NAME : PEDAKOTA KEERTHANA
REG NO: 20BCD7245

ADS 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
2. Load the dataset into the tool.

```
[21]: import numpy as np
      import pandas as pd
```

```
[3]: import matplotlib.pyplot as plt
```

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

```
[6]: data
```

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

[7]:

	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.

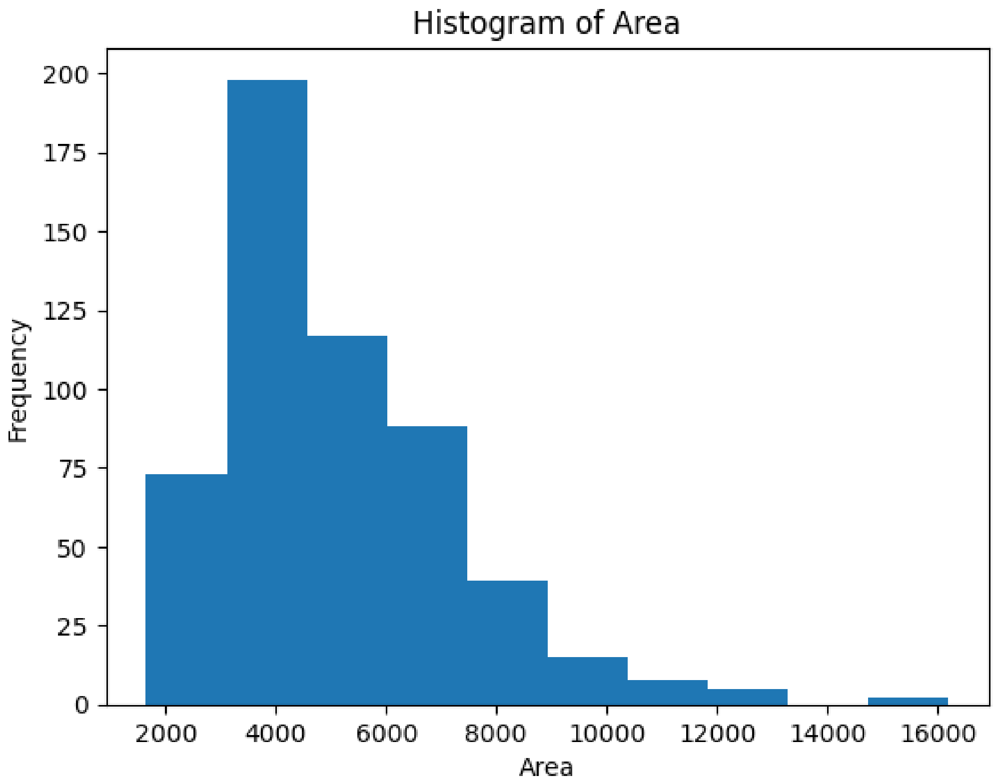
Univariate Analysis

```
[ ]: # Univariate analysis
```

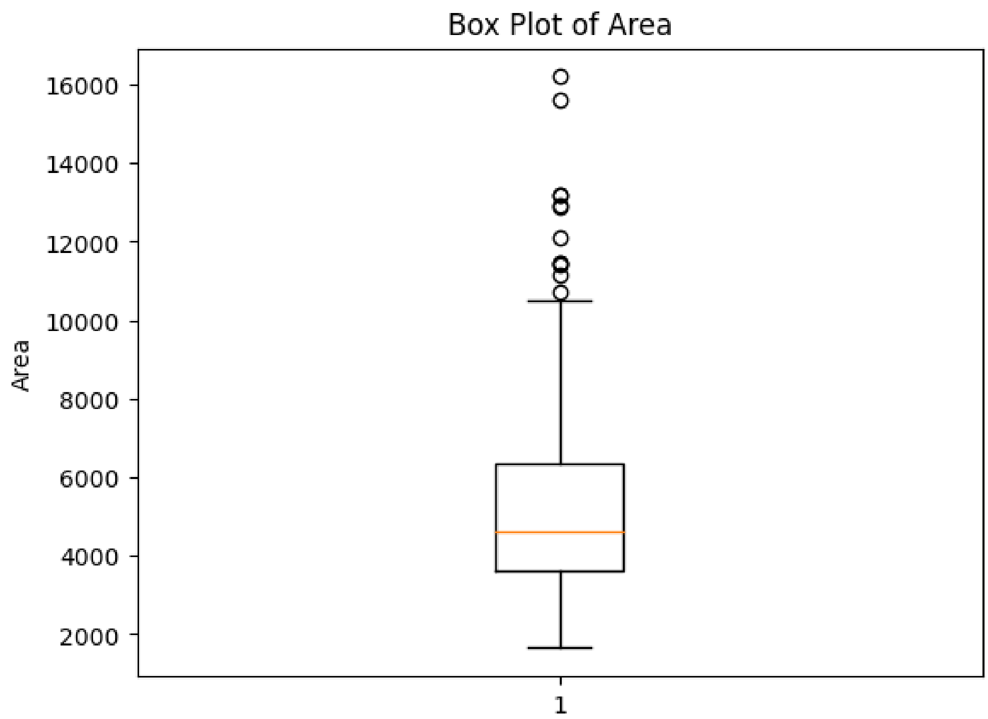
```
[17]: area=data['area']
      area
```

```
[17]: 0      7420
      1      8960
      2      9960
      3      7500
      4      7420
             ...
      540    3000
      541    2400
      542    3620
      543    2910
      544    3850
      Name: area, Length: 545, dtype: int64
```

```
[19]: plt.hist(area, bins=10)
      plt.xlabel('Area')
      plt.ylabel('Frequency')
      plt.title('Histogram of Area')
      plt.show()
```

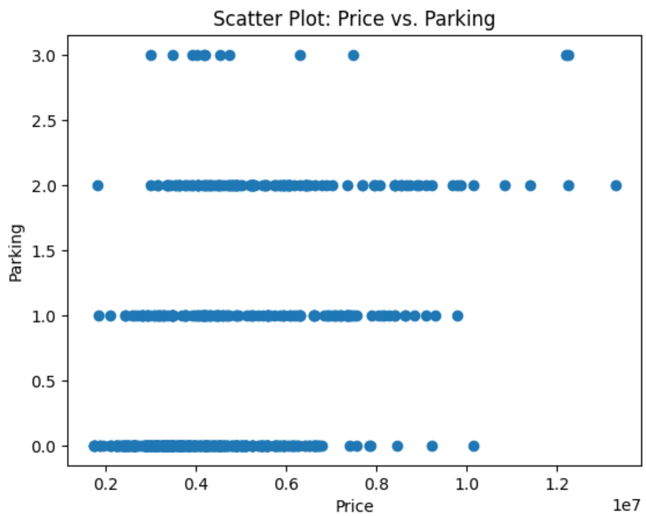


[20]:
```
# Box plot
plt.boxplot(area)
plt.ylabel('Area')
plt.title('Box Plot of Area')
plt.show()
```

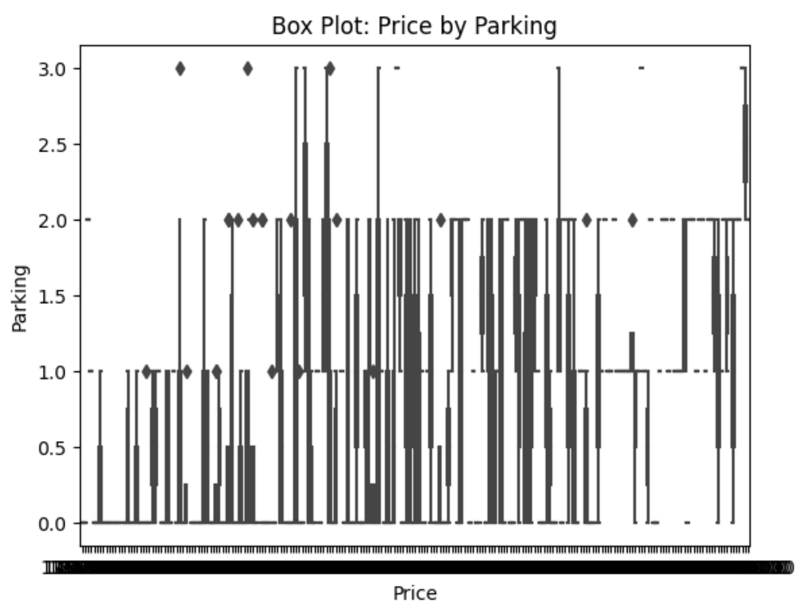


Bi-Variate Analysis

[32]:
```
import seaborn as sns

# Bivariate analysis - price vs.Parking
Price = data['price']
Parking = data['parking']
# Scatter plot
plt.scatter(Price, Parking)
plt.xlabel('Price')
plt.ylabel('Parking')
plt.title('Scatter Plot: Price vs. Parking')
plt.show()
```

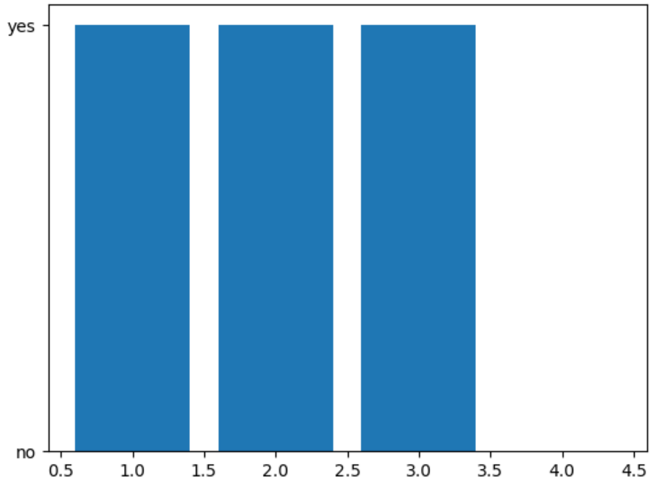


[36]:
```python
# Box plot
sns.boxplot(x=Price,y=Parking)
plt.xlabel('Price')
plt.ylabel('Parking')
plt.title('Box Plot: Price by Parking')
plt.show()
```



```
[45]: # Bar plot
bathrooms = data['bathrooms']
guestroom= data['guestroom']
plt.bar(Price,Parking)
```

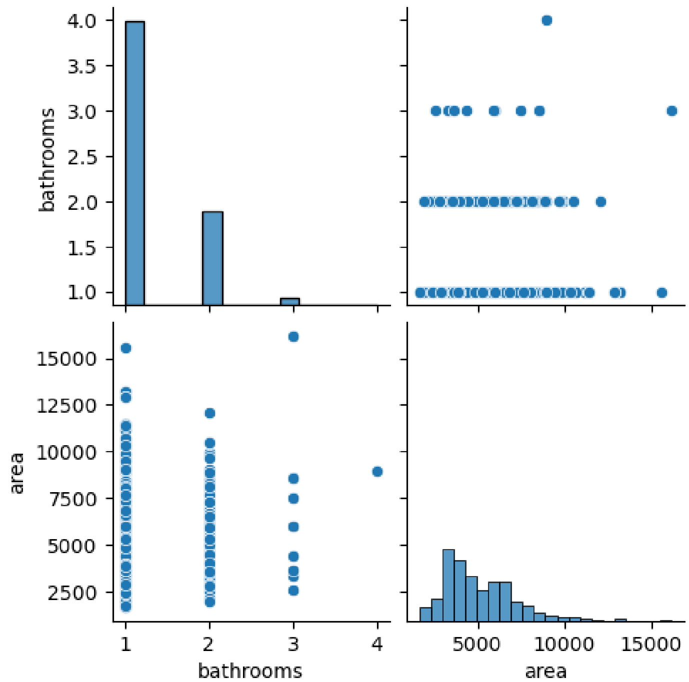
```
[45]: <BarContainer object of 545 artists>
```



Multi-Variate Analysis

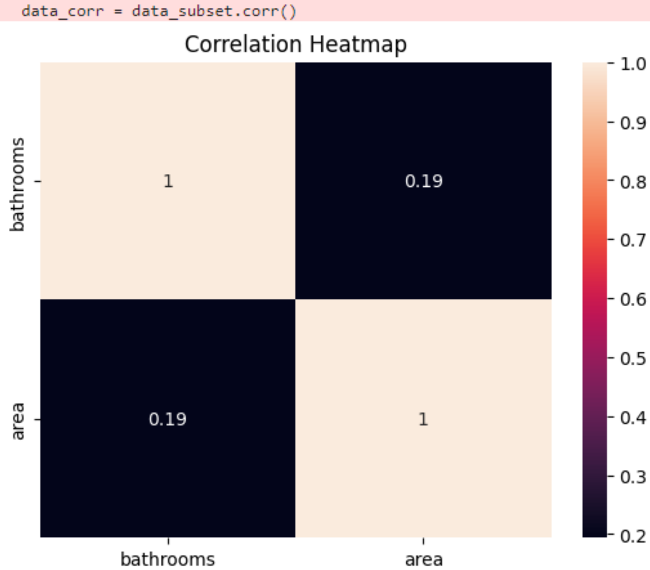
```
[48]: import seaborn as sns

# Multivariate analysis - bathrooms, area, and guestroom
bathrooms = data['bathrooms']
area=data['area']
guestroom= data['guestroom']
```

```
[49]: # Pair plot
data_subset = data[['bathrooms', 'area', 'guestroom']]
sns.pairplot(data_subset)
plt.show()
```




```
[50]: # Heatmap
data_corr = data_subset.corr()
sns.heatmap(data_corr, annot=True)
plt.title('Correlation Heatmap')
plt.show()
```

```
<ipython-input-50-c6042bfba707>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a futur
e version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  data_corr = data_subset.corr()
```



4. Perform descriptive statistics on the dataset.

[51]:
```
# Perform descriptive statistics
statistics = data.describe()
# Print the descriptive statistics
print(statistics)
```

```
              price          area    bedrooms   bathrooms     stories  \
count  5.450000e+02    545.000000  545.000000  545.000000  545.000000
mean   4.766729e+06   5150.541284    2.965138    1.286239    1.805505
std    1.870440e+06   2170.141023    0.738064    0.502470    0.867492
min    1.750000e+06   1650.000000    1.000000    1.000000    1.000000
25%    3.430000e+06   3600.000000    2.000000    1.000000    1.000000
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

          parking
count  545.000000
mean     0.693578
std      0.861586
min      0.000000
25%      0.000000
50%      0.000000
75%      1.000000
max      3.000000
```

5. Check for Missing values and deal with them.

[53]:
```
# Check for missing values
missing_values = data.isnull().sum()
missing_values
```

[53]:
```
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

[54]:
```
# Define a function to detect and replace outliers
def replace_outliers(column):
    q1 = column.quantile(0.25)
    q3 = column.quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    column = column.mask((column < lower_bound) | (column > upper_bound), column.median())
    return column
# Apply the replace_outliers function to each numeric column in the dataset
numeric_columns = data.select_dtypes(include='number').columns
data[numeric_columns] = data[numeric_columns].apply(replace_outliers)
# Print the dataset with replaced outliers
print(data)
```

```
     price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0    4340000  7420         4          2        3      yes        no       no
1    4340000  8960         4          1        2      yes        no       no
2    4340000  9960         3          2        2      yes        no      yes
3    4340000  7500         4          2        2      yes        no      yes
4    4340000  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

    hotwaterheating airconditioning  parking furnishingstatus
0                no             yes        2        furnished
1                no             yes        0        furnished
2                no              no        2   semi-furnished
3                no             yes        0        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

[55]:

```
# Identify categorical columns
categorical_columns = data.select_dtypes(include='object').columns
# Perform one-hot encoding
data_encoded = pd.get_dummies(data, columns=categorical_columns)
# Print the encoded dataset
data_encoded
```

[55]:

	price	area	bedrooms	bathrooms	stories	parking	mainroad_no	mainroad_yes	guestroom_no	guestroom_yes	basement_no	ba
0	4340000	7420	4	2	3	2	0	1	1	0	1	
1	4340000	8960	4	1	2	0	0	1	1	0	1	
2	4340000	9960	3	2	2	2	0	1	1	0	0	
3	4340000	7500	4	2	2	0	0	1	1	0	0	
4	4340000	7420	4	1	2	2	0	1	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	
540	1820000	3000	2	1	1	2	0	1	1	0	0	
541	1767150	2400	3	1	1	0	1	0	1	0	1	
542	1750000	3620	2	1	1	0	0	1	1	0	1	
543	1750000	2910	3	1	1	0	1	0	1	0	1	
544	1750000	3850	3	1	2	0	0	1	1	0	1	

545 rows × 19 columns

ent_no	basement_yes	hotwaterheating_no	hotwaterheating_yes	airconditioning_no	airconditioning_yes	furnishingstatus_furnished	furnishi
1	0	1	0	0	1	1	
1	0	1	0	0	1	1	
0	1	1	0	1	0	0	
0	1	1	0	0	1	1	
0	1	1	0	0	1	1	
...	...	...	...	...	...	...	
0	1	1	0	1	0	0	
1	0	1	0	1	0	0	
1	0	1	0	1	0	0	
1	0	1	0	1	0	1	
1	0	1	0	1	0	0	

8. Split the data into dependent and independent variables

[58]:
```
x=data.iloc[:,0:1]
x.head()
```

[58]:

	price
0	4340000
1	4340000
2	4340000
3	4340000
4	4340000

[59]:
```
y=data.iloc[:,1:]
y.head()
```

[59]:

	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	1	2	yes	no	no	no	yes	0	furnished
2	9960	3	2	2	yes	no	yes	no	no	2	semi-furnished
3	7500	4	2	2	yes	no	yes	no	yes	0	furnished
4	7420	4	1	2	yes	yes	yes	no	yes	2	furnished

[ ]:

9. Scale the independent variables

[14]:
```
from sklearn.preprocessing import StandardScaler
name=x.columns
scale=StandardScaler()
x=scale.fit_transform(x)
x=pd.DataFrame(x,columns=name)
x
```

[14]:

	price
0	4.566365
1	4.004484
2	4.004484
3	3.985755
4	3.554979
...	...
540	-1.576868
541	-1.605149
542	-1.614327
543	-1.614327
544	-1.614327

545 rows × 1 columns

10. Split the data into training and testing

[15]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=20,random_state=0)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(525, 1)
(20, 1)
(525, 18)
(20, 18)
```

11. Build the Model

```
[20]: #Build the model
      from sklearn.linear_model import LinearRegression
      lr=LinearRegression()
      lr
```

```
[20]: ▾ LinearRegression
      LinearRegression()
```

12. Train the Model

```
[22]: #train the model
      z=lr.fit(x_train,y_train)
      z
```

```
[22]: ▾ LinearRegression
      LinearRegression()
```

13. Test the model

```
[24]: #Test the Model
      pred=lr.predict(x_test)
      pred
```

```
[24]: array([[ 5.02584274e+03,  2.93515509e+00,  1.26455387e+00,
               1.76832790e+00,  6.62738137e-01,  1.49454397e-01,
               8.50545603e-01,  8.29372357e-01,  1.70627643e-01,
               6.59024284e-01,  3.40975716e-01,  9.56107526e-01,
               4.38924743e-02,  7.03620158e-01,  2.96379842e-01,
               2.41440638e-01,  4.15191919e-01,  3.43367443e-01],
             [ 5.93265669e+03,  3.15539406e+00,  1.47111374e+00,
               2.05568097e+00,  9.27174013e-01,  6.94675389e-02,
               9.30532461e-01,  7.50016061e-01,  2.49983939e-01,
               5.86003303e-01,  4.13996697e-01,  9.42105020e-01,
               5.78949805e-02,  5.36714864e-01,  4.63285136e-01,
               3.18208241e-01,  4.44826966e-01,  2.36964793e-01],
             [ 4.67625325e+03,  2.85024988e+00,  1.18492215e+00,
               1.65754926e+00,  5.60794401e-01,  1.80290452e-01,
               8.19709548e-01,  8.59965322e-01,  1.40034678e-01,
               6.87174896e-01,  3.12825104e-01,  9.61505688e-01,
               3.84943118e-02,  7.67964489e-01,  2.32035511e-01,
               2.11845651e-01,  4.03767192e-01,  3.84387156e-01],
             [ 6.44538794e+03,  3.27992170e+00,  1.58790693e+00,
               2.21815630e+00,  1.07669149e+00,  2.42413248e-02,
               9.75758675e-01,  7.05146380e-01,  2.94853620e-01,
               5.44715739e-01,  4.55284261e-01,  9.34187715e-01,
               6.58122854e-02,  4.42343179e-01,  5.57656821e-01,
               3.61614221e-01,  4.61583231e-01,  1.76802547e-01],
             [ 4.03004237e+03,  2.69330389e+00,  1.03772411e+00,
               1.45277663e+00,  3.72352947e-01,  2.37290433e-01,
               7.62709567e-01,  9.16515953e-01,  8.34840468e-02,
               7.39210876e-01,  2.60789124e-01,  9.71484110e-01,
               2.85158904e-02,  8.86904009e-01,  1.13095991e-01,
               1.57139767e-01,  3.82648759e-01,  4.60211474e-01],
             [ 6.00045586e+03,  3.17186053e+00,  1.48655747e+00,
```

#Actual values

[26]:
```
#actual values
y_test
```

[26]:

	area	bedrooms	bathrooms	stories	parking	mainroad_no	mainroad_yes	guestroom_no	guestroom_yes	basement_no	baser
239	4000	3	1	2	1	0	1	1	0	1	
113	9620	3	1	1	2	0	1	1	0	0	
325	3460	4	1	2	0	0	1	1	0	1	
66	13200	2	1	1	1	0	1	1	0	0	
479	3660	4	1	2	0	1	0	1	0	1	
103	6350	3	2	3	0	0	1	0	1	1	
386	3850	3	1	1	2	0	1	1	0	1	
480	3480	3	1	2	1	1	0	1	0	1	
400	3512	2	1	1	1	0	1	1	0	1	
37	9000	4	2	4	2	0	1	1	0	1	
71	6000	4	2	4	0	0	1	1	0	1	
329	3960	3	1	2	0	0	1	1	0	1	
450	3450	3	1	2	0	0	1	1	0	0	
432	6060	3	1	1	0	0	1	0	1	0	

[26]:

ating_no	hotwaterheating_yes	airconditioning_no	airconditioning_yes	furnishingstatus_furnished	furnishingstatus_semi-furnished	furnishingstat
1	0	1	0	1	0	
1	0	1	0	1	0	
1	0	0	1	0	1	
0	1	1	0	1	0	
1	0	1	0	0	0	
1	0	0	1	1	0	
1	0	1	0	0	0	
1	0	1	0	0	1	
1	0	1	0	0	0	
1	0	0	1	1	0	
1	0	0	1	0	0	
1	0	1	0	1	0	
1	0	1	0	0	1	
1	0	1	0	1	0	
1	0	1	0	0	1	
1	0	1	0	0	0	

14. Measure the performance using Metrics.

[47]:
```
from sklearn.metrics import r2_score
r2_score(pred,y_test)*100
```

[47]: -5490.664133484892

[ ]: