NM DATA ANALYTICS ASSIGNMENT 3 - House Price dataset of India

TEAM ID : NM2023TMID07102

DONE BY SRIRAM K

Importing the necessary libraries for EDA and data preprocessing

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

Converting csv file into dataframe

```
In [3]: df=pd.read_csv('C:/Users/surya/Downloads/House Price India.csv')
```

```
In [4]: df=df.drop(['Date'],axis=1)
```

```
In [5]: df
```

	id	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	...	Built Year	Renovation Year	Postal Code
0	6762810145	5	2.50	3650	9050	2.0	0	4	5	10	...	1921	0	122003
1	6762810635	4	2.50	2920	4000	1.5	0	0	5	8	...	1909	0	122004
2	6762810998	5	2.75	2910	9480	1.5	0	0	3	8	...	1939	0	122004
3	6762812605	4	2.50	3310	42998	2.0	0	0	3	9	...	2001	0	122005
4	6762812919	3	2.00	2710	4500	1.5	0	0	4	8	...	1929	0	122006
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14615	6762830250	2	1.50	1556	20000	1.0	0	0	4	7	...	1957	0	122066
14616	6762830339	3	2.00	1680	7000	1.5	0	0	4	7	...	1968	0	122072
14617	6762830618	2	1.00	1070	6120	1.0	0	0	3	6	...	1962	0	122056
14618	6762830709	4	1.00	1030	6621	1.0	0	0	4	6	...	1955	0	122042
14619	6762831463	3	1.00	900	4770	1.0	0	0	3	6	...	1969	2009	122018

14620 rows × 22 columns

In [6]: `df.head()`

Out[6]:

	id	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	...	Built Year	Renovation Year	Postal Code	Latt
0	6762810145	5	2.50	3650	9050	2.0	0	4	5	10	...	1921	0	122003	52
1	6762810635	4	2.50	2920	4000	1.5	0	0	5	8	...	1909	0	122004	52
2	6762810998	5	2.75	2910	9480	1.5	0	0	3	8	...	1939	0	122004	52
3	6762812605	4	2.50	3310	42998	2.0	0	0	3	9	...	2001	0	122005	52
4	6762812919	3	2.00	2710	4500	1.5	0	0	4	8	...	1929	0	122006	52

5 rows × 22 columns

In [7]: df.tail()

Out[7]:

	id	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	...	Built Year	Renovation Year	Postal Code
14615	6762830250	2	1.5	1556	20000	1.0	0	0	4	7	...	1957	0	122066
14616	6762830339	3	2.0	1680	7000	1.5	0	0	4	7	...	1968	0	122072
14617	6762830618	2	1.0	1070	6120	1.0	0	0	3	6	...	1962	0	122056
14618	6762830709	4	1.0	1030	6621	1.0	0	0	4	6	...	1955	0	122042
14619	6762831463	3	1.0	900	4770	1.0	0	0	3	6	...	1969	2009	122018

5 rows × 22 columns

Checking for null and duplicated values

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

```
Out[8]:  id                                     0
         number of bedrooms                     0
         number of bathrooms                    0
         living area                            0
         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 basement)  0
         Area of the basement                   0
         Built Year                             0
         Renovation Year                        0
         Postal Code                            0
         Lattitude                              0
         Longitude                              0
         living_area_renov                      0
         lot_area_renov                         0
         Number of schools nearby               0
         Distance from the airport              0
         Price                                  0
         dtype: int64
```

```
In [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
---  ------                                 --------------  -----
 0   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  Lattitude                              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()`

|	id	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	g
count	1.462000e+04	14620.000000	14620.000000	14620.000000	1.462000e+04	14620.000000	14620.000000	14620.000000	14620.000000	14
mean	6.762821e+09	3.379343	2.129583	2098.262996	1.509328e+04	1.502360	0.007661	0.233105	3.430506	
std	6.237575e+03	0.938719	0.769934	928.275721	3.791962e+04	0.540239	0.087193	0.766259	0.664151	
min	6.762810e+09	1.000000	0.500000	370.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	
25%	6.762815e+09	3.000000	1.750000	1440.000000	5.010750e+03	1.000000	0.000000	0.000000	3.000000	
50%	6.762821e+09	3.000000	2.250000	1930.000000	7.620000e+03	1.500000	0.000000	0.000000	3.000000	
75%	6.762826e+09	4.000000	2.500000	2570.000000	1.080000e+04	2.000000	0.000000	0.000000	4.000000	
max	6.762832e+09	33.000000	8.000000	13540.000000	1.074218e+06	3.500000	1.000000	4.000000	5.000000	

8 rows × 22 columns
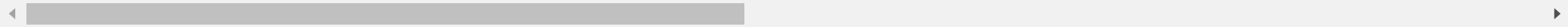
UNIVARIATE ANALYSIS

Checking for outliers

```
sns.boxplot(df['number of bedrooms'])
```

`<AxesSubplot:>`
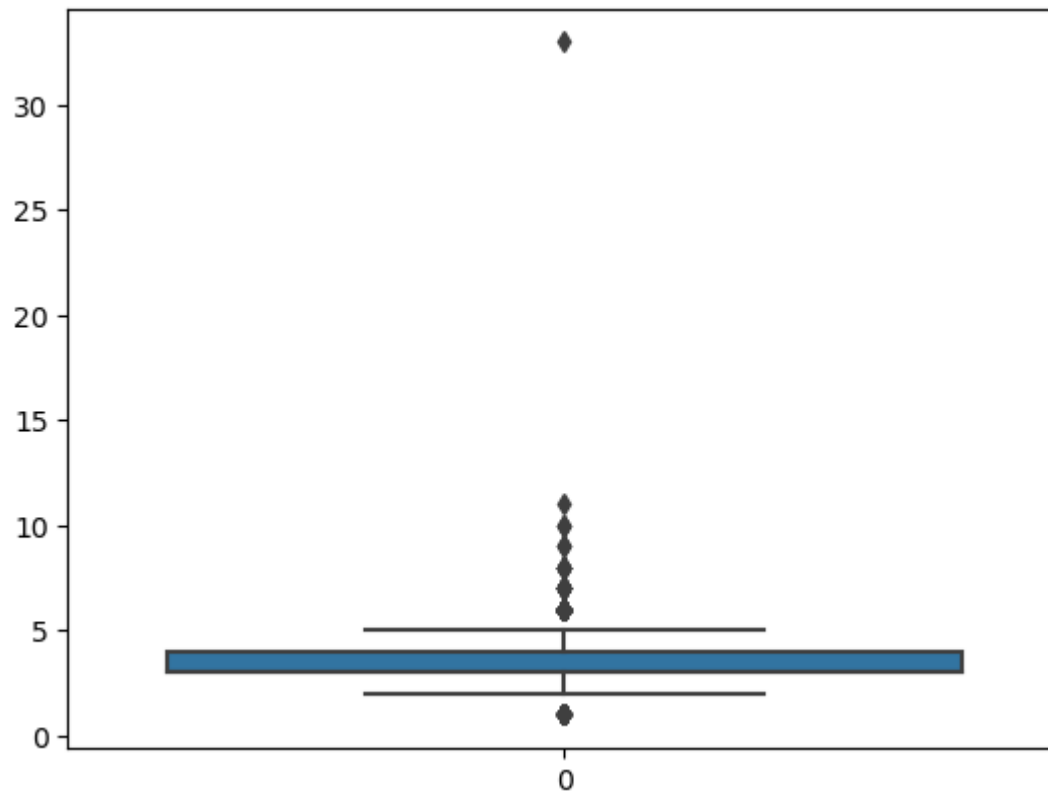
```
In [13]: z=np.abs(stats.zscore(df['number of bedrooms']))
```

```
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,   3322,   3532,   3600,   4207,   4486,
          4658,   4680,   6591,   6596,   6730,   6982,   6998,   7003,   7454,
          8559,   8650,   9282,   9629,   9810,   9955,  10168,  10177,  10676,
         10748,  10916,  10944,  11247,  11441,  11547,  11877,  12273,  13048,
         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]:
```python
df1=df[(z< 3)]
```

In [17]:
```python
sns.boxplot(df1['number of bedrooms'])
```

Out[17]: <AxesSubplot:>
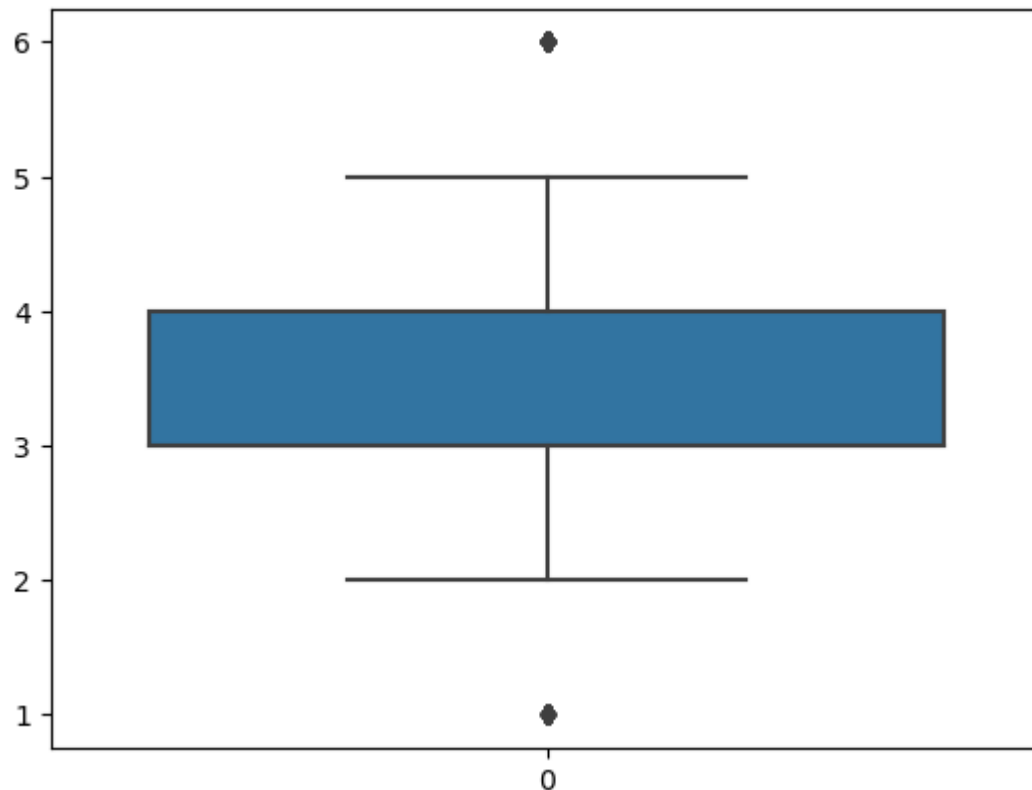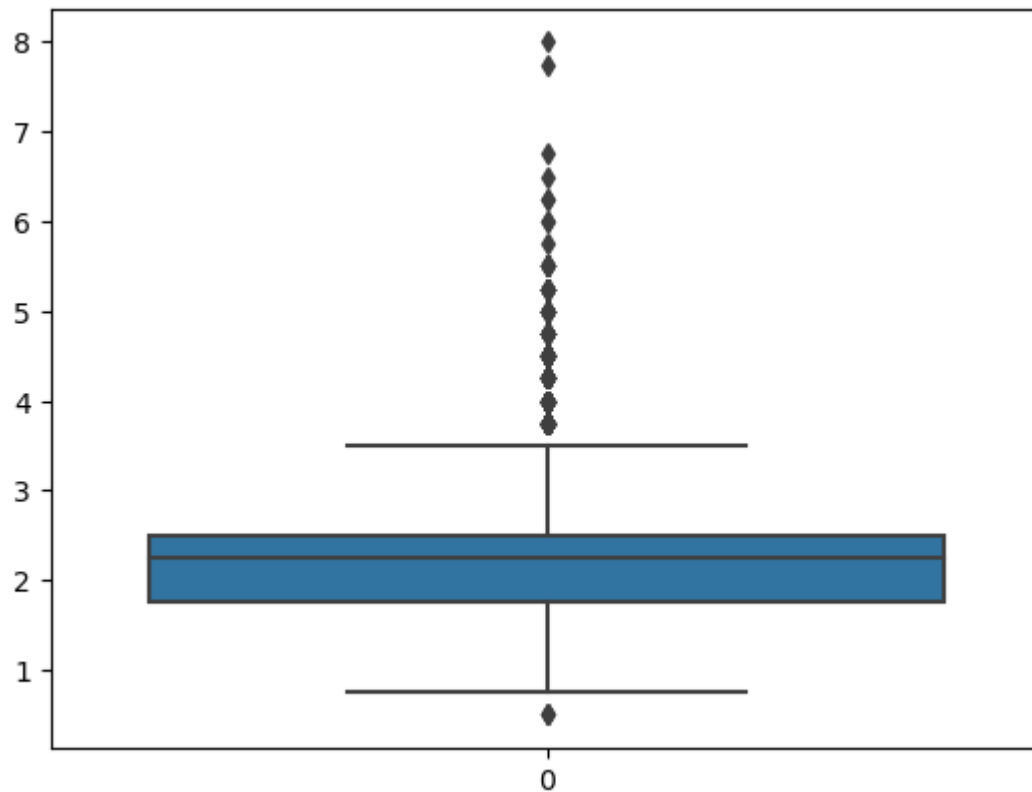

In [18]:
```python
df1
```

Out[18]:

	id	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	...	Built Year	Renovation Year	Postal Code
0	6762810145	5	2.50	3650	9050	2.0	0	4	5	10	...	1921	0	122003
1	6762810635	4	2.50	2920	4000	1.5	0	0	5	8	...	1909	0	122004
2	6762810998	5	2.75	2910	9480	1.5	0	0	3	8	...	1939	0	122004
3	6762812605	4	2.50	3310	42998	2.0	0	0	3	9	...	2001	0	122005
4	6762812919	3	2.00	2710	4500	1.5	0	0	4	8	...	1929	0	122006
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14615	6762830250	2	1.50	1556	20000	1.0	0	0	4	7	...	1957	0	122066
14616	6762830339	3	2.00	1680	7000	1.5	0	0	4	7	...	1968	0	122072
14617	6762830618	2	1.00	1070	6120	1.0	0	0	3	6	...	1962	0	122056
14618	6762830709	4	1.00	1030	6621	1.0	0	0	4	6	...	1955	0	122042
14619	6762831463	3	1.00	900	4770	1.0	0	0	3	6	...	1969	2009	122018

14571 rows × 22 columns

In [19]: `sns.boxplot(df1['number of bathrooms'])`

Out[19]: <AxesSubplot:>

```
In [20]: z=np.abs(stats.zscore(df1['number of bathrooms']))
```
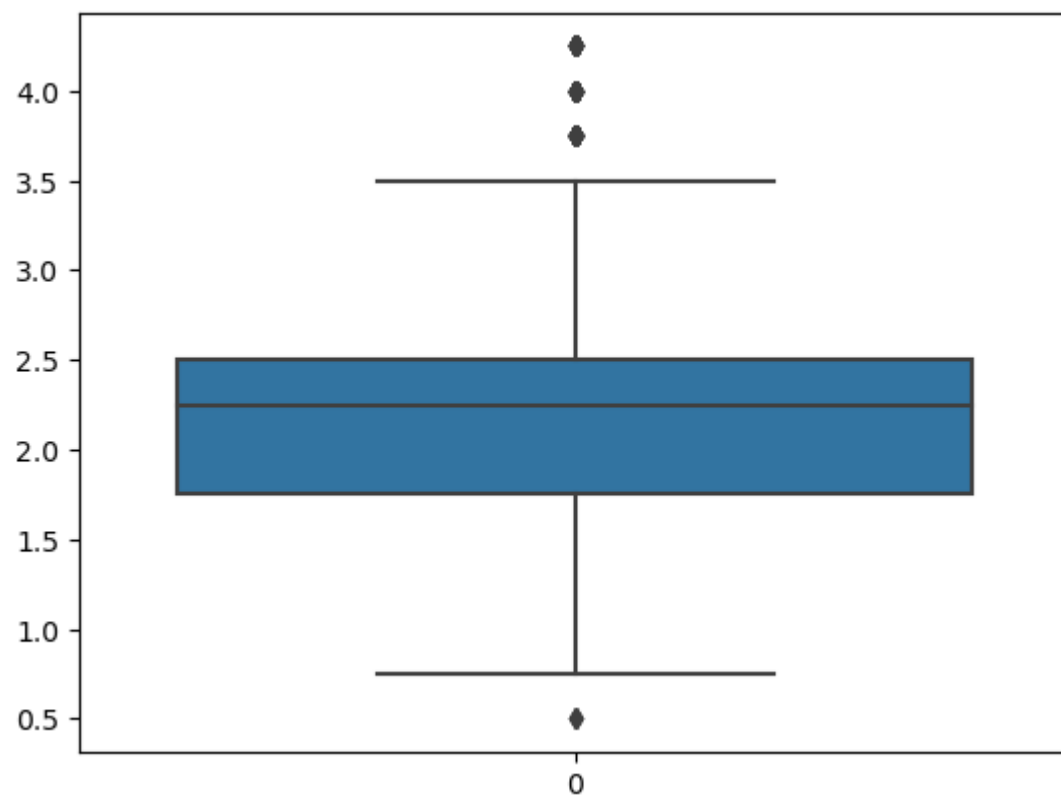
```
In [21]: len(np.where(z>3)[0])
```

Out[21]: 124

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

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

```
In [23]: df1=df1[(z< 3)]
```

```
In [24]: sns.boxplot(df1['number of bathrooms'])
```

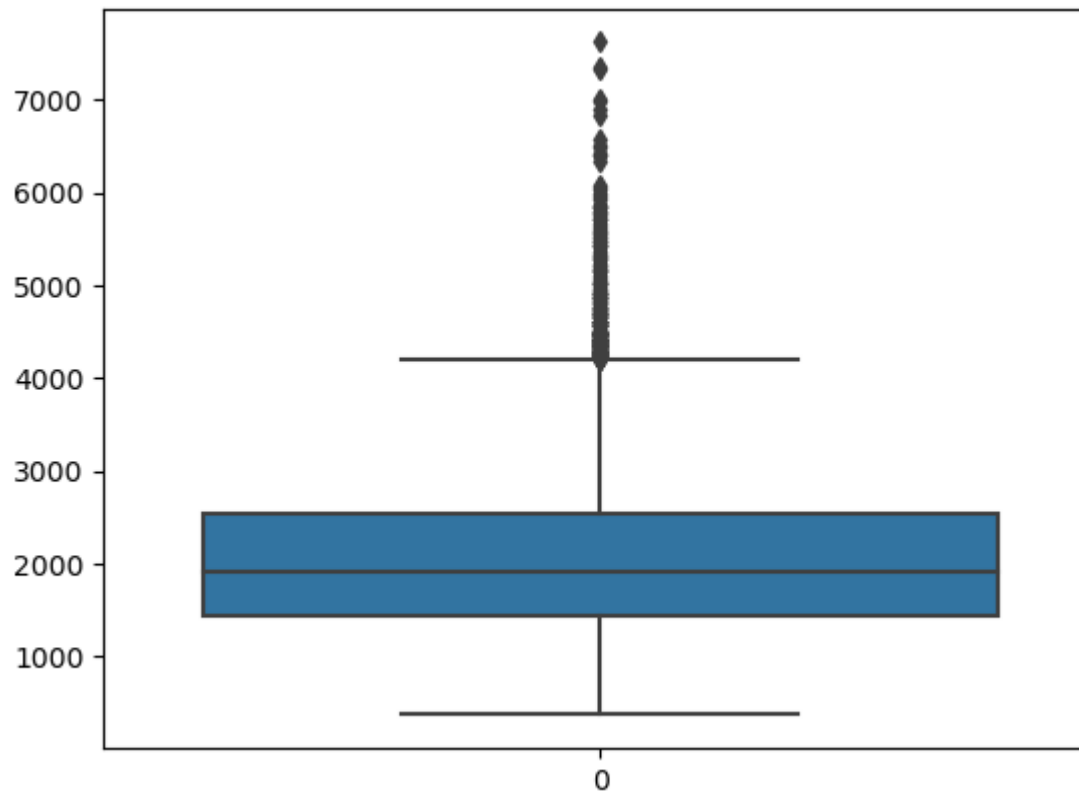|	id	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	grade of the house	...	Built Year	Renovation Year	Postal Code
0	6762810145	5	2.50	3650	9050	2.0	0	4	5	10	...	1921	0	122003
1	6762810635	4	2.50	2920	4000	1.5	0	0	5	8	...	1909	0	122004
2	6762810998	5	2.75	2910	9480	1.5	0	0	3	8	...	1939	0	122004
3	6762812605	4	2.50	3310	42998	2.0	0	0	3	9	...	2001	0	122005
4	6762812919	3	2.00	2710	4500	1.5	0	0	4	8	...	1929	0	122006
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14615	6762830250	2	1.50	1556	20000	1.0	0	0	4	7	...	1957	0	122066
14616	6762830339	3	2.00	1680	7000	1.5	0	0	4	7	...	1968	0	122072
14617	6762830618	2	1.00	1070	6120	1.0	0	0	3	6	...	1962	0	122056
14618	6762830709	4	1.00	1030	6621	1.0	0	0	4	6	...	1955	0	122042
14619	6762831463	3	1.00	900	4770	1.0	0	0	3	6	...	1969	2009	122018

14447 rows × 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

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

Out[26]: `<AxesSubplot:>`

```
In [27]:  z=np.abs(stats.zscore(df1['living area']))
```

```
In [28]:  len(np.where(z>3)[0])
```
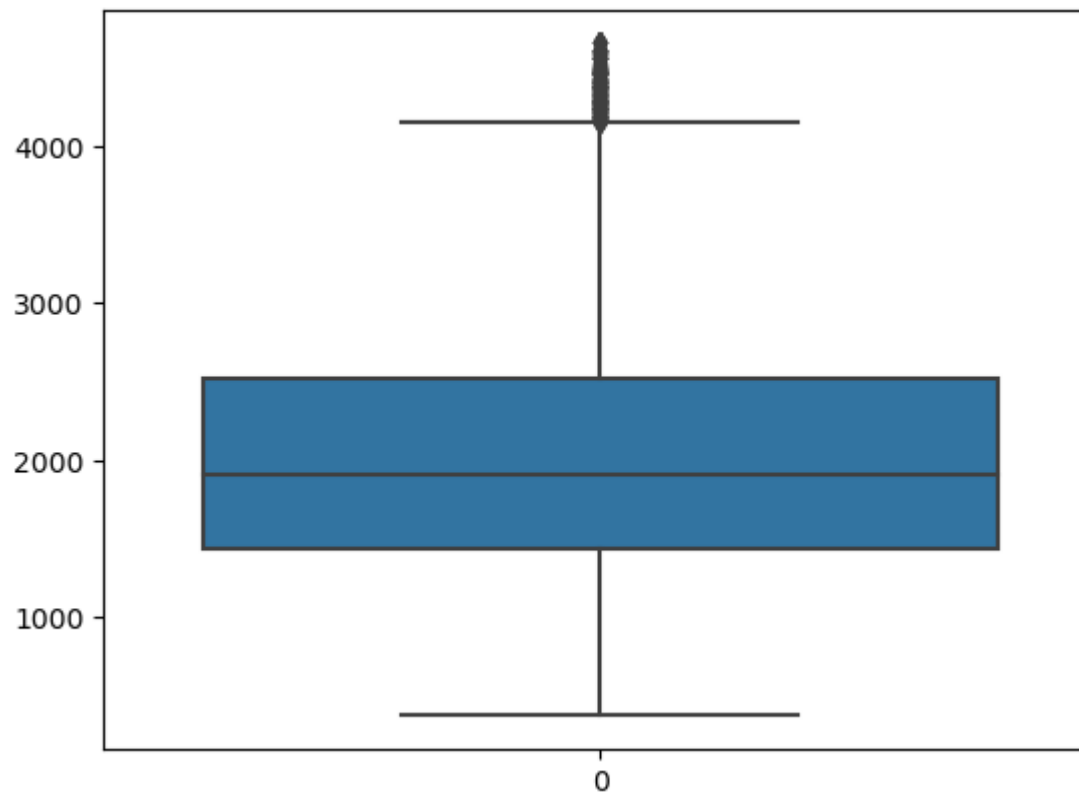
Out[28]: 136

```
In [29]:  len(np.where(z<-3)[0])
```

Out[29]: 0

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

```
In [31]:  sns.boxplot(df1['living area'])
```

Out[31]: <AxesSubplot:>


```
In [32]: z=np.abs(stats.zscore(df1['living area']))
```
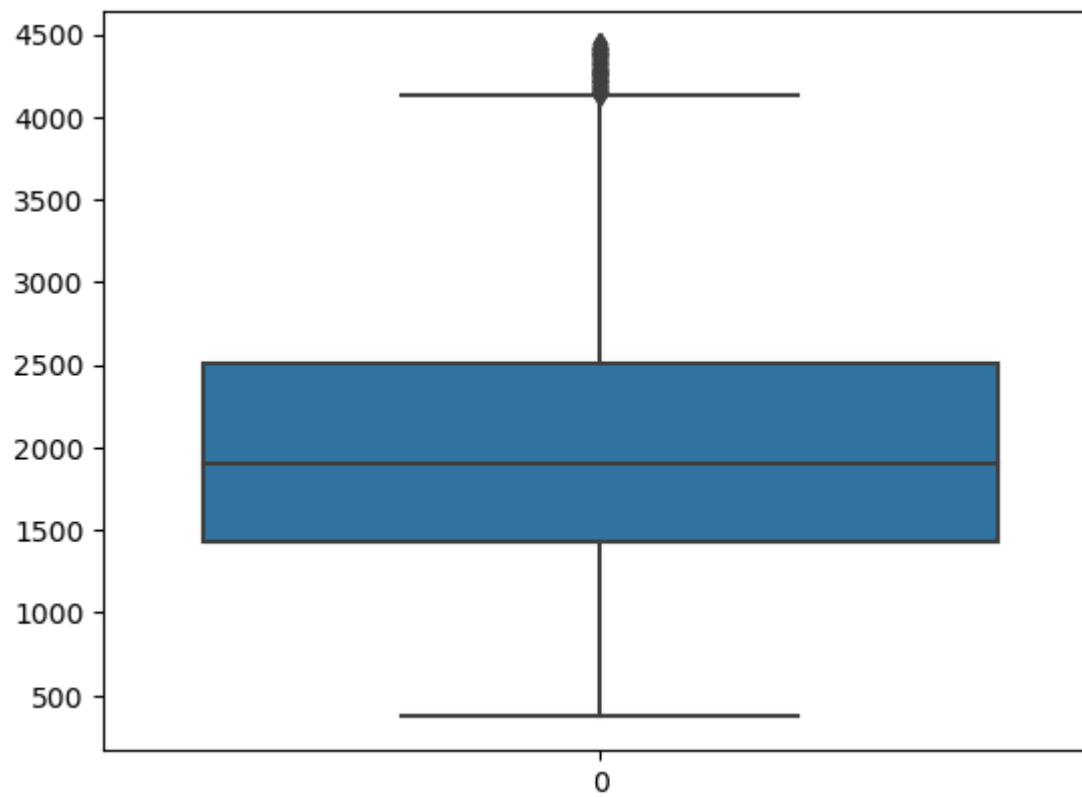
```
In [33]: len(np.where(z>3)[0])
```

Out[33]: 67

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

```
In [35]: sns.boxplot(df1['living area'])
```

Out[35]: <AxesSubplot:>

In [36]: df1

|	id	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	...	Built Year	Renovation Year	Postal Code
0	6762810145	5	2.50	3650	9050	2.0	0	4	5	10	...	1921	0	122003
1	6762810635	4	2.50	2920	4000	1.5	0	0	5	8	...	1909	0	122004
2	6762810998	5	2.75	2910	9480	1.5	0	0	3	8	...	1939	0	122004
3	6762812605	4	2.50	3310	42998	2.0	0	0	3	9	...	2001	0	122005
4	6762812919	3	2.00	2710	4500	1.5	0	0	4	8	...	1929	0	122006
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14615	6762830250	2	1.50	1556	20000	1.0	0	0	4	7	...	1957	0	122066
14616	6762830339	3	2.00	1680	7000	1.5	0	0	4	7	...	1968	0	122072
14617	6762830618	2	1.00	1070	6120	1.0	0	0	3	6	...	1962	0	122056
14618	6762830709	4	1.00	1030	6621	1.0	0	0	4	6	...	1955	0	122042
14619	6762831463	3	1.00	900	4770	1.0	0	0	3	6	...	1969	2009	122018

14244 rows × 22 columns

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]:
```python
sns.boxplot(df1['number of floors'])
```

Out[37]: <AxesSubplot:>
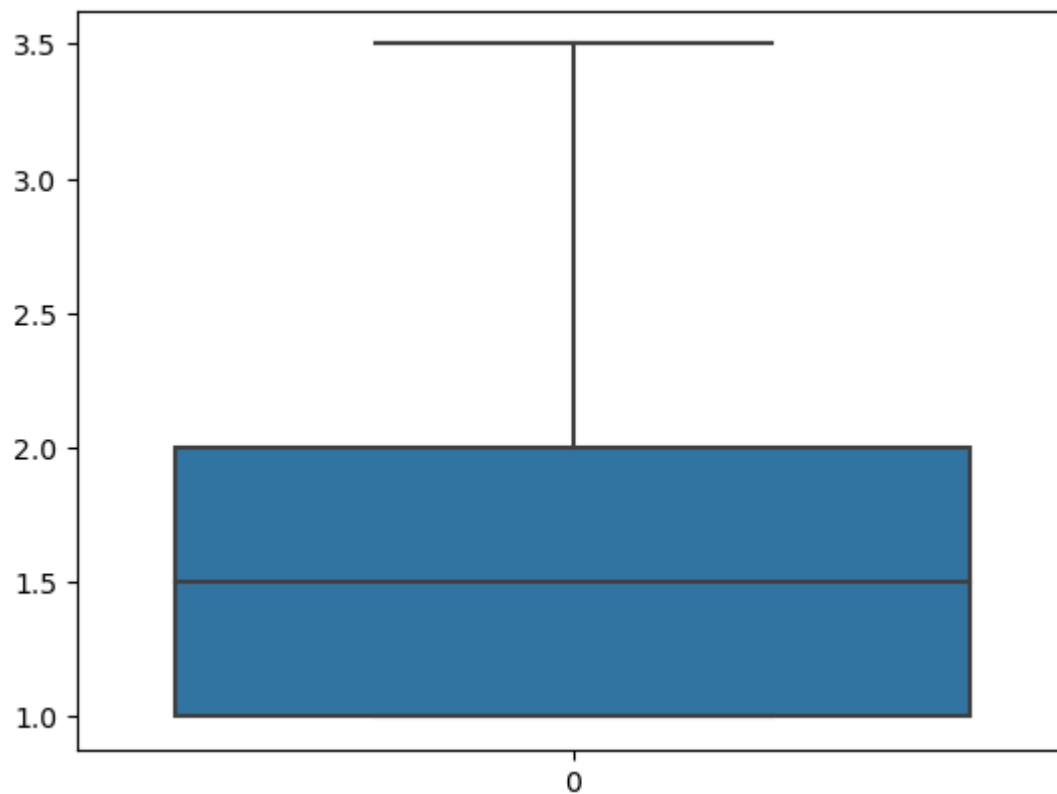
```
In [38]: z=np.abs(stats.zscore(df1['number of floors']))
```
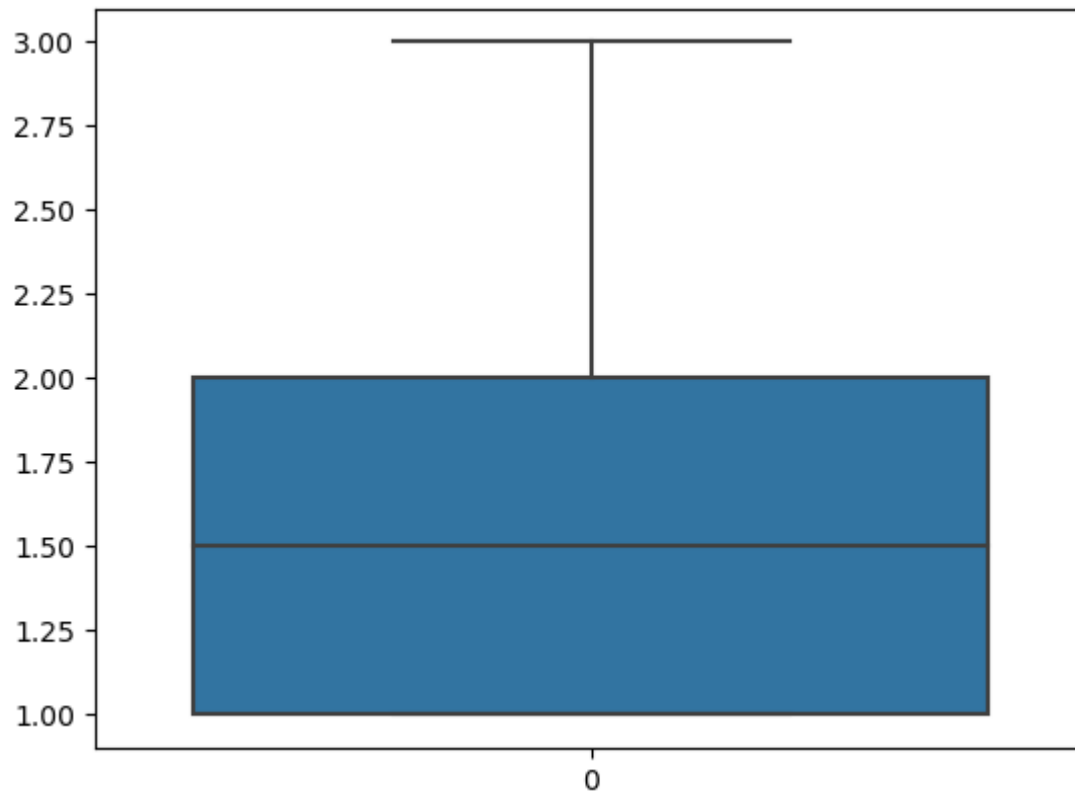
```
In [39]: len(np.where(z>3)[0])
```

Out[39]: 3

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

```
In [41]: sns.boxplot(df1['number of floors'])
```

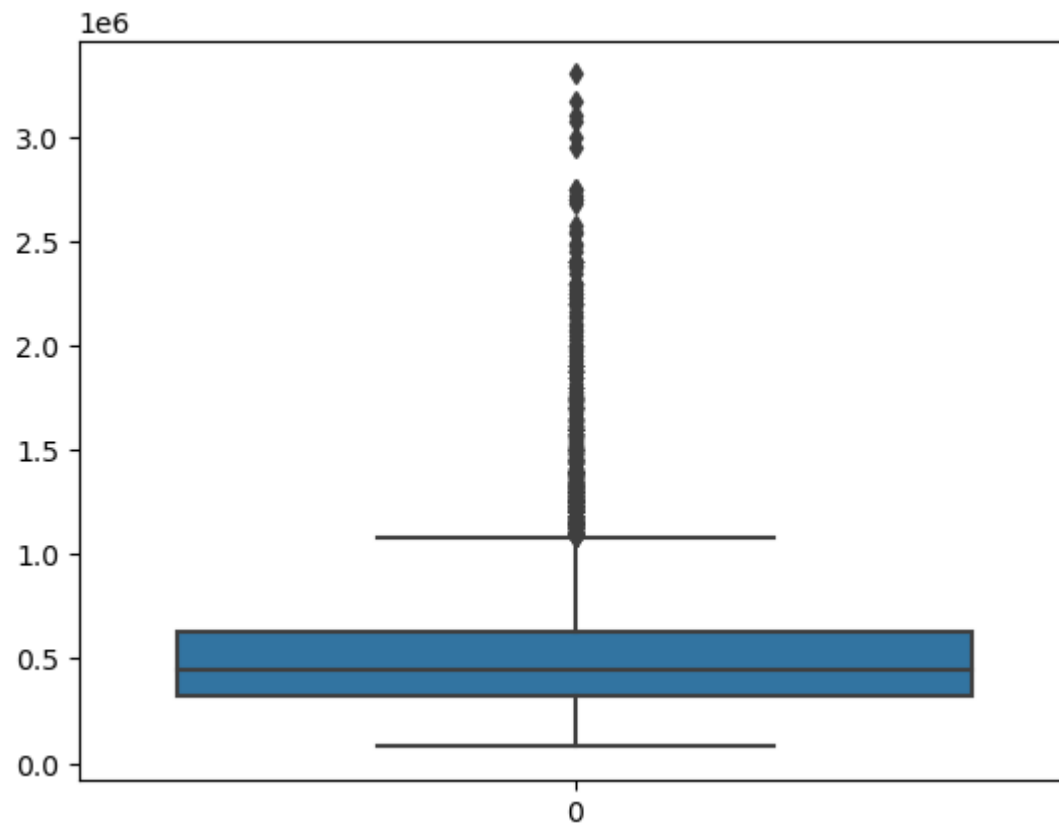Out[41]: <AxesSubplot:>

There are 3 outliers in number of floors

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

Out[42]: <AxesSubplot:>
```

In [43]: `z=np.abs(stats.zscore(df1['Price']))`

In [44]: `len(np.where(z>3)[0])`

Out[44]: 259

In [45]: `df1=df1[(z<3)]`

In [46]: `df1`

Out[46]:

	id	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	...	Built Year	Renovation Year	Postal Code
2	6762810998	5	2.75	2910	9480	1.5	0	0	3	8	...	1939	0	122004
3	6762812605	4	2.50	3310	42998	2.0	0	0	3	9	...	2001	0	122005
4	6762812919	3	2.00	2710	4500	1.5	0	0	4	8	...	1929	0	122006
5	6762813105	3	2.50	2600	4750	1.0	0	0	4	9	...	1951	0	122007
6	6762813157	5	3.25	3660	11995	2.0	0	2	3	10	...	2006	0	122008
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14615	6762830250	2	1.50	1556	20000	1.0	0	0	4	7	...	1957	0	122066
14616	6762830339	3	2.00	1680	7000	1.5	0	0	4	7	...	1968	0	122072
14617	6762830618	2	1.00	1070	6120	1.0	0	0	3	6	...	1962	0	122056
14618	6762830709	4	1.00	1030	6621	1.0	0	0	4	6	...	1955	0	122042
14619	6762831463	3	1.00	900	4770	1.0	0	0	3	6	...	1969	2009	122018

13982 rows × 22 columns

In [47]: `df1=df1.drop(['Renovation Year'],axis=1)`

In [48]: `df1`

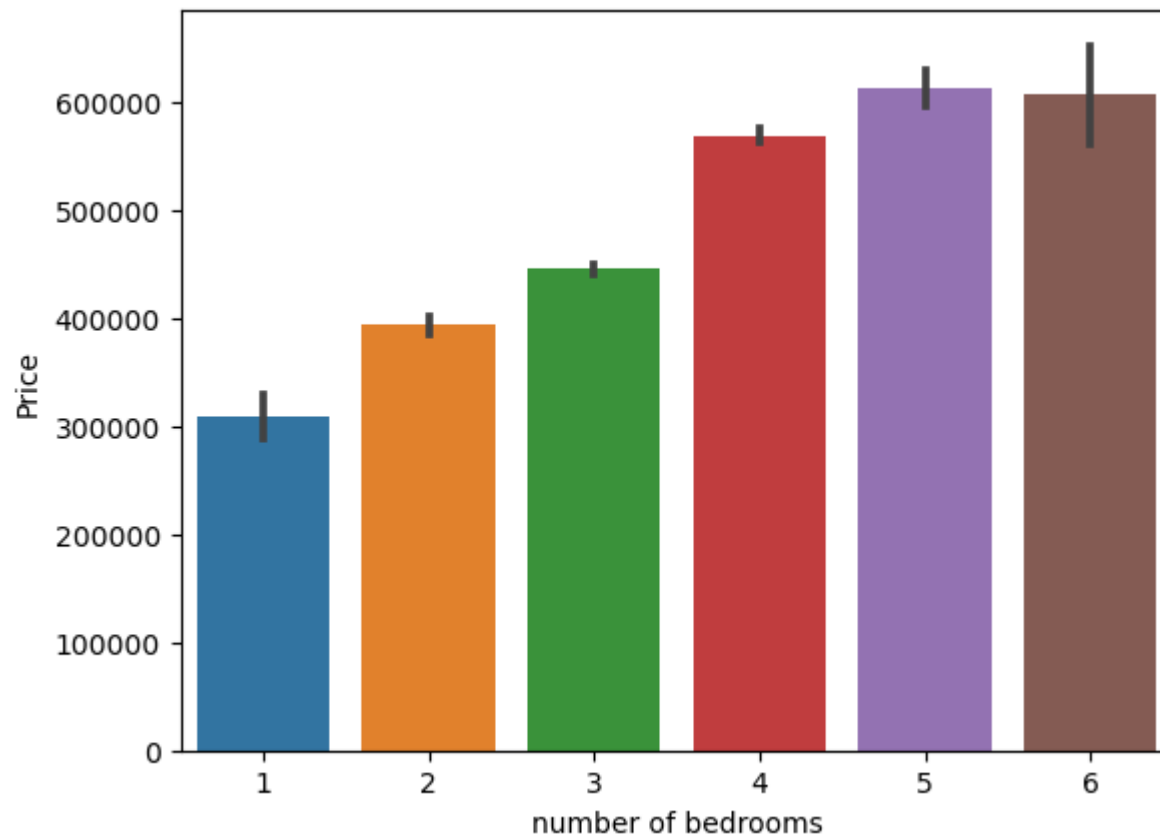|	id	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	grade of the house	...	Area of the basement	Built Year	Postal Code
2	6762810998	5	2.75	2910	9480	1.5	0	0	3	8	...	0	1939	122004
3	6762812605	4	2.50	3310	42998	2.0	0	0	3	9	...	0	2001	122005
4	6762812919	3	2.00	2710	4500	1.5	0	0	4	8	...	830	1929	122006
5	6762813105	3	2.50	2600	4750	1.0	0	0	4	9	...	900	1951	122007
6	6762813157	5	3.25	3660	11995	2.0	0	2	3	10	...	0	2006	122008
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14615	6762830250	2	1.50	1556	20000	1.0	0	0	4	7	...	0	1957	122066
14616	6762830339	3	2.00	1680	7000	1.5	0	0	4	7	...	0	1968	122072
14617	6762830618	2	1.00	1070	6120	1.0	0	0	3	6	...	0	1962	122056
14618	6762830709	4	1.00	1030	6621	1.0	0	0	4	6	...	0	1955	122042
14619	6762831463	3	1.00	900	4770	1.0	0	0	3	6	...	0	1969	122018

13982 rows × 21 columns

BI – VARIATE ANALYSIS

The column Renovation year have been removed. This is because most of the Renovation Year are 0 and proves to be of no use to the model

```
sns.barplot(data=df1,x='number of bedrooms',y='Price')
```
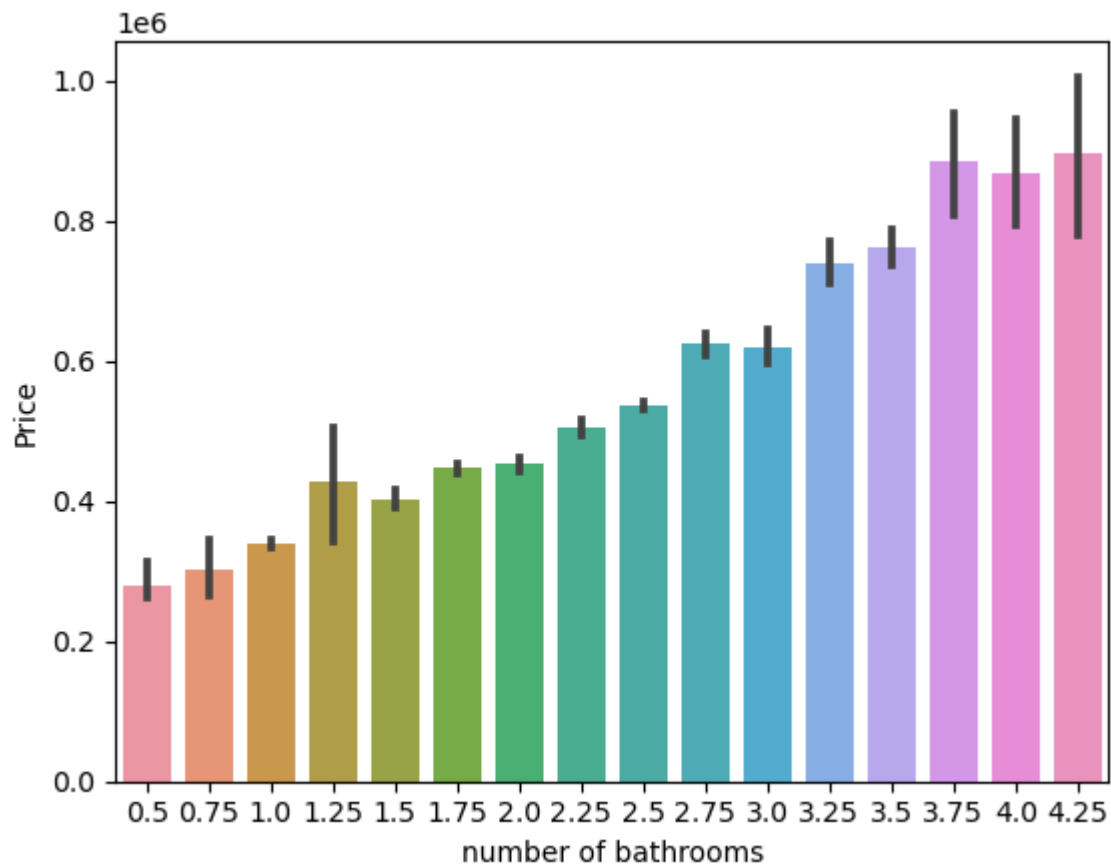
`<AxesSubplot:xlabel='number of bedrooms', ylabel='Price'>`

Clear indication of Price increasing with number of bedrooms

```
sns.barplot(data=df1,x='number of bathrooms',y='Price')
```
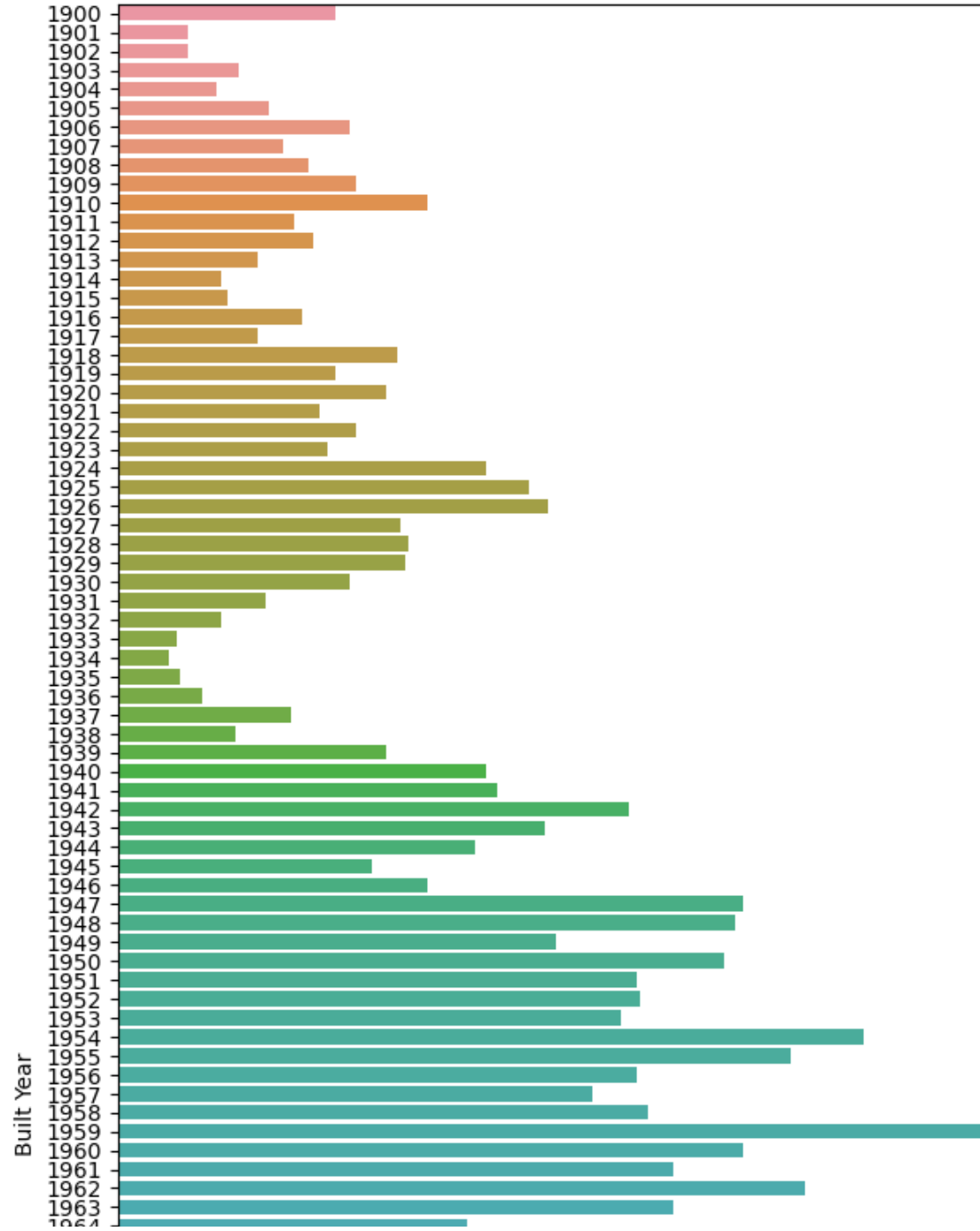
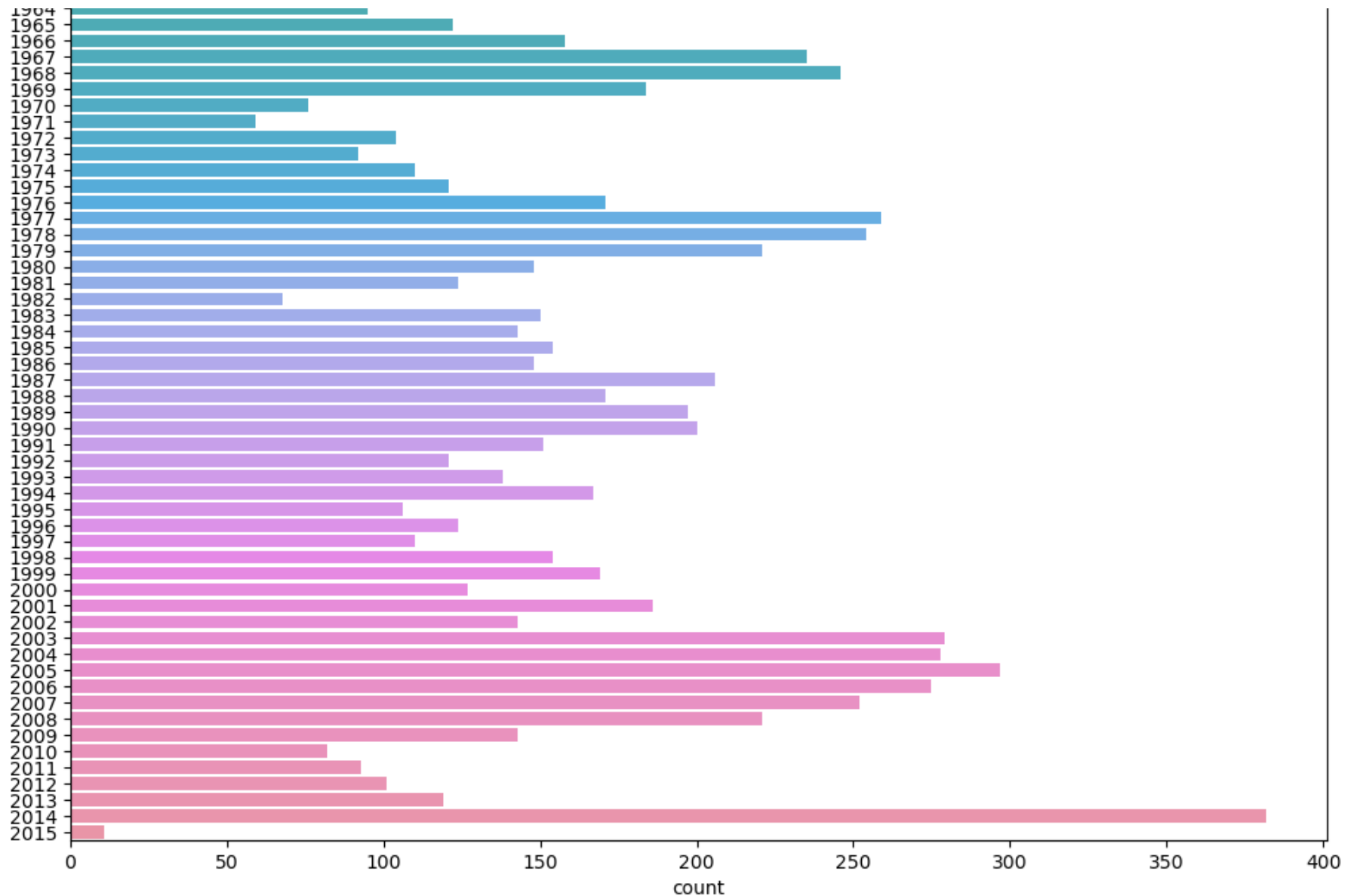`<AxesSubplot:xlabel='number of bathrooms', ylabel='Price'>`

Clear indication of Price increasing with number of bathrooms

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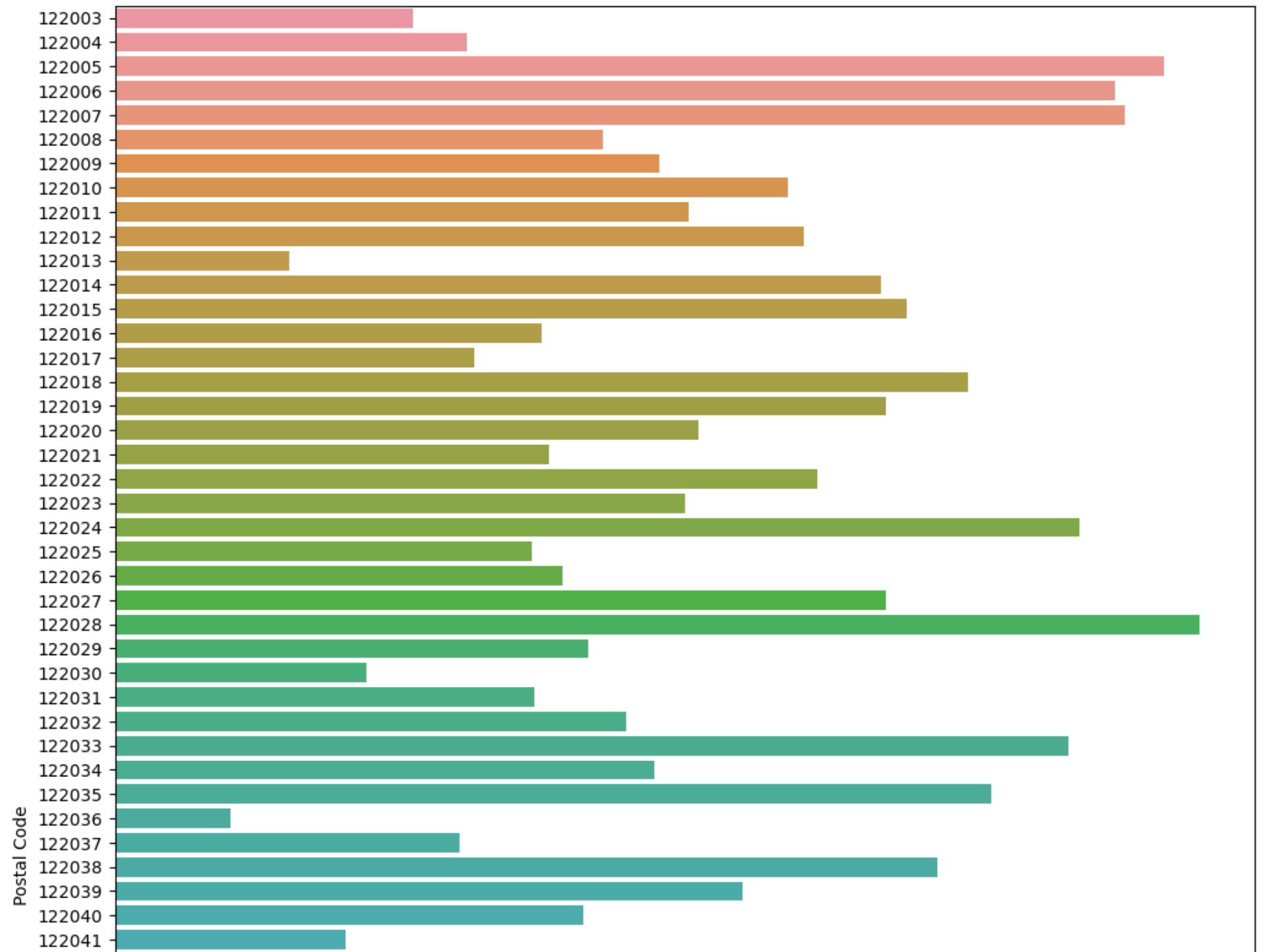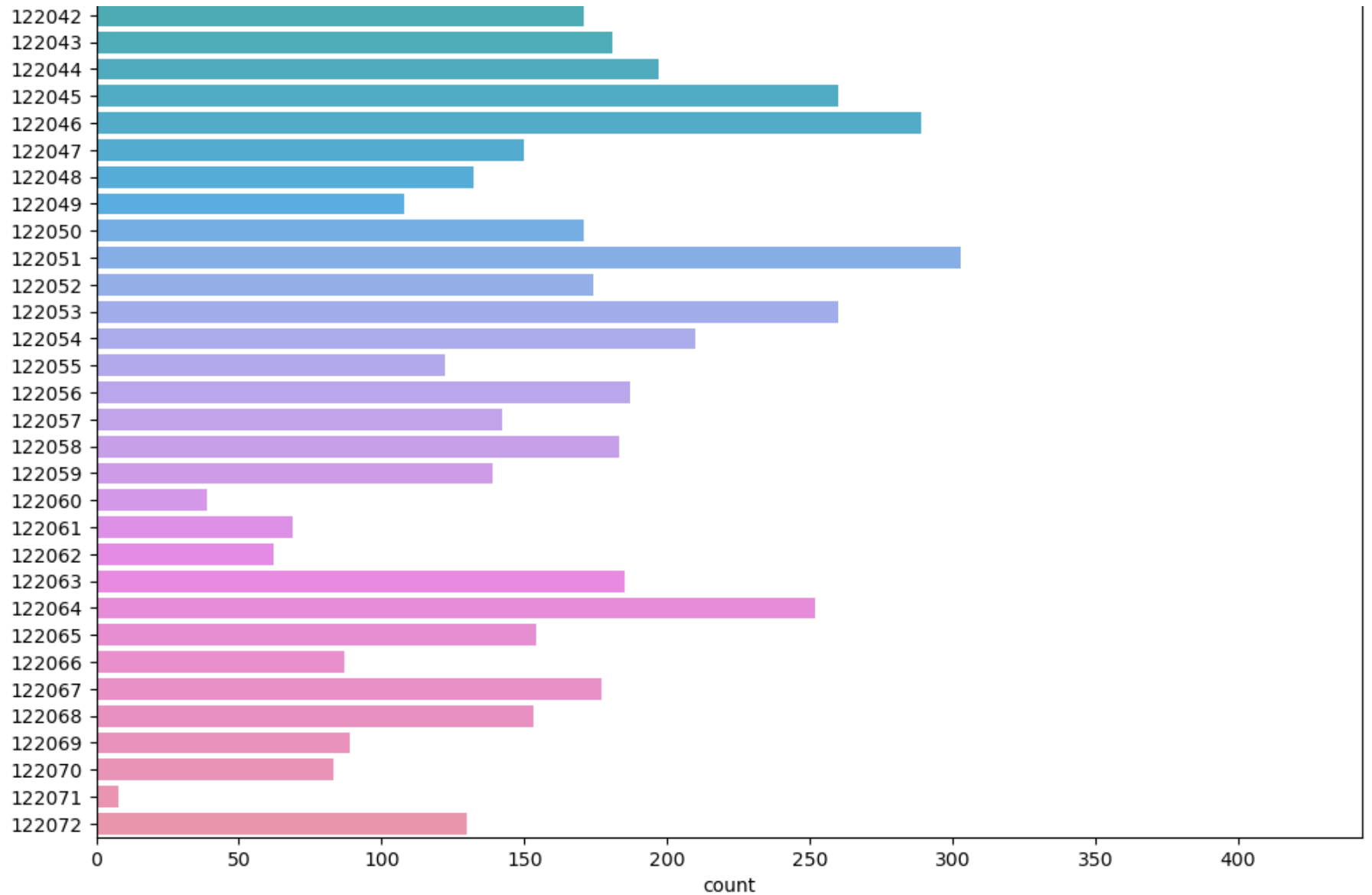
Most of the houses were listed for sale in 2017

```
In [52]: plt.figure(figsize=(12,18))
         sns.countplot(data=df1,y='Postal Code')
         plt.show()
```

Most of the houses listed for sale are from the Pincode 122028

```
In [53]:  df1[df1['Built Year']==2014]['Lattitude'].mean()

Out[53]:  52.77583376963351

In [54]:  df1[df1['Built Year']==2014]['Longitude'].mean()

Out[54]:  -114.38898952879582

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

          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.Icon(colo
          m
```
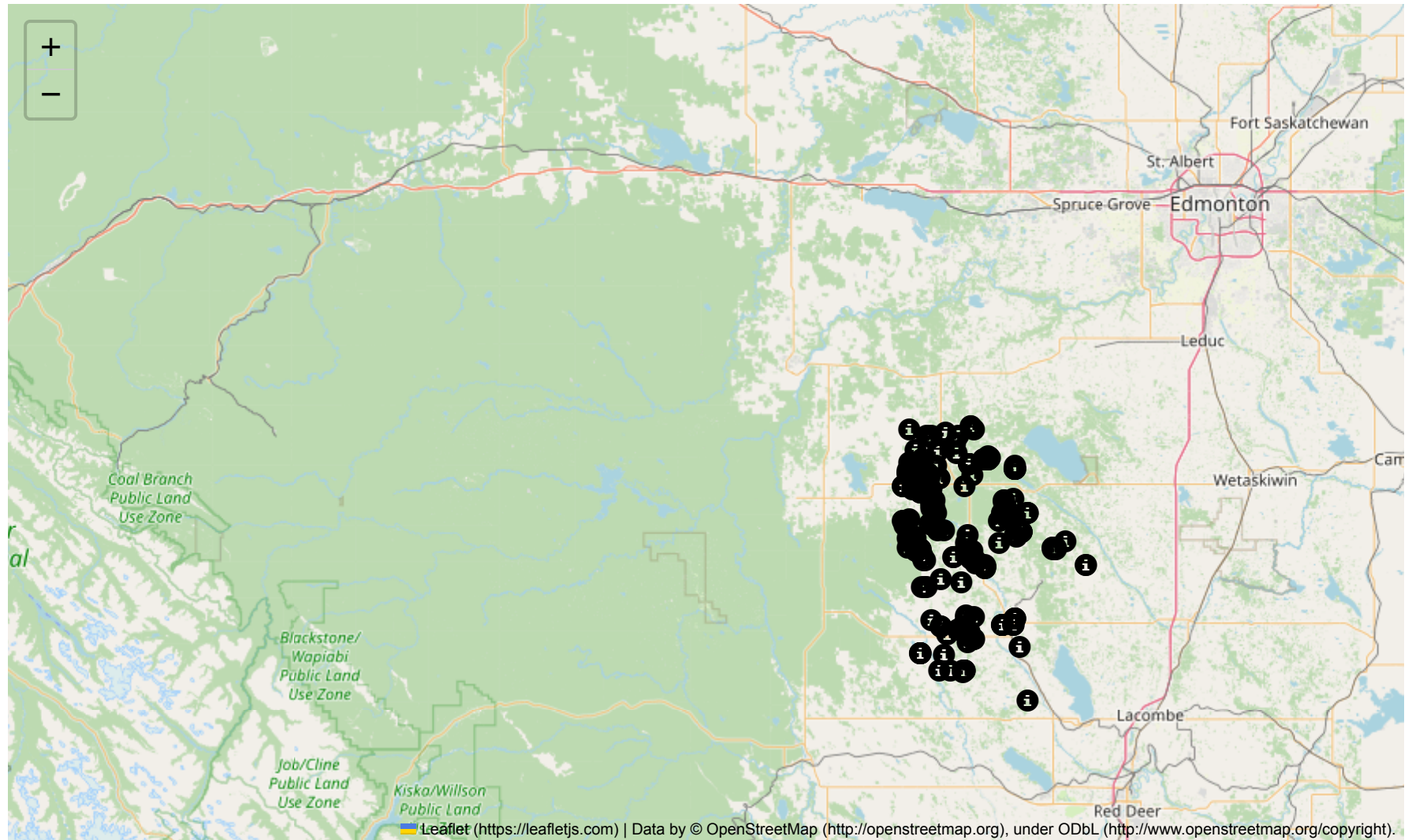
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

In [56]:
```python
df1[df1['Built Year']>=2014]['Lattitude'].mean()
```

Out[56]: 52.77850305343512

In [57]:
```python
df1[df1['Built Year']>=2014]['Longitude'].mean()
```
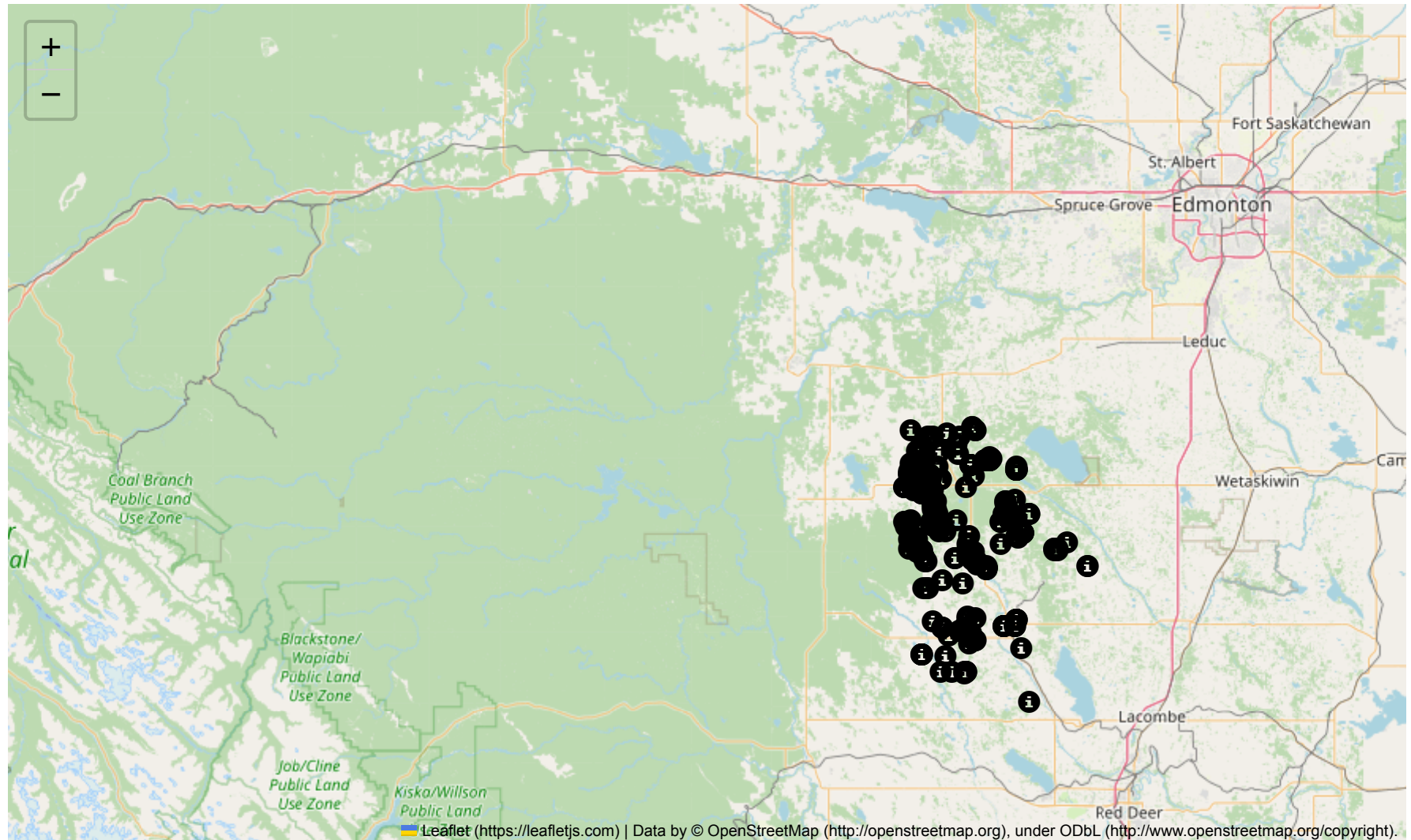
Out[57]: -114.39186768447837

```
In [58]: m = folium.Map(location = [52.77, -114.4], tiles ='OpenStreetMap',
             zoom_start=8)

         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.Icon(col
         m
```

Out[58]:

The houses listed for sale in this dataset are located in Alberta, Canada

```
In [59]:   df1=df1.drop(['id'],axis=1)
```
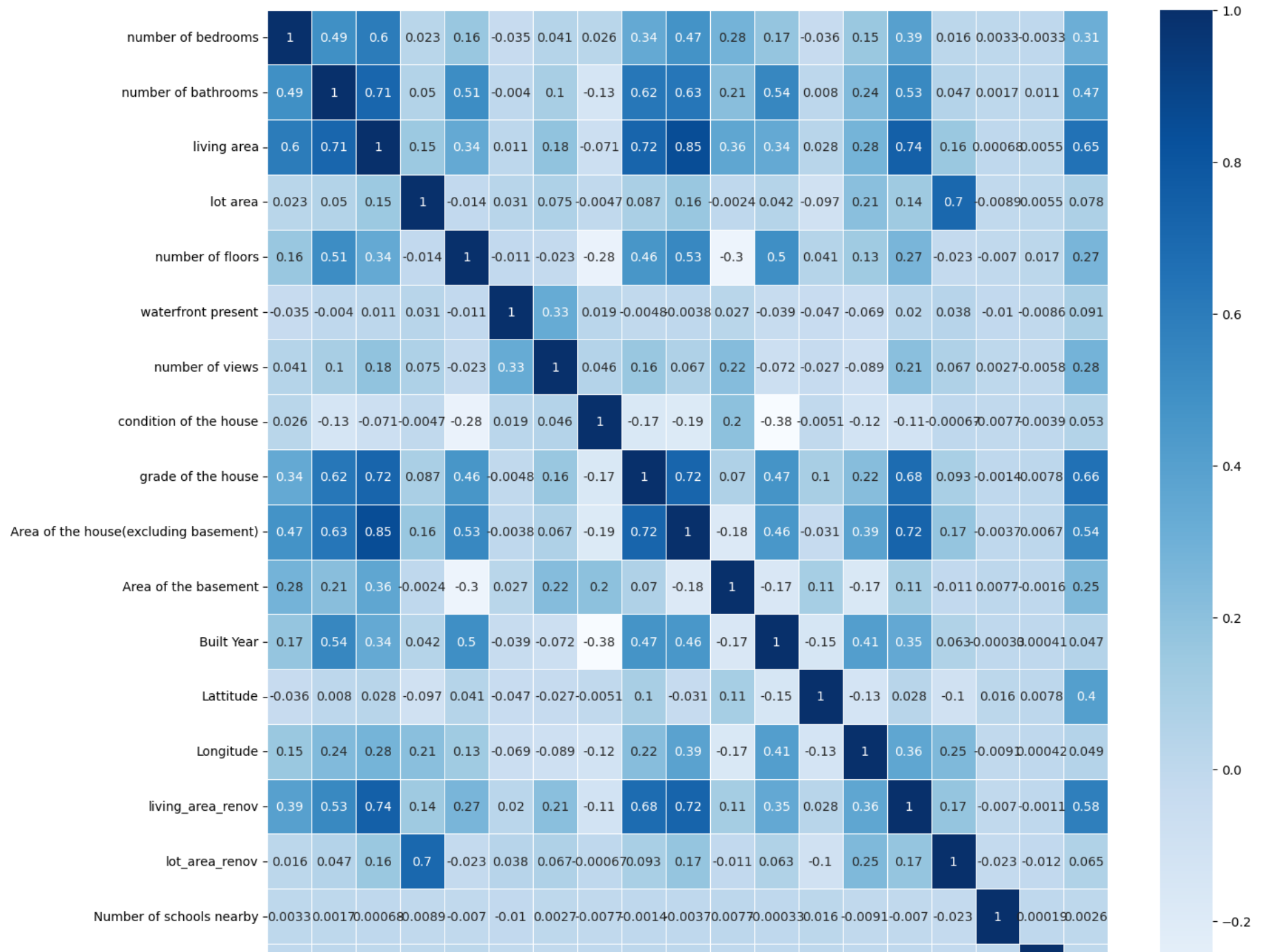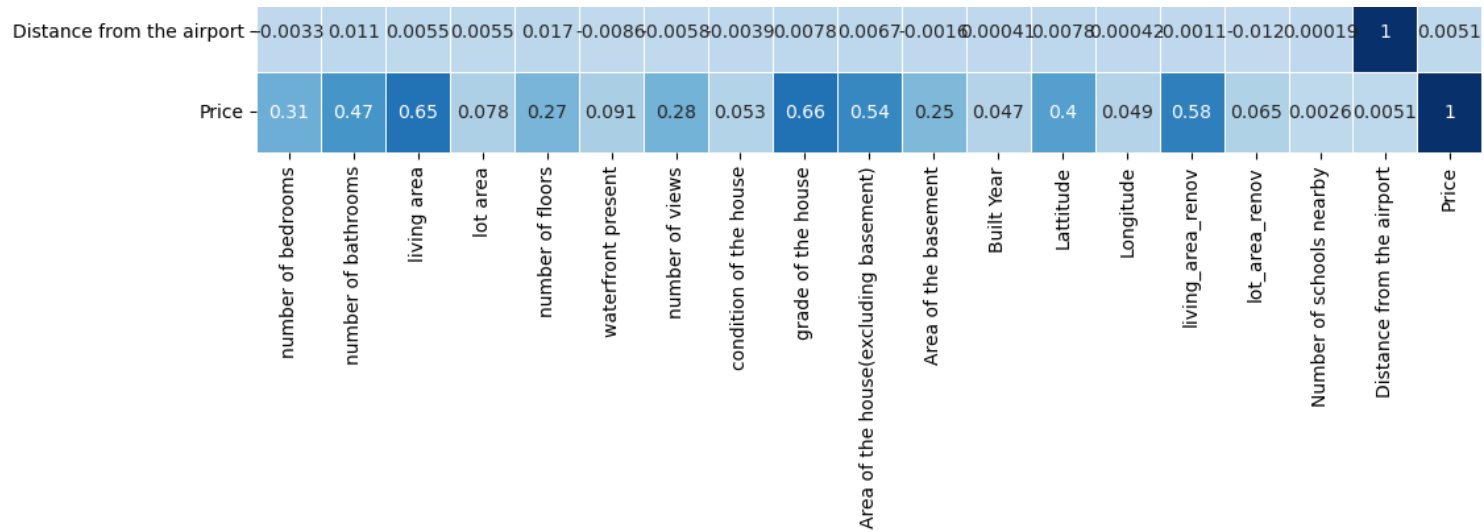
```
In [60]:   df1=df1.drop(['Postal Code'],axis=1)
```

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(df1.corr(),linewidths=0.5,annot=True,cmap='Blues')
           plt.show()
```

	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
Distance from the airport	-0.0033	0.011	0.0055	0.0055	0.017	-0.0086	0.0058	0.0039	0.0078	0.0067	-0.0016	0.00041	0.0078	0.00042	0.0011	-0.012	0.00019	1	0.0051
Price	0.31	0.47	0.65	0.078	0.27	0.091	0.28	0.053	0.66	0.54	0.25	0.047	0.4	0.049	0.58	0.065	0.0026	0.0051	1

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 from the
```

```
In [63]: plt.figure(figsize=(15,15))
         sns.heatmap(df1.corr(),linewidths=0.5,annot=True,cmap='Blues')
         plt.show()
```
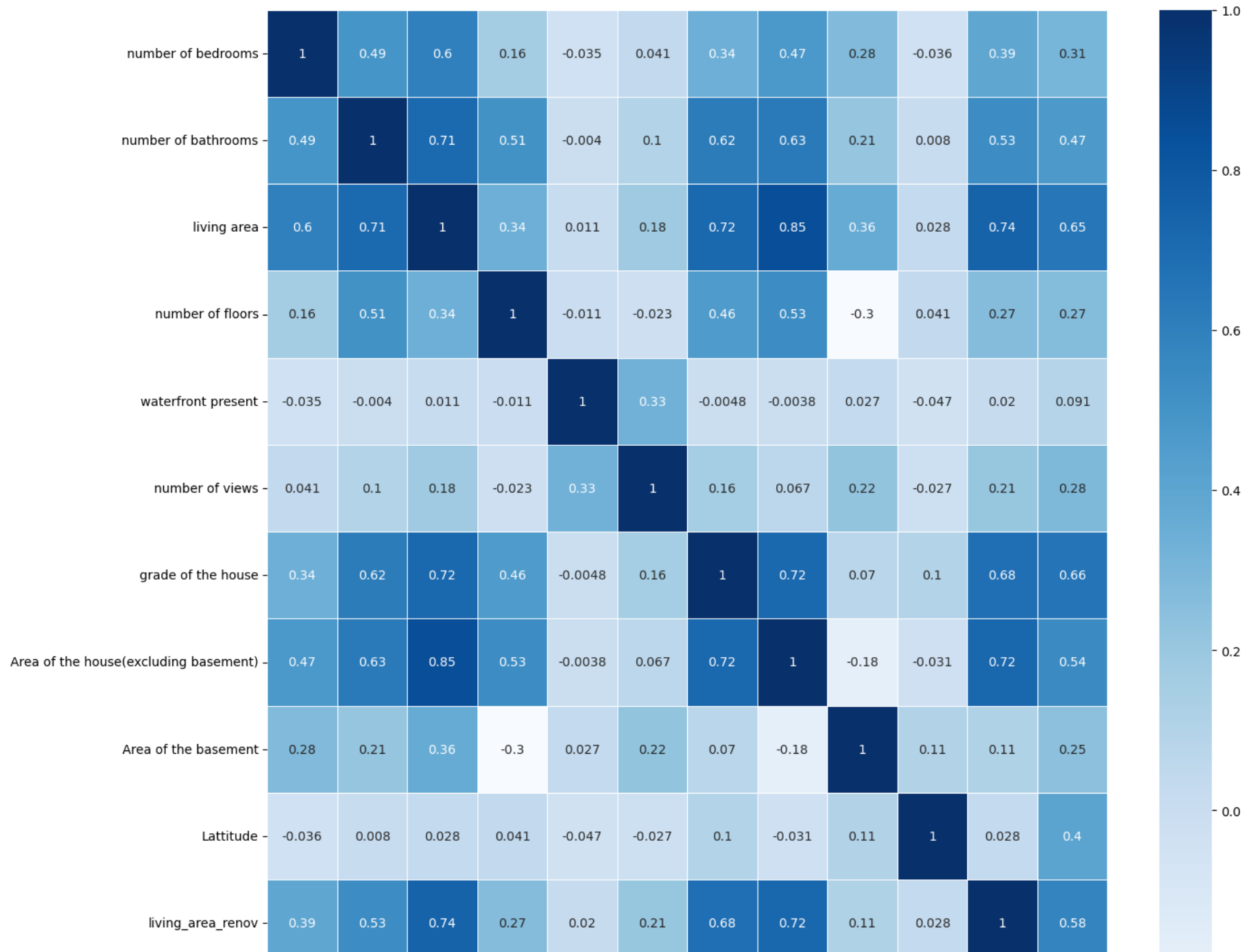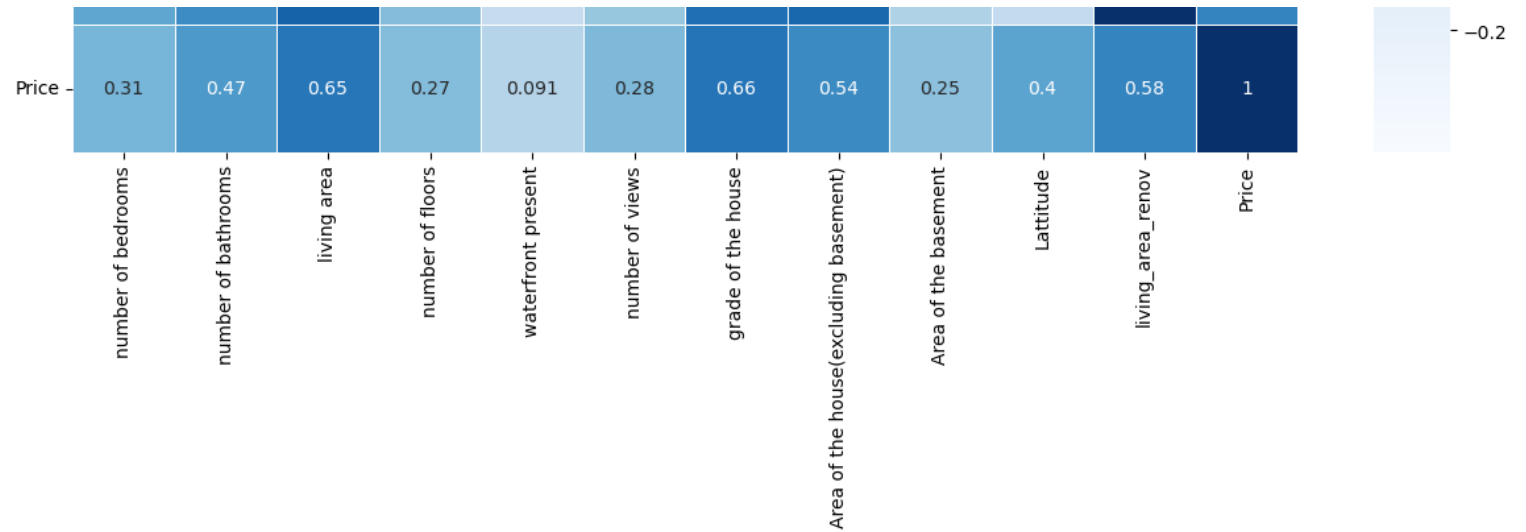
Training of Model, Splitting of Dataset into Train and Test Set

```
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,y_train,y_test= train_test_split(X,y,test_size=0.2,random_state=11)
```

```
In [70]: X_train.shape
```

```
Out[70]:  (11185, 11)

In [71]:  X_test.shape

Out[71]:  (2797, 11)

In [72]:  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
          from xgboost import XGBRegressor
          from sklearn.tree import DecisionTreeRegressor
          from sklearn.ensemble import StackingRegressor
          from sklearn.svm import SVR

In [73]:  pipelines = {
              'en':make_pipeline(StandardScaler(), ElasticNet()),
              'lasso':make_pipeline(StandardScaler(), Lasso()),
              'Rcv':make_pipeline(StandardScaler(), RidgeCV()),
              'CatB':make_pipeline(StandardScaler(), CatBoostRegressor(eval_metric='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]:  fit_models = {}
          for algo, pipeline in pipelines.items():
              model = pipeline.fit(X_train, y_train)
              fit_models[algo] = model
```

/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWarning: Objective did not c
onverge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisa
tion. Duality gap: 4.781e+12, tolerance: 5.929e+10
  coef_, l1_reg, l2_reg, X, y, max_iter, tol, rng, random, positive

```
Learning rate set to 0.05996
0:      learn: 221490.1496581    total: 61.4ms    remaining: 1m 1s
999:    learn: 77595.2298921     total: 2.85s     remaining: 0us
```

In [75]:
```python
from sklearn.metrics import mean_absolute_error, mean_squared_error
maes=[]
al=[]
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
lr MEAN ABSOLUTE ERROR 97574.48622571728
lr 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]:
```python
plt.figure(figsize=(5,5))
plt.xlabel('ML Algorithms...')
plt.ylabel('Root Mean Squared Errors...')
ax=sns.barplot(x=al,y=maes)
plt.show()
```
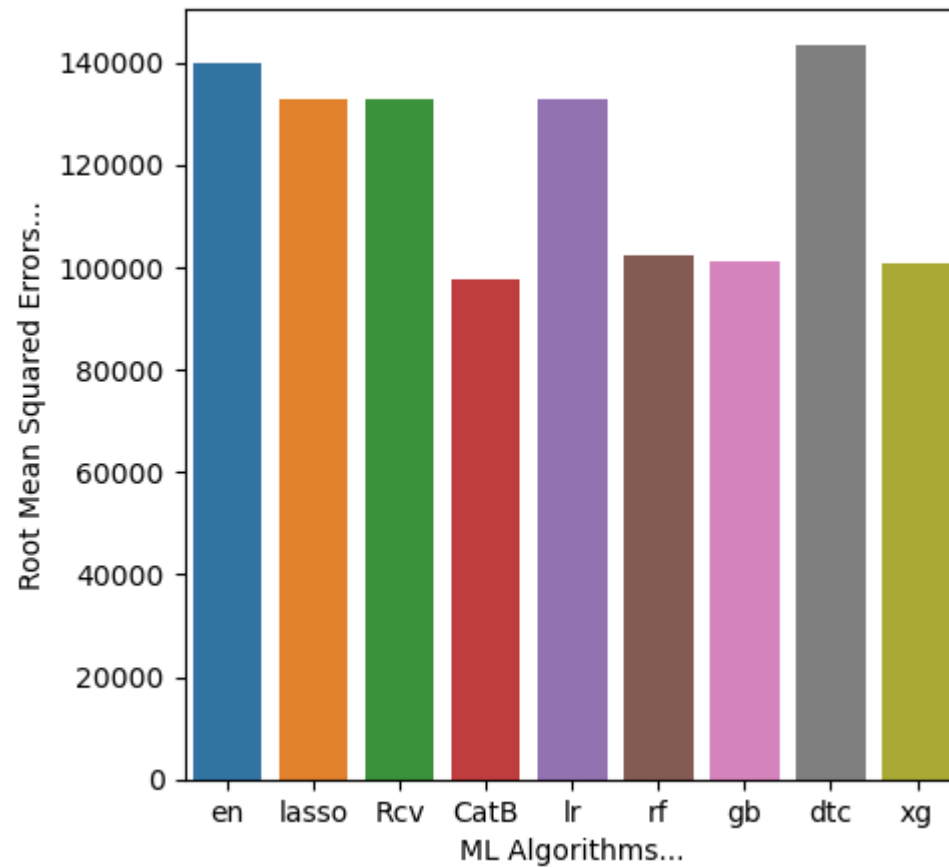
```
In [ ]:  CatB = CatBoostRegressor(verbose=1000,eval_metric='RMSE')
         rf = RandomForestRegressor()
         gb = GradientBoostingRegressor()
         xg = XGBRegressor()
         lr=LinearRegression()

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

         pipeline = make_pipeline(
             StandardScaler(),
             stregr
         )
```

```
pipeline.fit(X_train, y_train)

# 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
0:      learn: 221490.1496581   total: 4.18ms   remaining: 4.18s
999:    learn: 77595.2298921    total: 2.81s    remaining: 0us
Learning rate set to 0.057883
0:      learn: 222091.4863333   total: 3.52ms   remaining: 3.51s
999:    learn: 76337.1933964    total: 2.52s    remaining: 0us
Learning rate set to 0.057883
0:      learn: 222546.8538661   total: 2.94ms   remaining: 2.94s
999:    learn: 75466.5961681    total: 2.51s    remaining: 0us
Learning rate set to 0.057883
0:      learn: 223455.5230951   total: 3.2ms    remaining: 3.2s
999:    learn: 75656.3661258    total: 2.52s    remaining: 0us
Learning rate set to 0.057883
0:      learn: 221606.9467960   total: 3.71ms   remaining: 3.7s
999:    learn: 75195.9699196    total: 2.46s    remaining: 0us
Learning rate set to 0.057883
0:      learn: 219316.0911020   total: 2.47ms   remaining: 2.47s
```

In [ ]:
```
mean_squared_error(y_test,y_pred)**0.5
```

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

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

In [ ]:
```
plt.figure(figsize=(9,5))
plt.xlabel('ML Algorithms...')
plt.ylabel('Root Mean Squared Errors...')
ax=sns.barplot(x=al,y=maes)
plt.show()
```

ALL DONE BY SRIRAM AS NAAN MUDALVAN IBM SMARTINTERNZ ASSIGNMENT 3