Name : Mohanty Pratham

Reg.No: 20BCE7176.

ASSIGNMENT-3

Problem Statement: House Price Prediction

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.

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

Building a Regression Model

1. Download the dataset: Dataset

```
[7] from google.colab import files
    uploaded = files.upload()
```

Choose Files Housing.csv

- **Housing.csv**(text/csv) - 28209 bytes, last modified: 6/7/2023 - 100% done

Saving Housing.csv to Housing (1).csv

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

2. Load the dataset into the tool.

2. Load the dataset into the tool.

```
[11] df=pd.read_csv('Housing.csv')
```

```
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
```

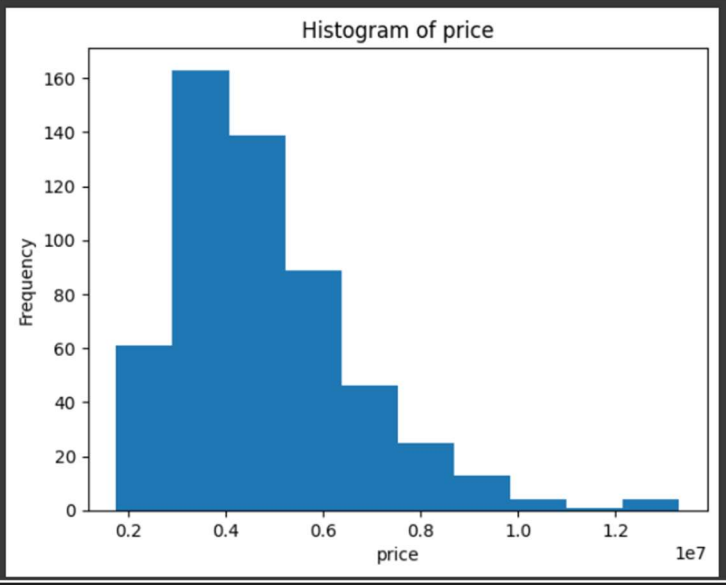
```
df.head()
```

	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. Perform Below Visualizations.

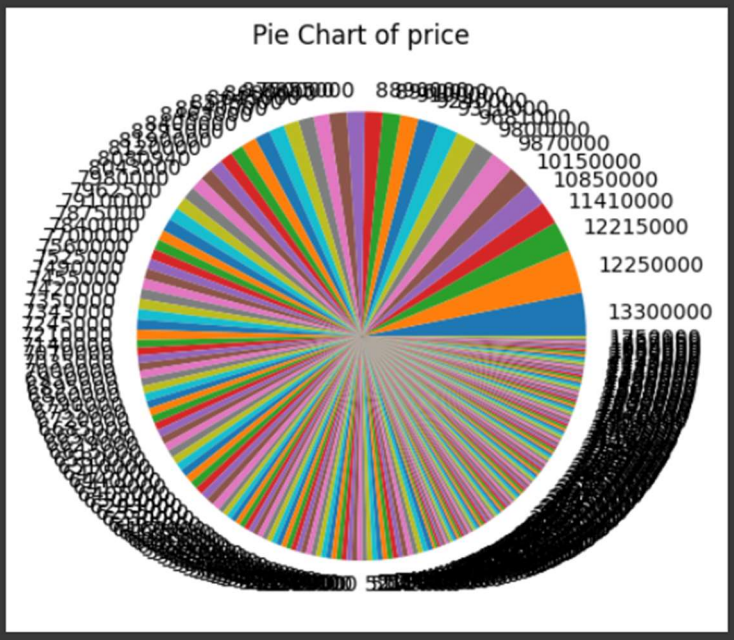
1. Univariate Analysis

```
plt.hist(df['price'], bins=10)
plt.title('Histogram of price')
plt.xlabel('price')
plt.ylabel('Frequency')
plt.show()
```



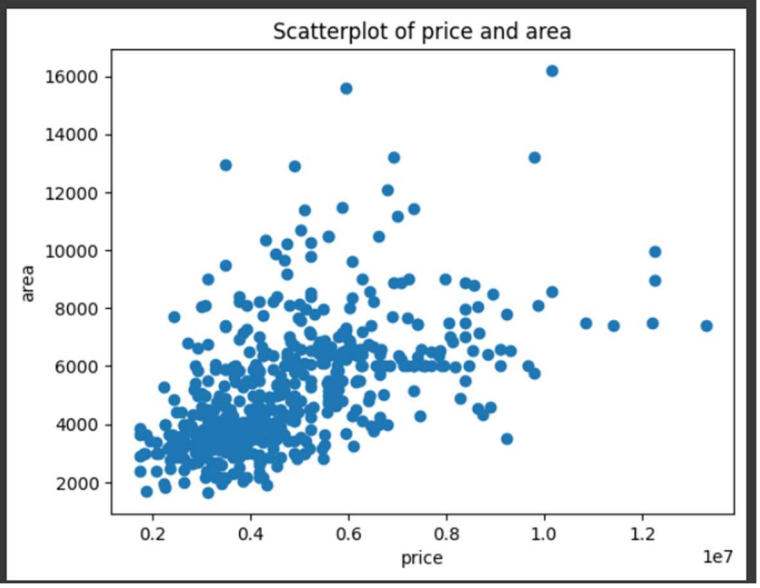

```
[15] # Boxplot
plt.boxplot(df['price'])
plt.title('Boxplot of price')
plt.xlabel('price')
plt.ylabel('Value')
plt.show()
```

```
[16] #Pie Chart
     plt.pie(df['price'].value_counts(), labels=df['price'].unique())
     plt.title('Pie Chart of price')
     plt.show()
```

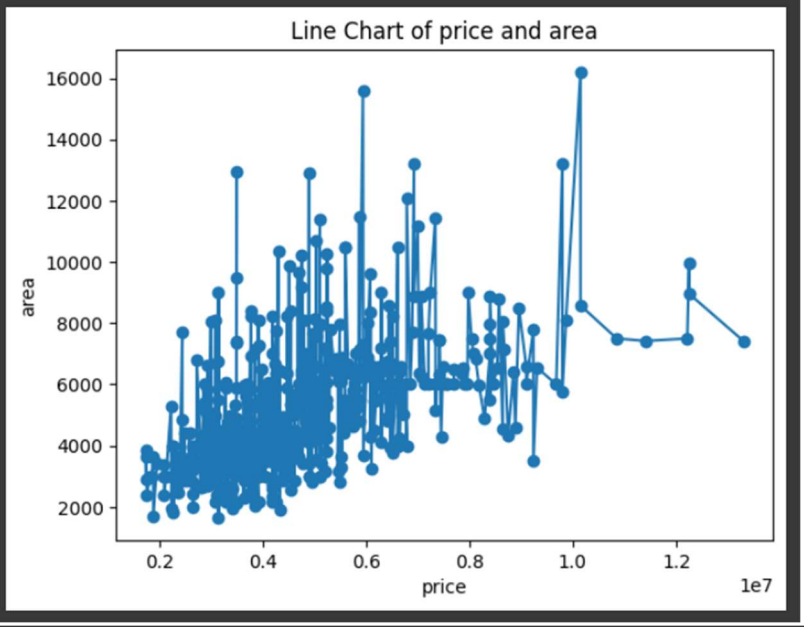


Bivariate analysis

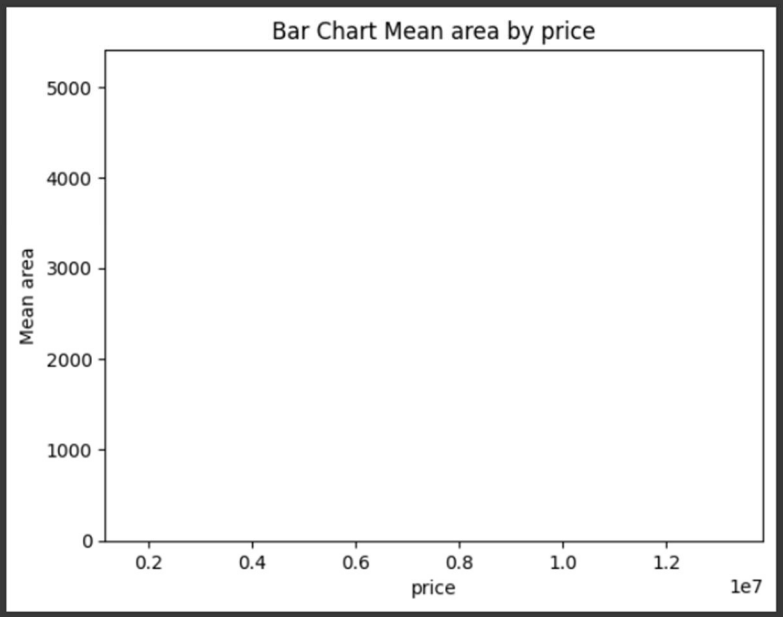
```
# Scatterplot
plt.scatter(df['price'], df['area'])
plt.title('Scatterplot of price and area')
plt.xlabel('price')
plt.ylabel('area')
plt.show()
```



[18]
```
# Line chart
plt.plot(df['price'], df['area'], 'o-')
plt.title('Line Chart of price and area')
plt.xlabel('price')
plt.ylabel('area')
plt.show()
```



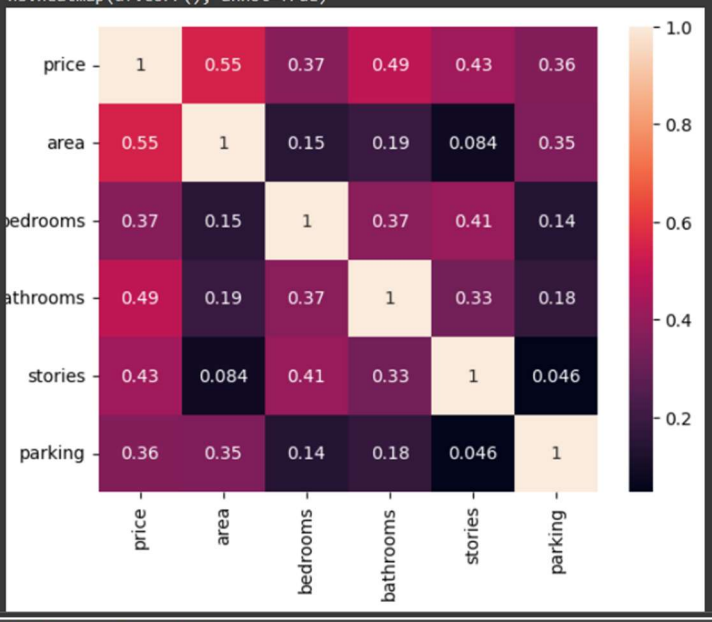
[19]
```
# Bar chart
plt.bar(df['price'].unique(), df['area'].mean(), align='center')
plt.title('Bar Chart Mean area by price')
plt.xlabel('price')
plt.ylabel('Mean area')
plt.show()
```



Multivariate analysis

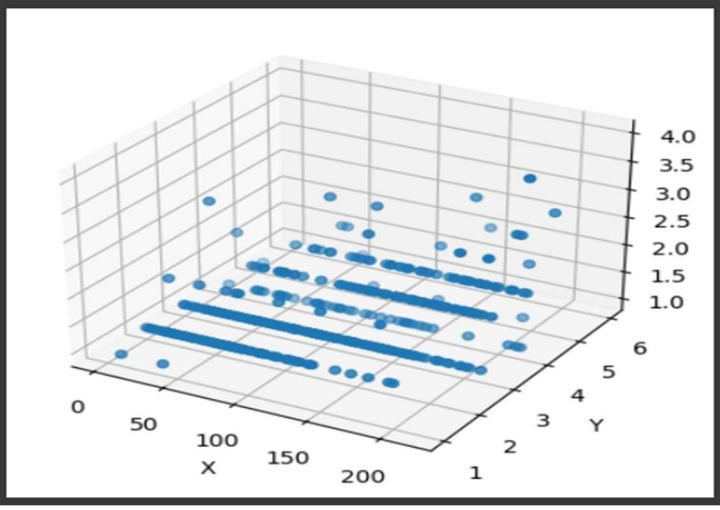
```
df['price'] = df['price'].astype('category').cat.codes
sns.heatmap(df.corr(), annot=True)
plt.show()
```

```
ython-input-20-de1da88191db>:4: FutureWarning: The default value of numeric_o
ns.heatmap(df.corr(), annot=True)
```

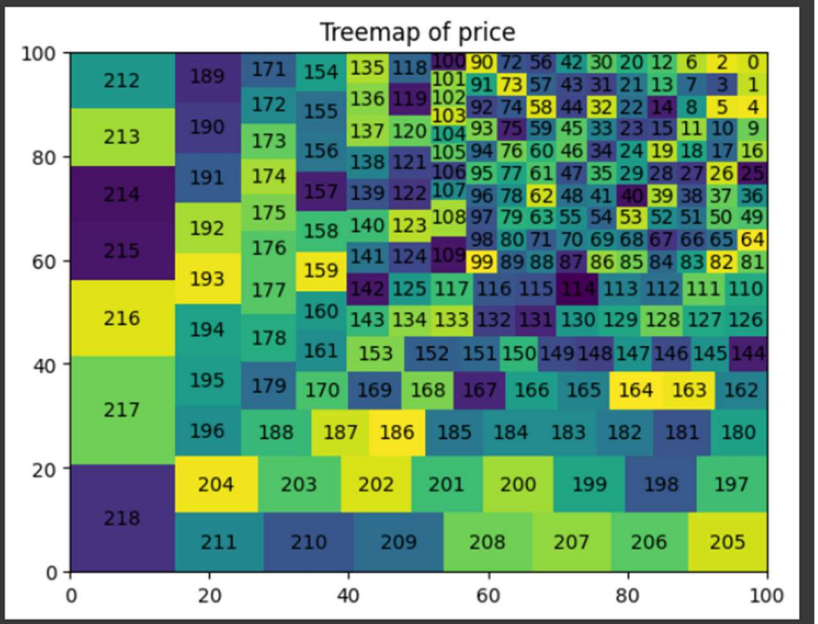


[21]

```
# 3D scatterplot
from mpl_toolkits.mplot3d import Axes3D
x = df['price']
y = df['bedrooms']
z = df['bathrooms']
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x, y, z)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.show()
```



```
[25] # Treemap
     import squarify
     plt.figure()
     squarify.plot(df['price'].value_counts(), label=df['price'].unique())
     plt.title('Treemap of price')
     plt.show()
```



4. Perform descriptive statistics on the dataset.

```
#4. Perform descriptive statistics on the dataset.
df.describe()
```

	price	area	bedrooms	bathrooms	stories	parking
count	545.000000	545.000000	545.000000	545.000000	545.000000	545.000000
mean	95.728440	5150.541284	2.965138	1.286239	1.805505	0.693578
std	56.256108	2170.141023	0.738064	0.502470	0.867492	0.861586
min	0.000000	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	51.000000	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	87.000000	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	137.000000	6360.000000	3.000000	2.000000	2.000000	1.000000
max	218.000000	16200.000000	6.000000	4.000000	4.000000	3.000000

```
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    int16 
 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: int16(1), int64(5), object(6)
memory usage: 48.0+ KB
```

5. Check for Missing values and deal with them.

```
#5. Check for Missing values and deal with them
df.isnull().sum()
```

```
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 them outliers

```
#6. Find the outliers and replace them outliers
target_column = 'price'
Q1 = df[target_column].quantile(0.25)
Q3 = df[target_column].quantile(0.75)
IQR = Q3 - Q1
IQR
```

```
86.0
```

+ Code + Text

```
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
lower_bound
```

```
-78.0
```

```
upper_bound
```

```
266.0
```

```
outliers = df[(df[target_column] < lower_bound) | (df[target_column] > upper_bound)]
median_value = df[target_column].median()
df.loc[(df[target_column] < lower_bound) | (df[target_column] > upper_bound), target_column] = median_value
median_value
```

```
87.0
```

```
df
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	furnishingstatus
0	218	7420	4	2	3	yes	no	no	no	yes	2	furnished
1	217	8960	4	4	4	yes	no	no	no	yes	3	furnished
2	217	9960	3	2	2	yes	no	yes	no	no	2	semi-furnished
3	216	7500	4	2	2	yes	no	yes	no	yes	3	furnished
4	215	7420	4	1	2	yes	yes	yes	no	yes	2	furnished
...	...	...	...	...	...	...	...	...	...	...	...	...
540	2	3000	2	1	1	yes	no	yes	no	no	2	unfurnished
541	1	2400	3	1	1	no	no	no	no	no	0	semi-furnished
542	0	3620	2	1	1	yes	no	no	no	no	0	unfurnished
543	0	2910	3	1	1	no	no	no	no	no	0	furnished
544	0	3850	3	1	2	yes	no	no	no	no	0	unfurnished

545 rows × 12 columns

```
print(df)
```

```
     price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0      218  7420         4          2        3      yes        no       no
1      217  8960         4          4        4      yes        no       no
2      217  9960         3          2        2      yes        no      yes
3      216  7500         4          2        2      yes        no      yes
4      215  7420         4          1        2      yes       yes      yes
..     ...   ...       ...        ...      ...      ...       ...      ...
540      2  3000         2          1        1      yes        no      yes
541      1  2400         3          1        1       no        no       no
542      0  3620         2          1        1      yes        no       no
543      0  2910         3          1        1       no        no       no
544      0  3850         3          1        2      yes        no       no

    hotwaterheating airconditioning  parking furnishingstatus
0                no              yes        2        furnished
1                no              yes        3        furnished
2                no               no        2   semi-furnished
3                no              yes        3        furnished
4                no              yes        2        furnished
..              ...              ...      ...              ...
540              no               no        2      unfurnished
541              no               no        0   semi-furnished
542              no               no        0      unfurnished
543              no               no        0        furnished
544              no               no        0      unfurnished

[545 rows x 12 columns]
```

7. Check for Categorical columns and perform encoding.

```
[42] #7. Check for Categorical columns and perform encoding.
     from sklearn.preprocessing import LabelEncoder
     df.dtypes
```

```
price               int16
area                int64
bedrooms            int64
bathrooms           int64
stories             int64
mainroad           object
guestroom          object
basement           object
hotwaterheating    object
airconditioning    object
parking             int64
furnishingstatus   object
dtype: object
```

```
[43] categorical_columns = df.select_dtypes(include=['object']).columns
     df_encoded = pd.get_dummies(df, columns=categorical_columns)
     categorical_columns
```

```
Index(['mainroad', 'guestroom', 'basement', 'hotwaterheating',
       'airconditioning', 'furnishingstatus'],
      dtype='object')
```

```
print(df_encoded)
```

```
     price  area  bedrooms  bathrooms  stories  parking  mainroad_no  \
0      218  7420         4          2        3        2            0
1      217  8960         4          4        4        3            0
2      217  9960         3          2        2        2            0
3      216  7500         4          2        2        3            0
4      215  7420         4          1        2        2            0
..     ...   ...       ...        ...      ...      ...          ...
540      2  3000         2          1        1        2            0
541      1  2400         3          1        1        0            1
542      0  3620         2          1        1        0            0
543      0  2910         3          1        1        0            1
544      0  3850         3          1        2        0            0

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

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

8. Split the data into dependent and independent variables.

```
#8. Split the data into dependent and independent variables.
dependent_variable = 'price'
independent_variables = df.drop(dependent_variable, axis=1)
dependent_variable = df[dependent_variable]
dependent_variable
```

```
0      218
1      217
2      217
3      216
4      215
      ...
540      2
541      1
542      0
543      0
544      0
Name: price, Length: 545, dtype: int16
```

```
independent_variables
```

	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	furnishingstatus
0	7420	4	2	3	yes	no	no	no	yes	2	furnished
1	8960	4	4	4	yes	no	no	no	yes	3	furnished
2	9960	3	2	2	yes	no	yes	no	no	2	semi-furnished
3	7500	4	2	2	yes	no	yes	no	yes	3	furnished
4	7420	4	1	2	yes	yes	yes	no	yes	2	furnished
...	...	...	...	...	...	...	...	...	...	...	...
540	3000	2	1	1	yes	no	yes	no	no	2	unfurnished
541	2400	3	1	1	no	no	no	no	no	0	semi-furnished
542	3620	2	1	1	yes	no	no	no	no	0	unfurnished
543	2910	3	1	1	no	no	no	no	no	0	furnished
544	3850	3	1	2	yes	no	no	no	no	0	unfurnished

545 rows × 11 columns

```
[47] print(independent_variables)
```

```
     area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0    7420         4          2        3      yes        no       no
1    8960         4          4        4      yes        no       no
2    9960         3          2        2      yes        no      yes
3    7500         4          2        2      yes        no      yes
4    7420         4          1        2      yes       yes      yes
..    ...       ...        ...      ...      ...       ...      ...
540  3000         2          1        1      yes        no      yes
541  2400         3          1        1       no        no       no
542  3620         2          1        1      yes        no       no
543  2910         3          1        1       no        no       no
544  3850         3          1        2      yes        no       no

    hotwaterheating airconditioning  parking furnishingstatus
0                no             yes        2        furnished
1                no             yes        3        furnished
2                no              no        2   semi-furnished
3                no             yes        3        furnished
4                no             yes        2        furnished
..              ...             ...      ...              ...
540              no              no        2      unfurnished
541              no              no        0   semi-furnished
542              no              no        0      unfurnished
543              no              no        0        furnished
544              no              no        0      unfurnished

[545 rows x 11 columns]
```

9. Scale the independent variables

```
#9. Scale the independent variables
from sklearn.preprocessing import StandardScaler
columns_to_scale = ['price', 'bedrooms', 'bathrooms', 'area', 'stories', 'parking']
scaler = StandardScaler()
df[columns_to_scale] = scaler.fit_transform(df[columns_to_scale])
df
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	furnishingstatus
0	2.175477	1.046726	1.403419	1.421812	1.378217	yes	no	no	no	yes	1.517692	furnished
1	2.157685	1.757010	1.403419	5.405809	2.532024	yes	no	no	no	yes	2.679409	furnished
2	2.157685	2.218232	0.047278	1.421812	0.224410	yes	no	yes	no	no	1.517692	semi-furnished
3	2.139893	1.083624	1.403419	1.421812	0.224410	yes	no	yes	no	yes	2.679409	furnished
4	2.122101	1.046726	1.403419	-0.570187	0.224410	yes	yes	yes	no	yes	1.517692	furnished
...	...	...	...	...	...	...	...	...	...	...	...	...
540	-1.667633	-0.991879	-1.308863	-0.570187	-0.929397	yes	no	yes	no	no	1.517692	unfurnished
541	-1.685425	-1.268613	0.047278	-0.570187	-0.929397	no	no	no	no	no	-0.805741	semi-furnished
542	-1.703217	-0.705921	-1.308863	-0.570187	-0.929397	yes	no	no	no	no	-0.805741	unfurnished
543	-1.703217	-1.033389	0.047278	-0.570187	-0.929397	no	no	no	no	no	-0.805741	furnished
544	-1.703217	-0.599839	0.047278	-0.570187	0.224410	yes	no	no	no	no	-0.805741	unfurnished

545 rows × 12 columns

```
[50] print(df)
```

```
        price      area  bedrooms  bathrooms   stories mainroad guestroom  \
0    2.175477  1.046726  1.403419   1.421812  1.378217      yes        no
1    2.157685  1.757010  1.403419   5.405809  2.532024      yes        no
2    2.157685  2.218232  0.047278   1.421812  0.224410      yes        no
3    2.139893  1.083624  1.403419   1.421812  0.224410      yes        no
4    2.122101  1.046726  1.403419  -0.570187  0.224410      yes       yes
..        ...       ...       ...        ...       ...      ...       ...
540 -1.667633 -0.991879 -1.308863  -0.570187 -0.929397      yes        no
541 -1.685425 -1.268613  0.047278  -0.570187 -0.929397       no        no
542 -1.703217 -0.705921 -1.308863  -0.570187 -0.929397      yes        no
543 -1.703217 -1.033389  0.047278  -0.570187 -0.929397       no        no
544 -1.703217 -0.599839  0.047278  -0.570187  0.224410      yes        no

    basement hotwaterheating airconditioning   parking furnishingstatus
0         no              no             yes  1.517692        furnished
1         no              no             yes  2.679409        furnished
2        yes              no              no  1.517692   semi-furnished
3        yes              no             yes  2.679409        furnished
4        yes              no             yes  1.517692        furnished
..       ...             ...             ...       ...              ...
540      yes              no              no  1.517692      unfurnished
541       no              no              no -0.805741   semi-furnished
542       no              no              no -0.805741      unfurnished
543       no              no              no -0.805741        furnished
544       no              no              no -0.805741      unfurnished

[545 rows x 12 columns]
```

10. Split the data into training and testing

```
#10.Split the data into training and testing
from sklearn.model_selection import train_test_split
X = df.drop('price', axis=1)
y = df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
X_train
```

	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	furnishingstatus
167	-0.253922	-1.308863	1.421812	-0.929397	yes	no	no	no	yes	1.517692	semi-furnished
368	0.225750	-1.308863	-0.570187	-0.929397	no	no	no	no	no	-0.805741	semi-furnished
301	-0.752043	0.047278	-0.570187	0.224410	yes	no	no	no	no	-0.805741	semi-furnished
527	-1.528742	-1.308863	-0.570187	-0.929397	no	no	yes	no	no	-0.805741	semi-furnished
382	-0.922695	0.047278	-0.570187	0.224410	yes	no	yes	no	no	-0.805741	furnished
...	...	...	...	...	...	...	...	...	...	...	...
71	0.391790	1.403419	1.421812	2.532024	yes	no	no	no	yes	-0.805741	unfurnished
106	0.138117	1.403419	1.421812	-0.929397	yes	no	yes	no	yes	-0.805741	semi-furnished
270	-0.300045	0.047278	1.421812	1.378217	yes	no	no	yes	no	0.355976	furnished
435	-0.512207	-1.308863	-0.570187	-0.929397	yes	no	no	no	no	-0.805741	unfurnished
102	0.161178	0.047278	1.421812	2.532024	yes	yes	no	no	yes	0.355976	semi-furnished

408 rows × 11 columns

```
X_test
```

	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	furnishingstatus
316	0.345668	1.403419	1.421812	0.224410	no	no	yes	no	no	0.355976	unfurnished
77	0.622401	0.047278	1.421812	1.378217	yes	no	no	no	yes	-0.805741	furnished
360	-0.512207	-1.308863	-0.570187	-0.929397	yes	no	no	no	no	-0.805741	semi-furnished
90	-0.069433	0.047278	-0.570187	0.224410	yes	no	no	no	yes	-0.805741	semi-furnished
493	-0.549105	0.047278	-0.570187	-0.929397	yes	no	no	no	no	-0.805741	furnished
...	...	...	...	...	...	...	...	...	...	...	...
172	1.498725	0.047278	-0.570187	0.224410	yes	yes	yes	no	yes	1.517692	unfurnished
124	0.633932	0.047278	1.421812	2.532024	yes	no	no	no	no	0.355976	furnished
388	-0.692084	0.047278	-0.570187	0.224410	yes	no	no	no	no	-0.805741	unfurnished
521	-0.699002	-1.308863	-0.570187	-0.929397	no	no	no	no	no	-0.805741	unfurnished
503	-0.530656	0.047278	-0.570187	-0.929397	yes	no	no	no	no	-0.805741	semi-furnished

137 rows × 11 columns

```
[53] y_train
```

```
167    0.520805
368   -0.635687
301   -0.262051
527   -1.525296
382   -0.706855
         ...
71     1.285868
106    0.983401
270   -0.155298
435   -0.920362
102    1.001194
Name: price, Length: 408, dtype: float64
```

```
[54] y_test
```

```
316   -0.386596
77     1.232492
360   -0.600102
90     1.125739
493   -1.276205
         ...
172    0.503013
124    0.876648
388   -0.742440
521   -1.454127
503   -1.329582
Name: price, Length: 137, dtype: float64
```

```
[55] from sklearn.preprocessing import LabelEncoder
     le=LabelEncoder()

     df['mainroad']=le.fit_transform(df['mainroad'])
     df['guestroom']=le.fit_transform(df['guestroom'])
     df['basement']=le.fit_transform(df['basement'])
     df['hotwaterheating']=le.fit_transform(df['hotwaterheating'])
     df['airconditioning']=le.fit_transform(df['airconditioning'])
     df['furnishingstatus']=le.fit_transform(df['furnishingstatus'])

[57] df.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	furnishingstatus
0	2.175477	1.046726	1.403419	1.421812	1.378217	1	0	0	0	1	1.517692	0
1	2.157685	1.757010	1.403419	5.405809	2.532024	1	0	0	0	1	2.679409	0
2	2.157685	2.218232	0.047278	1.421812	0.224410	1	0	1	0	0	1.517692	1
3	2.139893	1.083624	1.403419	1.421812	0.224410	1	0	1	0	1	2.679409	0
4	2.122101	1.046726	1.403419	-0.570187	0.224410	1	1	1	0	1	1.517692	0

11. Build the Model

```
[58] #11. Build the Model
     from sklearn.linear_model import LinearRegression
     model=LinearRegression()
     X_train, X_test, y_train, y_test = train_test_split(df, df['price'], test_size=0.25)

[59] model.fit(X_train,y_train)

     ▾ LinearRegression
     LinearRegression()
```

12. Train the Model

```
X_train
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	furnishingstatus
523	-1.471919	-1.090119	1.403419	1.421812	0.224410	1	0	0	0	0	-0.805741	0
80	1.214700	0.391790	0.047278	-0.570187	0.224410	1	0	0	1	0	0.355976	1
500	-1.329582	-1.084123	0.047278	-0.570187	-0.929397	1	0	0	0	0	-0.805741	2
482	-1.205036	-0.922695	0.047278	-0.570187	0.224410	0	0	0	0	0	-0.805741	2
406	-0.778024	0.078158	-1.308863	-0.570187	-0.929397	1	0	0	0	0	0.355976	2
...	...	...	...	...	...	...	...	...	...	...	...	...
483	-1.205036	0.675442	0.047278	-0.570187	0.224410	1	0	0	0	0	-0.805741	1
148	0.698727	0.557830	0.047278	-0.570187	1.378217	1	0	0	0	0	-0.805741	1
136	0.734311	0.115056	1.403419	1.421812	0.224410	1	0	0	0	1	1.517692	2
143	0.698727	-0.161678	2.759560	1.421812	1.378217	0	0	1	1	0	-0.805741	2
22	1.873010	1.337297	0.047278	-0.570187	-0.929397	1	1	1	0	1	0.355976	0

408 rows × 12 columns

```
[61] y_train

523   -1.471919
80     1.214700
500   -1.329582
482   -1.205036
406   -0.778024
          ...
483   -1.205036
148    0.698727
136    0.734311
143    0.698727
22     1.873010
Name: price, Length: 408, dtype: float64
```

13. Test the Model

```
[62] score = model.score(X_test, y_test)
```

X_test

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	furnishingstatus
274	-0.155298	0.599340	1.403419	-0.570187	0.224410	1	0	0	0	0	-0.805741	1
478	-1.205036	-0.715145	0.047278	-0.570187	0.224410	0	0	0	0	0	0.355976	2
244	0.022624	0.078158	0.047278	-0.570187	0.224410	1	1	1	0	0	-0.805741	1
192	0.378467	0.668524	0.047278	-0.570187	-0.929397	1	1	1	0	0	-0.805741	0
303	-0.262051	-0.300045	0.047278	-0.570187	-0.929397	1	0	1	0	0	-0.805741	0
...	...	...	...	...	...	...	...	...	...	...	...	...
117	0.876648	-0.669023	1.403419	-0.570187	0.224410	1	1	0	0	1	-0.805741	0
341	-0.493349	-0.908859	-1.308863	1.421812	-0.929397	1	0	1	0	0	1.517692	1
538	-1.632049	-0.692545	-1.308863	-0.570187	-0.929397	1	0	0	0	0	-0.805741	2
182	0.431844	-0.802777	0.047278	-0.570187	0.224410	0	0	0	0	1	-0.805741	1
343	-0.493349	-0.493758	-1.308863	-0.570187	-0.929397	1	0	0	0	0	-0.805741	1

137 rows × 12 columns

```
y_test

274   -0.155298
478   -1.205036
244    0.022624
192    0.378467
303   -0.262051
          ...
117    0.876648
341   -0.493349
538   -1.632049
182    0.431844
343   -0.493349
Name: price, Length: 137, dtype: float64
```

```
[65] score

1.0
```

```
[67] predictions = model.predict(X_test)
     predictions
```

```
array([-0.15529797, -1.20503648,  0.02262382,  0.37846738, -0.26205104,
       -0.93815381,  1.23249194, -0.83140074,  0.55638916, -1.13386777,
       -1.13386777, -0.19088232,  2.10430867, -0.24425886,  0.09379253,
        0.11158471,  0.02262382,  0.84106402,  0.50301263, -0.31542757,
       -1.59646441, -1.0271147 , -0.29763539,  0.25392213,  0.69872659,
       -1.38295827, -1.15165995,  1.07236233, -1.06269906,  1.74846511,
        0.07600035, -0.92036163,  1.80184164,  1.07236233,  0.50301263,
        0.41405174,  0.21833778,  1.16132322, -1.16945213, -0.49334935,
       -0.92036163, -1.48971134,  1.81963382, -0.13750579, -0.74243985,
        0.73431095,  0.39625956, -0.22646668,  0.62755788,  0.78768748,
       -0.36880411,  0.44963609,  0.69872659,  0.50301263, -0.88477728,
        0.66314223,  0.69872659, -0.10192143, -0.10192143,  1.00119362,
        0.46742827,  1.44599808,  0.71651877,  0.73431095, -1.20503648,
       -0.26205104, -0.27984321, -0.92036163,  1.10794669,  1.12573887,
        0.2005456 , -0.84919292,  0.37846738, -0.457765  ,  2.03313996,
        1.23249194, -1.32958173,  1.41041372, -0.22646668,  0.93002491,
       -0.79581638,  0.2005456 ,  0.78768748,  0.52080481,  0.7698953 ,
       -0.47555718, -0.26205104,  1.0189858 ,  0.93002491, -1.64984094,
        0.93002491, -0.99153035, -1.22282866,  0.07600035, -0.54672589,
       -0.26205104, -0.65347896, -1.18724431, -1.09828341, -0.65347896,
       -1.41854262,  0.25392213, -1.45412698,  0.11158471, -0.457765  ,
       -1.32958173, -0.0485449 ,  2.05093214, -1.22282866,  0.2005456 ,
       -0.90256945,  0.85885619,  1.73067293,  0.32509085, -1.06269906,
       -0.77802421,  0.44963609, -0.36880411, -0.65347896, -0.65347896,
       -0.40438846, -0.84919292, -1.61425658, -0.38659628,  1.07236233,
        1.65950422, -0.03075272, -0.26205104,  0.25392213, -0.84919292,
        1.23249194,  1.97976342,  0.87664837, -0.49334935, -1.63204876,
        0.43184392, -0.49334935])
```

14. Measure the performance using Metrics.

```
[68] #14. Measure the performance using Metrics
     from sklearn.metrics import mean_squared_error,r2_score, mean_absolute_error
     y_pred = model.predict(X_test)
```

```
     error=y_test-y_pred
     error
```

```
274    4.440892e-16
478   -6.661338e-16
244   -3.400058e-16
192   -5.551115e-17
303   -4.440892e-16
              ...
117    2.220446e-16
341   -4.996004e-16
538   -8.881784e-16
182   -1.665335e-16
343   -6.106227e-16
Name: price, Length: 137, dtype: float64
```

```
[71] se=error*error
     se
```

```
274    1.972152e-31
478    4.437343e-31
244    1.156039e-31
192    3.081488e-33
303    1.972152e-31
              ...
117    4.930381e-32
341    2.496005e-31
538    7.888609e-31
182    2.773339e-32
343    3.728600e-31
Name: price, Length: 137, dtype: float64
```

```
[72] mse=np.mean(se)
     mse
```

```
4.411557798949427e-31
```

```
[73] mse2=mean_squared_error(y_test,y_pred)
     mse2
```

```
4.411557798949427e-31
```

```
[74] mae=mean_absolute_error(y_test,y_pred)
     mae
```

```
5.337123687038133e-16
```

```
[75] rmse=np.sqrt(mse2)
     rmse
```

```
6.641955885843738e-16
```

```
[76] r2=r2_score(y_test,y_pred)
     r2
```

```
1.0
```