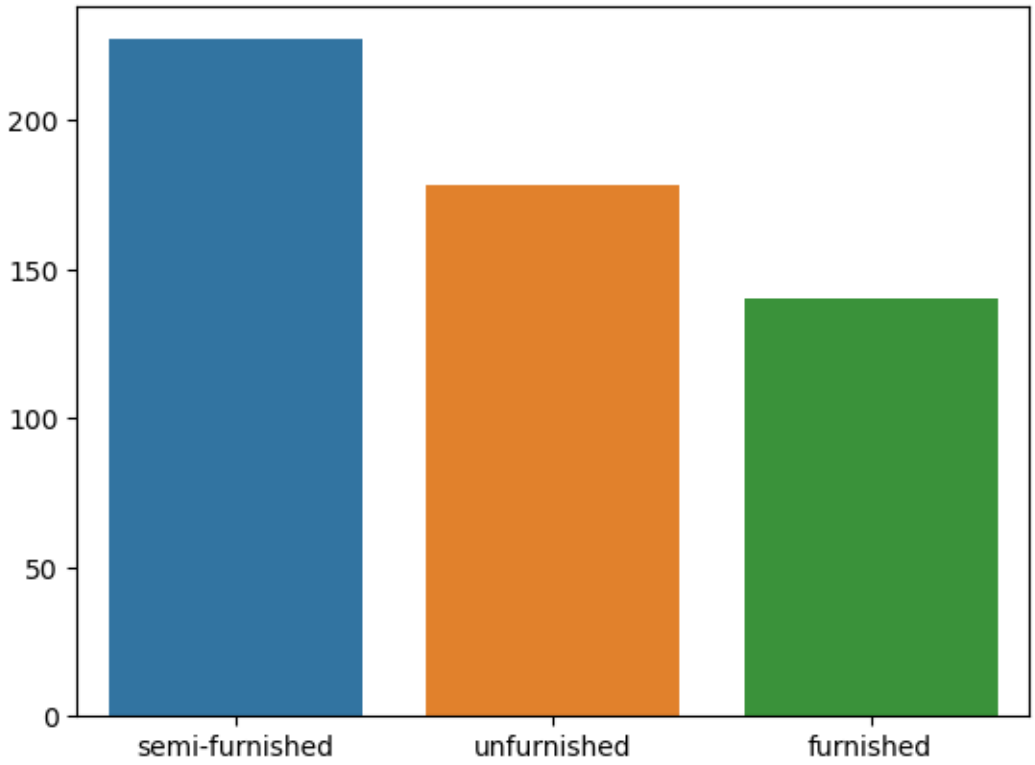
[8]:
```
<seaborn.axisgrid.JointGrid at 0x1429800d0>
```



Bar plot

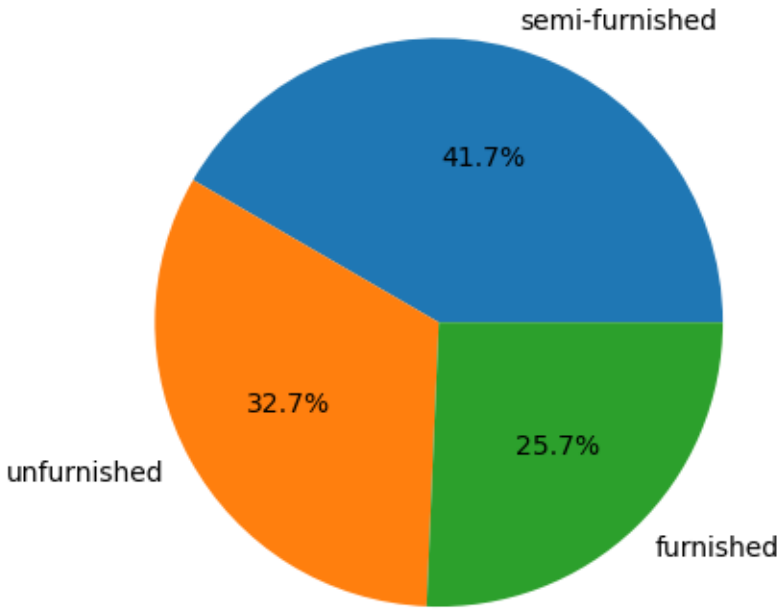
[9]:
```
x = data.furnishingstatus.value_counts()
sns.barplot(x=x.index, y=x.values)
```

[9]: <Axes: >



Pie plot

[10]:
```
x = data['furnishingstatus'].value_counts()
plt.pie(x.values,
        labels=x.index,
        autopct='%1.1f%%')
plt.show()
```

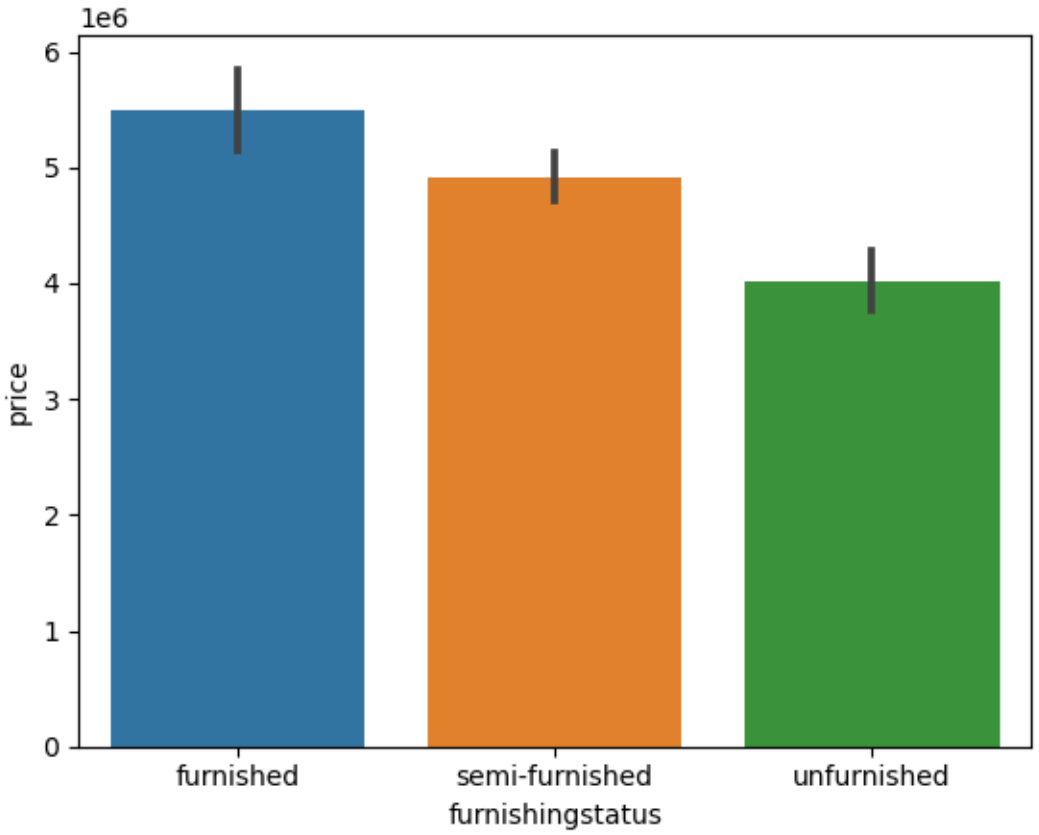


Bivariate analysis

Bar plot

[11]:
```
sns.barplot(x=data.furnishingstatus, y=data.price)
```

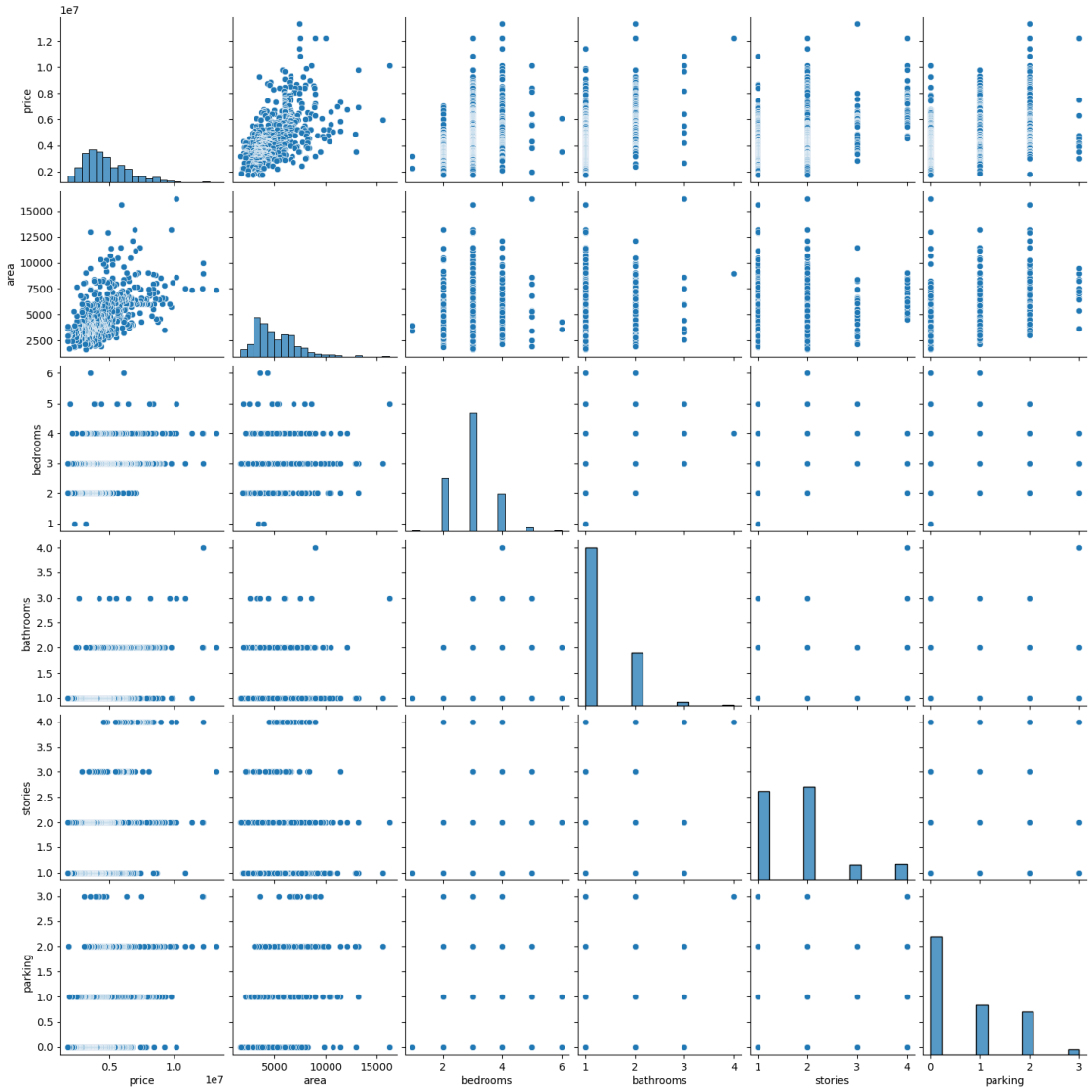
[11]:
```
<Axes: xlabel='furnishingstatus', ylabel='price'>
```



Pair plot

[12]:
```
sns.pairplot(data)
```

[12]:
```
<seaborn.axisgrid.PairGrid at 0x142d60820>
```

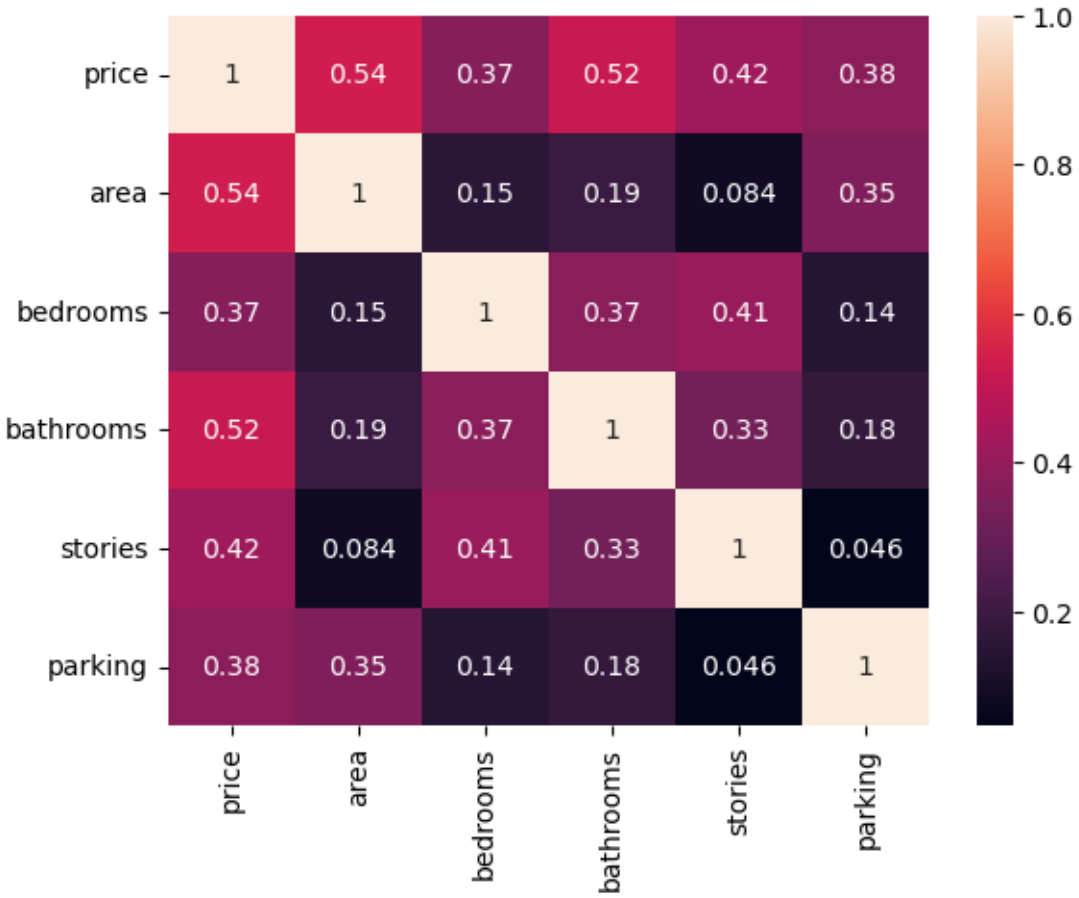


Multivariate Analysis

[13]:
```
sns.heatmap(data.corr(), annot=True)
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_10415/1119197534.py:1
: FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(data.corr(), annot=True)
```

[13]: <Axes: >



1.3.5 4. Perform descriptive statistics on the dataset.

Measure of central tendency - Mean, Median and Mode

[14]:
```
data.mean()
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_10415/531903386.py:1:
FutureWarning: The default value of numeric_only in DataFrame.mean is
deprecated. In a future version, it will default to False. In addition,
specifying 'numeric_only=None' is deprecated. Select only valid columns or
specify the value of numeric_only to silence this warning.
  data.mean()
```

[14]:
```
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
```

[15]:
```
data.median()
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_10415/4184645713.py:1
: FutureWarning: The default value of numeric_only in DataFrame.median is
deprecated. In a future version, it will default to False. In addition,
specifying 'numeric_only=None' is deprecated. Select only valid columns or
specify the value of numeric_only to silence this warning.
  data.median()
```

[15]:
```
price        4340000.0
area            4600.0
bedrooms           3.0
bathrooms          1.0
stories            2.0
parking            0.0
dtype: float64
```

[16]:
```
data.mode()
```

[16]:
```
      price    area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0  3500000  6000.0       3.0        1.0      2.0      yes         no       no
1  4200000     NaN       NaN        NaN      NaN      NaN       NaN      NaN

  hotwaterheating airconditioning  parking furnishingstatus
0              no              no      0.0   semi-furnished
1             NaN             NaN      NaN              NaN
```

Measure of variability:

Kurtosis

[17]:
```
data.kurt()
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_10415/2907027414.py:1
: FutureWarning: The default value of numeric_only in DataFrame.kurt is
deprecated. In a future version, it will default to False. In addition,
specifying 'numeric_only=None' is deprecated. Select only valid columns or
specify the value of numeric_only to silence this warning.
  data.kurt()
```

[17]:
```
price        1.960130
area         2.751480
bedrooms     0.728323
bathrooms    2.164856
stories      0.679404
parking     -0.573063
dtype: float64
```

Range

[18]:
```
data.max()
```

[18]:
```
price                 13300000
area                     16200
bedrooms                     6
bathrooms                    4
stories                      4
mainroad                   yes
guestroom                  yes
basement                   yes
hotwaterheating            yes
airconditioning            yes
parking                      3
furnishingstatus    unfurnished
dtype: object
```

[19]:
```
data.min()
```

[19]:
```
price                1750000
area                    1650
bedrooms                   1
bathrooms                  1
stories                    1
mainroad                  no
guestroom                 no
basement                  no
hotwaterheating           no
airconditioning           no
parking                    0
furnishingstatus    furnished
dtype: object
```

[20]:
```
Range = data.max()['price'] - data.min()['price']
print(Range)
```

```
11550000
```

Skewness

[21]:
```
data.skew()
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_10415/1188251951.py:1
: FutureWarning: The default value of numeric_only in DataFrame.skew is
deprecated. In a future version, it will default to False. In addition,
specifying 'numeric_only=None' is deprecated. Select only valid columns or
specify the value of numeric_only to silence this warning.
  data.skew()
```

[21]:
```
price         1.212239
area          1.321188
bedrooms      0.495684
bathrooms     1.589264
stories       1.082088
parking       0.842062
dtype: float64
```

Interquartile range - for price

[22]:
```
quantiles = data['price'].quantile(q=[0.75, 0.25])
quantiles
```

[22]:
```
0.75    5740000.0
0.25    3430000.0
Name: price, dtype: float64
```

[23]:
```
#Q3
quantiles.iloc[0]
```

[23]:
```
5740000.0
```

[24]:
```
#Q1
quantiles.iloc[1]
```

[24]:
```
3430000.0
```

[25]:
```
IQR = quantiles.iloc[0]-quantiles.iloc[1]
IQR
```

[25]:
```
2310000.0
```

Upper extreme Q3 + 1.5*IQR

[26]:
```
quantiles.iloc[0] + (1.5*IQR)
```

[26]:
```
9205000.0
```

Lower extreme Q1 - 1.5*IQR

[27]:
```
quantiles.iloc[1] - (1.5*IQR)
```

[27]:
```
-35000.0
```

Standard deviation

[28]:
```
data.std()
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_10415/2723740006.py:1
: FutureWarning: The default value of numeric_only in DataFrame.std is
deprecated. In a future version, it will default to False. In addition,
```

```
specifying 'numeric_only=None' is deprecated. Select only valid columns or
specify the value of numeric_only to silence this warning.
  data.std()
```

[28]:
```
price        1.870440e+06
area         2.170141e+03
bedrooms     7.380639e-01
bathrooms    5.024696e-01
stories      8.674925e-01
parking      8.615858e-01
dtype: float64
```

Variance

[29]:
```
data.var()
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_10415/445316826.py:1:
FutureWarning: The default value of numeric_only in DataFrame.var is deprecated.
In a future version, it will default to False. In addition, specifying
'numeric_only=None' is deprecated. Select only valid columns or specify the
value of numeric_only to silence this warning.
  data.var()
```

[29]:
```
price        3.498544e+12
area         4.709512e+06
bedrooms     5.447383e-01
bathrooms    2.524757e-01
stories      7.525432e-01
parking      7.423300e-01
dtype: float64
```

[30]:
```
data.describe()
```

[30]:

	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	
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	

```
50%      0.000000
75%      1.000000
max      3.000000
```

1.3.6 5. Check for Missing values and deal with them.

[31]:
```
data.isnull().sum()
```

[31]:
```
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
```

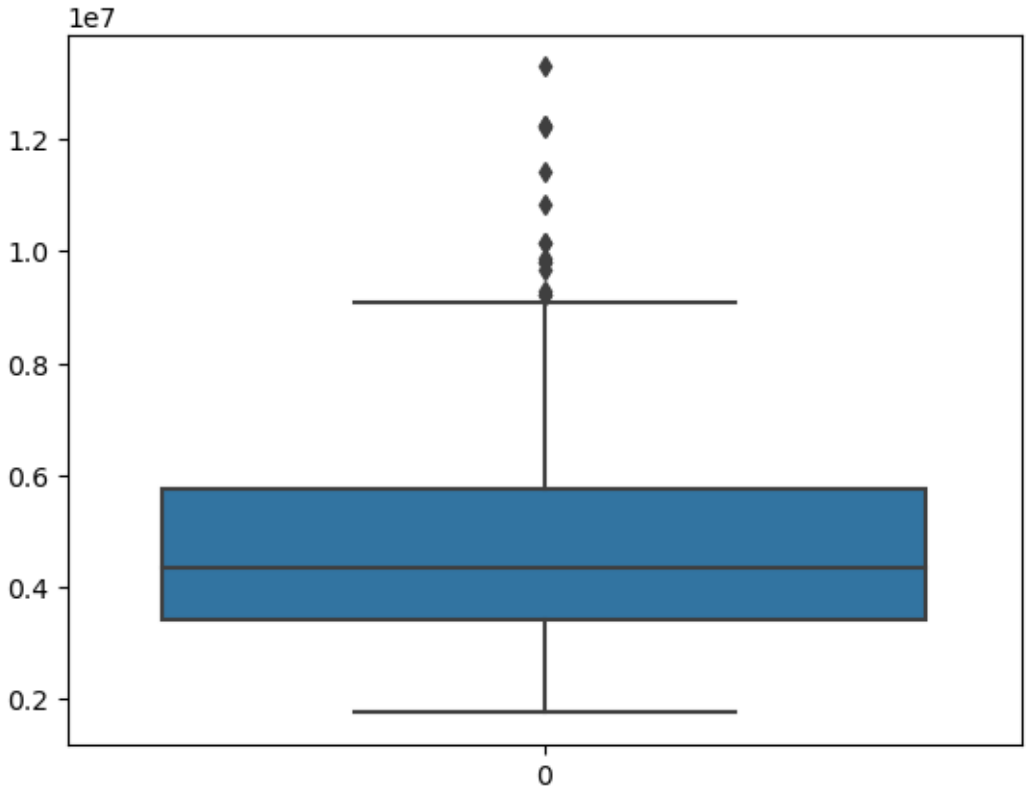
No missing values

1.3.7 6. Find the outliers and replace the outliers

Removing outliers

[32]:
```
sns.boxplot(data.price)
```

[32]:
```
<Axes: >
```



[33]:
```
quant99 = data.price.quantile(0.99)
upper_array = np.where(data.price>quant99)[0]

data.drop(index=upper_array, inplace=True)
```

[34]:
```
data.reset_index(drop = True, inplace=True)
data
```

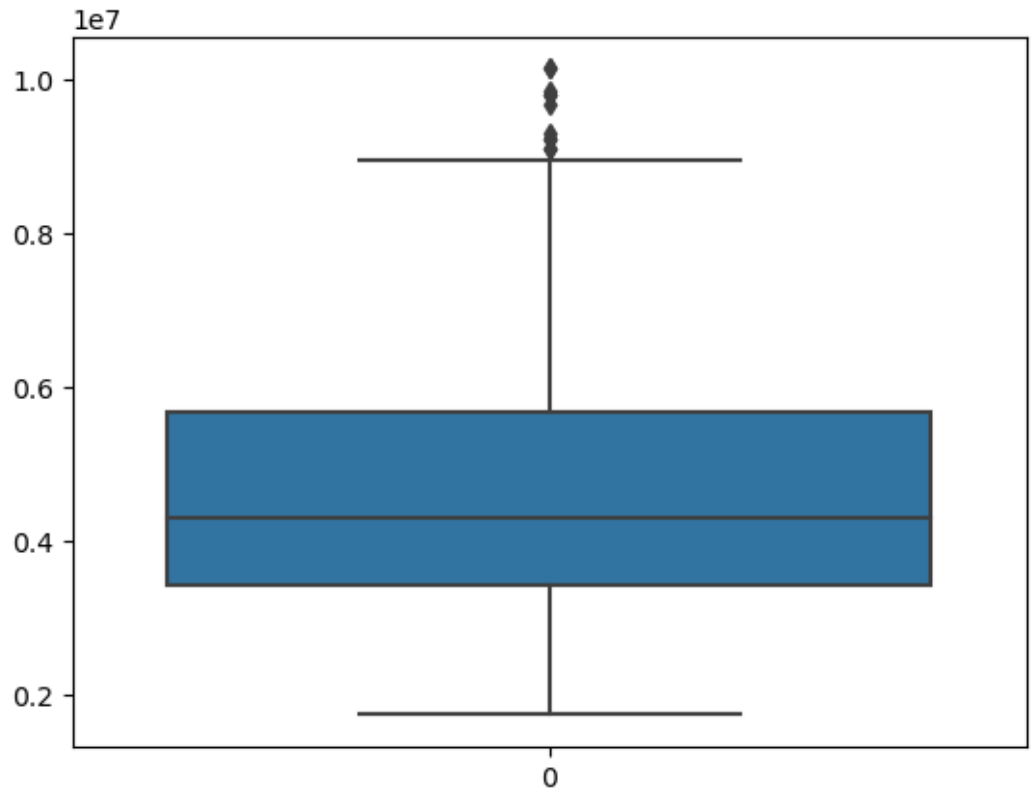
[34]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	\
0	10150000	8580	4	3	4	yes	no	
1	10150000	16200	5	3	2	yes	no	
2	9870000	8100	4	1	2	yes	yes	
3	9800000	5750	3	2	4	yes	yes	
4	9800000	13200	3	1	2	yes	no	
..	…	…	…	…	…	…	…	
534	1820000	3000	2	1	1	yes	no	
535	1767150	2400	3	1	1	no	no	
536	1750000	3620	2	1	1	yes	no	
537	1750000	2910	3	1	1	no	no	
538	1750000	3850	3	1	2	yes	no	

```
    basement hotwaterheating airconditioning  parking furnishingstatus
0         no              no             yes        2   semi-furnished
1         no              no              no        0      unfurnished
2        yes              no             yes        2        furnished
3         no              no             yes        1      unfurnished
4        yes              no             yes        2        furnished
..       …              …              …       …              …
534      yes              no              no        2      unfurnished
535       no              no              no        0   semi-furnished
536       no              no              no        0      unfurnished
537       no              no              no        0        furnished
538       no              no              no        0      unfurnished

[539 rows x 12 columns]
```

[35]:
```
sns.boxplot(data['price'])
```

[35]: <Axes: >



[36]:
```
data['price']
```

[36]:
```
0      10150000
1      10150000
2       9870000
3       9800000
4       9800000
         ...
534     1820000
535     1767150
536     1750000
537     1750000
538     1750000
Name: price, Length: 539, dtype: int64
```

1.3.8 7. Check for Categorical columns and perform encoding

Encoding techniques

Label encoding

[37]:
```
from sklearn.preprocessing import LabelEncoder
```

[38]:
```
le = LabelEncoder()
```

[39]:
```
data.head()
```

[39]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	furnishingstatus
0	10150000	8580	4	3	4	yes	no	no	no	yes	2	semi-furnished
1	10150000	16200	5	3	2	yes	no	no	no	no	0	unfurnished
2	9870000	8100	4	1	2	yes	yes	yes	no	yes	2	furnished
3	9800000	5750	3	2	4	yes	yes	no	no	yes	1	unfurnished
4	9800000	13200	3	1	2	yes	no	yes	no	yes	2	furnished

[40]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 539 entries, 0 to 538
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             539 non-null    int64
 1   area              539 non-null    int64
 2   bedrooms          539 non-null    int64
```

```
 3   bathrooms          539 non-null    int64
 4   stories            539 non-null    int64
 5   mainroad           539 non-null    object
 6   guestroom          539 non-null    object
 7   basement           539 non-null    object
 8   hotwaterheating    539 non-null    object
 9   airconditioning    539 non-null    object
 10  parking            539 non-null    int64
 11  furnishingstatus   539 non-null    object
dtypes: int64(6), object(6)
memory usage: 50.7+ KB
```

[41]:
```
columns = ['mainroad', 'guestroom', 'basement', 'hotwaterheating',␣
 ↪'airconditioning']
for col in columns:
 data[col] = le.fit_transform(data[col])
```

[42]:
```
data.head()
```

[42]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	furnishingstatus
0	10150000	8580	4	3	4	1	0	0	0	1	2	semi-furnished
1	10150000	16200	5	3	2	1	0	0	0	0	0	unfurnished
2	9870000	8100	4	1	2	1	1	1	0	1	2	furnished
3	9800000	5750	3	2	4	1	1	0	0	1	1	unfurnished
4	9800000	13200	3	1	2	1	0	1	0	1	2	furnished

One Hot Encoding

[43]:
```
data = pd.get_dummies(data, columns=['furnishingstatus'])
```

[44]:
```
data
```

[44]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom
0	10150000	8580	4	3	4	1	0
1	10150000	16200	5	3	2	1	0
2	9870000	8100	4	1	2	1	1
3	9800000	5750	3	2	4	1	1
4	9800000	13200	3	1	2	1	0
..	…	…	…	…	…	…	…
534	1820000	3000	2	1	1	1	0
535	1767150	2400	3	1	1	0	0

```
536   1750000   3620          2          1        1         1          0
537   1750000   2910          3          1        1         0          0
538   1750000   3850          3          1        2         1          0

     basement  hotwaterheating  airconditioning  parking  \
0           0                0                1        2
1           0                0                0        0
2           1                0                1        2
3           0                0                1        1
4           1                0                1        2
..        …                …                …       …
534         1                0                0        2
535         0                0                0        0
536         0                0                0        0
537         0                0                0        0
538         0                0                0        0

     furnishingstatus_furnished  furnishingstatus_semi-furnished  \
0                             0                                1
1                             0                                0
2                             1                                0
3                             0                                0
4                             1                                0
..                          …                                …
534                           0                                0
535                           0                                1
536                           0                                0
537                           1                                0
538                           0                                0

     furnishingstatus_unfurnished
0                               0
1                               1
2                               0
3                               1
4                               0
..                            …
534                             1
535                             0
536                             1
537                             0
538                             1

[539 rows x 14 columns]
```

1.3.9 8. Split the data into dependent and independent variables.

Dependent variable

[45]:
```
y = data.loc[:, 'price':'price']
y
```

[45]:
```
        price
0    10150000
1    10150000
2     9870000
3     9800000
4     9800000
..        ...
534   1820000
535   1767150
536   1750000
537   1750000
538   1750000

[539 rows x 1 columns]
```

Independent variable

[46]:
```
X = data.drop(columns=['price'], axis=1)
X
```

[46]:
```
      area  bedrooms  bathrooms  stories  mainroad  guestroom  basement  \
0     8580         4          3        4         1          0         0
1    16200         5          3        2         1          0         0
2     8100         4          1        2         1          1         1
3     5750         3          2        4         1          1         0
4    13200         3          1        2         1          0         1
..     ...       ...        ...      ...       ...        ...       ...
534   3000         2          1        1         1          0         1
535   2400         3          1        1         0          0         0
536   3620         2          1        1         1          0         0
537   2910         3          1        1         0          0         0
538   3850         3          1        2         1          0         0

     hotwaterheating  airconditioning  parking  furnishingstatus_furnished  \
0                  0                1        2                           0
1                  0                0        0                           0
2                  0                1        2                           1
3                  0                1        1                           0
4                  0                1        2                           1
..               ...              ...      ...                         ...
534                0                0        2                           0
535                0                0        0                           0
536                0                0        0                           0
537                0                0        0                           1
538                0                0        0                           0
```

```
     furnishingstatus_semi-furnished  furnishingstatus_unfurnished
0                                  1                             0
1                                  0                             1
2                                  0                             0
3                                  0                             1
4                                  0                             0
..                               …                             …
534                                0                             1
535                                1                             0
536                                0                             1
537                                0                             0
538                                0                             1

[539 rows x 13 columns]
```

1.3.10 9. Scale the independent variables

Scaling StandardScaler –> mean=0 std=1 MinMaxScaler –> scale between 0 to 1

[47]:
```
from sklearn.preprocessing import MinMaxScaler
scale = MinMaxScaler()
```

[48]:
```
name = X.columns
X_scaled = scale.fit_transform(X)
```

[49]:
```
X_scaled
```

[49]:
```
array([[0.47628866, 0.6       , 1.        , …, 0.        , 1.        ,
        0.        ],
       [1.        , 0.8       , 1.        , …, 0.        , 0.        ,
        1.        ],
       [0.44329897, 0.6       , 0.        , …, 1.        , 0.        ,
        0.        ],
       …,
       [0.13539519, 0.2       , 0.        , …, 0.        , 0.        ,
        1.        ],
       [0.08659794, 0.4       , 0.        , …, 1.        , 0.        ,
        0.        ],
       [0.15120275, 0.4       , 0.        , …, 0.        , 0.        ,
        1.        ]])
```

[50]:
```
X = pd.DataFrame(X_scaled, columns=name)
X
```

[50]:
```
       area  bedrooms  bathrooms   stories  mainroad  guestroom  basement  \
0  0.476289       0.6        1.0  1.000000       1.0        0.0       0.0
1  1.000000       0.8        1.0  0.333333       1.0        0.0       0.0
```

```
2    0.443299       0.6        0.0  0.333333       1.0        1.0       1.0
3    0.281787       0.4        0.5  1.000000       1.0        1.0       0.0
4    0.793814       0.4        0.0  0.333333       1.0        0.0       1.0
..        …       …          …       …        …        …       …
534  0.092784       0.2        0.0  0.000000       1.0        0.0       1.0
535  0.051546       0.4        0.0  0.000000       0.0        0.0       0.0
536  0.135395       0.2        0.0  0.000000       1.0        0.0       0.0
537  0.086598       0.4        0.0  0.000000       0.0        0.0       0.0
538  0.151203       0.4        0.0  0.333333       1.0        0.0       0.0

     hotwaterheating  airconditioning   parking  furnishingstatus_furnished  \
0                0.0              1.0  0.666667                         0.0
1                0.0              0.0  0.000000                         0.0
2                0.0              1.0  0.666667                         1.0
3                0.0              1.0  0.333333                         0.0
4                0.0              1.0  0.666667                         1.0
..               …              …       …                        …
534              0.0              0.0  0.666667                         0.0
535              0.0              0.0  0.000000                         0.0
536              0.0              0.0  0.000000                         0.0
537              0.0              0.0  0.000000                         1.0
538              0.0              0.0  0.000000                         0.0

     furnishingstatus_semi-furnished  furnishingstatus_unfurnished
0                                1.0                           0.0
1                                0.0                           1.0
2                                0.0                           0.0
3                                0.0                           1.0
4                                0.0                           0.0
..                               …                          …
534                              0.0                           1.0
535                              1.0                           0.0
536                              0.0                           1.0
537                              0.0                           0.0
538                              0.0                           1.0

[539 rows x 13 columns]
```

1.3.11 10. Split the data into training and testing

Train-Test Split

[51]:
```
from sklearn.model_selection import train_test_split
```

[52]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=0)
```

[53]:
```
X_train
```

[53]:

	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	\
470	0.288660	0.4	0.0	0.333333	1.0	0.0	1.0	
208	0.185567	0.2	0.0	0.000000	1.0	0.0	1.0	
250	0.161512	0.4	0.0	0.333333	1.0	0.0	0.0	
157	0.355670	0.4	0.0	0.000000	1.0	1.0	1.0	
118	0.335052	0.4	0.5	1.000000	1.0	0.0	0.0	
..	…	…	…	…	…	…	…	
70	0.327835	0.4	0.5	0.666667	1.0	0.0	0.0	
277	0.186254	0.6	0.0	0.333333	1.0	0.0	0.0	
9	0.298969	0.6	0.0	0.333333	1.0	0.0	1.0	
359	0.261168	0.2	0.0	0.000000	1.0	0.0	0.0	
192	0.295395	0.4	0.0	0.333333	1.0	0.0	0.0	

	hotwaterheating	airconditioning	parking	furnishingstatus_furnished	\
470	0.0	0.0	0.333333	0.0	
208	0.0	0.0	0.000000	0.0	
250	0.0	0.0	0.666667	1.0	
157	0.0	1.0	0.000000	0.0	
118	0.0	0.0	0.333333	1.0	
..	…	…	…	…	
70	0.0	1.0	0.000000	1.0	
277	0.0	0.0	0.000000	1.0	
9	0.0	0.0	0.666667	0.0	
359	0.0	0.0	0.000000	1.0	
192	0.0	1.0	0.000000	0.0	

	furnishingstatus_semi-furnished	furnishingstatus_unfurnished
470	0.0	1.0
208	0.0	1.0
250	0.0	0.0
157	1.0	0.0
118	0.0	0.0
..	…	…
70	0.0	0.0
277	0.0	0.0
9	1.0	0.0
359	0.0	0.0
192	1.0	0.0

[431 rows x 13 columns]

[54]:
```
y_train
```

[54]:

	price
470	2940000
208	4865000
250	4480000

```
157  5425000
118  5950000
..       …
70   6650000
277  4270000
9    9100000
359  3703000
192  4935000

[431 rows x 1 columns]
```

[55]:
```
X_test
```

[55]:
```
          area  bedrooms  bathrooms   stories  mainroad  guestroom  basement  \
172  0.373540       0.4        0.0  0.000000       1.0        1.0       1.0
469  0.092784       0.2        0.0  0.333333       1.0        0.0       0.0
196  0.169759       0.2        0.0  0.000000       1.0        0.0       1.0
417  0.144330       0.4        0.0  0.000000       1.0        0.0       0.0
535  0.051546       0.4        0.0  0.000000       0.0        0.0       0.0
..        …       …         …        …         …          …        …
494  0.079038       0.4        0.0  0.000000       1.0        0.0       0.0
225  0.183505       0.4        0.0  0.000000       1.0        0.0       0.0
337  0.167010       0.2        0.0  0.000000       1.0        0.0       0.0
318  0.195876       0.4        0.0  0.333333       0.0        0.0       1.0
10   0.340206       0.6        0.5  0.333333       1.0        1.0       1.0

     hotwaterheating  airconditioning   parking  furnishingstatus_furnished  \
172              0.0              0.0  0.666667                         0.0
469              0.0              0.0  0.000000                         0.0
196              0.0              0.0  0.333333                         0.0
417              0.0              0.0  0.000000                         0.0
535              0.0              0.0  0.000000                         0.0
..               …                …       …                           …
494              0.0              0.0  0.000000                         0.0
225              0.0              0.0  0.000000                         0.0
337              0.0              0.0  0.000000                         0.0
318              0.0              1.0  0.000000                         0.0
10               0.0              1.0  0.333333                         0.0

     furnishingstatus_semi-furnished  furnishingstatus_unfurnished
172                              1.0                           0.0
469                              1.0                           0.0
196                              1.0                           0.0
417                              0.0                           1.0
535                              1.0                           0.0
..                               …                           …
494                              0.0                           1.0
```

```
225                             1.0                             0.0
337                             1.0                             0.0
318                             1.0                             0.0
10                              0.0                             1.0

[108 rows x 13 columns]
```

[56]:
```
y_test
```

[56]:
```
       price
172  5229000
469  2961000
196  4900000
417  3360000
535  1767150
..       …
494  2660000
225  4690000
337  3850000
318  4007500
10   9100000

[108 rows x 1 columns]
```

1.3.12 11. Build the Model

[57]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

1.3.13 12. Train the Model

[58]:
```
#train the model
lr.fit(X_train,y_train)
```

[58]:
```
LinearRegression()
```

1.3.14 13. Test the Model

[59]:
```
#test the model
y_pred=lr.predict(X_test)
```

[60]:
```
y_pred #prediction
```

[60]:
```
array([[5281792.],
       [3493888.],
       [3919872.],
       [2830336.],
       [2392064.],
```

[4204544.],
[2787328.],
[4632576.],
[4763648.],
[5349376.],
[4687872.],
[6633472.],
[2289664.],
[3727360.],
[5038080.],
[4087808.],
[2799616.],
[2863104.],
[3217408.],
[5285888.],
[4990976.],
[3926016.],
[6561792.],
[2770944.],
[5769216.],
[3248128.],
[3549184.],
[3889152.],
[6043648.],
[6610944.],
[5998592.],
[5677056.],
[5738496.],
[3211264.],
[6684672.],
[4429824.],
[2723840.],
[4894720.],
[4349952.],
[4374528.],
[6273024.],
[3588096.],
[4759552.],
[5040128.],
[6402048.],
[2598912.],
[6078464.],
[5457920.],
[3825664.],
[6146048.],
[3424256.],
[5564416.],

[7649280.],
[3786752.],
[4186112.],
[7493632.],
[6588416.],
[4704256.],
[5550080.],
[1939456.],
[2977792.],
[5568512.],
[5128192.],
[5244928.],
[5376000.],
[5025792.],
[5046272.],
[4067328.],
[6256640.],
[4847616.],
[3174400.],
[4427776.],
[6277120.],
[4288512.],
[3284992.],
[2699264.],
[4919296.],
[3659776.],
[4370432.],
[2666496.],
[3506176.],
[3354624.],
[6205440.],
[3321856.],
[4241408.],
[2574336.],
[8169472.],
[2975744.],
[6414336.],
[5517312.],
[7084032.],
[3295232.],
[3235840.],
[7262208.],
[5378048.],
[4493312.],
[4532224.],
[6045696.],
[2826240.],

```
       [5965824.],
       [2795520.],
       [3303424.],
       [3530752.],
       [2603008.],
       [3428352.],
       [3266560.],
       [4796416.],
       [6889472.]])
```

[61]:
```
y_test # Actual outcome
```

[61]:
```
       price
172  5229000
469  2961000
196  4900000
417  3360000
535  1767150
..       …
494  2660000
225  4690000
337  3850000
318  4007500
10   9100000

[108 rows x 1 columns]
```

1.3.15 14. Measure the performance using Metrics.

[62]:
```
from sklearn.metrics import mean_squared_error,r2_score, mean_absolute_error
```

Error

[63]:
```
error=y_test-y_pred
```

[64]:
```
error
```

[64]:
```
         price
172   -52792.0
469  -532888.0
196   980128.0
417   529664.0
535  -624914.0
..         …
494    56992.0
225  1261648.0
337   583440.0
318  -788916.0
```

```
10   2210528.0

[108 rows x 1 columns]
```

Square error

[65]:
```
se=error*error
```

[66]:
```
se
```

[66]:
```
             price
172  2.786995e+09
469  2.839696e+11
196  9.606509e+11
417  2.805440e+11
535  3.905175e+11
..            …
494  3.248088e+09
225  1.591756e+12
337  3.404022e+11
318  6.223885e+11
10   4.886434e+12

[108 rows x 1 columns]
```

Mean square error

[67]:
```
mse=np.mean(se)
```

```
/Users/akashr/anaconda3/lib/python3.10/site-
packages/numpy/core/fromnumeric.py:3430: FutureWarning: In a future version,
DataFrame.mean(axis=None) will return a scalar mean over the entire DataFrame.
To retain the old behavior, use 'frame.mean(axis=0)' or just 'frame.mean()'
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
```

[68]:
```
mse
```

[68]:
```
price    1.003011e+12
dtype: float64
```

[69]:
```
mse2=mean_squared_error(y_test,y_pred)
```

[70]:
```
mse2
```

[70]:
```
1003011179983.2963
```

Mean absolute eroor

[71]:
```
mae=mean_absolute_error(y_test,y_pred)
```

[72]:
```
mae
```

[72]: 785114.574074074

[73]:
```
rmse=np.sqrt(mse2)
```

[74]:
```
rmse
```

[74]: 1001504.4582942686

R2 Score

[75]:
```
acc=r2_score(y_pred,y_test)
acc
```

[75]: 0.4931216347655545