

### Assignment 2 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.
3. Perform Below Visualizations. • Univariate Analysis • Bi-Variate Analysis • Multi-Variate Analysis
4. Perform descriptive statistics on the dataset.
5. Check for Missing values and deal with them.
6. Find the outliers and replace them outliers
7. Check for Categorical columns and perform encoding.
8. Split the data into dependent and independent variables.
9. Scale the independent variables
10. Split the data into training and testing
11. Build the Model
12. Train the Model
13. Test the Model
14. Measure the performance using Metrics.

```
In [ ]: from google.colab import files  
uploaded = files.upload()  
  
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.  
Saving House Price India.csv to House Price India.csv  
  
In [ ]: import pandas as pd  
import numpy as np  
  
In [ ]: # AUTHOR RESHMA FROM PRINCE DR K VASUDEVAN COLLEGE OF ENGINEERING AND TECHNOLOGY  
  
In [ ]: df = pd.read_csv('House Price India.csv')  
  
In [ ]: import pandas as pd  
import numpy as np  
  
# Load data into a pandas dataframe  
df = pd.read_csv('House Price India.csv')  
  
# Calculate measures of central tendency  
mean = df['number of bedrooms'].mean()  
median = df['number of bedrooms'].median()  
mode = df['number of bedrooms'].mode()  
  
# Calculate measures of dispersion  
range = df['number of bedrooms'].max() - df['number of bedrooms'].min()  
std_dev = df['number of bedrooms'].std()  
variance = df['number of bedrooms'].var()  
  
# Examine the distribution of the data  
histogram = df['number of bedrooms'].hist()  
boxplot = df.boxplot(column=['number of bedrooms'])  
  
# Identify outliers  
outliers = df[np.abs(df['number of bedrooms'] - df['number of bedrooms'].mean()) > 3 * df['number of bedrooms'].std()]  
  
# Print the results  
print("Mean: ", mean)  
print("Median: ", median)  
print("Mode: ", mode)  
print("Range: ", range)
```

```
print("Standard Deviation: ", std_dev)
print("Variance: ", variance)
print("Outliers: ", outliers)
```

Mean: 3.379343365253078  
 Median: 3.0  
 Mode: 0 3  
 Name: number of bedrooms, dtype: int64  
 Range: 32  
 Standard Deviation: 0.9387188525270168  
 Variance: 0.881193084089639  
 Outliers: id Date number of bedrooms number of bathrooms \

76	6762810164	42494	7	8.00
243	6762810052	42496	7	4.50
268	6762816384	42496	9	4.50
275	6762817937	42496	7	5.75
624	6762817573	42502	7	4.00
785	6762819926	42504	7	3.50
1512	6762810234	42517	8	3.50
1519	6762811513	42517	7	4.00
1553	6762817186	42517	7	4.50
1706	6762812569	42519	7	4.50
2814	6762812756	42537	7	4.25
3109	6762810241	42540	7	3.50
3114	6762810926	42540	7	5.50
3322	6762824851	42543	7	3.00
3532	6762815473	42545	33	1.75
3600	6762827935	42545	7	2.50
4207	6762825321	42553	8	2.75
4486	6762816413	42559	7	2.50
4658	6762810410	42561	8	2.75
4680	6762816797	42561	7	2.75
6591	6762810158	42589	7	4.75
6596	6762810849	42589	9	4.50
6730	6762820817	42592	9	7.50
6982	6762811117	42595	10	5.25
6998	6762813966	42595	7	3.75
7003	6762814707	42595	8	2.75
7454	6762818607	42602	11	3.00
8559	6762820832	42621	7	4.00
8650	6762822185	42622	7	3.25
9282	6762816452	42634	7	4.00
9629	6762810083	42638	7	3.00
9810	6762810131	42642	7	4.25
9955	6762813377	42644	7	2.25
10168	6762810199	42649	8	6.00
10177	6762812988	42649	7	6.75
10676	6762813920	42657	7	2.75
10748	6762812073	42658	8	4.00

10916	6762810049	42662	8	4.00
10944	6762818039	42662	7	2.25
11247	6762811840	42666	7	3.75
11441	6762816963	42670	7	1.50
11547	6762815290	42671	10	2.00
11877	6762827515	42677	7	1.00
12273	6762813642	42685	7	2.75
13048	6762816970	42698	8	3.00
13444	6762819515	42707	8	5.00
13825	6762821692	42714	8	3.25
14220	6762812912	42722	8	3.75
14481	6762815079	42732	10	3.00

	living area	lot area	number of floors	waterfront present \
76	13540	307752	3.0	0
243	6210	8856	2.5	0
268	3830	6988	2.5	0
275	3700	7647	2.0	0
624	3440	8100	2.0	0
785	2870	29699	1.0	0
1512	4440	6480	2.0	0
1519	3150	34830	1.0	0
1553	4140	9066	1.0	0
1706	4290	37607	1.5	0
2814	3670	4000	2.0	0
3109	4640	15235	2.0	0
3114	6630	13782	2.0	0
3322	2800	9569	1.0	0
3532	1620	6000	1.0	0
3600	1940	5458	2.0	0
4207	2790	6695	1.0	0
4486	2580	5750	1.0	0
4658	4040	20666	1.0	0
4680	2310	2400	1.5	0
6591	5310	8816	2.0	0
6596	3650	5000	2.0	0
6730	4050	6504	2.0	0
6982	4590	10920	1.0	0
6998	2310	5000	2.0	0
7003	2530	4800	2.0	0
7454	3000	4960	2.0	0
8559	3150	7800	2.0	0
8650	4340	8521	2.0	0
9282	2690	10880	1.0	0
9629	5350	14400	2.5	0

9810	4670	23115	2.0	0
9955	3260	8145	2.0	0
10168	4340	9415	2.0	0
10177	7480	41664	2.0	0
10676	3110	4400	1.5	0
10748	4020	7500	1.0	0
10916	7710	11750	3.5	0
10944	2620	6890	2.0	0
11247	5100	21802	2.0	0
11441	2670	11250	1.5	0
11547	3610	11914	2.0	0
11877	2350	8636	1.0	0
12273	3410	4056	1.5	0
13048	3840	15990	1.0	0
13444	2800	2580	2.0	0
13825	4300	10441	2.0	0
14220	3460	4600	2.0	0
14481	2920	3745	2.0	0

	number of views	condition of the house	...	Built	Year	\
76	4		3	...	1999	
243	2		5	...	1910	
268	0		3	...	1938	
275	1		3	...	1948	
624	0		3	...	1970	
785	0		3	...	1961	
1512	3		5	...	1959	
1519	0		3	...	1957	
1553	0		3	...	1978	
1706	0		5	...	1982	
2814	1		3	...	1964	
3109	1		3	...	1965	
3114	0		3	...	2004	
3322	2		3	...	1963	
3532	0		5	...	1947	
3600	0		3	...	1994	
4207	0		3	...	1977	
4486	0		4	...	1901	
4658	0		4	...	1962	
4680	0		3	...	1915	
6591	0		3	...	2013	
6596	0		3	...	1915	
6730	0		3	...	1996	
6982	2		3	...	2008	
6998	0		3	...	1984	

7003	0		4	...	1901
7454	0		3	...	1918
8559	0		3	...	2013
8650	0		3	...	1986
9282	0		4	...	1960
9629	0		4	...	1910
9810	2		3	...	1992
9955	0		5	...	1967
10168	0		3	...	1967
10177	2		3	...	1953
10676	0		5	...	1914
10748	0		3	...	1968
10916	0		5	...	1904
10944	0		4	...	1961
11247	0		3	...	2001
11441	0		4	...	1948
11547	0		4	...	1958
11877	0		3	...	1962
12273	0		4	...	1906
13048	0		3	...	1961
13444	0		3	...	1997
13825	0		4	...	1979
14220	0		3	...	1987
14481	0		4	...	1913

Renovation	Year	Postal	Code	Lattitude	Longitude	living_area_renov	\
76	0	122045	52.8975	-114.176		4850	
243	0	122061	52.8607	-114.544		2940	
268	0	122028	52.9227	-114.528		1460	
275	1984	122014	52.9693	-114.479		2510	
624	0	122028	52.9281	-114.539		1420	
785	0	122022	52.9453	-114.517		1380	
1512	0	122047	52.8610	-114.493		4440	
1519	2005	122030	52.8329	-114.337		2390	
1553	0	122022	52.9602	-114.481		1440	
1706	0	122012	52.7112	-114.223		2810	
2814	0	122032	52.8675	-114.578		2010	
3109	2003	122057	52.7966	-114.421		3230	
3114	0	122027	52.7699	-114.308		4470	
3322	0	122053	52.7402	-114.373		2150	
3532	0	122028	52.9178	-114.521		1330	
3600	0	122023	52.5491	-114.367		1710	
4207	0	122038	52.9865	-114.521		1760	
4486	0	122044	52.8325	-114.484		2280	
4658	0	122048	52.8640	-114.411		3670	

4680	0	122007	52.9075	-114.580	1340
6591	0	122048	52.8521	-114.398	2920
6596	2010	122004	52.8904	-114.479	2510
6730	0	122054	52.8223	-114.491	1448
6982	0	122048	52.8161	-114.303	2730
6998	0	122007	52.9081	-114.566	1360
7003	0	122047	52.8541	-114.495	1540
7454	1999	122034	52.7860	-114.553	1420
8559	0	122051	52.7559	-114.469	1880
8650	0	122034	52.7500	-114.528	1890
9282	0	122010	52.9087	-114.358	1840
9629	0	122047	52.8595	-114.475	3050
9810	0	122071	52.8483	-114.417	3240
9955	0	122029	52.8636	-114.305	2340
10168	0	122048	52.8616	-114.392	2050
10177	0	122031	52.6943	-114.558	2810
10676	0	122004	52.8984	-114.509	1240
10748	0	122026	52.9032	-114.553	1560
10916	0	122047	52.8563	-114.504	4210
10944	0	122030	52.8423	-114.324	2070
11247	0	122020	52.8250	-114.230	3350
11441	0	122038	52.9421	-114.522	2030
11547	0	122027	52.8005	-114.365	2040
11877	0	122051	52.7732	-114.467	1500
12273	0	122013	52.8754	-114.506	2510
13048	0	122033	52.9411	-114.401	1380
13444	0	122044	52.8386	-114.493	1800
13825	0	122015	52.7086	-114.321	1780
14220	0	122004	52.8917	-114.479	2170
14481	0	122004	52.8935	-114.510	1810

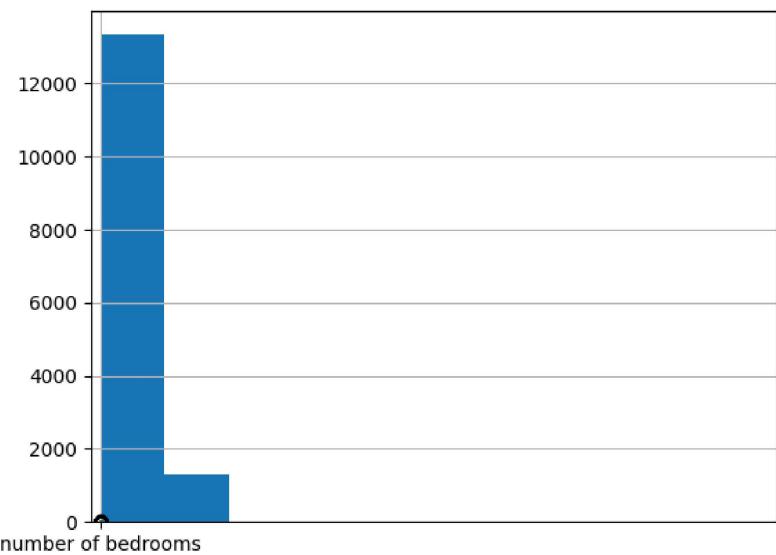
lot_area_renov	Number of schools nearby	Distance from the airport \
76	217800	55
243	5400	64
268	6291	62
275	7479	65
624	1560	62
785	7555	57
1512	8640	55
1519	12054	70
1553	1865	78
1706	40510	71
2814	4000	72
3109	20697	66
3114	8639	64

3322	7333	1	62
3532	4700	2	50
3600	5688	1	80
4207	7624	3	74
4486	5750	1	74
4658	20500	3	55
4680	3825	3	67
6591	10610	2	73
6596	5000	2	63
6730	3866	1	55
6982	10400	3	73
6998	1552	1	59
7003	4800	2	80
7454	4960	1	52
8559	6000	3	58
8650	8951	2	79
9282	10836	3	58
9629	7469	1	75
9810	13912	3	70
9955	8145	3	72
10168	9100	3	69
10177	33190	2	66
10676	4280	3	75
10748	3737	2	53
10916	8325	2	66
10944	7910	3	52
11247	10005	2	58
11441	9000	1	66
11547	11914	1	75
11877	7366	1	74
12273	4056	1	65
13048	8172	3	60
13444	2580	1	72
13825	10457	2	77
14220	3750	3	71
14481	3745	1	58

Price	
76	2280000
243	3200000
268	599999
275	540000
624	550000
785	475000
1512	1970000

1519	999000
1553	565000
1706	840000
2814	824000
3109	1950000
3114	1240000
3322	350000
3532	640000
3600	280000
4207	340000
4486	599000
4658	1650000
4680	580000
6591	2300000
6596	1280000
6730	450000
6982	1150000
6998	727160
7003	680000
7454	520000
8559	450000
8650	419000
9282	597157
9629	2890000
9810	2450000
9955	770000
10168	2150000
10177	800000
10676	730000
10748	900000
10916	3300000
10944	539000
11247	936000
11441	575000
11547	650000
11877	291000
12273	750000
13048	575000
13444	490000
13825	430000
14220	808000
14481	660000

[49 rows x 23 columns]



```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14620 entries, 0 to 14619
Data columns (total 23 columns):
 #   Column           Non-Null Count Dtype  
 ---  -- 
 0   id               14620 non-null  int64   
 1   Date              14620 non-null  int64   
 2   number of bedrooms 14620 non-null  int64   
 3   number of bathrooms 14620 non-null  float64 
 4   living area        14620 non-null  int64   
 5   lot area           14620 non-null  int64   
 6   number of floors    14620 non-null  float64 
 7   waterfront present 14620 non-null  int64   
 8   number of views     14620 non-null  int64   
 9   condition of the house 14620 non-null  int64   
 10  grade of the house   14620 non-null  int64   
 11  Area of the house(excluding basement) 14620 non-null  int64   
 12  Area of the basement 14620 non-null  int64   
 13  Built Year          14620 non-null  int64   
 14  Renovation Year      14620 non-null  int64   
 15  Postal Code          14620 non-null  int64   
 16  Latitude             14620 non-null  float64 
 17  Longitude            14620 non-null  float64 
 18  living_area_renov    14620 non-null  int64   
 19  lot_area_renov       14620 non-null  int64   
 20  Number of schools nearby 14620 non-null  int64   
 21  Distance from the airport 14620 non-null  int64   
 22  Price                14620 non-null  int64   
dtypes: float64(4), int64(19)
memory usage: 2.6 MB

```

```
In [ ]: # replace outliers
```

```

In [ ]: import pandas as pd
import numpy as np

# Load data into a pandas dataframe
df = pd.read_csv('House Price India.csv')

# Identify outliers using the Z-score method
outliers = df[np.abs(df['number of bedrooms']) - df['number of bedrooms'].mean() > 3 * df['number of bedrooms'].std()]

# Replace outliers with the median of the column
median = df['number of bedrooms'].median()

```

```
df['number of bedrooms'] = np.where(np.abs(df['number of bedrooms'] - df['number of bedrooms'].mean()) > 3 * df['number of bedrooms'].std(), None, df['number of bedrooms'])

# Print the updated dataframe
print(df)
```

	<code>id</code>	<code>Date</code>	<code>number of bedrooms</code>	<code>number of bathrooms</code>	\	
0	6762810145	42491	5.0	2.50		
1	6762810635	42491	4.0	2.50		
2	6762810998	42491	5.0	2.75		
3	6762812605	42491	4.0	2.50		
4	6762812919	42491	3.0	2.00		
...	...	...	...	...		
14615	6762830250	42734	2.0	1.50		
14616	6762830339	42734	3.0	2.00		
14617	6762830618	42734	2.0	1.00		
14618	6762830709	42734	4.0	1.00		
14619	6762831463	42734	3.0	1.00		
	<code>living area</code>	<code>lot area</code>	<code>number of floors</code>	<code>waterfront present</code>	\	
0	3650	9050	2.0	0		
1	2920	4000	1.5	0		
2	2910	9480	1.5	0		
3	3310	42998	2.0	0		
4	2710	4500	1.5	0		
...	...	...	...	...		
14615	1556	20000	1.0	0		
14616	1680	7000	1.5	0		
14617	1070	6120	1.0	0		
14618	1030	6621	1.0	0		
14619	900	4770	1.0	0		
	<code>number of views</code>	<code>condition of the house</code>	...	<code>Built Year</code>	\	
0	4	5	...	1921		
1	0	5	...	1909		
2	0	3	...	1939		
3	0	3	...	2001		
4	0	4	...	1929		
...	...	...	...	...		
14615	0	4	...	1957		
14616	0	4	...	1968		
14617	0	3	...	1962		
14618	0	4	...	1955		
14619	0	3	...	1969		
	<code>Renovation Year</code>	<code>Postal Code</code>	<code>Lattitude</code>	<code>Longitude</code>	<code>living_area_renov</code>	\
0	0	122003	52.8645	-114.557	2880	
1	0	122004	52.8878	-114.470	2470	
2	0	122004	52.8852	-114.468	2940	
3	0	122005	52.9532	-114.321	3350	
4	0	122006	52.9047	-114.485	2060	

```

...
14615      ...    0    122066   52.6191  -114.472    2250
14616      ...    0    122072   52.5075  -114.393    1540
14617      ...    0    122056   52.7289  -114.507    1130
14618      ...    0    122042   52.7157  -114.411    1420
14619      ...  2009    122018   52.5338  -114.552     900

      lot_area_renov  Number of schools nearby  Distance from the airport \
0            5400                  2                   58
1            4000                  2                   51
2            6600                  1                   53
3            42847                 3                   76
4            4500                  1                   51
...
14615      ...    17286                 3                   76
14616      ...    7480                  3                   59
14617      ...    6120                  2                   64
14618      ...    6631                  3                   54
14619      ...    3480                  2                   55

      Price
0    2380000
1    1400000
2    1200000
3    838000
4    805000
...
14615    221700
14616    219200
14617    209000
14618    205000
14619    146000

```

[14620 rows x 23 columns]

In [ ]: # checking for any other outliers

In [ ]: # Identify outliers  
outliers = df[np.abs(df['number of bedrooms'] - df['number of bedrooms'].mean()) > 3 \* df['number of bedrooms'].std()]

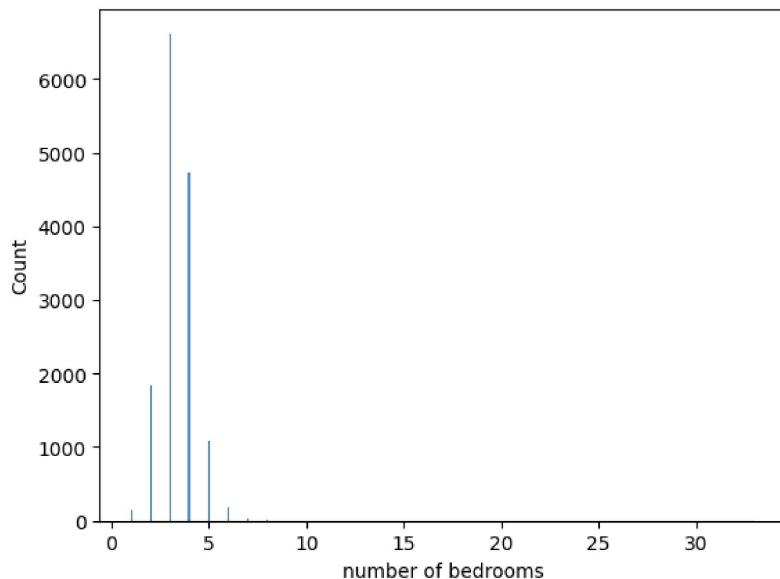
In [ ]: # we get null, hence we sucessfully replplaced the outliers.

In [ ]: import pandas as pd  
import seaborn as sns

```
# Load data into a pandas dataframe
df = pd.read_csv('House Price India.csv')

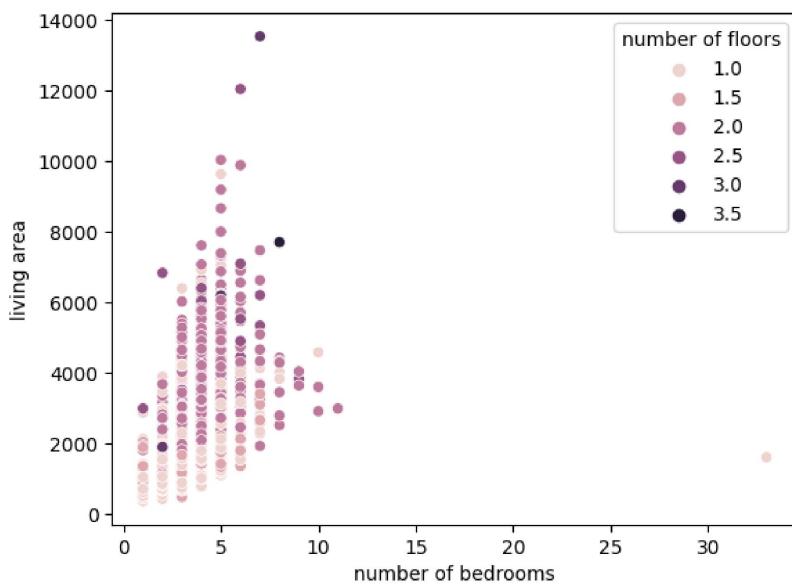
# Univariate analysis - histogram
sns.histplot(data=df, x='number of bedrooms')
```

```
Out[ ]: <Axes: xlabel='number of bedrooms', ylabel='Count'>
```



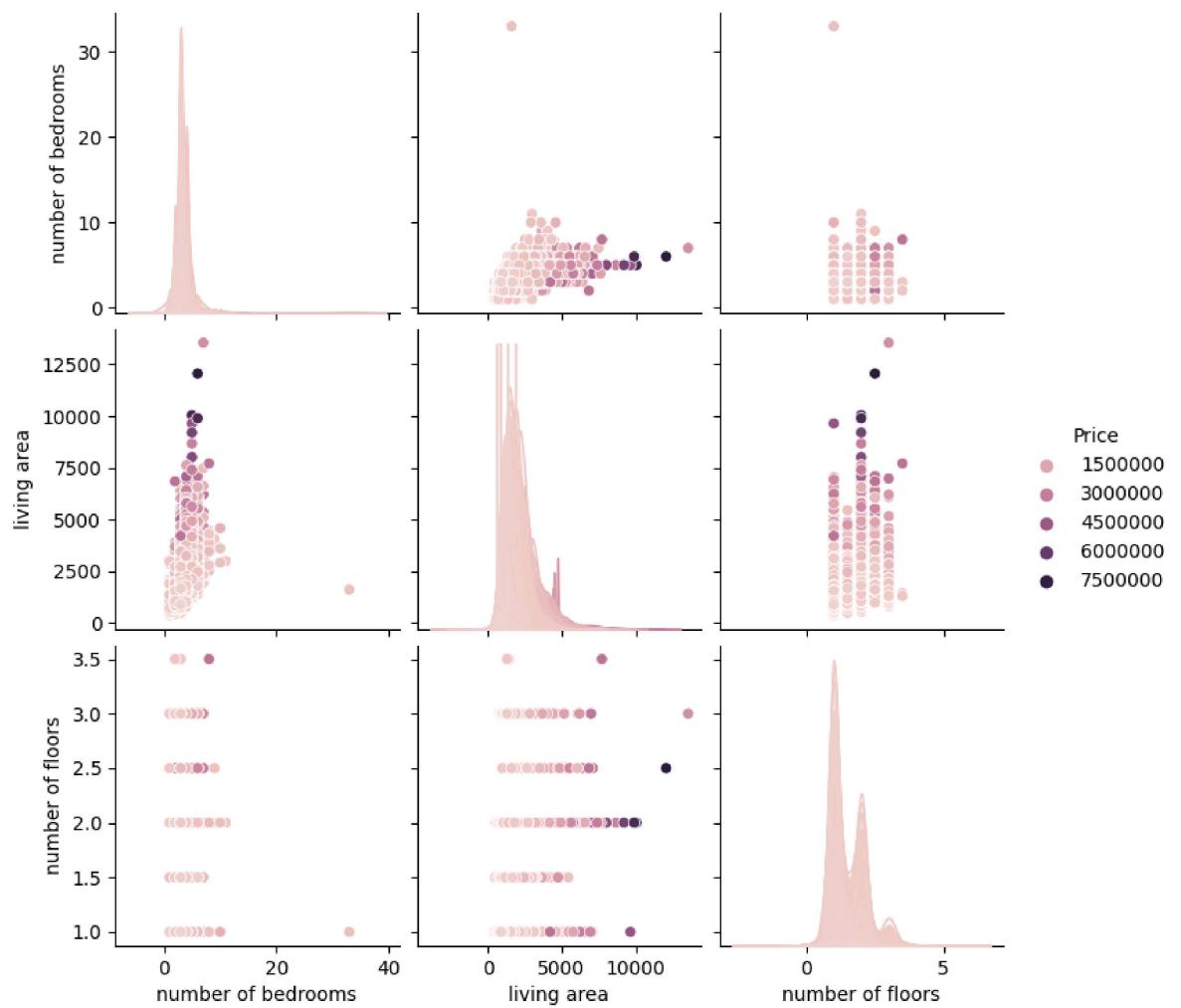
```
In [ ]: # Bi-variate analysis - scatter plot
sns.scatterplot(data=df, x='number of bedrooms', y='living area', hue='number of floors')
```

```
Out[ ]: <Axes: xlabel='number of bedrooms', ylabel='living area'>
```



```
In [ ]: # Multi-variate analysis - pair plot
sns.pairplot(data=df, vars=['number of bedrooms', 'living area', 'number of floors'], hue='Price')
```

```
Out[ ]: <seaborn.axisgrid.PairGrid at 0x7fe092701c60>
```



```
In [ ]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14620 entries, 0 to 14619
Data columns (total 23 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   id               14620 non-null   int64  
 1   Date              14620 non-null   int64  
 2   number of bedrooms 14620 non-null   int64  
 3   number of bathrooms 14620 non-null   float64 
 4   living area       14620 non-null   int64  
 5   lot area          14620 non-null   int64  
 6   number of floors   14620 non-null   float64 
 7   waterfront present 14620 non-null   int64  
 8   number of views    14620 non-null   int64  
 9   condition of the house 14620 non-null   int64  
 10  grade of the house 14620 non-null   int64  
 11  Area of the house(excluding basement) 14620 non-null   int64  
 12  Area of the basement 14620 non-null   int64  
 13  Built Year        14620 non-null   int64  
 14  Renovation Year   14620 non-null   int64  
 15  Postal Code        14620 non-null   int64  
 16  Latitude            14620 non-null   float64 
 17  Longitude           14620 non-null   float64 
 18  living_area_renov  14620 non-null   int64  
 19  lot_area_renov     14620 non-null   int64  
 20  Number of schools nearby 14620 non-null   int64  
 21  Distance from the airport 14620 non-null   int64  
 22  Price              14620 non-null   int64  
dtypes: float64(4), int64(19)
memory usage: 2.6 MB
```

```
In [ ]: # we have no null values
```

```
In [ ]: import pandas as pd

# Load data into a pandas dataframe
df = pd.read_csv('House Price India.csv')

# Identify categorical columns
cat_cols = df.select_dtypes(include=['object']).columns.tolist()

# Perform one-hot encoding for categorical columns
```

```
df = pd.get_dummies(df, columns=cat_cols)

# Print the updated dataframe
print(df)
```

						\
0	6762810145	42491	5	2.50		
1	6762810635	42491	4	2.50		
2	6762810998	42491	5	2.75		
3	6762812605	42491	4	2.50		
4	6762812919	42491	3	2.00		
...	...	...	...	...		
14615	6762830250	42734	2	1.50		
14616	6762830339	42734	3	2.00		
14617	6762830618	42734	2	1.00		
14618	6762830709	42734	4	1.00		
14619	6762831463	42734	3	1.00		
						\
0	3650	9050	2.0	0		
1	2920	4000	1.5	0		
2	2910	9480	1.5	0		
3	3310	42998	2.0	0		
4	2710	4500	1.5	0		
...	...	...	...	...		
14615	1556	20000	1.0	0		
14616	1680	7000	1.5	0		
14617	1070	6120	1.0	0		
14618	1030	6621	1.0	0		
14619	900	4770	1.0	0		
						\
0	number of views	condition of the house	...	Built Year	\	
1	4	5	...	1921		
2	0	5	...	1909		
3	0	3	...	1939		
4	0	3	...	2001		
...	...	4	...	1929		
14615	0	4	...	1957		
14616	0	4	...	1968		
14617	0	3	...	1962		
14618	0	4	...	1955		
14619	0	3	...	1969		
						\
0	Renovation Year	Postal Code	Lattitude	Longitude	living_area_renov	\
1	0	122003	52.8645	-114.557	2880	
2	0	122004	52.8878	-114.470	2470	
3	0	122004	52.8852	-114.468	2940	
4	0	122005	52.9532	-114.321	3350	

```

...
14615      0    122066   52.6191  -114.472      2250
14616      0    122072   52.5075  -114.393      1540
14617      0    122056   52.7289  -114.507      1130
14618      0    122042   52.7157  -114.411      1420
14619    2009    122018   52.5338  -114.552      900

      lot_area_renov  Number of schools nearby  Distance from the airport \
0            5400                  2                   58
1            4000                  2                   51
2            6600                  1                   53
3            42847                 3                   76
4            4500                  1                   51
...
14615    17286                 3                   76
14616    7480                  3                   59
14617    6120                  2                   64
14618    6631                  3                   54
14619    3480                  2                   55

      Price
0    2380000
1    1400000
2    1200000
3    838000
4    805000
...
14615    221700
14616    219200
14617    209000
14618    205000
14619    146000

```

[14620 rows x 23 columns]

```
In [ ]: import pandas as pd

# Load data into a pandas dataframe
df = pd.read_csv('House Price India.csv')

# Split the data into dependent and independent variables
X = df.drop('number of bedrooms', axis=1)
y = df['Price']

# Print the shapes of the X and y variables
```

```
print('Independent_variable:', X.shape)
print('dependent_variable:', y.shape)
```

```
Independent_variable: (14620, 22)
dependent_variable: (14620,)
```

```
In [ ]: import pandas as pd
from sklearn.preprocessing import StandardScaler

# Load data into a pandas dataframe
df = pd.read_csv('House Price India.csv')

# Split the data into dependent and independent variables
X = df.drop('number of bedrooms', axis=1)

# Scale the independent variables using StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Print the scaled data
print(X_scaled)
```

[[ -1.71314837 -1.68590818 0.48111873 ... -0.01498123 -0.77788599  
  5.0094382 ]  
[-1.63458951 -1.68590818 0.48111873 ... -0.01498123 -1.56126035  
  2.34291528]  
[-1.57639183 -1.68590818 0.80583278 ... -1.23858786 -1.33743911  
  1.79872693]  
...
[ 1.56916901 1.92234067 -1.46716559 ... -0.01498123 -0.10642226  
  -0.89772635]  
[ 1.58375852 1.92234067 -1.46716559 ... 1.2086254 -1.22552848  
  -0.90861012]  
[ 1.70464296 1.92234067 -1.46716559 ... -0.01498123 -1.11361786  
  -1.06914568]]

```
In [ ]: import pandas as pd
from sklearn.model_selection import train_test_split
# Load data into a pandas dataframe
df = pd.read_csv('House Price India.csv')

# Split the data into dependent and independent variables
X = df.drop('number of bedrooms', axis=1)
y = df['Price']

# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Print the shapes of the training and testing sets
print('Training set shape:', X_train.shape, y_train.shape)
print('Testing set shape:', X_test.shape, y_test.shape)
```

Training set shape: (11696, 22) (11696,)  
Testing set shape: (2924, 22) (2924,)

```
In [ ]: import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load data into a pandas dataframe
df = pd.read_csv('House Price India.csv')

# Split the data into dependent and independent variables
X = df.drop('number of bedrooms', axis=1)
y = df['Price']

# Scale the independent variables using StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Build a Linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Print the coefficients of the model
print('Coefficients:', model.coef_)

# Predict the target variable for the test set
y_pred = model.predict(X_test)

# Print the mean squared error of the model
from sklearn.metrics import mean_squared_error
print('Mean squared error:', mean_squared_error(y_test, y_pred))
```

```
Coefficients: [ 2.48053844e-10  0.0000000e+00 -2.19755645e-10 -1.69150617e-10  
-6.58161947e-11 -1.49083521e-10  1.02334038e-10 -5.80226402e-11  
2.83806532e-10 -2.86978606e-10 -1.18451701e-10 -1.43294350e-10  
-2.44295998e-10  1.19270580e-10 -3.39268519e-11 -5.63918396e-11  
8.62988441e-11 -7.27595761e-12 -2.03726813e-10  7.90123522e-11  
-2.00088834e-11  3.67519811e+05]  
Mean squared error: 2.143431357174532e-18
```

```
In [ ]: import pandas as pd  
from sklearn.linear_model import LinearRegression  
from sklearn.model_selection import train_test_split  
from sklearn.preprocessing import StandardScaler  
  
# Load data into a pandas dataframe  
df = pd.read_csv('House Price India.csv')  
  
# Split the data into dependent and independent variables  
X = df.drop('number of bedrooms', axis=1)  
y = df['Price']  
  
# Scale the independent variables using StandardScaler  
scaler = StandardScaler()  
X_scaled = scaler.fit_transform(X)  
  
# Split the data into training and testing sets  
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)  
  
# Build a Linear regression model  
model = LinearRegression()  
  
# Train the model using the training data  
model.fit(X_train, y_train)
```

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

```
In [ ]: from sklearn.metrics import mean_squared_error  
  
# Use the trained model to make predictions on the testing data  
y_pred = model.predict(X_test)  
  
# Calculate the mean squared error between the predicted values and the actual values
```

```
mse = mean_squared_error(y_test, y_pred)
print('Mean squared error:', mse)
```

Mean squared error: 2.143431357174532e-18

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

# Use the trained model to make predictions on the testing data
y_pred = model.predict(X_test)

# Calculate the R-squared value
r2 = r2_score(y_test, y_pred)
print('R-squared:', r2)

# Calculate the mean absolute error
mae = mean_absolute_error(y_test, y_pred)
print('Mean absolute error:', mae)
```

R-squared: 1.0

Mean absolute error: 1.1375178385490269e-09

```
In [ ]: # AUTHOR RESHMA FROM PRINCE DR K VASUDEVAN COLLEGE OF ENGINEERING AND TECHNOLOGY
```