data_science_-_python

May 19, 2020

Project Title: "Melbourne house price data" Name: "Vikash Pateshwari" Email: "vikash.pateshwari@wipro.com" Company : "Wipro Technology" output: html_document , .ipynb , .py

Business Understanding

Melbourne house price data

Data Understanding

You can download the data from https://www.kaggle.com/anthonypino/melbourne-housing-market#Melbourne_housing_FULL.csv You can understand the data by looking at the data dictionary provided @ https://rpubs.com/kunaljubce/mlb_housing_data

Import the data set in Python.

View the dataset

See the structure and the summary of the dataset to understand the data.

Find out the number of:

```
 <ul>
 <li>Numeric attributes:</li>
 <li>Categorical attributes:</li>
 </ul>
</h4>
```

[1]:
```
#1.Import the data set in Python.
#downloaded the data sets
import numpy as np
import pandas as pd
full = pd.read_csv(r'C:\Users\imvik\Wipro␣
 ↪Assignment\AI\data_science_-_python\melbourne-housing-market\Melbourne_housing_FULL.
 ↪csv')
```

[2]:
```
#2.View the dataset
#head for top 5 rows or tail for 5 last items
#full.tail()
full.head()
```

[2]:
```
        Suburb             Address  Rooms Type      Price Method SellerG  \
0  Abbotsford       68 Studley St      2    h        NaN     SS  Jellis
1  Abbotsford        85 Turner St      2    h  1480000.0      S  Biggin
2  Abbotsford     25 Bloomburg St      2    h  1035000.0      S  Biggin
3  Abbotsford  18/659 Victoria St      3    u        NaN     VB  Rounds
4  Abbotsford        5 Charles St      3    h  1465000.0     SP  Biggin

        Date  Distance  Postcode      …       Bathroom  Car  Landsize  \
0  3/09/2016       2.5    3067.0      …            1.0  1.0     126.0
1  3/12/2016       2.5    3067.0      …            1.0  1.0     202.0
2  4/02/2016       2.5    3067.0      …            1.0  0.0     156.0
3  4/02/2016       2.5    3067.0      …            2.0  1.0       0.0
4  4/03/2017       2.5    3067.0      …            2.0  0.0     134.0

   BuildingArea  YearBuilt         CouncilArea Lattitude  Longtitude  \
0           NaN        NaN  Yarra City Council  -37.8014    144.9958
1           NaN        NaN  Yarra City Council  -37.7996    144.9984
2          79.0     1900.0  Yarra City Council  -37.8079    144.9934
3           NaN        NaN  Yarra City Council  -37.8114    145.0116
4         150.0     1900.0  Yarra City Council  -37.8093    144.9944

              Regionname Propertycount
0  Northern Metropolitan        4019.0
1  Northern Metropolitan        4019.0
2  Northern Metropolitan        4019.0
3  Northern Metropolitan        4019.0
4  Northern Metropolitan        4019.0

[5 rows x 21 columns]
```

[3]:
```python
#See the structure and the summary of the dataset to understand the data.
full.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34857 entries, 0 to 34856
Data columns (total 21 columns):
Suburb           34857 non-null object
Address          34857 non-null object
Rooms            34857 non-null int64
Type             34857 non-null object
Price            27247 non-null float64
Method           34857 non-null object
SellerG          34857 non-null object
Date             34857 non-null object
Distance         34856 non-null float64
Postcode         34856 non-null float64
Bedroom2         26640 non-null float64
```

```
Bathroom         26631 non-null float64
Car              26129 non-null float64
Landsize         23047 non-null float64
BuildingArea     13742 non-null float64
YearBuilt        15551 non-null float64
CouncilArea      34854 non-null object
Lattitude        26881 non-null float64
Longtitude       26881 non-null float64
Regionname       34854 non-null object
Propertycount    34854 non-null float64
dtypes: float64(12), int64(1), object(8)
memory usage: 5.6+ MB
```

[4]:
```
full.shape
```

[4]:
```
(34857, 21)
```

[5]:
```
full.columns
```

[5]:
```
Index(['Suburb', 'Address', 'Rooms', 'Type', 'Price', 'Method', 'SellerG',
       'Date', 'Distance', 'Postcode', 'Bedroom2', 'Bathroom', 'Car',
       'Landsize', 'BuildingArea', 'YearBuilt', 'CouncilArea', 'Lattitude',
       'Longtitude', 'Regionname', 'Propertycount'],
      dtype='object')
```

[6]:
```
full.describe()
```

[6]:
```
              Rooms         Price      Distance      Postcode      Bedroom2  \
count  34857.000000  2.724700e+04  34856.000000  34856.000000  26640.000000
mean       3.031012  1.050173e+06     11.184929   3116.062859      3.084647
std        0.969933  6.414671e+05      6.788892    109.023903      0.980690
min        1.000000  8.500000e+04      0.000000   3000.000000      0.000000
25%        2.000000  6.350000e+05      6.400000   3051.000000      2.000000
50%        3.000000  8.700000e+05     10.300000   3103.000000      3.000000
75%        4.000000  1.295000e+06     14.000000   3156.000000      4.000000
max       16.000000  1.120000e+07     48.100000   3978.000000     30.000000

           Bathroom           Car       Landsize  BuildingArea     YearBuilt  \
count  26631.000000  26129.000000   23047.000000   13742.00000  15551.000000
mean       1.624798      1.728845     593.598993     160.25640   1965.289885
std        0.724212      1.010771    3398.841946     401.26706     37.328178
min        0.000000      0.000000       0.000000       0.00000   1196.000000
25%        1.000000      1.000000     224.000000     102.00000   1940.000000
50%        2.000000      2.000000     521.000000     136.00000   1970.000000
75%        2.000000      2.000000     670.000000     188.00000   2000.000000
max       12.000000     26.000000  433014.000000   44515.00000   2106.000000
```

```
          Lattitude    Longtitude  Propertycount
count  26881.000000  26881.000000   34854.000000
mean     -37.810634    145.001851    7572.888306
std        0.090279      0.120169    4428.090313
min      -38.190430    144.423790      83.000000
25%      -37.862950    144.933500    4385.000000
50%      -37.807600    145.007800    6763.000000
75%      -37.754100    145.071900   10412.000000
max      -37.390200    145.526350   21650.000000
```

[7]:
```
#Find out the number of:
#    Numeric attributes:
numeric = full.select_dtypes(exclude='object')
len(numeric.columns)
```

[7]: 13

[8]:
```
#    Categorical attributes:
categorical= full.select_dtypes(include='object')
len(categorical.columns)
```

[8]: 8

Data Preparation : Data Cleaning

Duplicate values: Identify if the datasets have duplicate values or not and remove the duplicate values.

```
<li>Find out the number of rows present in the dataset</li>
```

Before removing duplicate values

After removing duplicate values

Variable type: Check if all the variables have the correct variable type, based on the data dictionary. If not, then change them.

```
<li>For how many attributes did you need to change the data type?</li>
```

Missing value treatment: Check which variables have missing values and use appropriate treatments.

```
<li>    For each of the variables, find the number of missing values and provide the value tha
```

Outlier Treatment:

```
<li>Identify the varibales : Make a subset of the dataset with all the numeric variables. </li
<li>Outliers : For each variable of this subset, carry out the outlier detection. Find out the
```

[9]:
```
#Duplicate values: Identify if the datasets have duplicate values or not and␣
 ↪remove the duplicate values.
full[full.duplicated()]
```

[9]:
```
            Suburb        Address  Rooms Type  Price Method SellerG  \
15858  Nunawading  1/7 Lilian St      3    t    NaN     SP  Jellis

             Date  Distance  Postcode     …      Bathroom  Car  Landsize  \
15858  17/06/2017      15.4    3131.0     …           3.0  2.0     405.0

      BuildingArea  YearBuilt             CouncilArea Lattitude  Longtitude  \
15858        226.0     2000.0  Manningham City Council -37.82678   145.16777

                Regionname Propertycount
15858  Eastern Metropolitan        4973.0

[1 rows x 21 columns]
```

[10]:
```python
#Find out the number of rows present in the dataset
#Before removing duplicate values
duplicate = full.duplicated()
duplicate.count()
```

[10]:
```
34857
```

[11]:
```python
#After removing duplicate values
remdupl = full.drop_duplicates()
len(remdupl)
```

[11]:
```
34856
```

[12]:
```python
#Variable type: Check if all the variables have the correct variable type,
#               based on the data dictionary. If not, then change them.
full.dtypes
```

[12]:
```
Suburb             object
Address            object
Rooms               int64
Type               object
Price             float64
Method             object
SellerG            object
Date               object
Distance          float64
Postcode          float64
Bedroom2          float64
Bathroom          float64
Car               float64
Landsize          float64
BuildingArea      float64
YearBuilt         float64
```

```
CouncilArea       object
Lattitude        float64
Longtitude       float64
Regionname        object
Propertycount    float64
dtype: object
```

[13]:
```
#as object is not  datatype so it need to converted in cateogical data type.
objectdtye = full.select_dtypes({object}).columns
objectdtye
```

[13]:
```
Index(['Suburb', 'Address', 'Type', 'Method', 'SellerG', 'Date', 'CouncilArea',
       'Regionname'],
      dtype='object')
```

[14]:
```
full[objectdtye] = full[objectdtye].astype('category')
```

[15]:
```
#For how many attributes did you need to change the data type?
full.info()
#from below info we can check there are 8 categorical data.
#dtypes: category(8), float64(12), int64(1)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34857 entries, 0 to 34856
Data columns (total 21 columns):
Suburb           34857 non-null category
Address          34857 non-null category
Rooms            34857 non-null int64
Type             34857 non-null category
Price            27247 non-null float64
Method           34857 non-null category
SellerG          34857 non-null category
Date             34857 non-null category
Distance         34856 non-null float64
Postcode         34856 non-null float64
Bedroom2         26640 non-null float64
Bathroom         26631 non-null float64
Car              26129 non-null float64
Landsize         23047 non-null float64
BuildingArea     13742 non-null float64
YearBuilt        15551 non-null float64
CouncilArea      34854 non-null category
Lattitude        26881 non-null float64
Longtitude       26881 non-null float64
Regionname       34854 non-null category
Propertycount    34854 non-null float64
dtypes: category(8), float64(12), int64(1)
```

```
memory usage: 5.4 MB
```

[16]:
```
#Missing value treatment: Check which variables have missing values and use␣
↪appropriate treatments.
#For each of the variables, find the number of missing values and provide the␣
↪value that they have been imputed with.
full.isnull().sum()
```

[16]:
```
Suburb               0
Address              0
Rooms                0
Type                 0
Price             7610
Method               0
SellerG              0
Date                 0
Distance             1
Postcode             1
Bedroom2          8217
Bathroom          8226
Car               8728
Landsize         11810
BuildingArea     21115
YearBuilt        19306
CouncilArea          3
Lattitude         7976
Longtitude        7976
Regionname           3
Propertycount        3
dtype: int64
```

[17]:
```
#droping missing value columns
full1 = full
len(full1), len(full1.dropna())
```

[17]:
```
(34857, 8887)
```

[18]:
```
#method 2 imputing the values
full1.isna().sum()
```

[18]:
```
Suburb               0
Address              0
Rooms                0
Type                 0
Price             7610
Method               0
SellerG              0
```

```
Date                 0
Distance             1
Postcode             1
Bedroom2          8217
Bathroom          8226
Car               8728
Landsize         11810
BuildingArea     21115
YearBuilt        19306
CouncilArea          3
Lattitude         7976
Longtitude        7976
Regionname           3
Propertycount        3
dtype: int64
```

[19]:

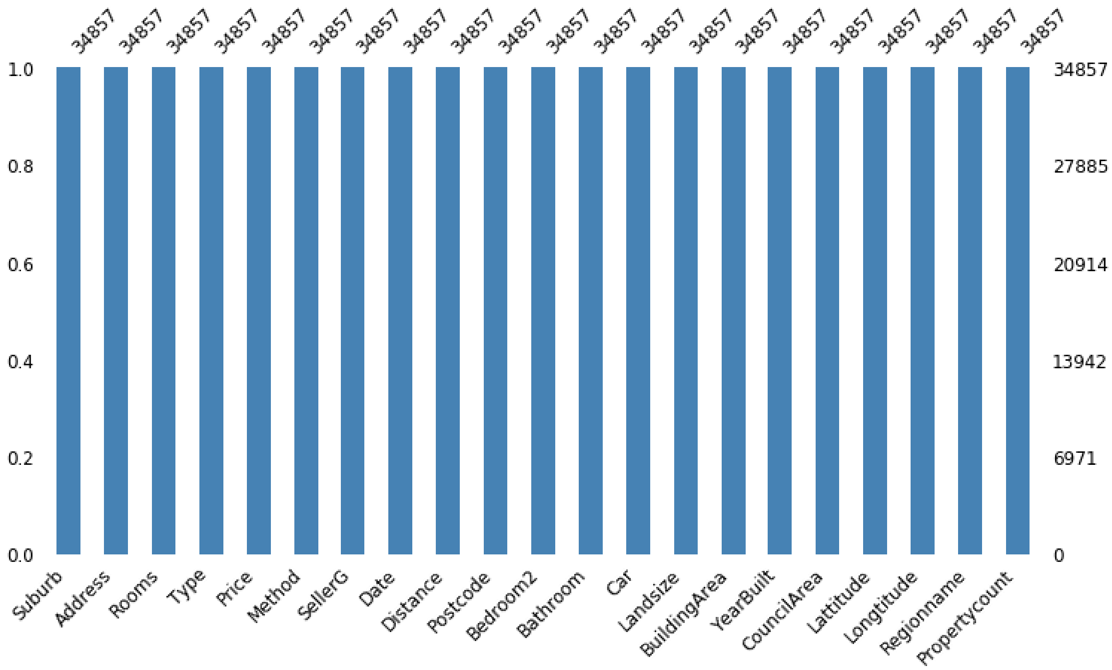
```
Price_mean = full1.Price.mean()
Distance_mean = full1.Distance.mean()
Postcode_mean = full1.Postcode.mean()
Bedroom2_mean = full1.Bedroom2.mean()
Bathroom_mean = full1.Bathroom.mean()
Car_mean = full1.Car.mean()
Landsize_mean = full1.Landsize.mean()
BuildingArea_mean = full1.BuildingArea.mean()
YearBuilt_mean = full1.YearBuilt.mean()
Lattitude_mean = full1.Lattitude.mean()
Longtitude_mean = full1.Longtitude.mean()
Propertycount_mean = full1.Propertycount.mean()
full1.fillna(value = {'Price':Price_mean
                      ,'Distance':Distance_mean
                      ,'Postcode':int(Postcode_mean)
                      ,'Bedroom2':Bedroom2_mean
                      ,'Car':Car_mean
                      ,'Bathroom':Bathroom_mean
                      ,'Landsize':Landsize_mean
                      ,'BuildingArea':BuildingArea_mean
                      ,'YearBuilt':int(YearBuilt_mean)
                      ,'Lattitude':Lattitude_mean
                      ,'Longtitude':Longtitude_mean
                      ,'Propertycount':Propertycount_mean
                      ,'CouncilArea' : 'Boroondara City Council'#we are taking␣
↪most used categroical value_counts()
                      ,'Regionname' :'Southern Metropolitan'} #we are taking␣
↪most used categroical value_counts()
                      ,inplace = True
                      )
#full1["Price"].fillna( method ='ffill')
```

[20]:
```
full1.isna().sum()
```

[20]:
```
Suburb           0
Address          0
Rooms            0
Type             0
Price            0
Method           0
SellerG          0
Date             0
Distance         0
Postcode         0
Bedroom2         0
Bathroom         0
Car              0
Landsize         0
BuildingArea     0
YearBuilt        0
CouncilArea      0
Lattitude        0
Longtitude       0
Regionname       0
Propertycount    0
dtype: int64
```

[21]:
```
#missing no library offers a very nice way to visualize the distribution of NaN␣
↪values.
import missingno as msno
import matplotlib.pyplot as plt
%matplotlib inline
msno.bar(full1, figsize=(12, 6), fontsize=12, color='steelblue')
```

[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x186607872b0>
```



[24]:
```
#Outlier Treatment:
#Identify the varibales : Make a subset of the dataset with all the numeric␣
↪variables.
full1_numeric=full1.select_dtypes(['int64','float64'])
full1_numeric.head()
```

[24]:
```
   Rooms         Price  Distance  Postcode  Bedroom2  Bathroom  Car  Landsize  \
0      2  1.050173e+06       2.5    3067.0       2.0       1.0  1.0     126.0
1      2  1.480000e+06       2.5    3067.0       2.0       1.0  1.0     202.0
2      2  1.035000e+06       2.5    3067.0       2.0       1.0  0.0     156.0
3      3  1.050173e+06       2.5    3067.0       3.0       2.0  1.0       0.0
4      3  1.465000e+06       2.5    3067.0       3.0       2.0  0.0     134.0

   BuildingArea  YearBuilt  Lattitude  Longtitude  Propertycount
0      160.2564     1965.0   -37.8014    144.9958         4019.0
1      160.2564     1965.0   -37.7996    144.9984         4019.0
2       79.0000     1900.0   -37.8079    144.9934         4019.0
3      160.2564     1965.0   -37.8114    145.0116         4019.0
4      150.0000     1900.0   -37.8093    144.9944         4019.0
```

[27]:
```
#Outliers : For each variable of this subset, carry out the outlier detection.
#          Find out the percentile distribution of each #variable and carry out␣
↪capping and flooring for outlier values
full1_numeric.describe()
#75% is Q3
```

```
#25% is Q1
#IQR=Q3-Q1
#floor = Q1 -1.5*IQR
#cap=Q3+1.5*IQR
#we can also use full1_numeric['Rooms'].quantile(0.25) to get Q1 Value.
```

[27]:

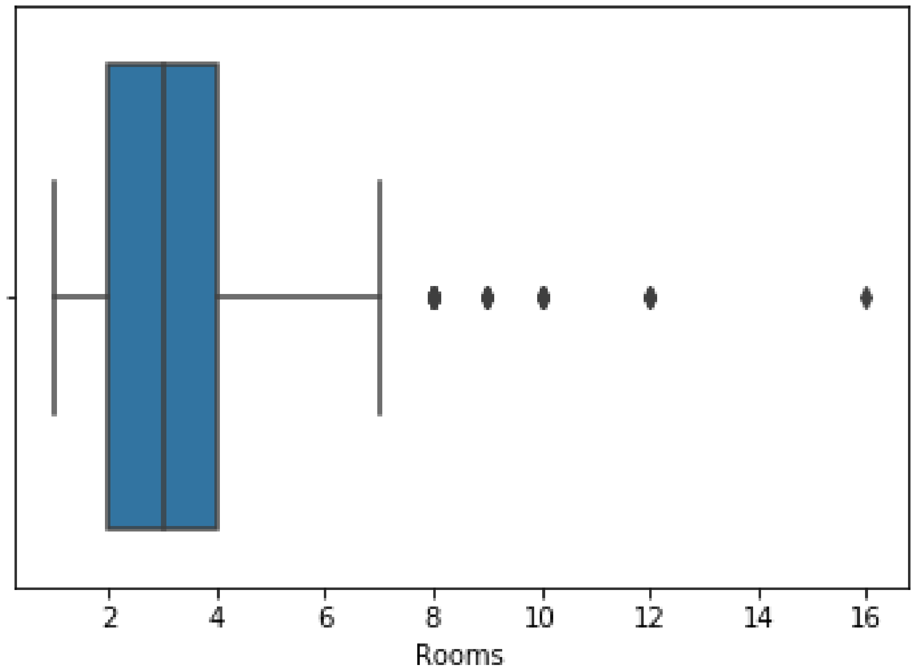
```
              Rooms         Price      Distance      Postcode      Bedroom2  \
count  34857.000000  3.485700e+04  34857.000000  34857.000000  34857.000000
mean       3.031012  1.050173e+06     11.184929   3116.062857      3.084647
std        0.969933  5.671357e+05      6.788795    109.022339      0.857337
min        1.000000  8.500000e+04      0.000000   3000.000000      0.000000
25%        2.000000  6.950000e+05      6.400000   3051.000000      3.000000
50%        3.000000  1.050173e+06     10.300000   3103.000000      3.000000
75%        4.000000  1.150000e+06     14.000000   3156.000000      3.084647
max       16.000000  1.120000e+07     48.100000   3978.000000     30.000000

           Bathroom           Car       Landsize  BuildingArea     YearBuilt  \
count  34857.000000  34857.000000   34857.000000  34857.000000  34857.000000
mean       1.624798      1.728845     593.598993    160.256400   1965.129328
std        0.633013      0.875119    2763.694121    251.943934     24.932766
min        0.000000      0.000000       0.000000      0.000000   1196.000000
25%        1.000000      1.000000     357.000000    160.000000   1965.000000
50%        1.624798      1.728845     593.598993    160.256400   1965.000000
75%        2.000000      2.000000     598.000000    160.256400   1965.000000
max       12.000000     26.000000  433014.000000  44515.000000   2106.000000

          Lattitude    Longtitude  Propertycount
count  34857.000000  34857.000000   34857.000000
mean     -37.810634    145.001851    7572.888306
std        0.079280      0.105528    4427.899750
min      -38.190430    144.423790      83.000000
25%      -37.846900    144.964400    4385.000000
50%      -37.810634    145.001851    6763.000000
75%      -37.770900    145.051750   10412.000000
max      -37.390200    145.526350   21650.000000
```

[29]:

```
#BoxPlot for variable "Rooms"
import seaborn as sns
sns.boxplot(x=full1_numeric['Rooms'])
```

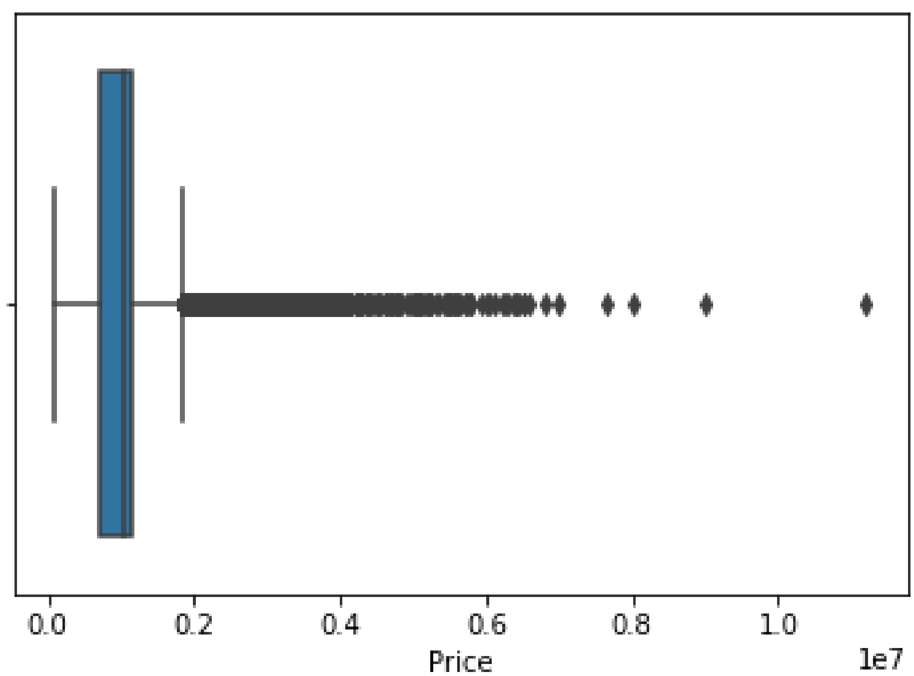
[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18660b94da0>
```



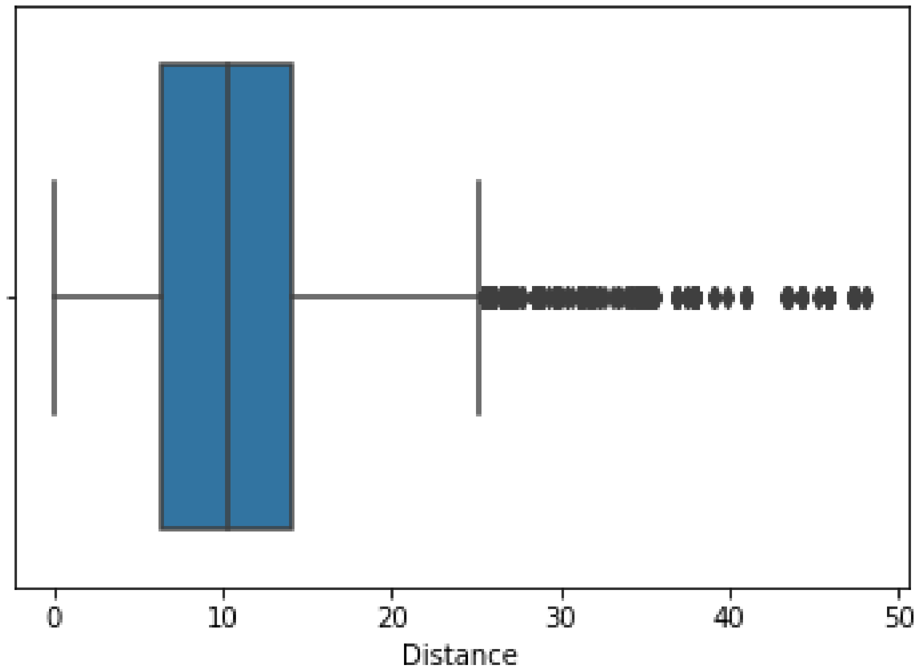
[32]:
```
sns.boxplot(x=full1_numeric['Price'])
```

[32]: <matplotlib.axes._subplots.AxesSubplot at 0x18660ca2a20>



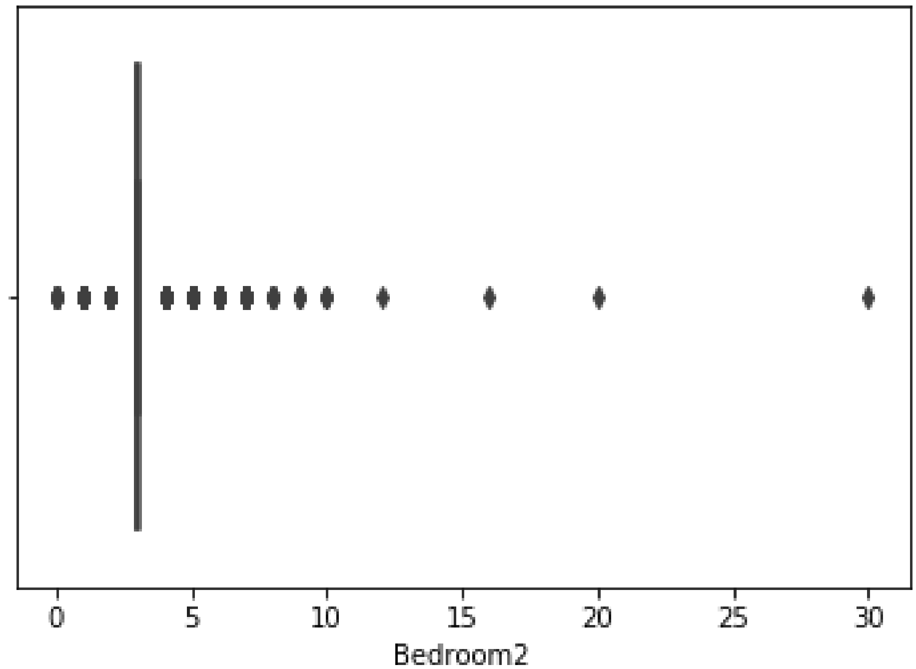
[33]:
```
sns.boxplot(x=full1_numeric['Distance'])
```

[33]: <matplotlib.axes._subplots.AxesSubplot at 0x186618a9860>



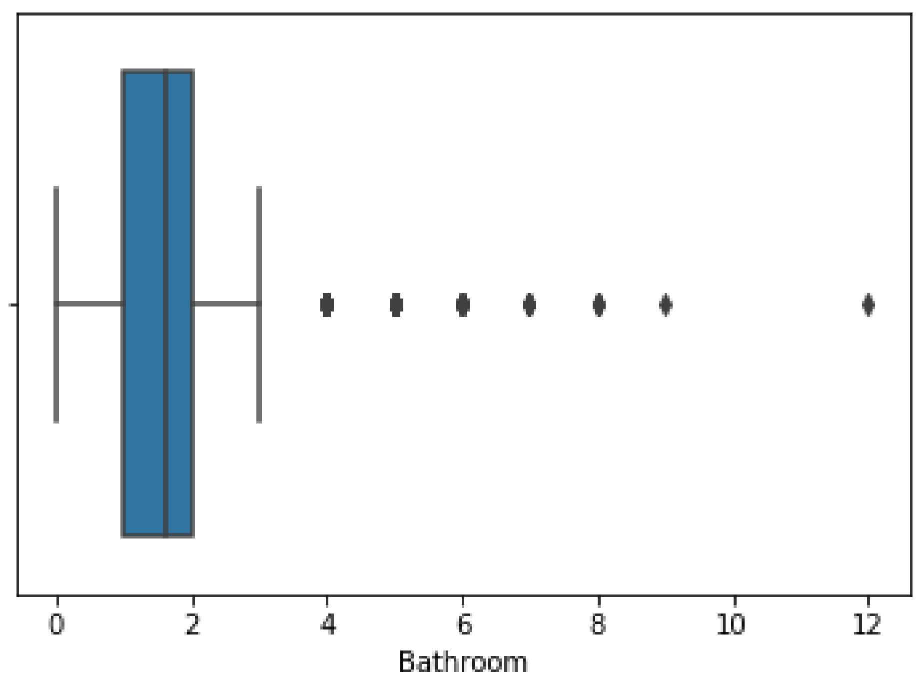
[35]:
```
sns.boxplot(x=full1_numeric['Bedroom2'])
```

[35]: <matplotlib.axes._subplots.AxesSubplot at 0x1866194f908>



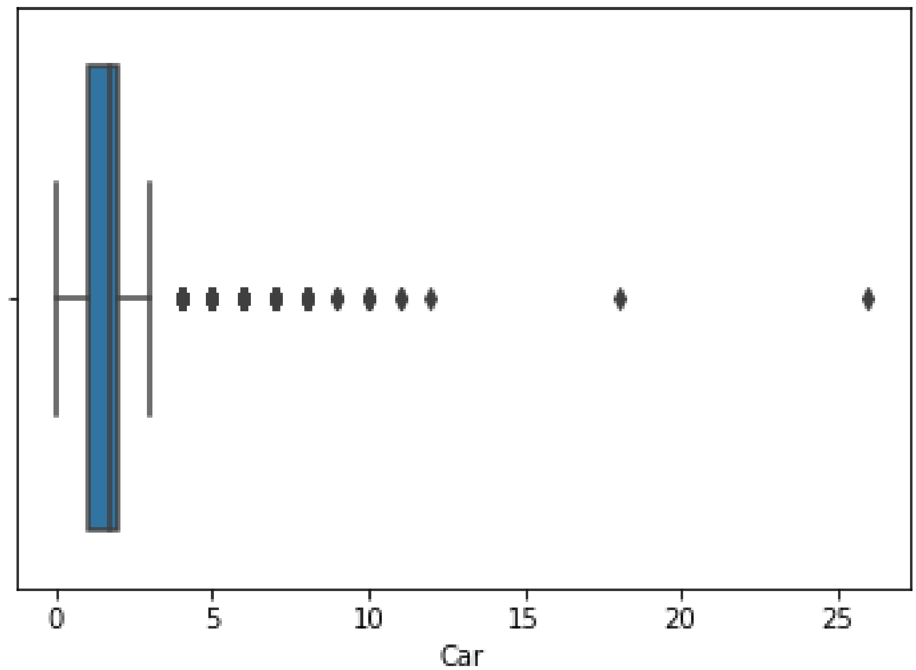
[36]:
```
sns.boxplot(x=full1_numeric['Bathroom'])
```

[36]: <matplotlib.axes._subplots.AxesSubplot at 0x1866189f780>



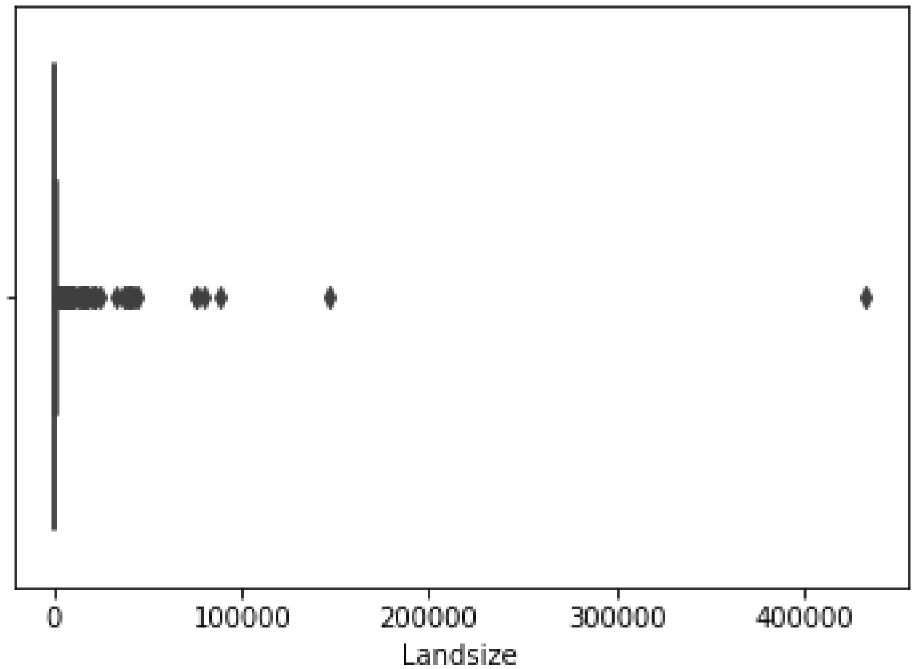
[37]:
```
sns.boxplot(x=full1_numeric['Car'])
```

[37]: <matplotlib.axes._subplots.AxesSubplot at 0x186618f7160>



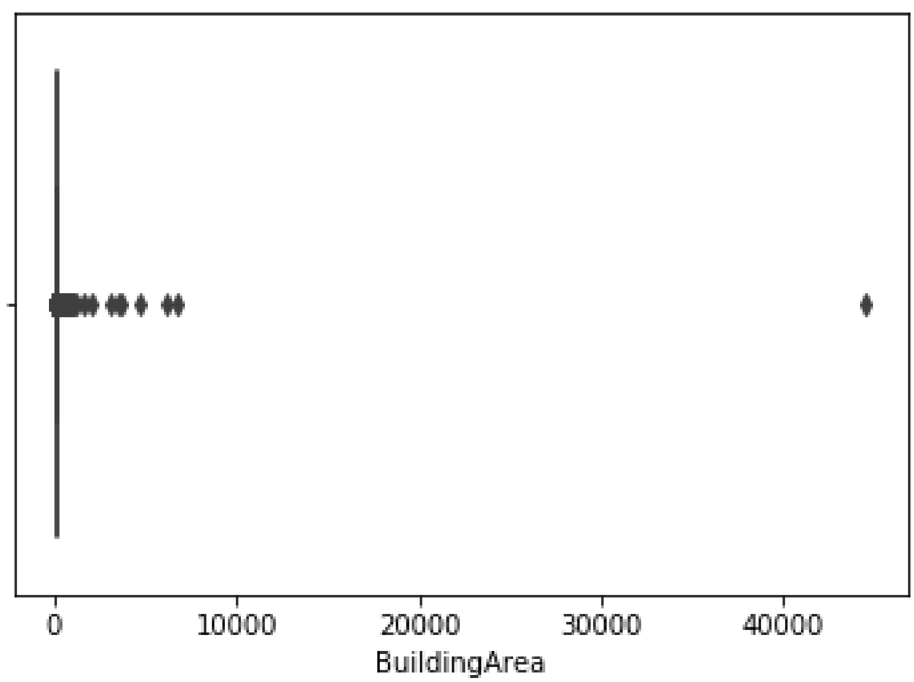
[38]:
```
sns.boxplot(x=full1_numeric['Landsize'])
```

[38]: <matplotlib.axes._subplots.AxesSubplot at 0x18661a68e80>



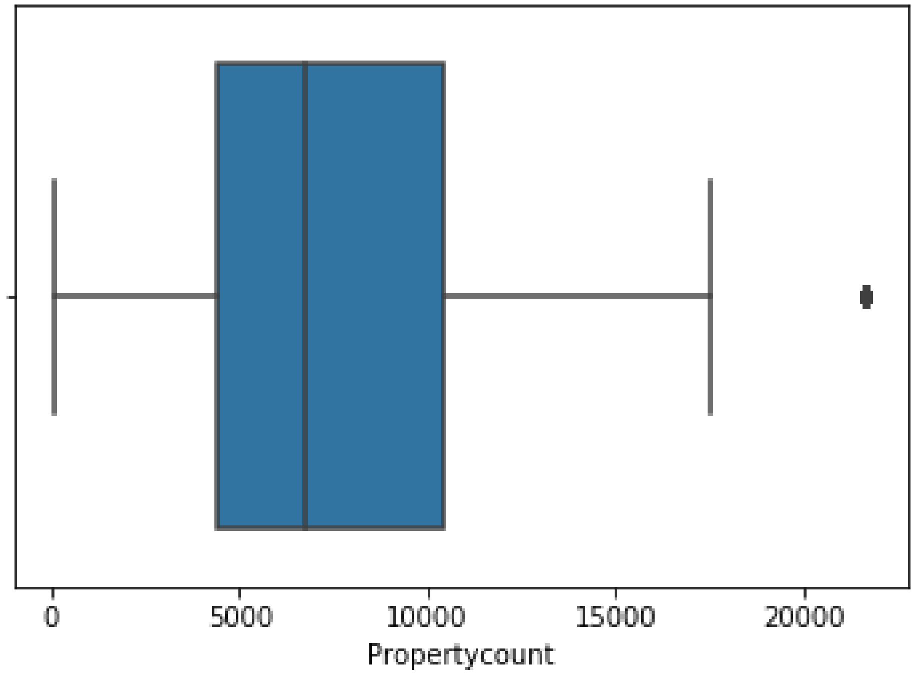
[39]:
```
sns.boxplot(x=full1_numeric['BuildingArea'])
```

[39]: <matplotlib.axes._subplots.AxesSubplot at 0x18661ab8128>



[40]:
```
sns.boxplot(x=full1_numeric['Propertycount'])
```

[40]: <matplotlib.axes._subplots.AxesSubplot at 0x18660b3ec18>



[41]:
```
for col in full1_numeric:
    if col!='Lattitude' and col!='Longtitude':# not considering Lattitude &␣
↪Longitude as Outlier treatment
        print("\n\nOutlier for ",col)
        # Cap=IQ3+1.5*IQR & Floor=1.5*IQR-IQ1
        Q1 = full1_numeric[col].quantile(0.25)
        Q3 = full1_numeric[col].quantile(0.75)
        IQR = Q3 - Q1
        print("IQR for ",col,"=",IQR)
        limit=1.5*IQR
        floor = limit - Q1
        print("Floor for ",col,"=",floor)
        Cap = Q3 + limit
        print("Cap for ",col,"=",Cap)
        #values less than (1.5*IQR-Q1) and more than (1.5*IQR+Q3) are removed.
        #Removing Outlier values for Price due to Large number
        if col=='Price':
            ␣
↪full1_numeric=full1_numeric[(full1_numeric[col]>=floor)&(full1_numeric[col]<=Cap)]
```

```python
            print(full1_numeric.shape)
```

```
Outlier for  Rooms
IQR for  Rooms = 2.0
Floor for  Rooms = 1.0
Cap for  Rooms = 7.0


Outlier for  Price
IQR for  Price = 455000.0
Floor for  Price = -12500.0
Cap for  Price = 1832500.0
(32301, 13)


Outlier for  Distance
IQR for  Distance = 7.799999999999999
Floor for  Distance = 5.299999999999999
Cap for  Distance = 25.9


Outlier for  Postcode
IQR for  Postcode = 107.0
Floor for  Postcode = -2885.5
Cap for  Postcode = 3313.5


Outlier for  Bedroom2
IQR for  Bedroom2 = 0.08464714714714727
Floor for  Bedroom2 = -2.873029279279279
Cap for  Bedroom2 = 3.211617867867868


Outlier for  Bathroom
IQR for  Bathroom = 1.0
Floor for  Bathroom = 0.5
Cap for  Bathroom = 3.5


Outlier for  Car
IQR for  Car = 1.0
Floor for  Car = 0.5
Cap for  Car = 3.5
```

```
Outlier for  Landsize
IQR for  Landsize = 250.598993361392
Floor for  Landsize = 32.898490042087985
Cap for  Landsize = 969.4974834034799


Outlier for  BuildingArea
IQR for  BuildingArea = 7.256400356571106
Floor for  BuildingArea = -142.11539946514335
Cap for  BuildingArea = 171.14100089142778


Outlier for  YearBuilt
IQR for  YearBuilt = 1.0
Floor for  YearBuilt = -1963.5
Cap for  YearBuilt = 1967.5


Outlier for  Propertycount
IQR for  Propertycount = 6118.0
Floor for  Propertycount = 4883.0
Cap for  Propertycount = 19589.0
```

Data Preparation Feature Engineering:

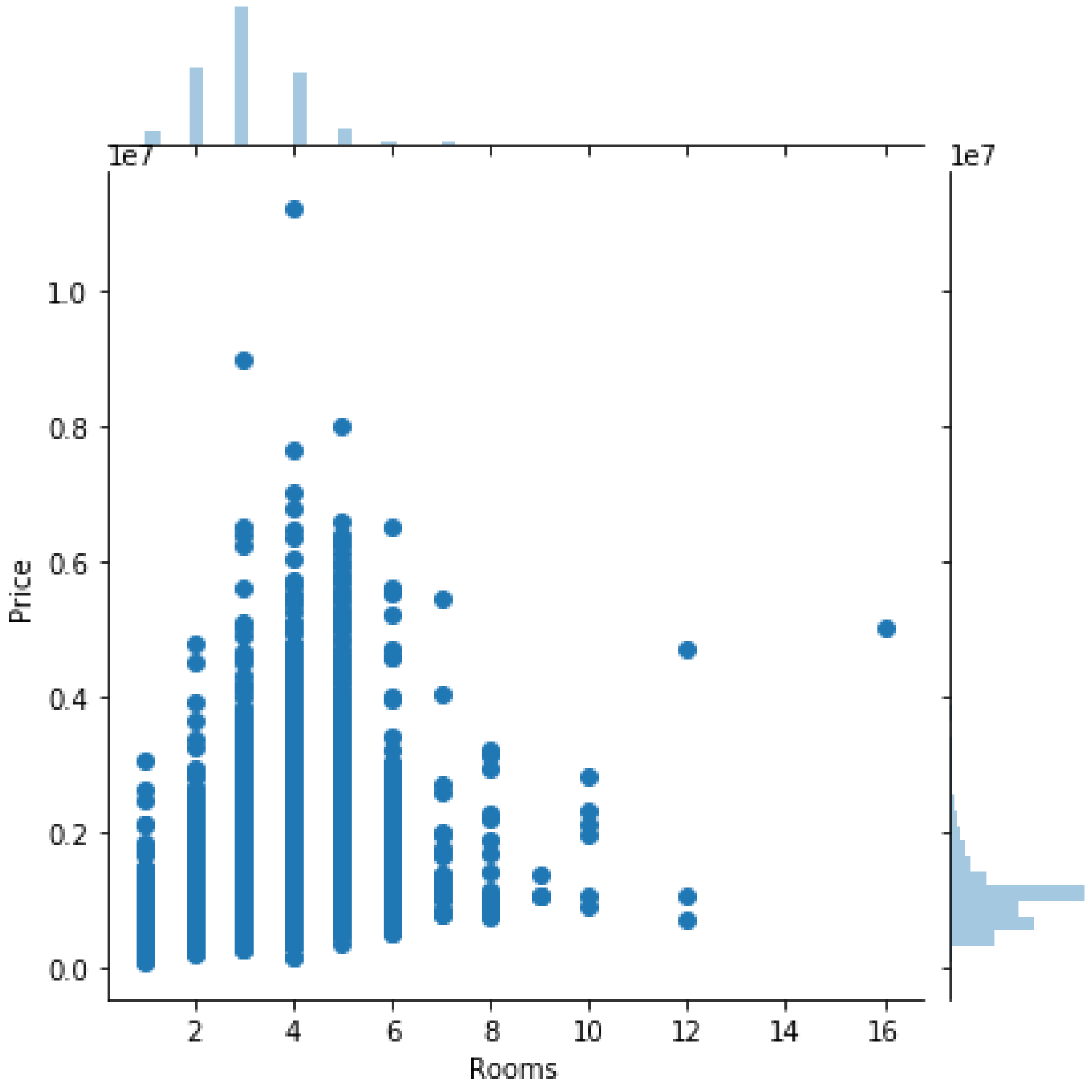
Feature Transformation:

Identify variables that have non-linear trends. How many variables have non-linear trends? Transform them (as required)

[42]:
```
#Identify variables that have non-linear trends
sns.jointplot(x='Rooms',y='Price',data=full1,kind='scatter')
```

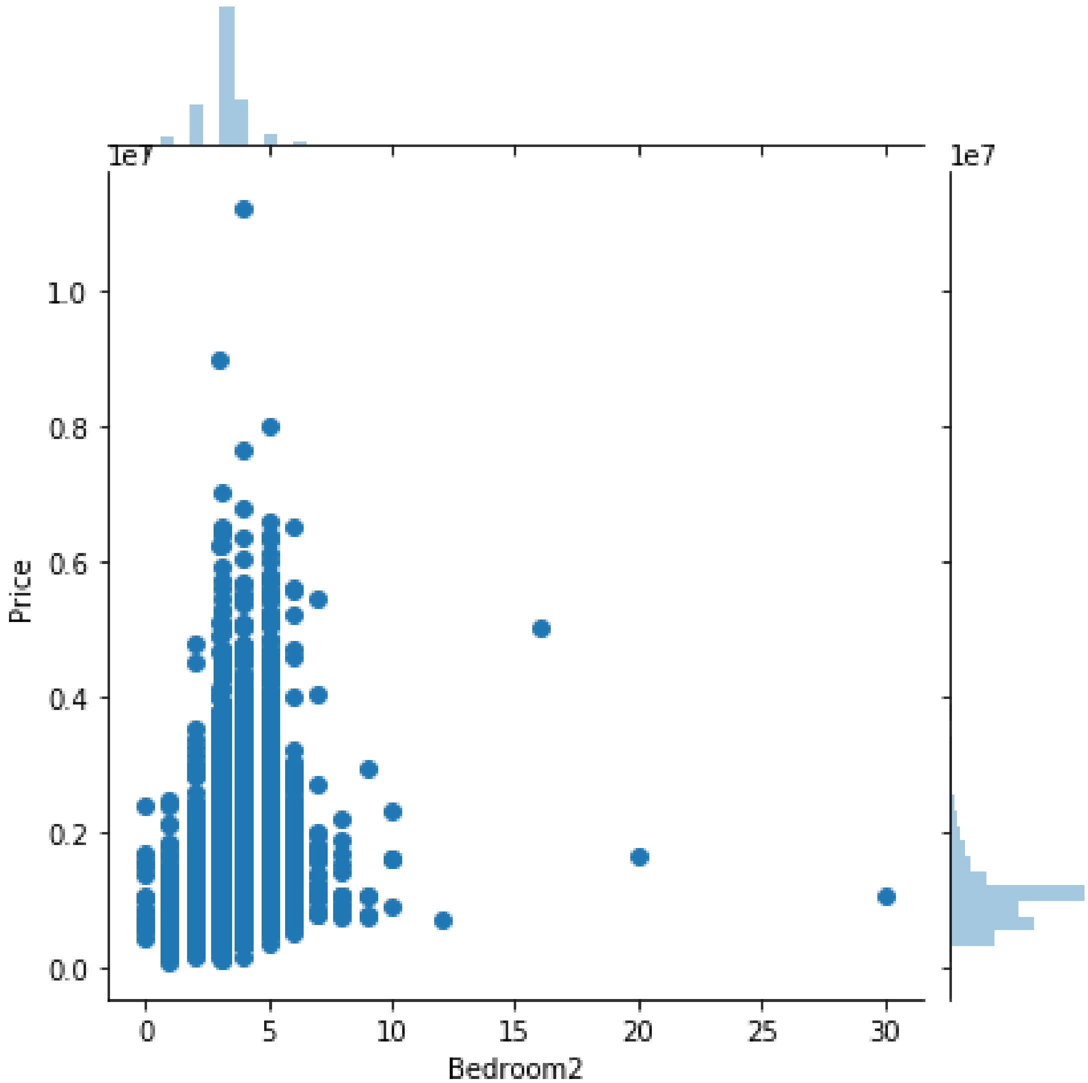
```
C:\Users\imvik\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will
be interpreted as an array index, `arr[np.array(seq)]`, which will result either
in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

[42]:
```
<seaborn.axisgrid.JointGrid at 0x18661b42eb8>
```



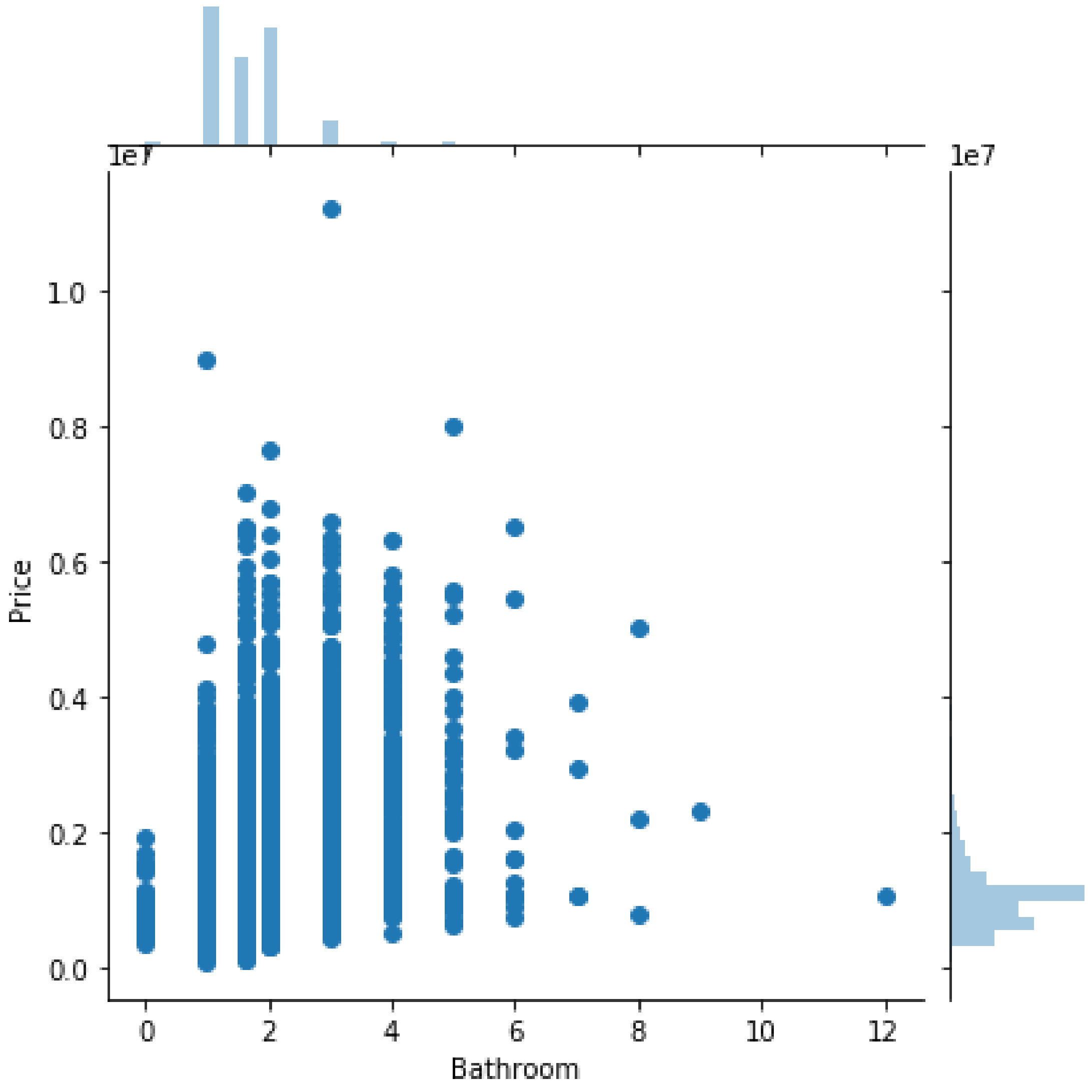
[43]:
```
sns.jointplot(x='Bedroom2',y='Price',data=full1,kind='scatter')
```

[43]: <seaborn.axisgrid.JointGrid at 0x18662f6bf60>



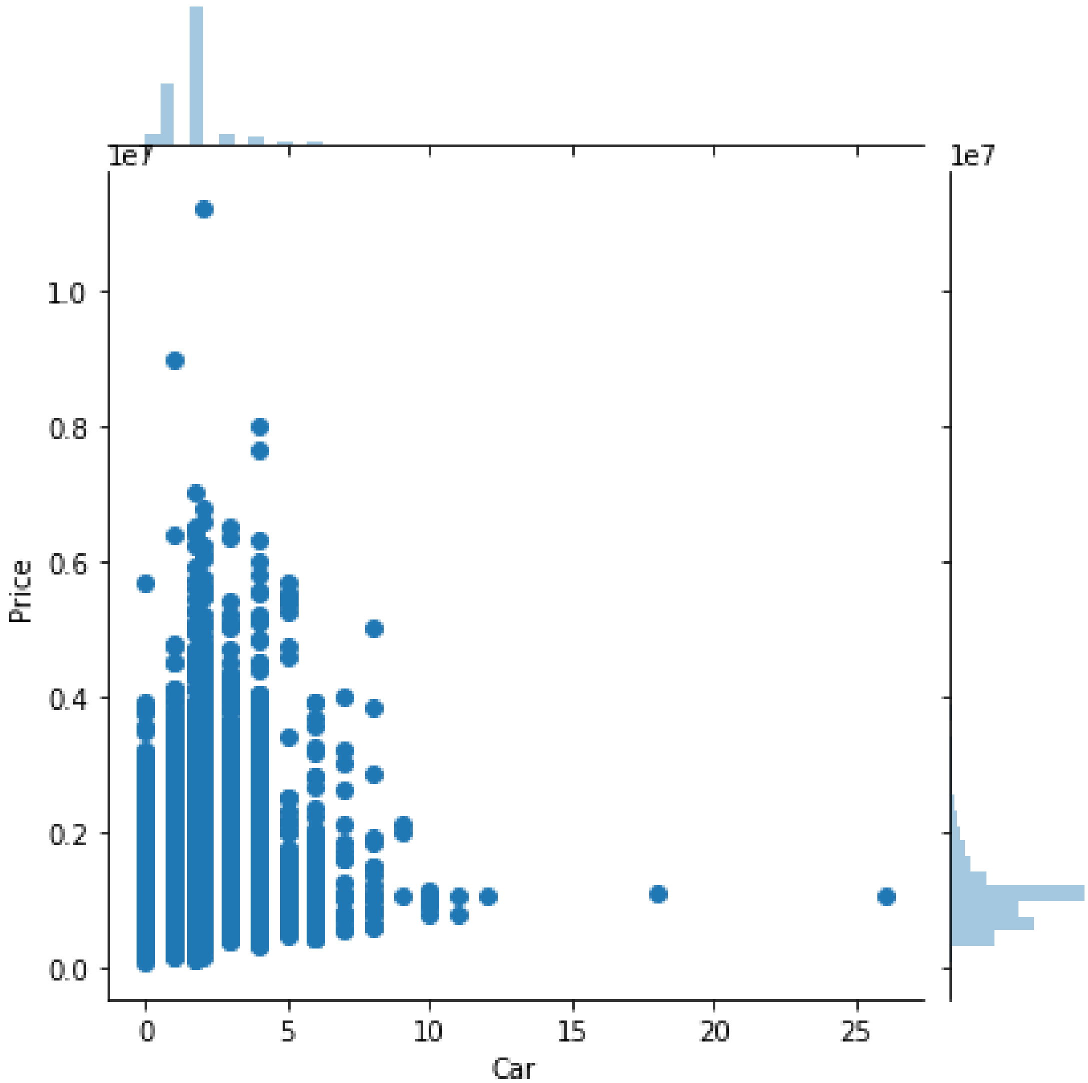

```
[44]: sns.jointplot(x='Bathroom',y='Price',data=full1,kind='scatter')
```

```
[44]: <seaborn.axisgrid.JointGrid at 0x18663238860>
```



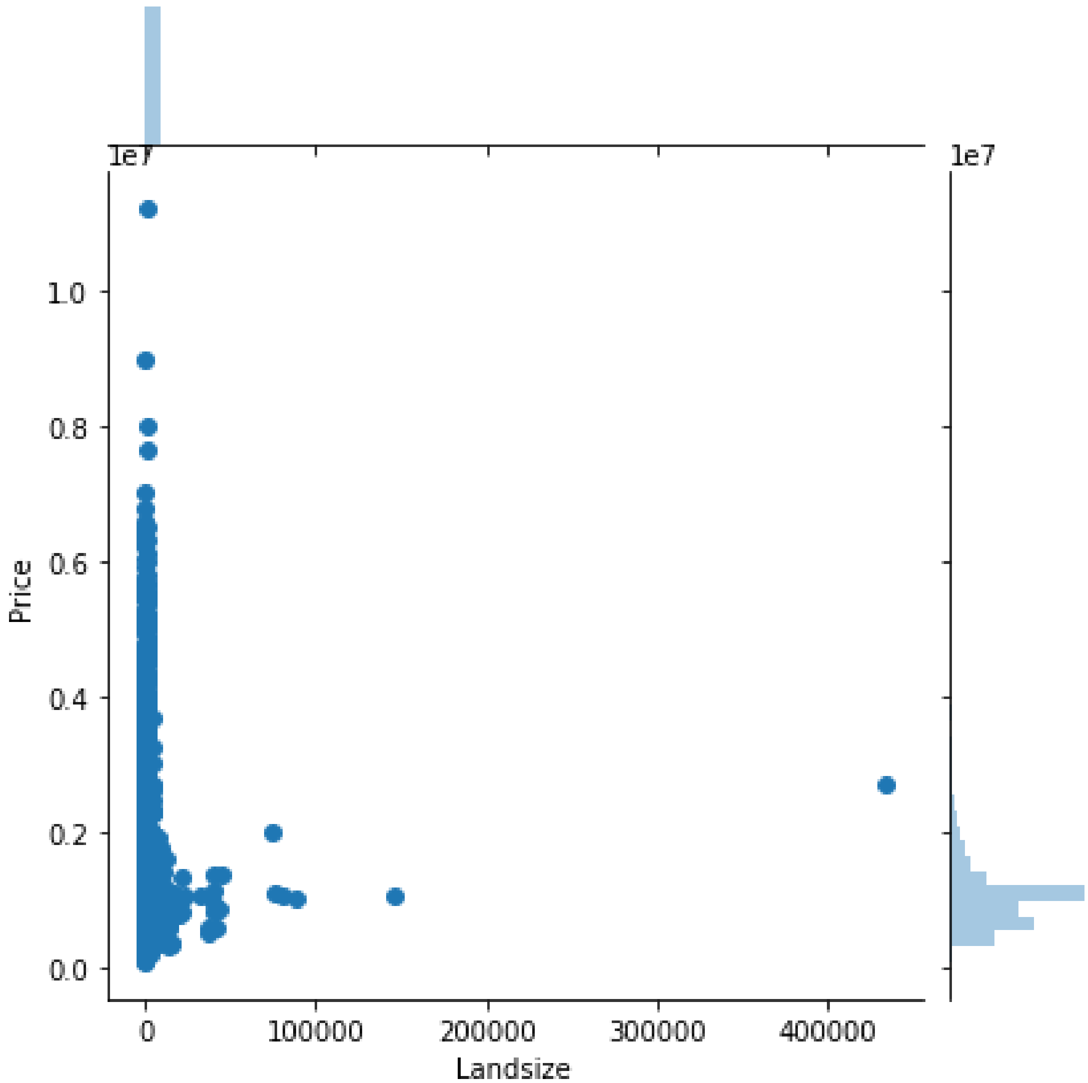
[45]:
```
sns.jointplot(x='Car',y='Price',data=full1,kind='scatter')
```

[45]: <seaborn.axisgrid.JointGrid at 0x18663313cc0>



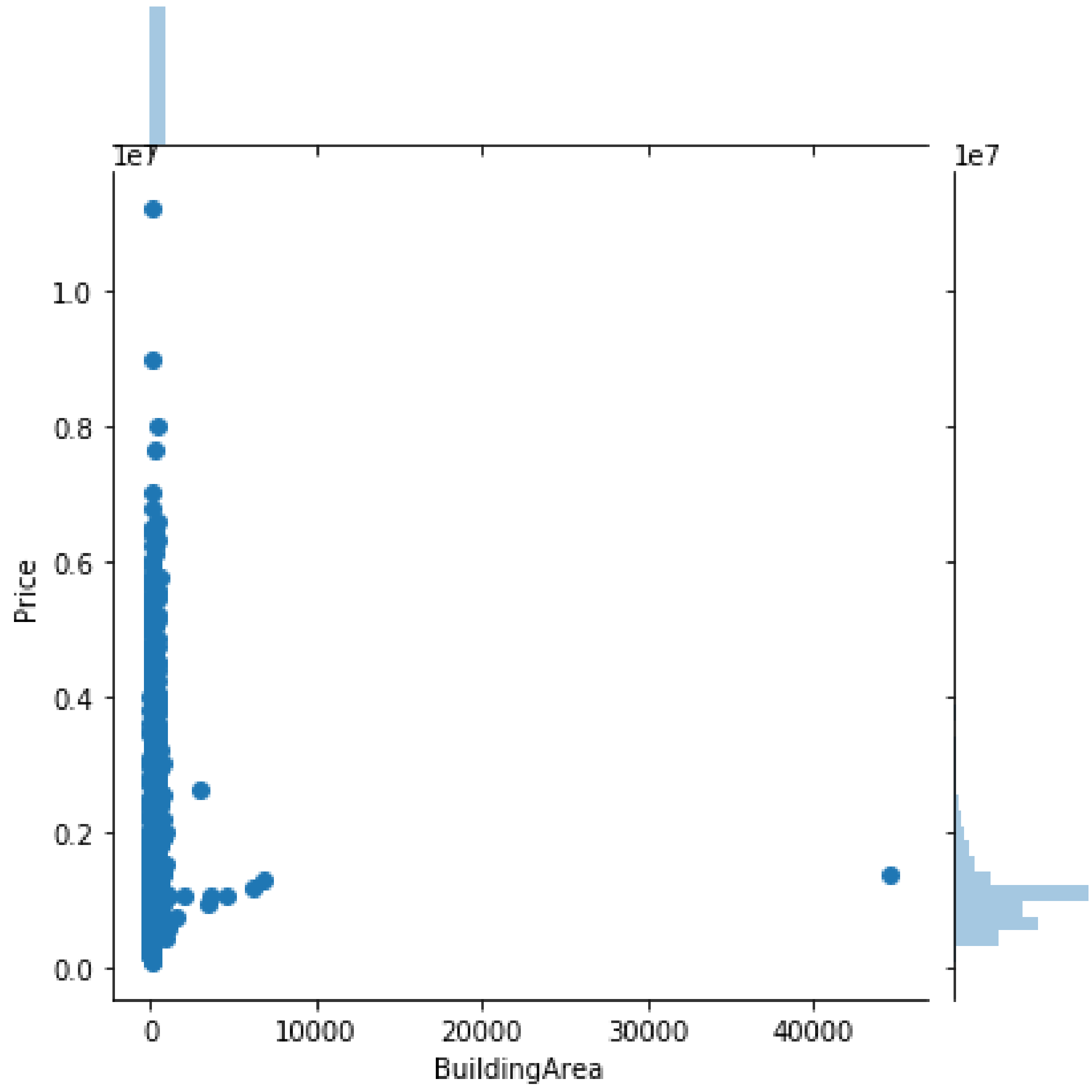
[46]:
```
sns.jointplot(x='Landsize',y='Price',data=full1,kind='scatter')
```

[46]: <seaborn.axisgrid.JointGrid at 0x186634ecf98>



[47]:
```
sns.jointplot(x='BuildingArea',y='Price',data=full1,kind='scatter')
```

[47]: <seaborn.axisgrid.JointGrid at 0x1866469a6a0>



[49]:
```
# Matrix form for correlation data
full1.corr()
```

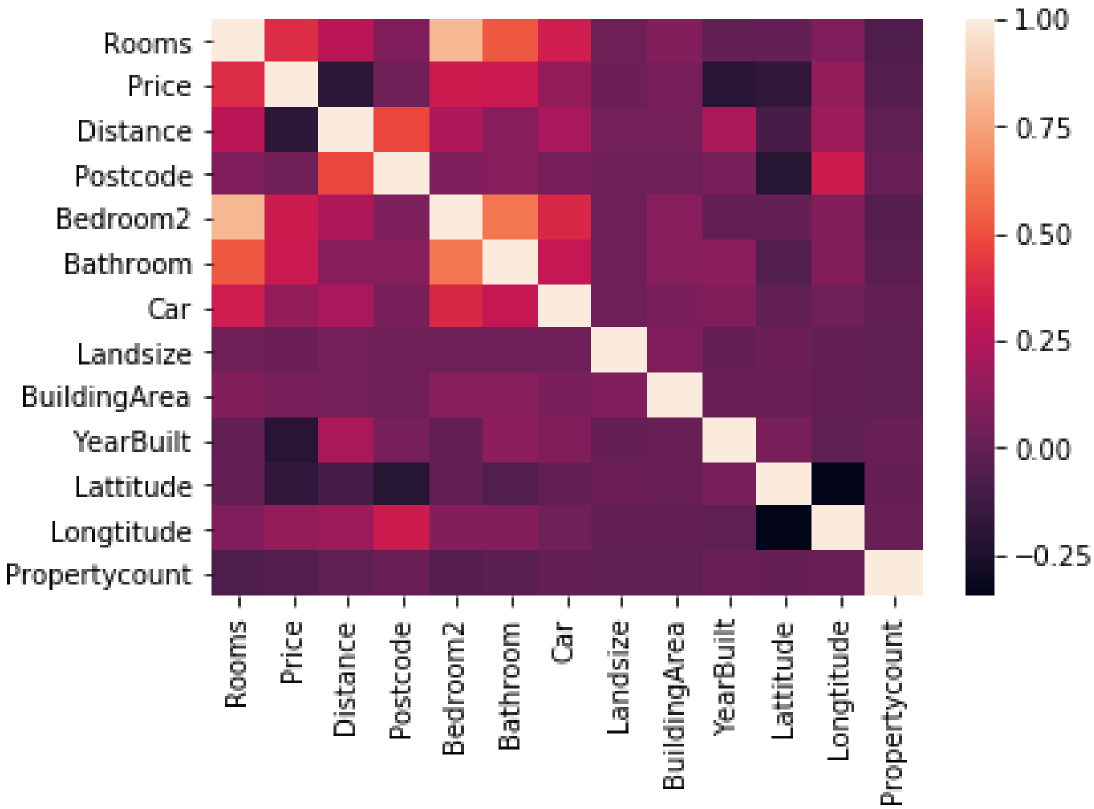
[49]:

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom \
Rooms	1.000000	0.404908	0.271511	0.085890	0.819099	0.529191
Price	0.404908	1.000000	-0.186848	0.040511	0.327485	0.324631
Distance	0.271511	-0.186848	1.000000	0.481566	0.239091	0.111939
Postcode	0.085890	0.040511	0.481566	1.000000	0.080398	0.108110
Bedroom2	0.819099	0.327485	0.239091	0.080398	1.000000	0.614737
Bathroom	0.529191	0.324631	0.111939	0.108110	0.614737	1.000000
Car	0.337780	0.154545	0.211768	0.060746	0.385459	0.305530
Landsize	0.030136	0.026460	0.048717	0.032452	0.034578	0.034007
BuildingArea	0.098468	0.065301	0.050110	0.029252	0.110089	0.107991

```
YearBuilt     -0.008246 -0.200594  0.220729  0.063470 -0.001555  0.130616
Lattitude      0.004254 -0.170840 -0.089506 -0.208542  0.003431 -0.058905
Longtitude     0.090140  0.154873  0.179113  0.327576  0.105682  0.106059
Propertycount -0.071675 -0.052934 -0.018140  0.017108 -0.045785 -0.028168

                    Car  Landsize  BuildingArea  YearBuilt  Lattitude  \
Rooms          0.337780  0.030136      0.098468  -0.008246   0.004254
Price          0.154545  0.026460      0.065301  -0.200594  -0.170840
Distance       0.211768  0.048717      0.050110   0.220729  -0.089506
Postcode       0.060746  0.032452      0.029252   0.063470  -0.208542
Bedroom2       0.385459  0.034578      0.110089  -0.001555   0.003431
Bathroom       0.305530  0.034007      0.107991   0.130616  -0.058905
Car            1.000000  0.034846      0.074810   0.095065  -0.008943
Landsize       0.034846  1.000000      0.085636   0.010427   0.022734
BuildingArea   0.074810  0.085636      1.000000   0.013803   0.012526
YearBuilt      0.095065  0.010427      0.013803   1.000000   0.069606
Lattitude     -0.008943  0.022734      0.012526   0.069606   1.000000
Longtitude     0.046613 -0.002323     -0.001562  -0.016935  -0.345589
Propertycount -0.008171 -0.014453     -0.015003   0.014488   0.009574

               Longtitude  Propertycount
Rooms            0.090140      -0.071675
Price            0.154873      -0.052934
Distance         0.179113      -0.018140
Postcode         0.327576       0.017108
Bedroom2         0.105682      -0.045785
Bathroom         0.106059      -0.028168
Car              0.046613      -0.008171
Landsize        -0.002323      -0.014453
BuildingArea    -0.001562      -0.015003
YearBuilt       -0.016935       0.014488
Lattitude       -0.345589       0.009574
Longtitude       1.000000       0.014067
Propertycount    0.014067       1.000000
```

[53]:
```
sns.heatmap(full1.corr())
#Rooms, Bedroom2, Bathroom & Car have weak positive correlation with Price.
#Distance have weak weak negative correlation with Price
#Landsize & Building Area do not have much correlation with Price and it is␣
 ↪having non-linear trend.
```

[53]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x18665314d30>
```



Standardization:

Name the variables to be standardised before using a distance-based algorithm

[56]:
```
#All numeric variables is standardised before applying KNN method(distance␣
↪based algorithm)
#Z-Score
int_col=['Rooms','Price','Distance','Bedroom2','Bathroom','Car','Landsize','BuildingArea','Yea
from scipy.stats import zscore
full1_z=full1[int_col].apply(zscore)
full1_z
```

[56]:

	Rooms	Price	Distance	Bedroom2	Bathroom	Car \
0	-1.062988	0.000000	-1.279322	-1.265153	-9.870370e-01	-8.328650e-01
1	-1.062988	0.757901	-1.279322	-1.265153	-9.870370e-01	-8.328650e-01
2	-1.062988	-0.026755	-1.279322	-1.265153	-9.870370e-01	-1.975583e+00
3	-0.031974	0.000000	-1.279322	-0.098734	5.927324e-01	-8.328650e-01
4	-0.031974	0.731452	-1.279322	-0.098734	5.927324e-01	-1.975583e+00
5	-0.031974	-0.352960	-1.279322	-0.098734	5.927324e-01	-8.328650e-01
6	0.999040	0.969494	-1.279322	-0.098734	-9.870370e-01	3.098534e-01
7	0.999040	0.000000	-1.279322	-0.098734	5.927324e-01	3.098534e-01
8	-1.062988	0.000000	-1.279322	1.067685	-9.870370e-01	3.098534e-01

9	-1.062988	0.000000	-1.279322	-0.098734	5.927324e-01	-8.328650e-01
10	-1.062988	-0.192502	-1.279322	-1.265153	-9.870370e-01	-1.975583e+00
11	-0.031974	1.456157	-1.279322	1.067685	5.927324e-01	-1.975583e+00
12	-1.062988	0.000000	-1.279322	-1.265153	5.927324e-01	-8.328650e-01
13	0.999040	0.000000	-1.279322	3.400522	5.927324e-01	-1.975583e+00
14	-1.062988	1.032972	-1.279322	-1.265153	-9.870370e-01	3.098534e-01
15	-0.031974	-0.088469	-1.279322	0.000000	3.507793e-16	2.537345e-16
16	-1.062988	-0.538104	-1.279322	0.000000	3.507793e-16	2.537345e-16
17	-2.094002	-1.322759	-1.279322	-2.431571	-9.870370e-01	-8.328650e-01
18	-1.062988	0.082568	-1.279322	-0.098734	-9.870370e-01	3.098534e-01
19	-1.062988	-0.896048	-1.279322	0.000000	3.507793e-16	2.537345e-16
20	-1.062988	0.000000	-1.279322	-1.265153	-9.870370e-01	-8.328650e-01
21	-1.062988	-0.511654	-1.279322	0.000000	3.507793e-16	2.537345e-16
22	-2.094002	-1.003607	-1.279322	0.000000	3.507793e-16	2.537345e-16
23	-1.062988	-0.617451	-1.279322	-1.265153	5.927324e-01	-8.328650e-01
24	-0.031974	0.528676	-1.279322	-0.098734	5.927324e-01	3.098534e-01
25	-1.062988	-0.529287	-1.279322	-1.265153	5.927324e-01	-8.328650e-01
26	0.999040	1.648353	-1.279322	0.000000	3.507793e-16	2.537345e-16
27	-2.094002	-0.970105	-1.279322	0.000000	3.507793e-16	2.537345e-16
28	-1.062988	0.215695	-1.279322	-1.265153	-9.870370e-01	-8.328650e-01
29	-2.094002	-1.074138	-1.279322	-2.431571	-9.870370e-01	-8.328650e-01
…	…	…	…	…	…	…
34827	0.999040	0.080805	-0.704838	1.067685	-9.870370e-01	1.452572e+00
34828	-0.031974	-0.264796	-0.704838	-0.098734	-9.870370e-01	3.098534e-01
34829	-0.031974	0.000000	-1.190940	0.000000	3.507793e-16	2.537345e-16
34830	-0.031974	0.176022	-1.190940	-0.098734	5.927324e-01	3.098534e-01
34831	0.999040	-0.661533	0.782929	1.067685	5.927324e-01	3.738009e+00
34832	0.999040	-0.636847	0.782929	0.000000	3.507793e-16	2.537345e-16
34833	0.999040	-0.201319	0.812390	1.067685	5.927324e-01	3.098534e-01
34834	0.999040	0.000000	0.812390	1.067685	5.927324e-01	3.098534e-01
34835	-0.031974	0.000000	0.812390	-0.098734	5.927324e-01	3.098534e-01
34836	2.030054	0.616839	0.812390	2.234103	5.927324e-01	3.098534e-01
34837	2.030054	1.710068	-0.645916	2.234103	2.172502e+00	3.098534e-01
34838	-1.062988	-1.058269	-0.645916	0.000000	3.507793e-16	2.537345e-16
34839	-1.062988	-1.014187	-0.645916	0.000000	3.507793e-16	2.537345e-16
34840	-1.062988	-0.934840	-0.645916	0.000000	3.507793e-16	2.537345e-16
34841	-1.062988	-0.194266	-0.645916	-1.265153	5.927324e-01	-8.328650e-01
34842	-0.031974	0.616839	-0.645916	-0.098734	5.927324e-01	2.537345e-16
34843	-0.031974	-0.388225	-0.645916	-0.098734	-9.870370e-01	3.098534e-01
34844	-2.094002	-1.075901	-0.969984	0.000000	3.507793e-16	2.537345e-16
34845	0.999040	0.000000	-0.969984	1.067685	2.172502e+00	3.098534e-01
34846	0.999040	-0.740880	2.108662	1.067685	5.927324e-01	3.098534e-01
34847	-0.031974	-0.970105	2.108662	-0.098734	5.927324e-01	3.098534e-01
34848	0.999040	-0.756749	2.108662	1.067685	5.927324e-01	3.098534e-01
34849	-0.031974	-0.846676	2.108662	-0.098734	5.927324e-01	3.098534e-01
34850	-0.031974	0.000000	2.108662	-0.098734	5.927324e-01	3.098534e-01
34851	-0.031974	0.089621	-0.719568	-0.098734	-9.870370e-01	2.537345e-16

34852	0.999040	0.757901	-0.719568	1.067685	-9.870370e-01	1.452572e+00
34853	-1.062988	-0.285956	-0.719568	-1.265153	5.927324e-01	-8.328650e-01
34854	-1.062988	-0.608634	-0.719568	-1.265153	-9.870370e-01	3.098534e-01
34855	-0.031974	0.158389	-0.719568	0.000000	3.507793e-16	2.537345e-16
34856	-1.062988	-0.053204	-0.719568	-1.265153	-9.870370e-01	-1.975583e+00

	Landsize	BuildingArea	YearBuilt	Lattitude	Longtitude
0	-0.169196	1.128113e-16	-0.005187	1.164791e-01	-5.734233e-02
1	-0.141696	1.128113e-16	-0.005187	1.391838e-01	-3.270396e-02
2	-0.158341	-3.225224e-01	-2.612236	3.448971e-02	-8.008545e-02
3	-0.214788	1.128113e-16	-0.005187	-9.658401e-03	9.238319e-02
4	-0.166301	-4.070964e-02	-2.612236	1.683047e-02	-7.060915e-02
5	-0.180775	1.128113e-16	-0.005187	1.732409e-01	-4.691840e-02
6	-0.171367	-7.246319e-02	1.960126	4.331933e-02	-7.345204e-02
7	-0.070052	2.371339e-01	1.639259	1.782864e-01	-5.070892e-02
8	-0.142058	1.128113e-16	-2.612236	1.404452e-01	-4.218025e-02
9	-0.141696	1.128113e-16	-2.612236	1.391838e-01	-2.796581e-02
10	-0.149295	1.128113e-16	-0.005187	8.242195e-02	-6.208048e-02
11	-0.126137	1.974420e-01	-2.211151	1.038653e-01	-2.417529e-02
12	1.338228	-3.106148e-01	1.759584	3.575109e-02	-5.070892e-02
13	-0.131565	-5.261722e-02	-4.216573	5.088758e-02	-7.819019e-02
14	-0.122157	-2.113850e-01	-3.013320	5.845583e-02	-6.113285e-02
15	0.000000	1.128113e-16	-0.005187	2.688782e-13	3.231991e-12
16	0.000000	1.128113e-16	-0.005187	2.688782e-13	3.231991e-12
17	-0.214788	1.128113e-16	-0.005187	1.240473e-01	-4.312788e-02
18	-0.135183	-3.383992e-01	-2.612236	1.215246e-01	-2.796581e-02
19	0.000000	1.128113e-16	-0.005187	2.688782e-13	3.231991e-12
20	-0.151104	-3.185532e-01	-1.609525	1.366611e-01	-4.407551e-02
21	0.000000	1.128113e-16	-0.005187	2.688782e-13	3.231991e-12
22	0.000000	1.128113e-16	-0.005187	2.688782e-13	3.231991e-12
23	-0.214788	1.128113e-16	-0.005187	-4.612903e-03	4.594933e-02
24	-0.137354	1.180581e-01	1.599150	2.692146e-02	-5.165655e-02
25	-0.214788	-2.629845e-01	1.759584	3.575109e-02	-5.070892e-02
26	0.000000	1.128113e-16	-0.005187	2.818284e-02	-7.250441e-02
27	0.000000	1.128113e-16	-0.005187	2.688782e-13	3.231991e-12
28	-0.144229	1.128113e-16	-0.005187	2.818284e-02	-4.312788e-02
29	-0.214788	1.128113e-16	-0.005187	1.139563e-01	-2.891344e-02
…	…	…	…	…	…
34827	-0.081631	1.128113e-16	-0.005187	2.411029e-01	-1.223211e+00
34828	-0.048341	-1.915390e-01	-1.007898	1.166052e-01	-1.300822e+00
34829	0.000000	1.128113e-16	-0.005187	2.688782e-13	3.231991e-12
34830	0.030540	1.128113e-16	-0.005187	3.486812e-02	-5.544690e-01
34831	-0.012157	-4.864803e-02	0.596439	1.636562e+00	-1.211271e+00
34832	0.000000	1.128113e-16	-0.005187	2.688782e-13	3.231991e-12
34833	0.063829	1.128113e-16	-0.005187	-1.333723e+00	1.694067e+00
34834	0.043566	1.128113e-16	-0.005187	-1.163438e+00	1.783997e+00
34835	0.030178	1.128113e-16	-0.005187	-1.367528e+00	1.792905e+00

```
34836  0.052612  5.268851e-01   0.596439 -1.039445e+00  1.733204e+00
34837 -0.051960  3.204870e-01   2.000235 -6.453912e-01 -1.168438e+00
34838  0.000000  1.128113e-16  -0.005187  2.688782e-13  3.231991e-12
34839  0.000000  1.128113e-16  -0.005187  2.688782e-13  3.231991e-12
34840  0.000000  1.128113e-16  -0.005187  2.688782e-13  3.231991e-12
34841 -0.170643  1.128113e-16   1.358500 -6.726369e-01 -9.139050e-01
34842 -0.097190 -8.956091e-03   1.198066 -7.325522e-01 -9.534212e-01
34843 -0.089953  1.128113e-16  -0.646922 -5.652939e-01 -1.143516e+00
34844  0.000000  1.128113e-16  -0.005187  2.688782e-13  3.231991e-12
34845 -0.113834  3.046102e-01   1.759584 -5.492745e-01 -4.625506e-02
34846 -0.074394  1.128113e-16  -0.005187  2.533147e+00  3.743031e-01
34847 -0.076203 -1.677238e-01   2.040343  2.412181e+00  3.568667e-01
34848 -0.079098  1.128113e-16  -0.005187  2.488999e+00  3.057894e-01
34849 -0.068604 -8.956091e-03   1.879909  2.526840e+00  3.039890e-01
34850 -0.117815 -1.002476e-01   2.040343  2.518893e+00  3.881385e-01
34851 -0.110578  1.128113e-16  -0.005187 -3.982215e-03 -1.105800e+00
34852 -0.000217  1.128113e-16  -0.005187  1.315558e-03 -1.110443e+00
34853 -0.179327 -2.232926e-01   2.120560 -6.150090e-02 -1.076424e+00
34854 -0.135183 -1.597855e-01   1.398608 -1.542119e-01 -1.168344e+00
34855  0.000000  1.128113e-16  -0.005187  2.688782e-13  3.231991e-12
34856 -0.124328 -2.272618e-01  -1.408983 -9.417050e-02 -1.026673e+00

[34857 rows x 11 columns]
```

Dummy encoding :

Identify the number of dummy variables to be created for the variable steel. Submit the Python script with Outputs in a document

[59]:
```
full1.Suburb.value_counts()
```

[59]:
```
Reservoir              844
Bentleigh East         583
Richmond               552
Glen Iris              491
Preston                485
Kew                    467
Brighton               456
Brunswick              444
South Yarra            435
Hawthorn               428
Northcote              424
Camberwell             423
Balwyn North           420
Essendon               409
Coburg                 405
Glenroy                400
Brighton East          393
```

```
Pascoe Vale            378
St Kilda               374
Port Melbourne         371
Malvern East           369
Prahran                336
Thornbury              322
Balwyn                 319
Bentleigh              319
Yarraville             304
Surrey Hills           293
Elwood                 288
Moonee Ponds           285
Hawthorn East          284
                      …
Upwey                    2
Silvan                   2
Wattle Glen              2
Darley                   2
Beaconsfield Upper       2
Montrose                 2
Gisborne South           2
Lynbrook                 2
Hurstbridge              2
Healesville              2
Wandin North             1
Yarra Glen               1
Eynesbury                1
Fawkner Lot              1
Wildwood                 1
Ferny Creek              1
Guys Hill                1
Cranbourne East          1
Bulla                    1
Hopetoun Park            1
Olinda                   1
croydon                  1
Kalkallo                 1
Botanic Ridge            1
Menzies Creek            1
Avonsleigh               1
Monbulk                  1
Belgrave                 1
Coldstream               1
viewbank                 1
Name: Suburb, Length: 351, dtype: int64
```