NM DATA ANALYTICS ASSIGNMENT 3 - House Price dataset of India

DONE BY Saranya R P

Importing the necessary libraries for EDA and data preprocessing

```
In [2]: import pandas as pd
```
import pandas as pd import numpy as np import matplotlib. pyplot as PIt import seaborn as sns import folium from scipy import stats

Converting csv file into dataframe

```
df=pd.read_csv('C:/Users/Reshma/Downloads/House Price India.csv')
```

In `df=df.drop([` 'Date'] , axis-I)

Out[5] :

id	number of floors	number of	number of views	number of the	condition of the	grade ...	number of bedrooms	bathrooms	living area	lot area	waterfront present	house	Built house Year	Renovatio Yee
o 6762810145	5		2.50	3650	9050	2.0				4	5		10...1921	
1 6762810635	4		2.50	2920	4000	1.5	5	8...1909						
2 6762810998	5		2.75	2910	9480	1.5	3	8...1939						
3 6762812605	4		2.50	3310	42998	2.0	3	9...2001						

	id	number of house	number of house	number of bathrooms	number of the ...	condition of the	grade ...	number of bathrooms	living area	lot area	waterfront present	Built Year	Renovation Year	bedrooms	floors	views
4	6762812919	3		2.00	2710	4500	1.5	4	8...1929							
14615	6762830250	2		1.50	1556	20000	1.0	4	7...1957							
14616	6762830339	3		2.00	1680	7000	1.5	4	7...1968							
14617	6762830618	2		1.00	1070	6120	1.0	3	6...1962							
14618	6762830709	4		1.00	1030	6621	1.0	4	6...1955							
14619	6762831463	3		1.00	900	4770	1.0	3	6...1969 200							

14620 rows x 22 columns

[6]: df.head()

Out[6]:

	id	number of house	number of house	number of the ...	condition of the ...	grade	...bathrooms	living area	lot area	waterfront present	Built Year	Renovation Year	bedrooms	floors	views
o	6762810145		5	2.50	3650	9050	2.0			4	5	10 ... 1921			
1	6762810635	4		2.50	2920	4000	1.5	0	5	8...1909					
2	6762810998	5		2.75	2910	9480	1.5	3		8...1939					
3	6762812605	4		2.50	3310	42998	2.0	0	3	9...2001 o 1:					
4	6762812919	3		2.00	2710	4500	1.5	4		8 ... 1929					

[7]: df.tail()

Out [7] :

	id	number of floors	number of views	number of the ...	condition of the	grade of the	bedrooms	bathrooms	living area	lot area	waterfront present	Built house	Renovatio house	Year	Yee
14615	6762830250	2	1.5	1556	20000	1.0	4	71957						...	
14616	6762830339	3	2.0	1680	7000	1.5	4	7...1968							
14617	6762830618	2	1.0	1070	6120	1.0	3	6...1962							
14618	6762830709	4	1.0	1030	6621	1.0	4	6...1955							
14619	6762831463	3	1.0	900	4770	1.0	3	6...1969 200							

5 rows x 22 columns

Checking for null and duplicated values

In [8] : df.isna() .sum()

Out [8] :

id	0
number of bedrooms	0

number of bathrooms	basement)	0	
living area lot area		0	
number of floors		0	
waterfront present		0	
number of views		0	
condition of the house		0	
grade of the house		0	
Area of the house(excluding		0	
Area of the basement Built		0	
Year		0	
Renovation Year		0	
Postal Code		0	
L attitude		0	
Longitude living		0	
area renov lot area		0	
renov		0	
Number of schools nearby		0	
Distance from the airport		0	
Price dtype: int64		0	
		0	

[9]: df.duplicated() . sum()

Out[9] : 0

In [10] : df.info()

```
<class 'pandas. core. frame. DataFrame' >
RangeIndex: 14620 entries, 0 to 14619
Data columns (total 22 columns) :
```

#	Column	Non-Null Count	Dtype
	------	--------------	-----
e	id	14620 non-null	int64
1	number of bedrooms	14620 non-null	int64
2	number of bathrooms	14620 non-null	float64
3	living area	14620 non-null	int64
4	lot area	14620 non-null	int64
5	number of floors	14620 non-null	float64

6	waterfront present	14620 non-null	int64
7	number of views	14620 non-null	int64
8	condition of the house	14620 non-null	int64
9	grade of the house	14620 non-null	int64
10	Area of the house(excluding basement)	14620 non-null	int64
11	Area of the basement	14620 non-null	int64
12	Built Year	14620 non-null	int64
13	Renovation Year	14620 non-null	int64
14	Postal Code	14620 non-null	int64
15	L attitude	14620 non-null	float64
16	Longitude	14620 non-null	float64
17	living area renov	14620 non-null	int64
18	lot area renov	14620 non-null	int64
19	Number of schools nearby	14620 non-null	int64
20	Distance from the airport	14620 non-null	int64
21	Price	14620 non-null	int64

dtypes: float64(4),   int64(18)
memory usage: 2.5 MB

In[11] : df.describe()

Out[11] :

	id	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condi the
count	1.462000e+04	14620.000000	14620.000000	14620.000000	1.462000e+04	14620.000000	14620.000000	14620.000000	14620.
mean	6.762821e+09	3.379343	2.129583	2098.262996	1.509328e+04	1.502360	0.007661	0.233105	3.
std	6.237575e+03	0.938719	0.769934	928.275721	3.791962e+04	0.540239	0.087193	0.766259	o.
min	6.762810e+09	1.000000	0.500000	370.000000	5.200000e+02	1.000000	0.000000	0.000000	1.
25%	6.762815e+09	3.000000	1.750000	1440.000000	5.010750e+03	1.000000	0.000000	0.000000	3.
50%	6.762821e+09	3.000000	2.250000	1930.000000	7.620000e+03	1.500000	0.000000	0.000000	3.
75%	6.762826e+09	4.000000	2.500000	2570.000000	1.080000e+04	2.000000	0.000000	0.000000	4.

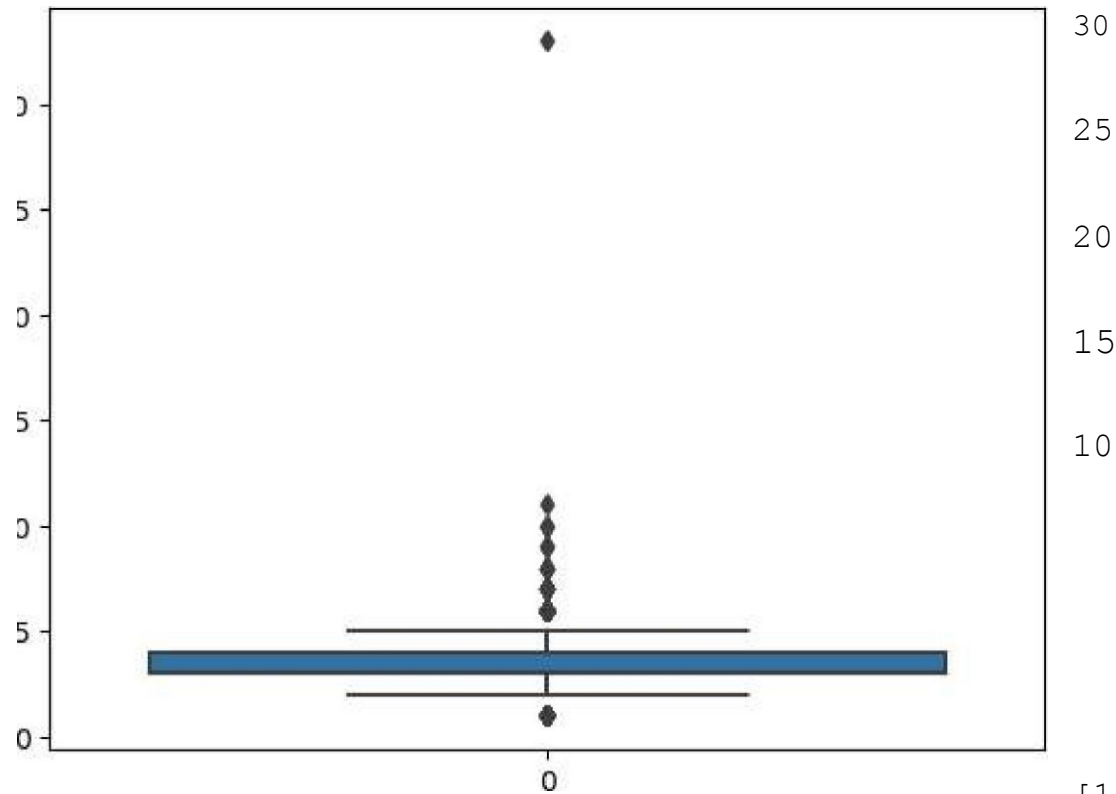| | max | 6.762832e+09 | 33.000000 | 8.000000 | 13540.000000 | 1.074218e+06 | 3.500000 | 1.000000 | 4.000000 | 5. |

8 rows x 22 columns

UNIVARIATE ANALYSIS

Checking for outliers

In [12] : sns.boxplot(df[' number of bedrooms '])

Out [12] : <AxesSubp10t : >

In[14] : threshold-3

```
print(np.where(z>3),len(np.where(z>3)[0]))
```

```
(array([  76,  243,  268,  275,  624,  785,1512,1519,1553,
       1706, 2814,  3109, 3114, 332 3532, 3600, 4207,  4486,
                      2,
       4658,  4680,   6591, 6596, 673 6982, 6998, 7003,  7454,
                      0,
       8559,  8650,  9282, 9629, 981 9955,10168,10177,10676,
                      0,
      10748,10916,10944,11247,   11547 11877,12273,13048,
                      11441,    ,
      13444,13825,14220, 14481]),
                      ) 49
```

```
In [15] : print(np.where(z<-3))
```

(array([], dtype=int64),)

There are 138 outliers in number of bedrooms as proved from the boxplot and the fact that there are observations whose z-score is beyond 3

```
In [16] :df1=df[(z
```

```
In [17] : sns.boxplot(dfl[ 'number of bedrooms ' ] )
```

Out[17] :   <AxesSubp10t : >


2

1

dfl

Out[18] :

id	number of bedrooms	number of bathrooms	number of the area	condition of the area	grade ...	number of floors	living present	lot views	waterfront house	Built house	Renovatio Year	Yee
0 6762810145	5	2.50	3650	9050	2.0				4	5	10...1921	
1 6762810635	4	2.50	2920	4000	1.5	5	8...1909					
2 6762810998	5	2.75	2910	9480	1.5	3	8...1939					
3 6762812605	4	2.50	3310	42998	2.0	3	9...2001					
4 6762812919	3	2.00	2710	4500	1.5	4	8...1929					
14615 6762830250	2	1.50	1556	20000	1.0	4	7...1957					
14616 6762830339	3	2.00	1680	7000	1.5	4	7...1968					
14617 6762830618	2	1.00	1070	6120	1.0	3	6...1962					
14618 6762830709	4	1.00	1030	6621	1.0	4	6...1955					
14619 6762831463	3	1.00	900	4770	1.0	3	6...1969	200				

14571 rows x 22 columns

```
In [ 19]  :  sns.boxplot(dfl[ 'number of bathrooms' ])
```

Out[19] : <AxesSubp10t : >
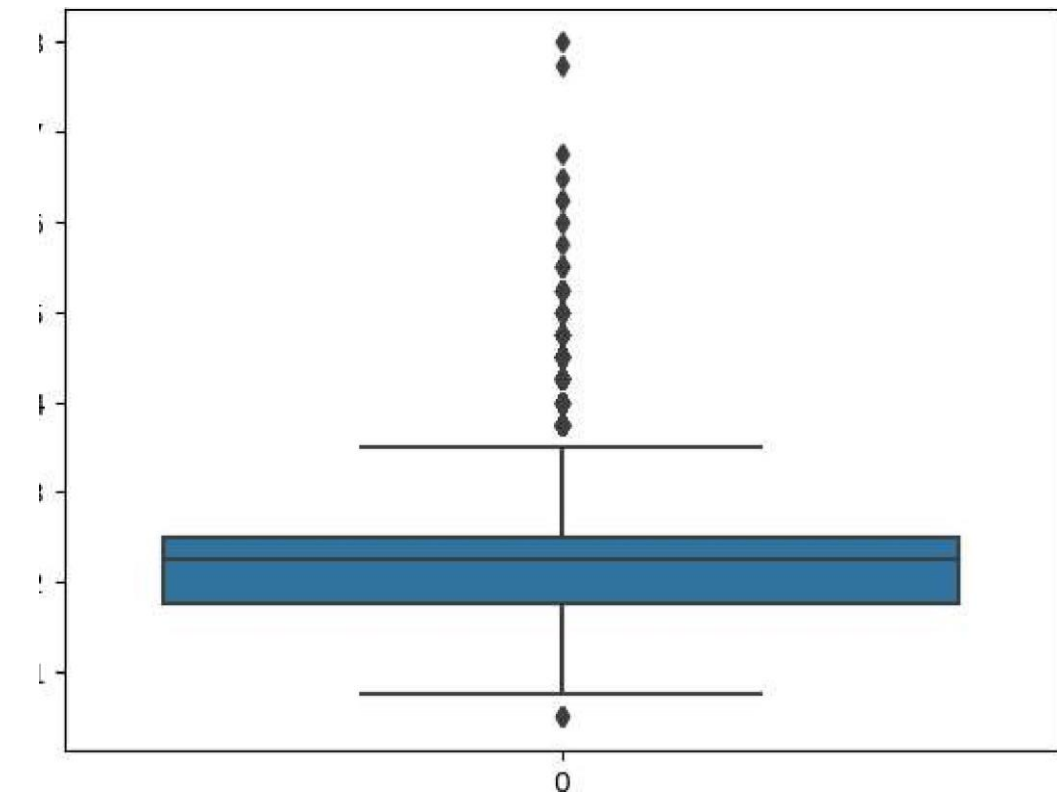
8

7

6

5

4

3

: <AxesSubp10t : >

2

1

In [20] :
```
z=np.abs(stats.zscore(df1['number of bathrooms' ]
))
```

[21]:
```
len(np.where(z>3)
[0] )
```

Out[21] : 124

In [22]
:
```
print(np.where(z<-
3))
```

(array([], dtype=int64),)

```
df1=df1[(z< 3)]
```

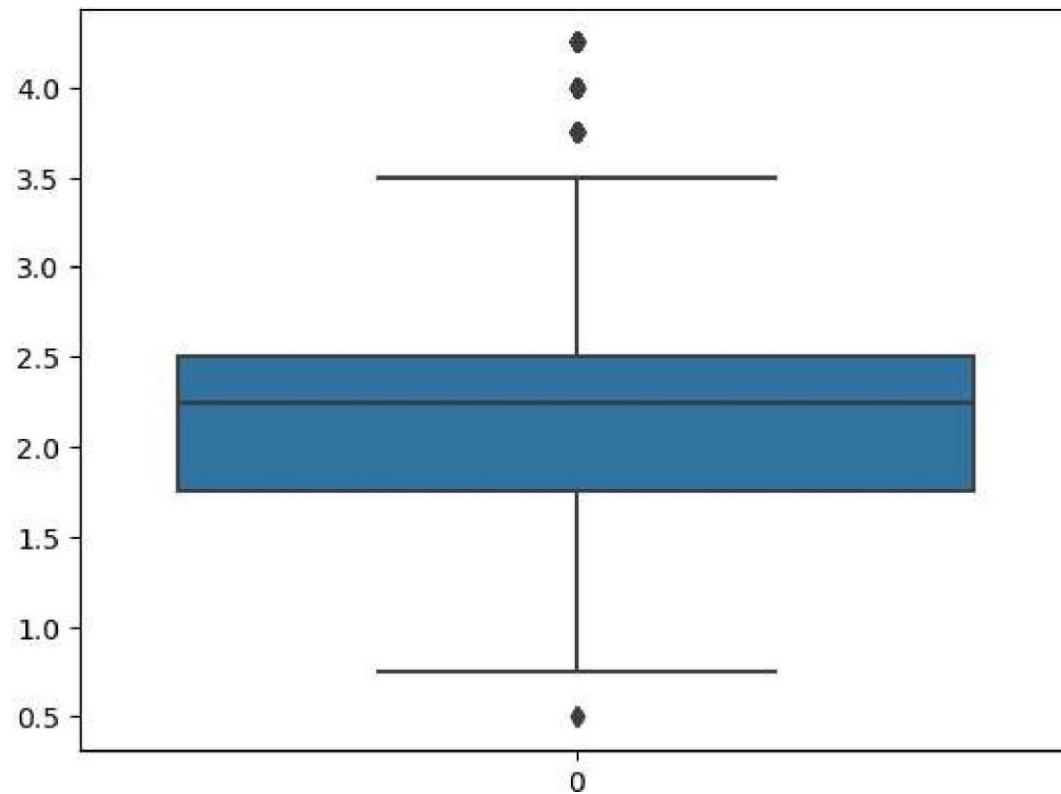I n [24] : sns.boxplot(dfl['number of bathrooms'])
Out[24]

sns.boxplot(dfl[' living area])

: <AxesSubp10t : >

In[25] : dfl

Out[25] :

	id	number of floors	number of views	number of	condition of the of the	grade ...bedrooms	number of living	number bathrooms	living area	lot area	waterfront present	house	Built house	Renovatio Year	Yee
o	6762810145	5		2.50	3650	9050	2.0			4	5	10...1921			
1	6762810635	4		2.50	2920	4000	1.5	5	8...1909						

2	6762810998	5	2.75	2910	9480	1.5	3	8...1939
3	6762812605	4	2.50	3310	42998	2.0	3	9...2001
4	6762812919	3	2.00	2710	4500	1.5	4	8...1929
14615	6762830250	2	1.50	1556	20000	1.0	4	7...1957
14616	6762830339	3	2.00	1680	7000	1.5	4	7...1968
14617	6762830618	2	1.00	1070	6120	1.0	3	6...1962
14618	6762830709	4	1.00	1030	6621	1.0	4	6...1955
14619	6762831463	3	1.00	900	4770	1.0	3	6...1969 200

14447 rows x 22 columns

There are 124 outliers in number of bathrooms as proved from the boxplot and the fact that there are observations whose z-score is beyond 3

```
[26]: sns.boxplot(df1['living area'])
```

Out [26]

sns.boxplot(dfl[' living area])

: <AxesSubp10t : >
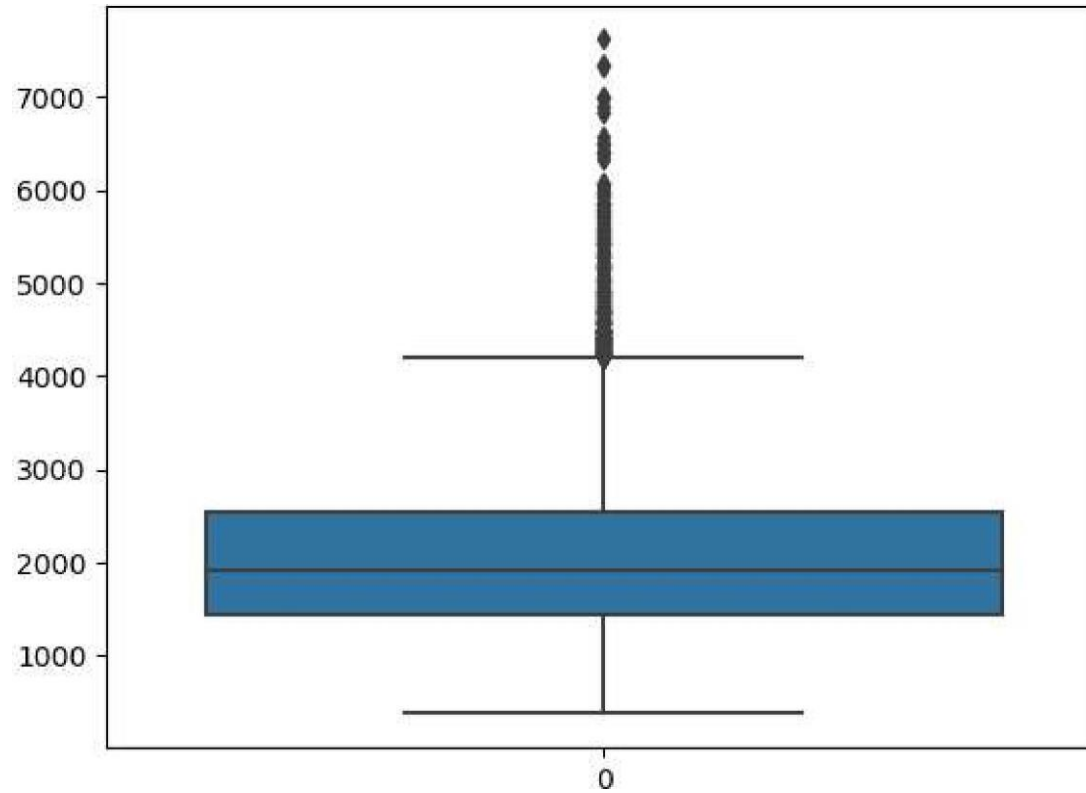
`z=np.abs(stats.zscore(df1[` ' living area']))

In [28] : len(np.where(z>3) [0])

Out[28] : 136

In [29] : len(np.where(z<-3) [e])
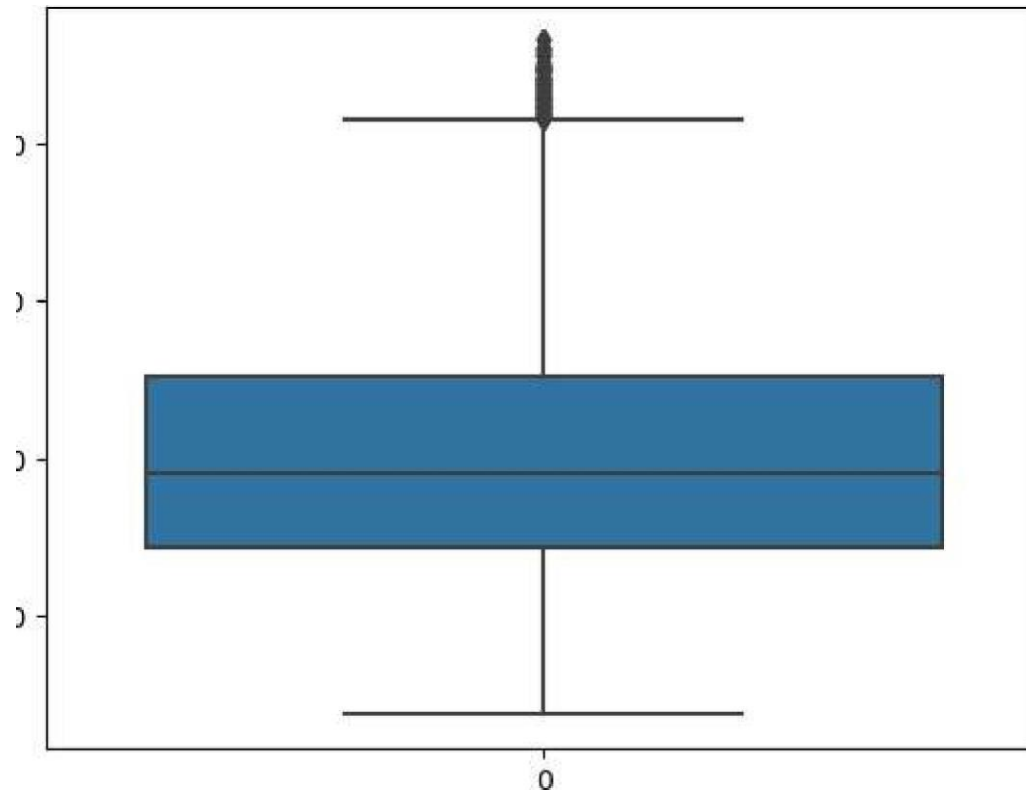
```
sns.boxplot(dfl[ ' living area ] )
```

: <AxesSubp10t : >

Out[29] : 0

---

[30] : df1=df1[(z<3)]

In [31] :

Out[31]

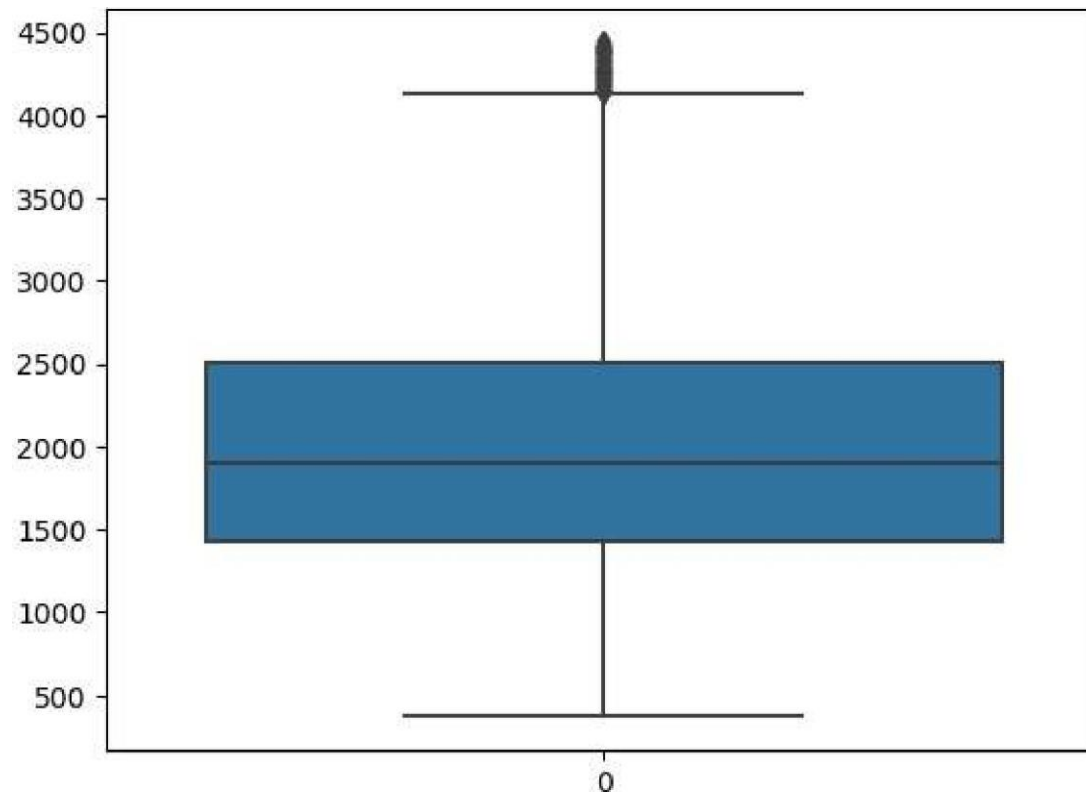4000

sns.boxplot(df1[' living area])

: <AxesSubp10t : >

3000

2000

1000

In [32] : `z=np.abs(stats.zscore(df1['living area']))`

In [33] : `len(np.where(z>3)[0])`

Out[33] : 67

In [34] : `df1=df1[(z<3)]`

In [35] : '

Out[35]

```
sns.boxplot(df1[ ' living area ] )
```

: <AxesSubp10t : >

Out[36] :

|	id	number of floors	number of views	number of ...	condition of the	grade of the	number of bedrooms	bathrooms	living area	lot area	waterfront present	Built house	Renovatio house Year	Yee
o	6762810145	5	2.50	3650	9050	2.0			4	5	10...1921			
1	6762810635	4	2.50	2920	4000	1.5	5	8...1909						
2	6762810998	5	2.75	2910	9480	1.5	3	8...1939						
3	6762812605	4	2.50	3310	42998	2.0	3	9...2001						
4	6762812919	3	2.00	2710	4500	1.5	4	8...1929						
14615	6762830250	2	1.50	1556	20000	1.0	4	7...1957						
14616	6762830339	3	2.00	1680	7000	1.5	4	7...1968						
14617	6762830618	2	1.00	1070	6120	1.0	3	6...1962						
14618	6762830709	4	1.00	1030	6621	1.0	4	6...1955						
14619	6762831463	3	1.00	900	4770	1.0	3	6...1969 200						

14244 rows x 22 columns

: <AxesSubp10t : >

There are 205 outliers in living as proved from the boxplot and the fact that there are observations whose z-score is beyond 3

In [37] : sns.boxplot(dfl['number of floors'])

Out[37]

3.5
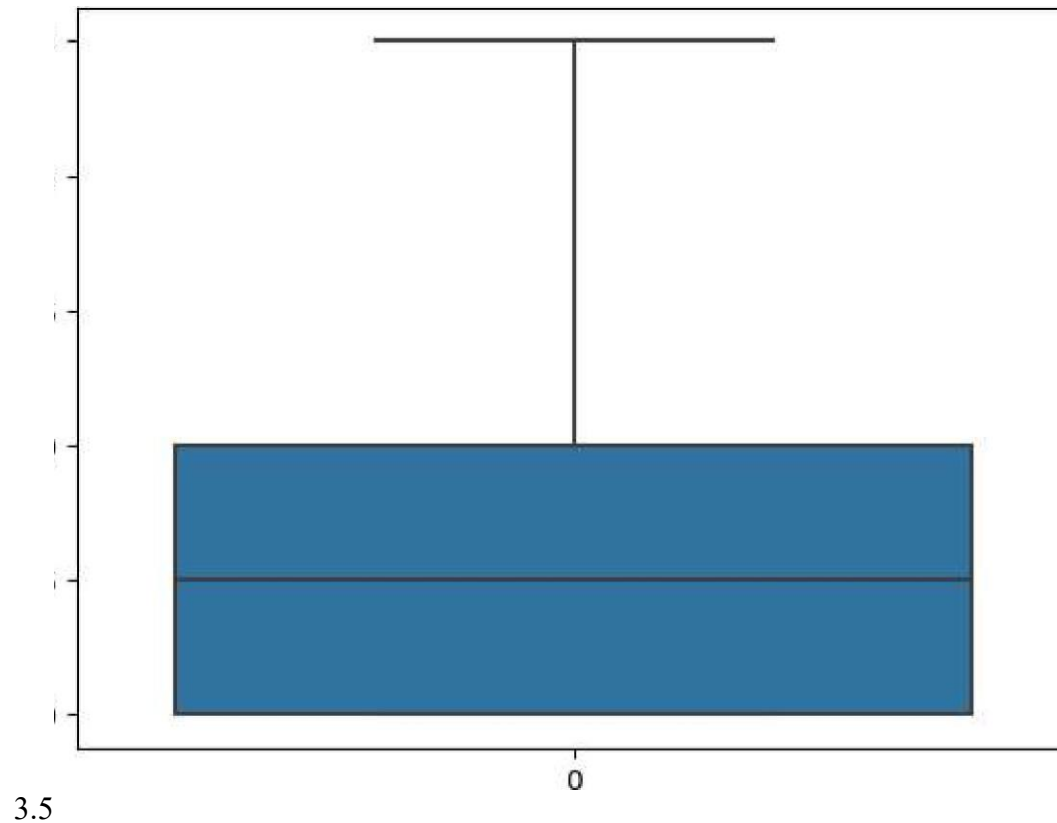
3.0

2.5

2.0

1.5

1.0

In [38]:
```
z=np.abs(stats.zscore(df1['number of floors']))
```

In [39]:
len(np.where(z>3) [0])

Out[39]: 3

In [40]:
```
df1=df1[(z<3)]
```

[41]:
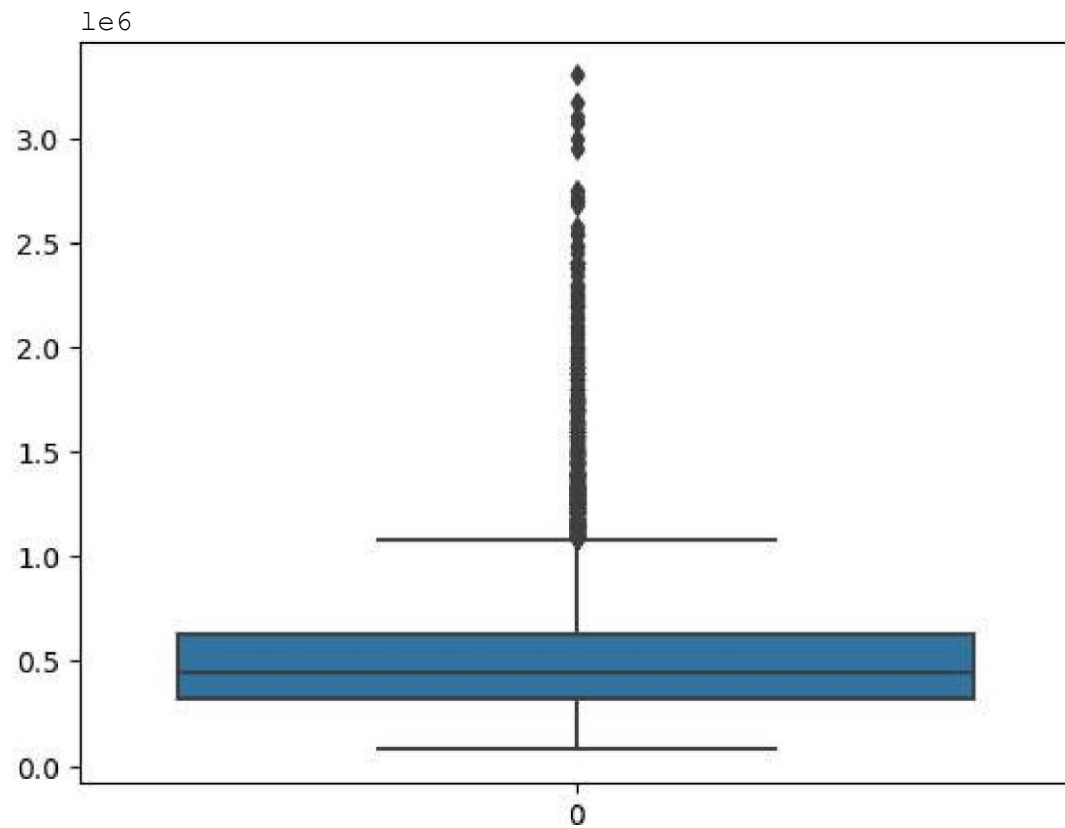sns.boxplot(dfl['number of floors'])

Out[41] : <AxesSubp10t : >
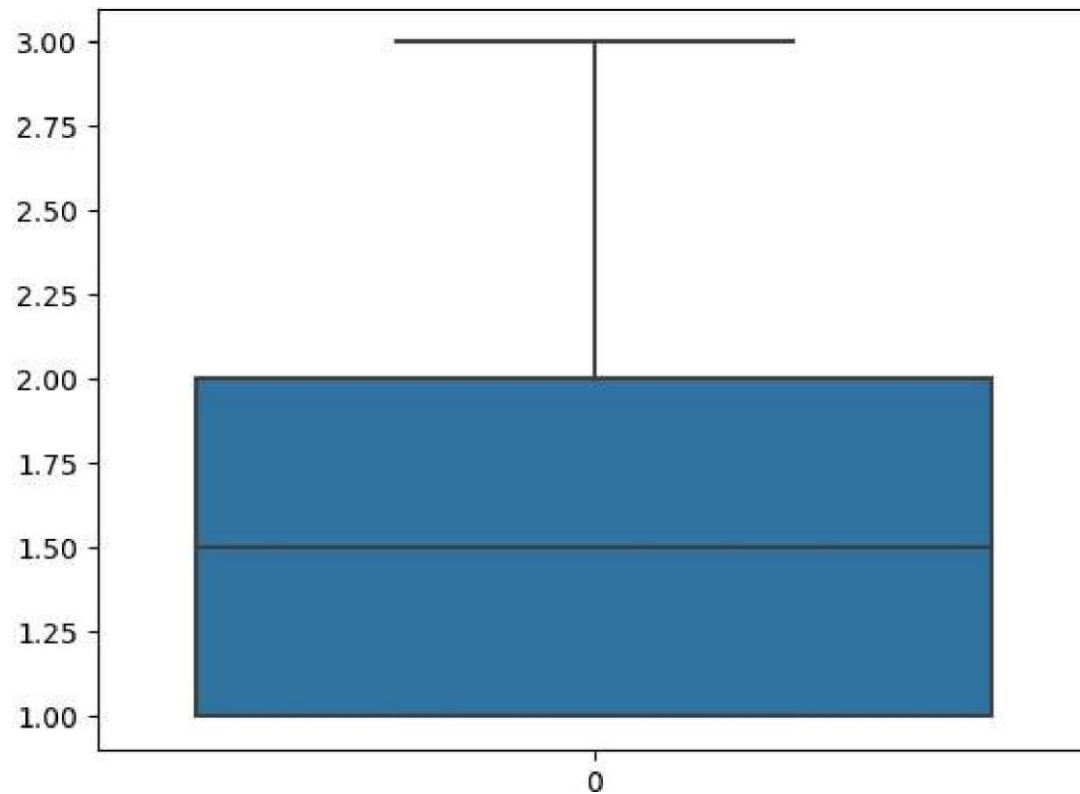
There are 3 outliers in number of floors

```
In [ 42 :
sns.boxplot(df1['Price'])
```

Out[42] : <AxesSubp10t : >

```
df1=df1[(z<3)]
```

```
z=np.  bs(stats.zscore(df1['Price']))
len(np.where(z>3)[0])
```

Out[44] : 259

dfl

Out [46] :

id	number of bedrooms	number of bathrooms	number of the area	condition area	grade floors	number of the present	living views	lot house	waterfront house	Built Year	Renovatio Yee
2 6762810998	5	2.75	2910	9480	1.5	3	8	...1939			
3 6762812605	4	2.50	3310	42998	2.0	3	9	...2001			
4 6762812919	3	2.00	2710	4500	1.5	4	8	...1929			
5 6762813105	3	2.50	2600	4750	1.0	4	9	...1951			
6 6762813157	5	3.25	3660	11995	2.0	2	3	10...2006			
14615 6762830250	2	1.50	1556	20000	1.0	4	7	...1957			
14616 6762830339	3	2.00	1680	7000	1.5	4	7	...1968			
14617 6762830618	2	1.00	1070	6120	1.0	3	6	...1962			
14618 6762830709	4	1.00	1030	6621	1.0	4	6	...1955			
14619 6762831463	3	1.00	900	4770	1.0	3	6	...1969	200		

13982 rows x 22 columns

In [47]

```
df1=df1.drop(['Renovation Year'],axis=1)
```

:

In [48] : dfl

	number of bedrooms	number of bathrooms	number of the area	condition area	grade floors	Area of the present	number of living views	lot house	waterfront	Built

|	id of house	of basement	of	of the of the...the	bathrooms	area	area	present	Year	bedrooms	floors	views	house
2	6762810998	5	2.75	2910	9480	1.5	3	8...o	1939				
3	6762812605	4	2.50	3310	42998	2.0	3	9...o	2001				
4	6762812919	3	2.00	2710	4500	1.5	4	8...830	1929				
5	6762813105	3	2.50	2600	4750	1.0	4	9...900	1951				
6	6762813157	5	3.25	3660	11995	2.0	2	3	10...o	2006			
14615	6762830250	2	1.50	1556	20000	1.0	4	7...o	1957				
14616	6762830339	3	2.00	1680	7000	1.5	4	7...o	1968				
14617	6762830618	2	1.00	1070	6120	1.0	3	6...o	1962				
14618	6762830709	4	1.00	1030	6621	1.0	4	6...o	1955				
14619	6762831463	3	1.00	900	4770	1.0	3	6...o	1969				

13982 rows x 21 columns

B1 - VARIATE ANALYSIS

The column Renovation year have been removed. This is because most of the Renovation Year are O and proves to be of no use to the model

In [49] :

```python
sns.barplot(data=df1,x='number of bedrooms',y='Price')
```

<AxesSubp10t:x1abe1='number of bedrooms '   ylabel=' Price' >
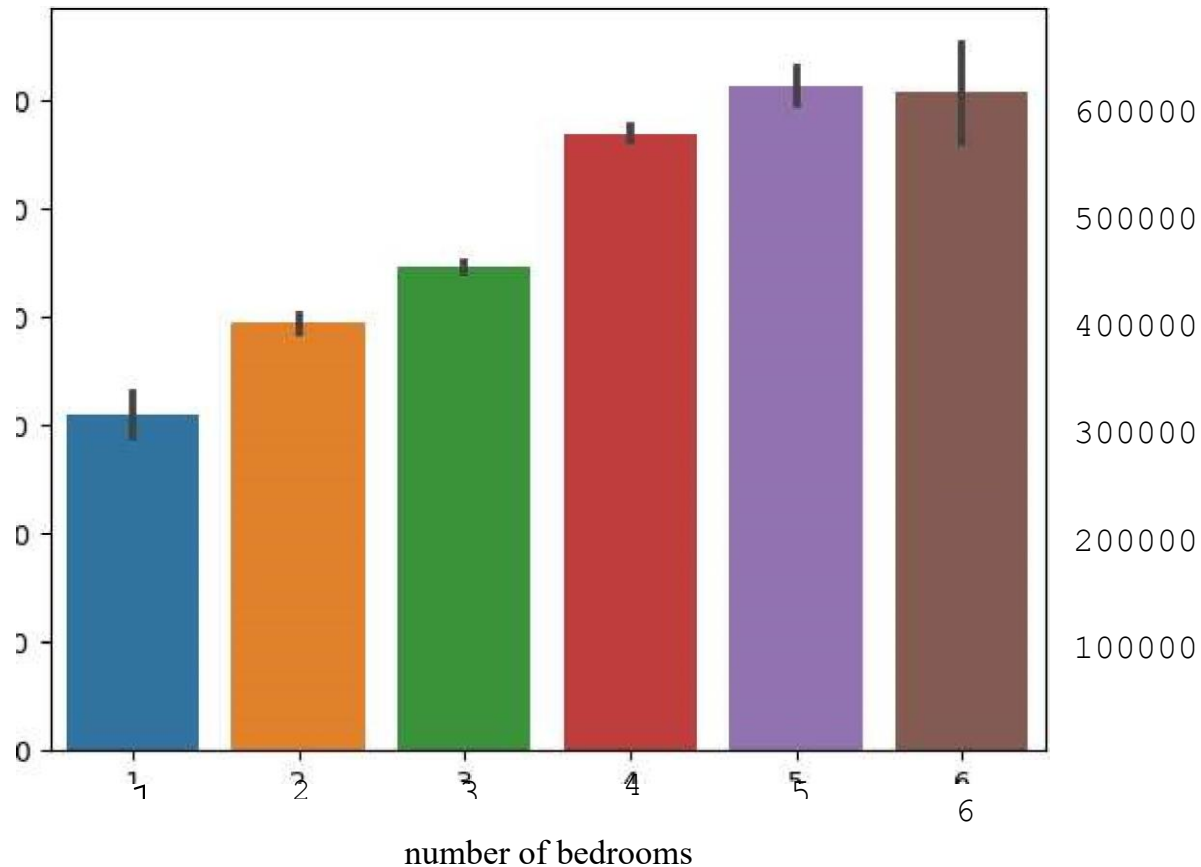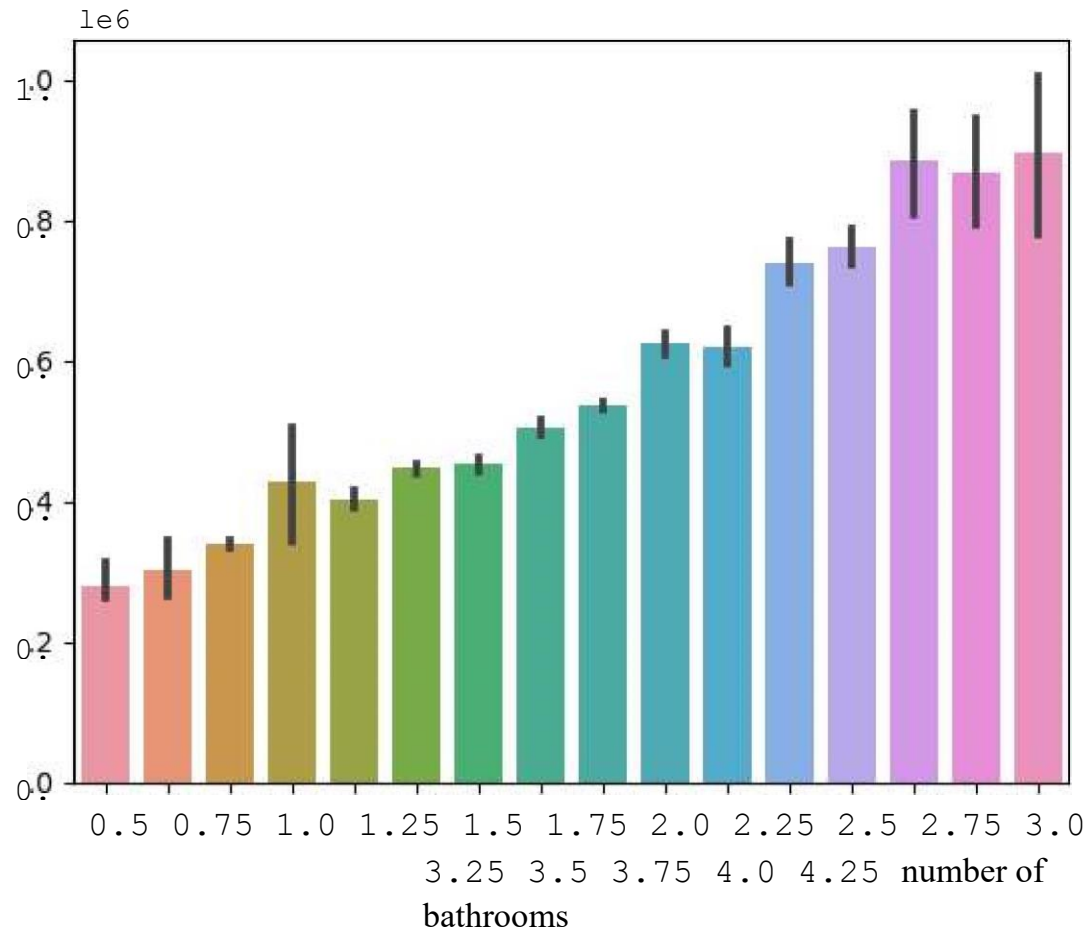

number of bedrooms

Out [46] :

Clear indication of Price increasing with number of `bedrooms`
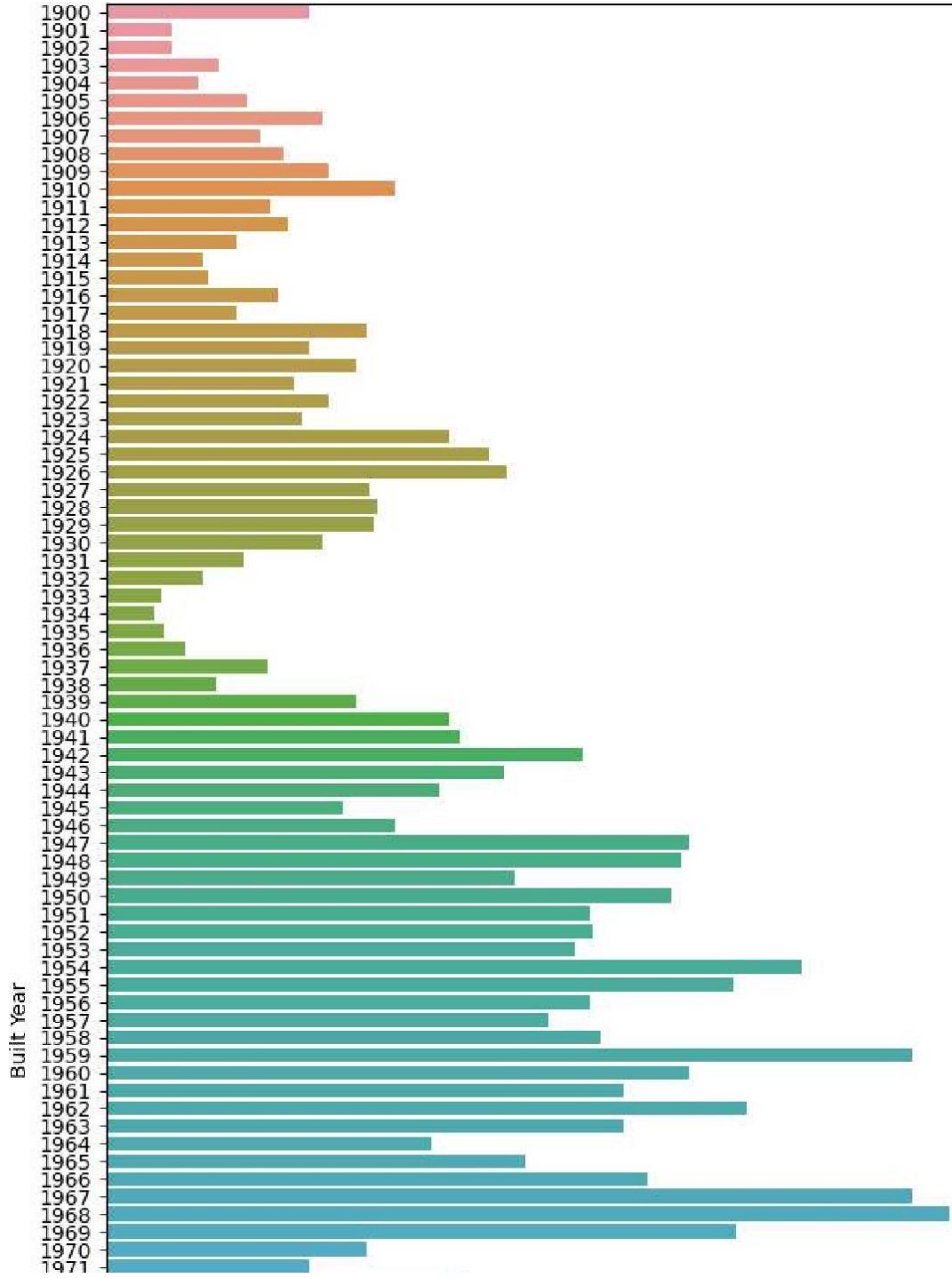
```
[50]:  sns.barplot(data=df1,x='number of bathrooms',y='Price')
```

Out[50] :<AxesSubplot:xlabel='number of bathrooms' ,ylabel='Price'>

Clear indication of Price increasing with number of bathrooms

In [51]
```
plt.figure(figsize=(12,18  :    18))
sns.countplot(data=df1,y=  ' Built Year' )
Plt.show()
```
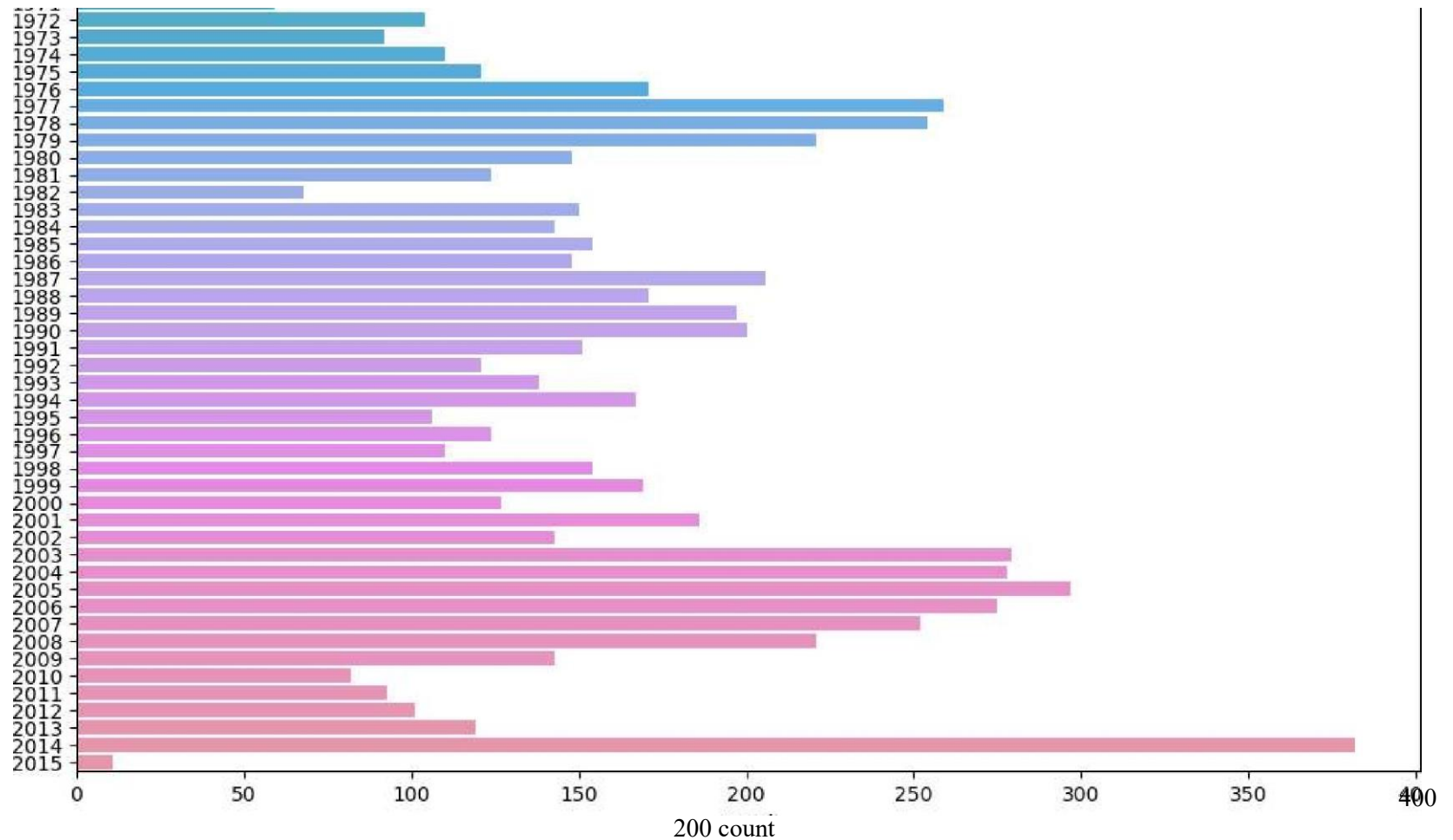
Most of the houses were listed for sale in 2017

---

ln [52] : plt.figure(figsize=(12,18)) sns.countplot
(data=dfl,y='Posta1 Code') Pl t . show()
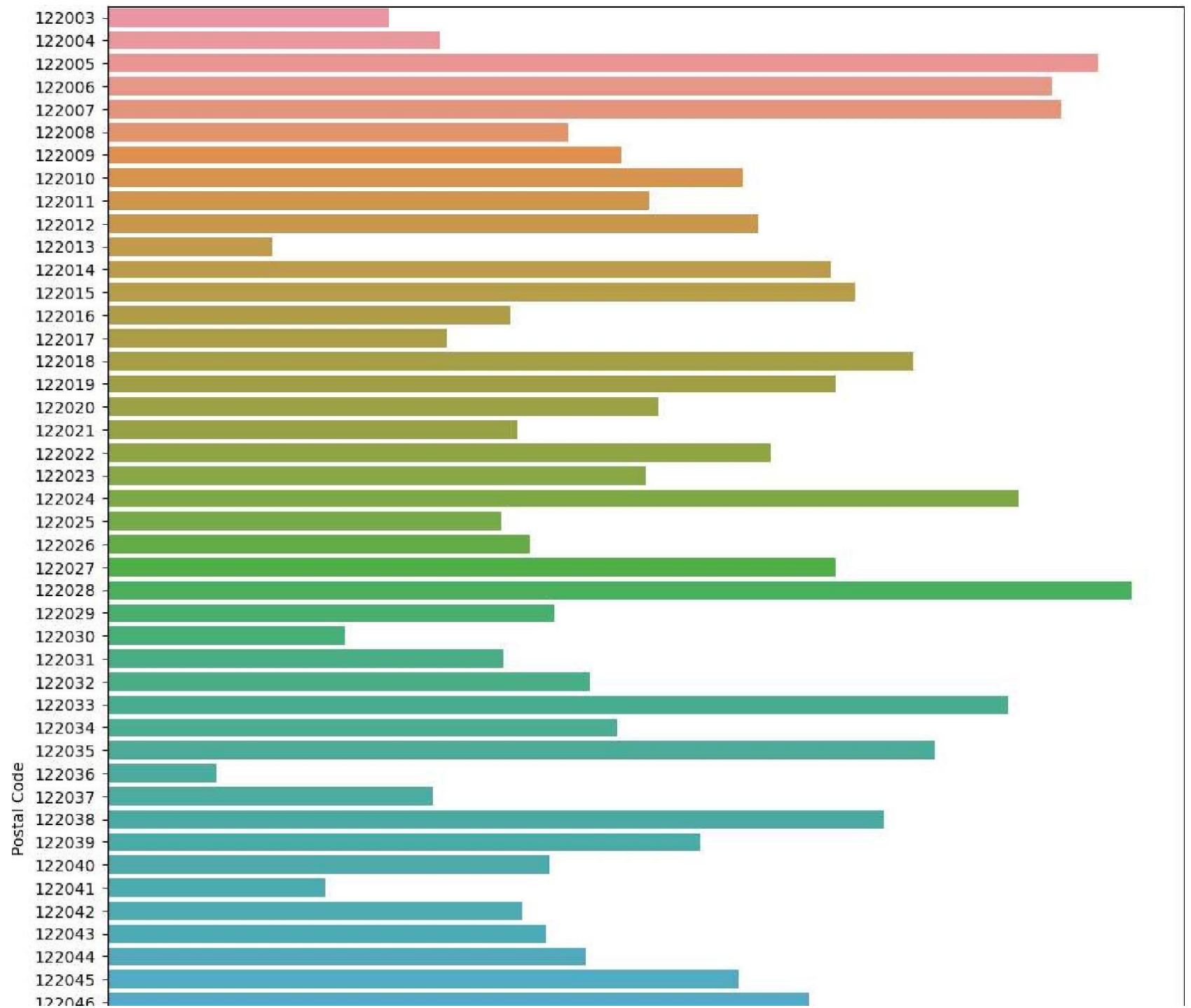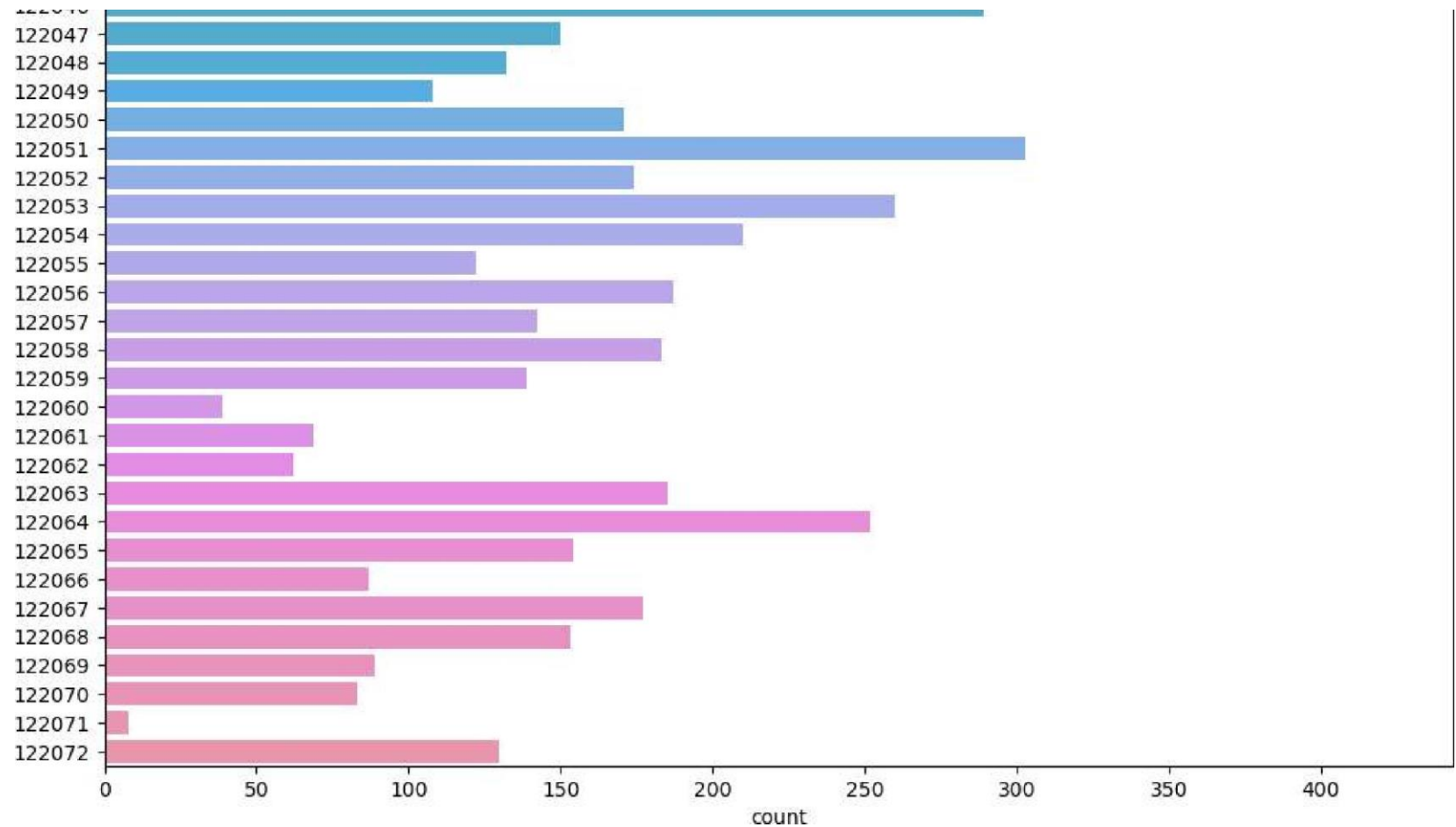
Most of the houses listed for sale are from the Pincode 122028

In [53] : df1[d .Built Year -2014] [' L attitude'] . mean()

Out[54] :

Out [53]

:   52.77583376963351

In [54] :

```python
df1[df1['Built Year']==2014]['Longitude'].mean()
```
          -114.38898952879582

[55] m = folium. Map(location   [52.77, -114.4], tiles = 'OpenStreetMap' , zoom_start=8)

Out[55] :

```
for index, location_info in df1[(df1['Built Year']==2014) & (df1['Distance from the airport']<=70)].iterrows():
    folium.Marker([location_info["Lattitude"], location_info["Longitude"]], popup=location_info["Price"],icon=folium
m
```
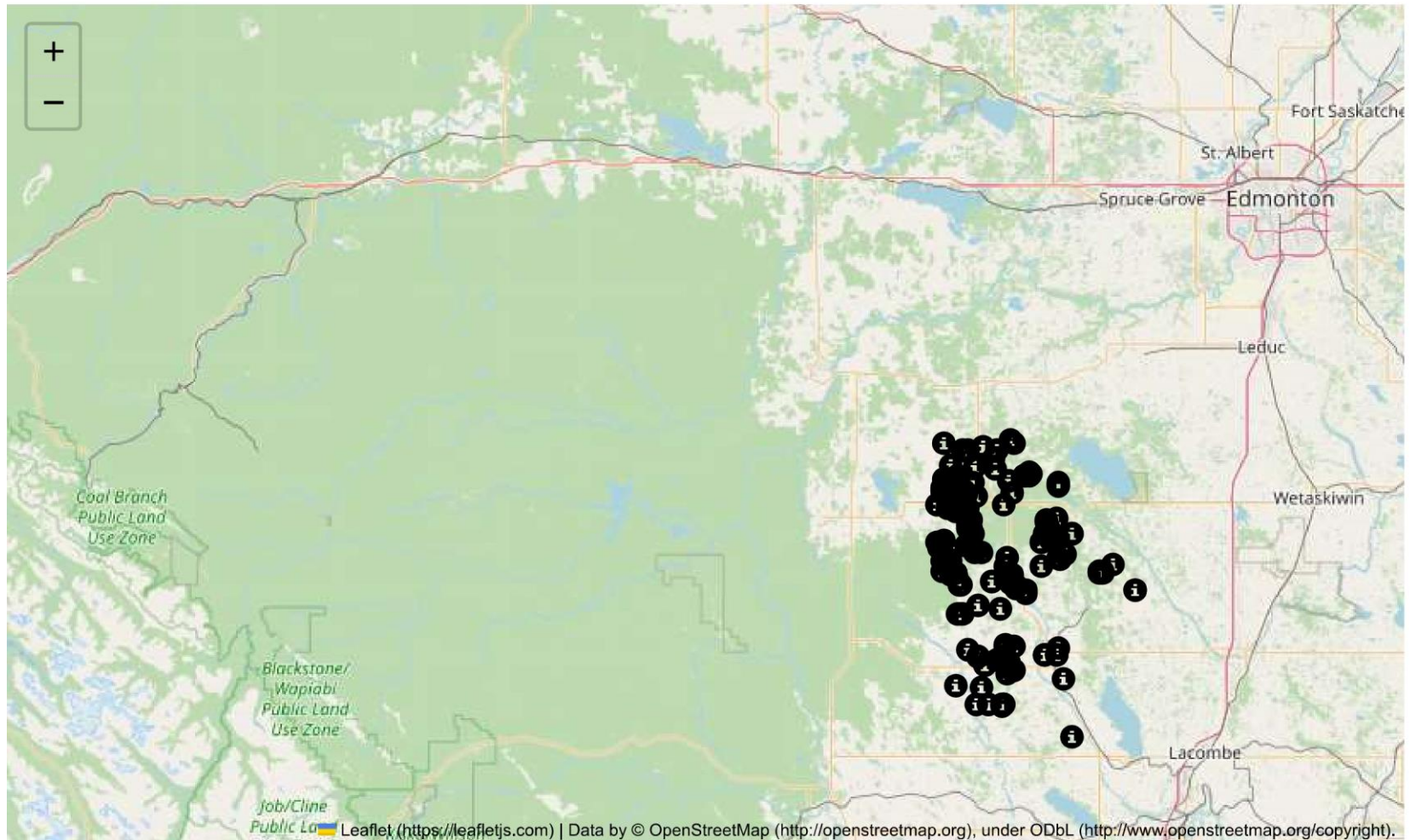

In [56] :

```
df1[df1['Built Year']>=2014]['Lattitude'].mean()
```

52 .77850305343512

Out[54] :

In [57] : `df1[df1    [ 'Built Year'|>=2014][ ' Longitude' ] . mean()`

-114.39186768447837

Out[57] :

```
m =folium. Map (location = [52.77, -114.4], tiles = 'OpenStreetMap' ,
    zoom_start=8)

for index, location _info in df1[(df1[ 'Built Year' ]>=2014) & (dfl[ 'Distance from the airport'
folium.Marker([10cation_info["Lattitude"], location_info["Longitude"] ], m
```
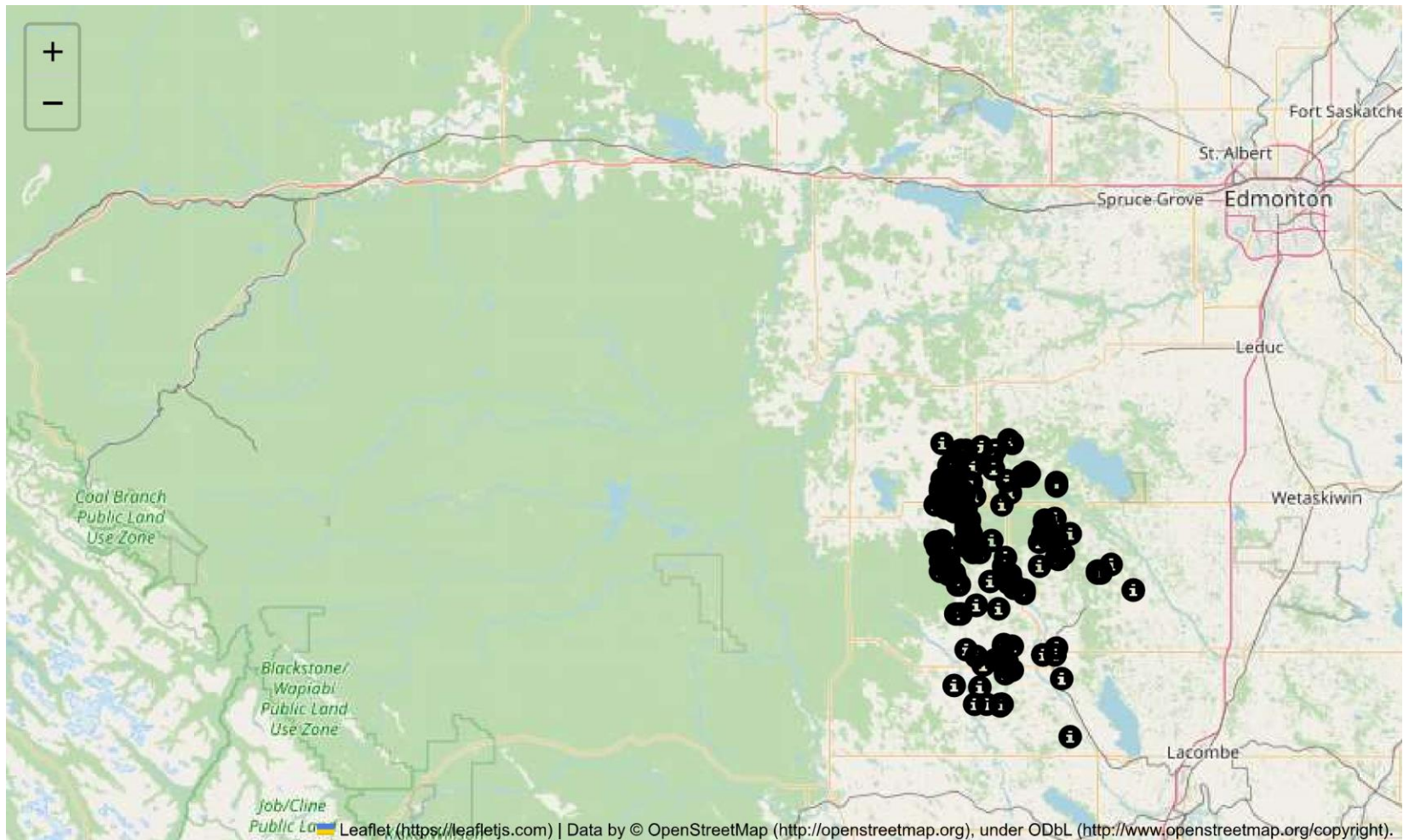
The houses listed for sale in this dataset are located in Alberta, Canada

```
df1=df1.drop(['id'],axis=1)
```

Out[54]:

In [60]:
```python
df1=df1.drop(['Postal Code'],axis=l)
```

MULTI - VARIATE ANALYSIS

Columns ID and Postal Code have been dropped from df as an increase or decrease in Postal Code shall not directly impact the Price of the property

In [61] : plt.figure(figsize=(15,15)) sns . heatmap(dfl . corr() ,
    linewidths=0.5, annot=True, cmap= ' Blues ') PIt . show()

number of bedrooms			0.023		'-0,035 0,041 01026			0.28	0.17 -0.036 0.15		0.016 0.0033-0.003	
number of bathrooms	0.49		0.05			0.62	0.63	0.21	0.008 0.24		0.047 0.0017 0.011	

1.0

liVing area   0.6   0.71   0.15   0.72   0.85   .36   0.34   0.16 0.00060.0055

                                                                                                                        0.8

        o. 05   0.15   -0.014 0.031 0.075 -0.0047 0.087 0.16 -0.00240.042 -0.097 0.21 0.14   0.00890.0055 0.078

number of floors 0.16   -0,014   -0,011 40.023 0.28   -0.3   0,041 0.13   0.27 -0.023 -0,007 0,017 0.2

waterfront present•-O.035 -0.004 0.011 0.03i -0.011   01019-0.00460.0038 0.02? -0.039 -0.047 -0.0" 0.02 0.038 -0.01 -0.0086 0.090.6

number of views - 0.041 0-1   0.18 0.075 -0.023   0.046 0.16 0,067 0.22 -0.072-0.027 -O.OB9 0.21 0.067 0.0027-0.0058 0.2

condition of the house - 0.02& -0.13 -0.071-0.0047 -0.23 0,019 0,046   -0.17 -0.19   0.2   -0.38 -0.0051 -0.12

                0.4 grade of the house 0.34   0.62   0.72   0.004& o.lí -0.17   0.72   0.07   0.1   o. 093 -0.00140.0078

Area of the house(excluding basement)   0116   0,0038 0,067 -0.19   -0.18   -0,031   0.17 -0.00370.0067

Area of the basement   .28 0.21   .0024 -0.3 0,027 0.22   0.2   0.07 -0.18   -0.17 0.1L -0.17 '0.11 -0.011 0.0077-0.0016, 0.2

                                                                                                                        0.2

        Built year 0.17   0.042   *0.039   -0.38   0.47   0.46   0.06+0.00038000410.047

Lattitude -90.036 0.008 0028 -0,097 0.041 -0,047   0.1   -0.031 0.11 -0.15   -0.13 0.028 -0.1   O.OL6 0.0078

Longitude 0.15   0.24   0.28   0.21 0.13 0069 -0.089 -0.12 0.22   0.39   0.41   0.25 -0.00911000420.04

                                                                                                                        0.0

livi   0.39   0.53   0.74   0.68   0.72   0,028   0,17 -0.007-0.001

IOt area renov -0.016 0.047 0.16   -0.023 0,038 0,067-0.000670.093 0.17 -0,011 0.063 -0.1   0.25 0.17   -0.023 -0.012 0.065

Number ot schools nearbY -0.00330.00170.00058).0089-0.007 -0.01   0.023   -000100026

Distance from the airport -0.0033 0.011 0.00550.0055 0.017   0.0051

Price   0.07B 0.27   0.091 00.28   0.053   0.065 0.00260.0051   5'   0.2S o.

047   0.049

number of bedrooms
number of bathrooms
living area
lot area
number of floors
waterfront present
number of views
condition of the house
grade of the house
Area of the house(excluding basement)
Area of the basement
Built Year
Lattitude
Longitude
living_area_renov
lot_area_renov
Number of schools nearby
Distance from the airport
Price

Columns like 'lot area','condition of the house','Built Year','lot_area_renov','Number of schools nearby','Distance from the airport','Longitude' contribute minimal to Price which is the Target variable. Hence it is removed before training

In [62] : `df1=df1.drop([` 'lot area' , ' condition of the house' , ' Built Year' , 'lot area renov' , 'Number of schools nearby' , 'Distance

In [63] : plt.figure(figsize=(15,15)) sns .heatmap(dfl . corr() ,
linewidths=0.5, annot=True, cmap= ' Blues ') PIt . show()

	bedrooms	bathrooms	livingarea	floors	waterfront	views	grade	Area(excl. basement)	Area basement	Lattitude	living_area renov	Price
number of bedrooms	1	0.49	0.6	0.16	-0.035	0.041	0.34	0.47	0.28	-0.036	0.39	0.31
number of bathrooms	0.49	1	0.71	0.51	-0.004	0.1	0.62	0.63	0.21	0.008	0.53	0.47
livingarea	0.6	0.71	1	0.34	0.011	0.18	0.72	0.85	0.36	0.028	0.74	0.65
number of floors	0.16	0.51	0.34	1	-0.011	-0.023	0.46	0.53	-0.3	0.041	0.27	0.27
watertront present	-0.035	-0.004	0.011	-0.011	1	0.33	-0.0048	-0.0038	0.027	-0.047	0.02	0.091
number ot views	0.041	0.1	0.18	-0.023	0.33	1	0.16	0.067	0.22	-0.027	0.21	0.28
grade of the house	0.34	0.62	0.72	0.46	-0.0048	0.16	1	0.72	0.07	0.1	0.68	0.66
Area of the house(excluding basement)	0.47	0.63	0.85	0.53	-0.0038	0.067	0.72	1	-0.18	-0.031	0.72	0.54
Area of the basement	0.28	0.21	0.36	-0.3	0.027	0.22	0.07	-0.18	1	0.11	0.11	0.25
Lattitude	-0.036	0.008	0.028	0.041	-0.047	-0.027	0.1	-0.031	0.11	1	0.028	0.4
living_area renov	0.39	0.53	0.74	0.27	0.02	0.21	0.68	0.72	0.11	0.028	1	0.58
Price	0.31	0.47	0.65	0.27	0.091	0.28	0.66	0.54	0.25	0.4	0.58	1

number of bedrooms · number of bathrooms · living area · number of floors · waterfront present · number of views · grade of the house · Area of the house(excluding basement) · Area of the basement · Lattitude · living_area_renov

Training of Model, Splitting of Dataset into Train and Test Set

In [64] :
```
from   sklearn   .   model_selection
import train_test_split
```

In [65] :
```
X=df1.drop( [ ' Price' ],axis =1)
```

In [66] : X. shape

Out[66] : (13982, 11)

In [67] : y_df1[' Price ']

In [68] : y. shape

Out[68] : (13982,)

In [69] :
```
X_train, X_test,   train_test_split   (X,   y,
y_train,           test_size=0.2, random_state=11)
```

Out[71] : (2797, 11)

721:

```python
from sklearn.pipeline import make_pipeline from sklearn.preprocessing import
StandardScaler from sklearn.linear_model import ElasticNet, Lasso,
LinearRegression, RidgeCV from catboost import CatBoostRegressor from
sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor f rom
xgboost import XGBRegressor from sklearn.tree import DecisionTreeRegressor from
sklearn.ensemble import StackingRegressor from sklearn.svm import SVR
```

In [73] :
```python
pipelines __ {en' :make_pipeline(StandardScaler(),
    ElasticNet()),
    ' lasso' :make_pipeline(StandardScaler(), Lasso())'
    'Rcv ':make_pipeline(StandardScaler(), RidgeCV()),
    'CatB' : make_ pipeline(StandardScaler(), CatBoostRegressor(eva1_metr1c• ='RMSE ' ,verbose-
    1000) ) ,
    ' lr ' :make_pipeline(StandardScaler(), LinearRegression()),
    ' rf' :make_pipeline(StandardScaler(), RandomForestRegressor()),
    'gb' :make_pipeline(StandardScaler(),  GradientBoostingRegressor()), '
    dtc :make_pipeline (StandardScaler()  ,  DecisionTreeRegressor()),  . xg'
    :make_pipeline(StandardScaler() ,XGBRegressor())
```

In [74] :

```python
fit_models = {}
```

In [70] : X_train . shape

Out[70]
:
(11185, 11)

In [71] : X_test . shape

```python
for algo, pipeline in pipelines. items() :
    model      pipeline.fit(X_train, y_train)
    fit_models[algo] - model
```

/opt/conda/1ib/python3.7/site-packages/sk1earn/1inear_mode1/_coordinate_descent.py:648: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider inc reasing regularisation. Duality gap: 4.781e+12, tolerance: 5.929e+1e coef 11_ reg, 12_ reg, X, y, max_iter, tol, rng, random, positive Learning rate set to 0.05996 e: learn: 221490.1496581 total: 61.4ms remaining: 1m Is

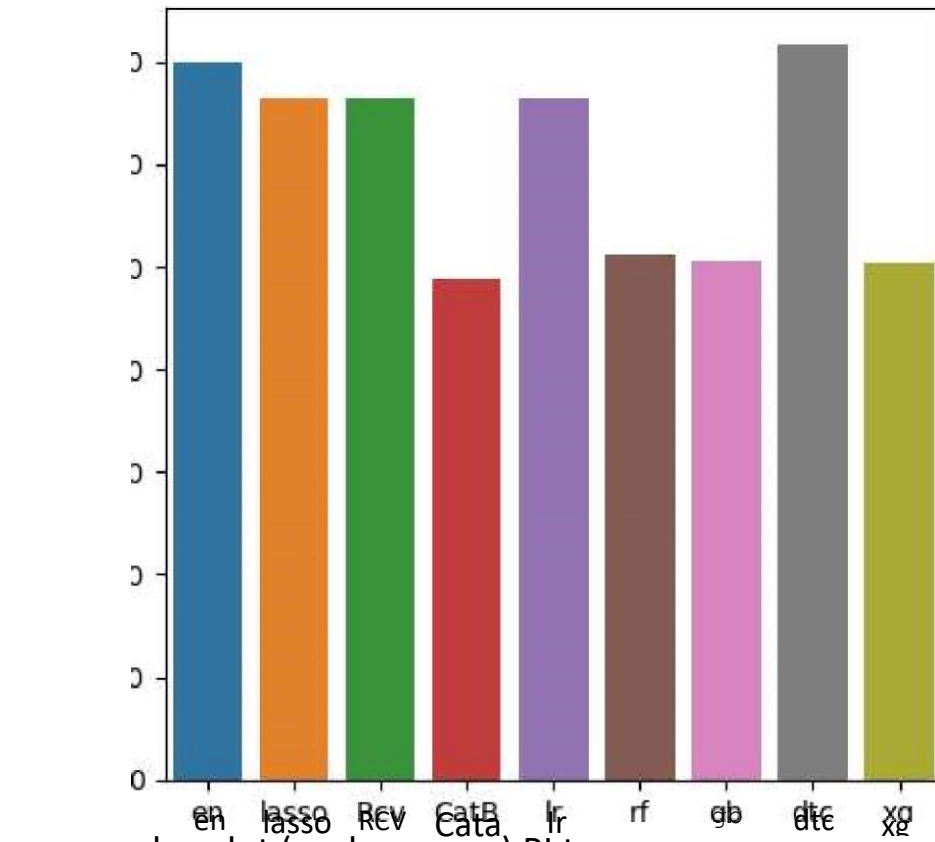999:   learn: 77595.2298921   total: 2.85s   remaining: eus

```
[75]: from sklearn . metrics import mean_absolute_error, mean_ squared _error

      for algo, model in fit_models.items() :
          Yhat  model . predict (X_test)  al . append (algo) maes . append
          (mean_squared_error (y_test , yhat) * *0.5) print(algo, 'MEAN ABSOLUTE ERROR'
          , mean_absolute_error(y_test, yhat) ) print(algo, 'ROOT MEAN SQUARED ERROR'
          ,mean_squared_error(y_test,yhat)**0.5)
```

en MEAN ABSOLUTE ERROR 104444.32355671145 en ROOT MEAN SQUARED ERROR 140011.53917862213 lasso MEAN ABSOLUTE ERROR 97479.23118789196 lasso ROOT MEAN SQUARED ERROR 132916.1566456281 Rcv MEAN ABSOLUTE ERROR 97481.91673717603

Rcv ROOT MEAN SQUARED ERROR 132918.333682342

CatB MEAN ABSOLUTE ERROR 66637.30790160663

CatB ROOT MEAN SQUARED ERROR 97508.34029611414 Ir MEAN ABSOLUTE ERROR 97574.48622571728 Ir ROOT MEAN SQUARED ERROR 132952.7515959945 rf MEAN ABSOLUTE ERROR 69217.89879907611 rf ROOT MEAN SQUARED ERROR 102292.3632979867 gb MEAN ABSOLUTE ERROR 69874.84067217445 gb ROOT MEAN SQUARED ERROR 101056.41447857216 dtc MEAN ABSOLUTE ERROR 96944.72285782386 dtc ROOT MEAN SQUARED ERROR 143316.21683052482 xg MEAN ABSOLUTE ERROR 69035.05210660976 xg ROOT MEAN SQUARED ERROR 100694.41040458805

```
In [76] : plt.figure(figsize=(5, 5)) plt.xlabel(
          'ML Algorithms. . . ') plt.ylabel(
```

'Root Mean Squared Errors. ')

ax=sns . barplot (x=al, y=maes) Pl t .
show()

140000

120000

2 100000

LLI

80000

60000

Root Me a 40000

200001

ML Algorithmsv..

[ ]:

```
CatB = CatBoostRegressor(verbose=1000,eval_metric=' RMSE
' )    rf    =    RandomForestRegressor()    gb    =
GradientBoostingRegressor() xg = XGBRegressor()
1r=LinearRegression()

stregr - StackingRegressor(estimators=[( 'catb' ,CatB), ( 'xg' , xg),('gb',gb)],
                          final_estimator=lr)

pipeline - make_pipeline(
    StandardSca1er(),
    stregr

pipeline. fit (X_train, y_train)
```

```python
# Generate predictions on the test set y_pred
pipeline. predict (X_test)

# Evaluate the model print( " Root Mean Squared Error: %.4f" %
mean_squared_error(y_test,y_pred)**0.5)
```

Learning rate   set to 0.05996
e :   learn : 221490. 1496581   total : 4.18ms   remaining: 4.18s
999:   learn : 77595.2298921   total : 2.81s   remaining: eus
Learning rate   set to 0.057883
e: learn: 222091.4863333   total : 3. 52ms   remaining: 3. 51s
999 : learn : 76337 . 1933964 total : 2.52s   remaining: eus
Learning rate   set to 0.057883
e :   learn : 222546. 8538661   total : 2 .94ms   remaining: 2. 94s
999 : learn : 75466.   5961681   total : 2. 51s   remaining: eus
Learning rate   set to 0.057883
e :   learn : 223455 . 5230951   total : 3.2ms   remaining: 3.2s
999:   learn : 75656.   3661258   total : 2.52s   remaining: eus
Learning rate   set to 0.057883
e: learn : 221606.9467960   total : 3.71ms   remaining: 3.7s
999:   learn : 75195   .9699196   total : 2.46s   remaining: eus
Learning rate   set to 0.057883
e: learn : 219316.0911020   total : 2.47ms   remaining: 2.47s

[ ] mean_ squared _error (y_test , y_pred)   .5

[ ] al . append(stacked model') maes . append (mean_squared_error
    (y_test, y_pred) * *0.5)

[ ] for i in range(10) :
        print("The RMSE of" , al [i] , ' is ' ,maes[i])

[ ] plt.figure(figsize=(9,5))
    plt.xlabel('ML Algorithms . . .
     ')
    plt.ylabel(' Root Mean Squared Errors . ') ax=sns
    . barplot (x=al, y=maes) PI t . show()

ALL DONE BY RESHMA J AS NAAN MUDALVAN IBM SMARTINTERNZ ASSIGNMENT 3