MelbourneHousingDataAnalysis

May 25, 2023

[97]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sbn
from sklearn import linear_model
from sklearn.model_selection import train_test_split # Sklearn package's␣
 ↪randomized data splitting function
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import NearestNeighbors, KNeighborsClassifier,␣
 ↪KNeighborsRegressor
```

[98]:
```python
import warnings
warnings.filterwarnings('ignore')
```

[39]:
```python
#import housing dataset and display first five rows
mhDataset = pd.read_csv("Melbourne_housing_FULL.csv")
mhDataset.head()
```

[39]:
```
       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  BuildingArea  \
0  3/09/2016       2.5    3067.0  …       1.0  1.0     126.0           NaN
1  3/12/2016       2.5    3067.0  …       1.0  1.0     202.0           NaN
2  4/02/2016       2.5    3067.0  …       1.0  0.0     156.0          79.0
3  4/02/2016       2.5    3067.0  …       2.0  1.0       0.0           NaN
4  4/03/2017       2.5    3067.0  …       2.0  0.0     134.0         150.0

   YearBuilt         CouncilArea Lattitude  Longtitude             Regionname  \
0        NaN  Yarra City Council  -37.8014    144.9958  Northern Metropolitan
1        NaN  Yarra City Council  -37.7996    144.9984  Northern Metropolitan
2     1900.0  Yarra City Council  -37.8079    144.9934  Northern Metropolitan
3        NaN  Yarra City Council  -37.8114    145.0116  Northern Metropolitan
4     1900.0  Yarra City Council  -37.8093    144.9944  Northern Metropolitan
```

```
  Propertycount
0        4019.0
1        4019.0
2        4019.0
3        4019.0
4        4019.0

[5 rows x 21 columns]
```

[4]:
```
mhDataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34857 entries, 0 to 34856
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         34857 non-null  object
 1   Address        34857 non-null  object
 2   Rooms          34857 non-null  int64
 3   Type           34857 non-null  object
 4   Price          27247 non-null  float64
 5   Method         34857 non-null  object
 6   SellerG        34857 non-null  object
 7   Date           34857 non-null  object
 8   Distance       34856 non-null  float64
 9   Postcode       34856 non-null  float64
 10  Bedroom2       26640 non-null  float64
 11  Bathroom       26631 non-null  float64
 12  Car            26129 non-null  float64
 13  Landsize       23047 non-null  float64
 14  BuildingArea   13742 non-null  float64
 15  YearBuilt      15551 non-null  float64
 16  CouncilArea    34854 non-null  object
 17  Lattitude      26881 non-null  float64
 18  Longtitude     26881 non-null  float64
 19  Regionname     34854 non-null  object
 20  Propertycount  34854 non-null  float64
dtypes: float64(12), int64(1), object(8)
memory usage: 5.6+ MB
```

[5]:
```
#Since the Date, Address, Postcode, Lattitude, Longitude,YearBuilt isnt going␣
 ↪to contribute much for our prediction, let us
#take only the required columns.
cols_to_use =␣
 ↪['Suburb','Rooms','Type','Price','Method','SellerG','Distance','Bedroom2','Bathroom','Car',
 ↪'CouncilArea', 'Regionname', 'Propertycount']
```

```
mhDataset = mhDataset[cols_to_use]
mhDataset.head()
```

```
[5]:        Suburb  Rooms Type      Price Method SellerG  Distance  Bedroom2  \
0  Abbotsford      2    h        NaN     SS  Jellis       2.5       2.0
1  Abbotsford      2    h  1480000.0      S  Biggin       2.5       2.0
2  Abbotsford      2    h  1035000.0      S  Biggin       2.5       2.0
3  Abbotsford      3    u        NaN     VB  Rounds       2.5       3.0
4  Abbotsford      3    h  1465000.0     SP  Biggin       2.5       3.0

   Bathroom  Car  Landsize  BuildingArea         CouncilArea  \
0       1.0  1.0     126.0           NaN  Yarra City Council
1       1.0  1.0     202.0           NaN  Yarra City Council
2       1.0  0.0     156.0          79.0  Yarra City Council
3       2.0  1.0       0.0           NaN  Yarra City Council
4       2.0  0.0     134.0         150.0  Yarra City Council

              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
```

In the previous step, we have filtered the columns required for our analysis.Below given is the type of data each column has.

Type of Data present in the columns

Method: S - property sold; SP - property sold prior; PI - property passed in; PN - sold prior not disclosed; SN - sold not disclosed; NB - no bid; VB - vendor bid; W - withdrawn prior to auction; SA - sold after auction; SS - sold after auction price not disclosed. N/A - price or highest bid not available.

Type: br - bedroom(s); h - house,cottage,villa, semi,terrace; u - unit, duplex; t - townhouse; dev site - development site; o res - other residential.

SellerG: Real Estate Agent

Date: Date sold

Distance: Distance from CBD in Kilometres

Regionname: General Region (West, North West, North, North east …etc)

Propertycount: Number of properties that exist in the suburb.

Bedroom2 : Scraped # of Bedrooms (from different source)

Bathroom: Number of Bathrooms

Car: Number of carspots

Landsize: Land Size in Metres

BuildingArea: Building Size in Metres

YearBuilt: Year the house was built

CouncilArea: Governing council for the area

Lattitude: Self explanitory

Longtitude: Self explanitory

Now for the Type and Method columns, we will replace the single variable entry with a meaningful value for better understanding

[6]:
```
mhDataset['Type'] = mhDataset['Type'].replace({'h': 'House/Villa', 'u': 'Unit/
↪Duplex', 't': 'TownHouse'})
mhDataset.head(10)


mhDataset['Method'] = mhDataset['Method'].replace({'SS':'Sold after auction␣
↪price not disclosed',
                                                    'S':'Property Sold',
                                                    'VB':'Vendor Bid',
                                                    'SP':'Property Sold Prior',
                                                    'PI':'Property passed in',
                                                    'SN':'Sold not disclosed',
                                                    'W':'Withdrawn Prior',
                                                    'PN':'Sold prior not␣
↪disclosed',
                                                    'SA':'Sold after auction'})
```

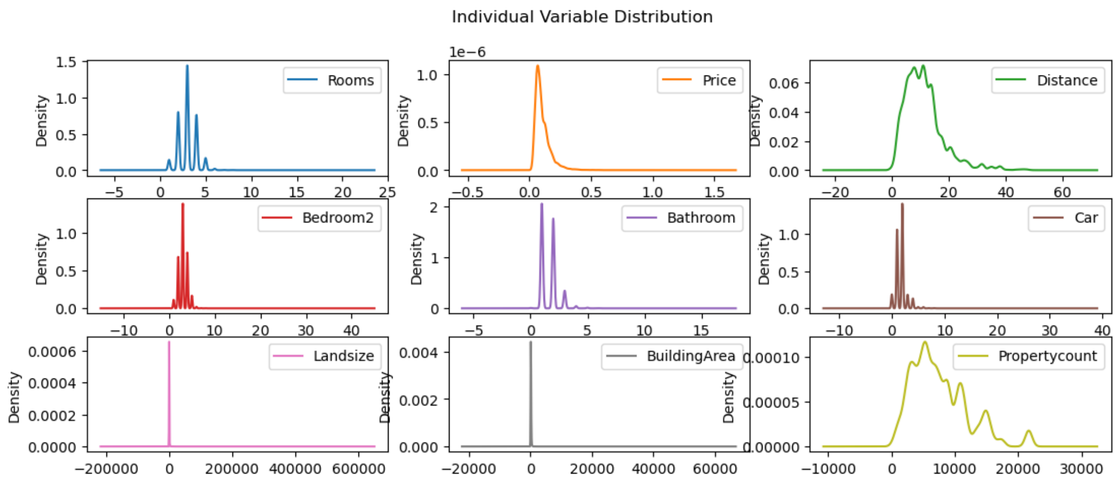
Data Visualization

Data visualization, in other words we can say it as exploratory data analysis, is the process of visualizing the pattern of individual data, relationships between two or more variable to come up with an overall idea of how the data is distributed , related and what kind of future prediction we could do with the given dataset.

Univariate Plots

This kind of plot summarizes only one value at a time and is basically used to check if the variable distribution is normal or skewed

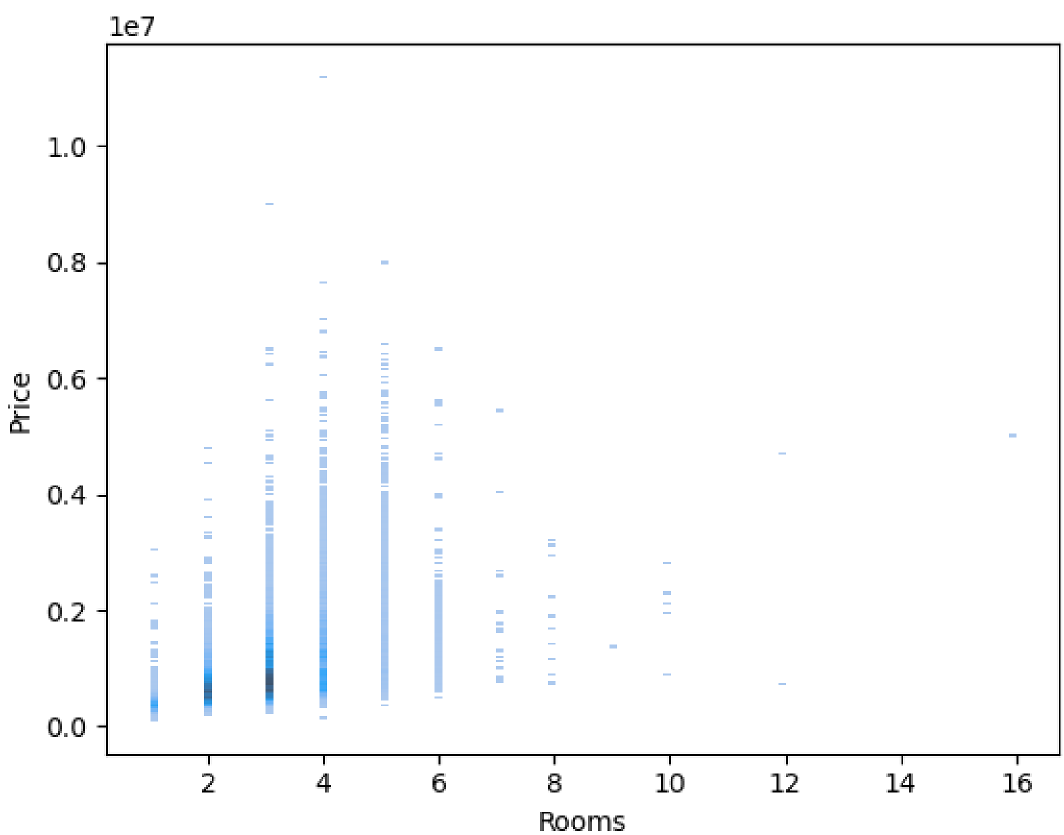
[13]:
```
mhDataset.plot.density(subplots=True, layout=(3,3), sharex=False,␣
↪sharey=False,figsize =(13,5),title="Individual Variable Distribution")
plt.show()
```



Bivariate Plots

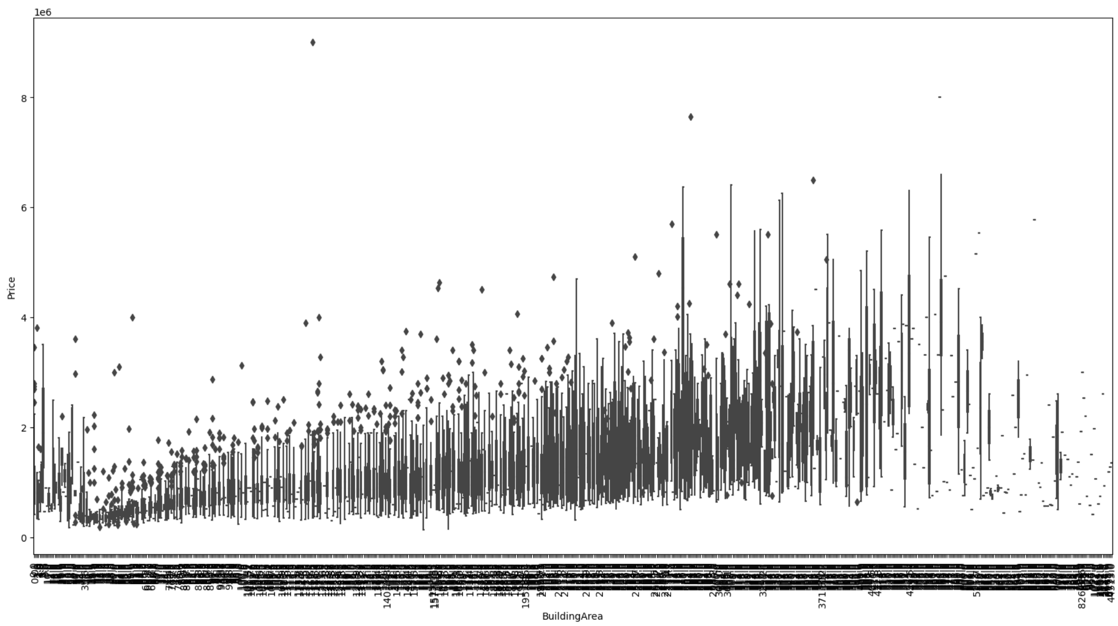
This kind of plotting is used to compare two variables to identify the relationship of those variables.

[18]:
```
sbn.histplot(mhDataset.dropna(), x=mhDataset['Rooms'],y=mhDataset['Price'])

plt.show()
```



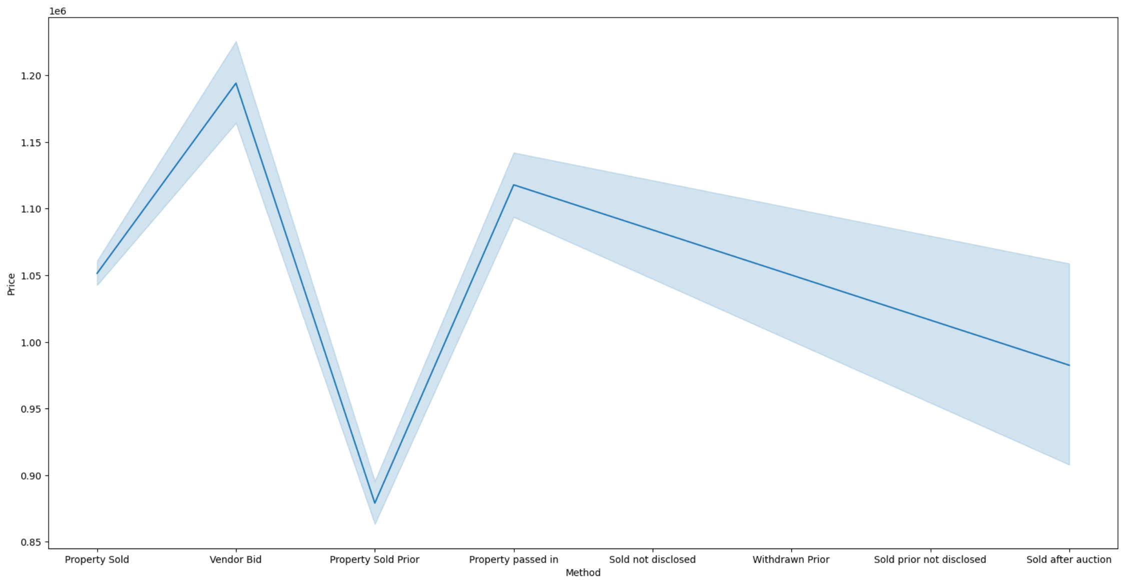
In the above graph, the data distributuion is at its peak when the number of rooms is a house is 4.

[46]:
```
fig, ax = plt.subplots(figsize=(20, 10))
sbn.boxplot(x=mhDataset['BuildingArea'],y=mhDataset['Price'])
plt.xticks(rotation=90)
plt.show()
```



As per the above graph, Building area and Price have a positive correlation. Price increases as the building area increases.

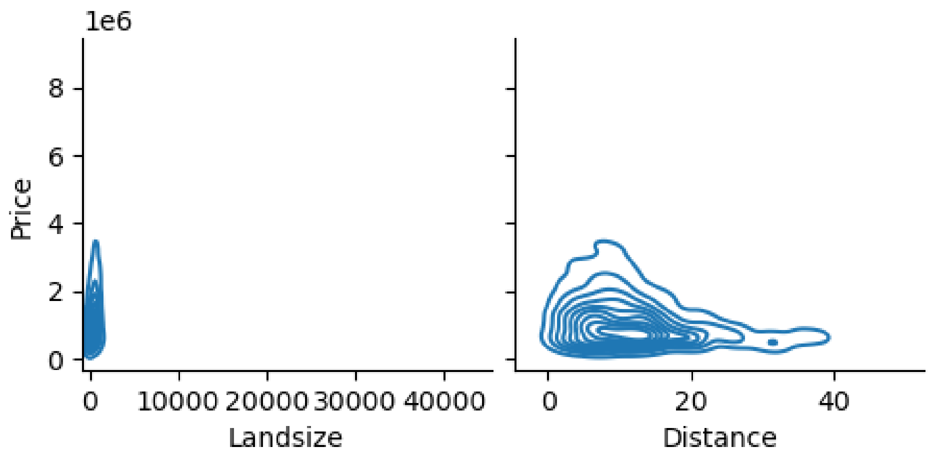
[78]:
```
fig, ax = plt.subplots(figsize=(20, 10))
sbn.lineplot(x=mhDataset['Method'],y=mhDataset['Price'])

plt.show()
```



In the above given graph we could visualize that the Price touched it low point with properties sold prior and at its peak for "Vendor Bid" category

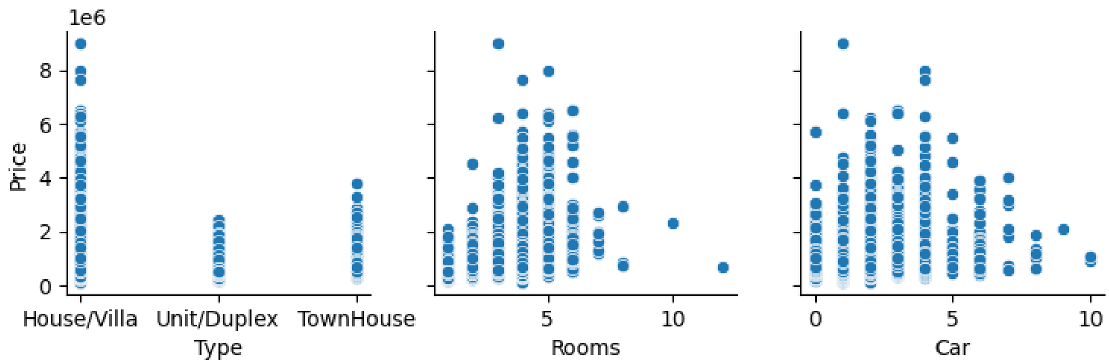
Multiple Pairwise Bivariate Plots

[22]:
```
sbn.pairplot(mhDataset.
  ↪dropna(),x_vars=['Landsize','Distance'],y_vars=['Price'],kind='kde')
plt.show()
```



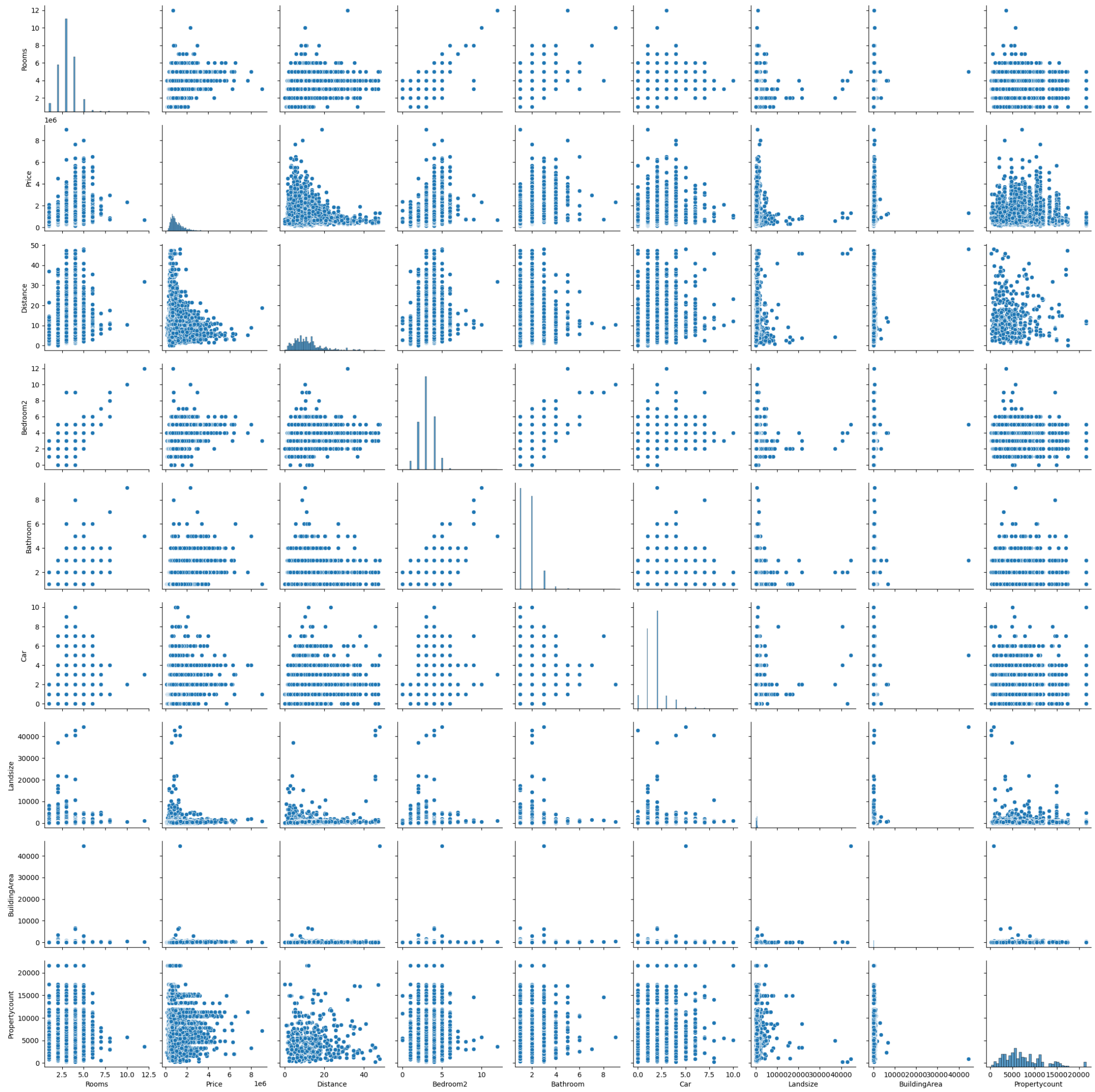
With the above graph, 1. We couldn't derive any relation between landsize and Price as all the parameter sticks very close to x=0 range. 2. With respect to distance, as it is in closer proximity, the price is higher.

[47]:
```
#Relationship between House type,car spots and price
sbn.pairplot(mhDataset.
 ↪dropna(),x_vars=['Type','Rooms','Car'],y_vars=['Price'],kind='scatter')
plt.show()
```



1. The above graph shows that the price of house/villa type property is comparatively higher than unit/Duplex or TownHouse Properties.
2. More number of Houses are available with >=1 rooms <= 5. The distribution is relatively higher in that range. And discarding the outliers, price of the houses with 5 bedroom is comparatively higher.
3. Prices of the houses with 3 or 5 car spots are higher and almost all the houses have car spots.

[70]:
```
sbn.pairplot(mhDataset.dropna())
plt.show()
```



Corelation Heatmap - a part of MultiVariate Plots

Let us see how the dependent variable and independent variables are correlated. Higher posiive correlation value shows strong correlation.

[12]:
```
correlation_matrix = mhDataset.corr()
correlation_matrix
```

[12]:

	Rooms	Price	Distance	Bedroom2	Bathroom	Car	\
Rooms	1.000000	0.465238	0.271511	0.946755	0.611826	0.393878	
Price	0.465238	1.000000	-0.211384	0.430275	0.429878	0.201803	
Distance	0.271511	-0.211384	1.000000	0.269524	0.126201	0.241835	
Bedroom2	0.946755	0.430275	0.269524	1.000000	0.614892	0.388491	
Bathroom	0.611826	0.429878	0.126201	0.614892	1.000000	0.307518	

```
Car           0.393878  0.201803  0.241835  0.388491  0.307518  1.000000
Landsize      0.037402  0.032748  0.060862  0.037019  0.036333  0.037829
BuildingArea  0.156229  0.100754  0.076301  0.154157  0.147558  0.104373
Propertycount -0.071677 -0.059017 -0.018140 -0.053451 -0.032887 -0.009617

              Landsize  BuildingArea  Propertycount
Rooms         0.037402      0.156229      -0.071677
Price         0.032748      0.100754      -0.059017
Distance      0.060862      0.076301      -0.018140
Bedroom2      0.037019      0.154157      -0.053451
Bathroom      0.036333      0.147558      -0.032887
Car           0.037829      0.104373      -0.009617
Landsize      1.000000      0.354530      -0.018195
BuildingArea  0.354530      1.000000      -0.024523
Propertycount -0.018195     -0.024523       1.000000
```

[13]:

```
correlation_matrix["Price"]
```

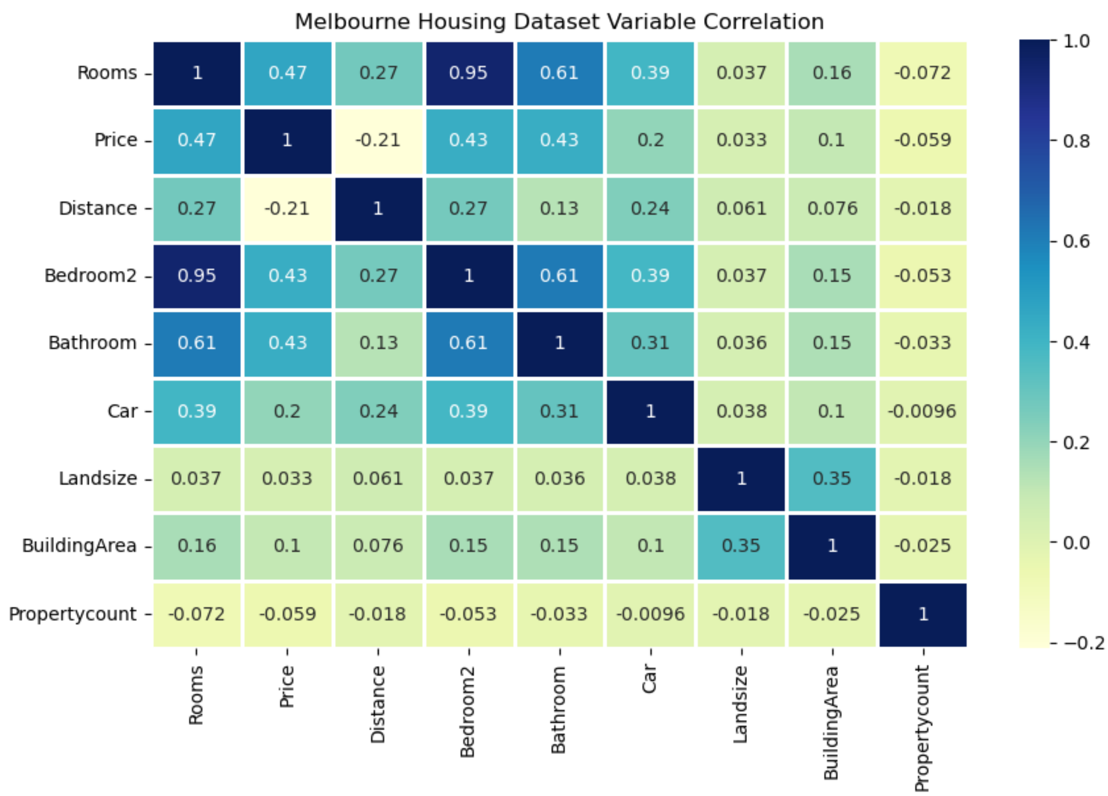
[13]:

```
Rooms            0.465238
Price            1.000000
Distance        -0.211384
Bedroom2         0.430275
Bathroom         0.429878
Car              0.201803
Landsize         0.032748
BuildingArea     0.100754
Propertycount   -0.059017
Name: Price, dtype: float64
```

When we look at the above data, we could see that except Distance and Propery Count, "Price" variable is positively correlated with other independent variables. Now we will see the correlation Heatmap.

[15]:

```
plt.figure(figsize = (10,6))
sbn.heatmap(correlation_matrix, cmap = 'YlGnBu', linewidth = 1, annot = True)
plt.title('Melbourne Housing Dataset Variable Correlation')
plt.show()
```



[16]:
```
mhDataset.isna().sum()
```

[16]:
```
Suburb               0
Rooms                0
Type                 0
Price             7610
Method               0
SellerG              0
Distance             1
Bedroom2          8217
Bathroom          8226
Car               8728
Landsize         11810
BuildingArea     21115
CouncilArea          3
Regionname           3
Propertycount        3
dtype: int64
```

[17]:
```
mhDataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 34857 entries, 0 to 34856
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         34857 non-null  object
 1   Rooms          34857 non-null  int64
 2   Type           34857 non-null  object
 3   Price          27247 non-null  float64
 4   Method         34857 non-null  object
 5   SellerG        34857 non-null  object
 6   Distance       34856 non-null  float64
 7   Bedroom2       26640 non-null  float64
 8   Bathroom       26631 non-null  float64
 9   Car            26129 non-null  float64
 10  Landsize       23047 non-null  float64
 11  BuildingArea   13742 non-null  float64
 12  CouncilArea    34854 non-null  object
 13  Regionname     34854 non-null  object
 14  Propertycount  34854 non-null  float64
dtypes: float64(8), int64(1), object(6)
memory usage: 4.0+ MB
```

[88]:
```
#Now that need to replace null values with 0 for numerical columns.
cols_to_be_replaced_with_Zero =␣
↪['Distance','Bedroom2','Bathroom','Car','Propertycount']
mhDataset[cols_to_be_replaced_with_Zero] =␣
↪mhDataset[cols_to_be_replaced_with_Zero].fillna(0)
```

[89]:
```
mhDataset.isna().sum()
```

[89]:
```
Suburb               0
Rooms                0
Type                 0
Price             7610
Method               0
SellerG              0
Distance             0
Bedroom2             0
Bathroom             0
Car                  0
Landsize         11810
BuildingArea     21115
CouncilArea          3
Regionname           3
Propertycount        0
dtype: int64
```

[90]:
```
#For Landsize and Building area, filling null values with the "mean" of the␣
↪respectiove fields would be appropriate.
mhDataset['Landsize'] = mhDataset['Landsize'].fillna(mhDataset['Landsize'].
↪mean())
mhDataset['BuildingArea'] = mhDataset['BuildingArea'].
↪fillna(mhDataset['BuildingArea'].mean())
```

[91]:
```
mhDataset.isna().sum()
```

[91]:
```
Suburb              0
Rooms               0
Type                0
Price            7610
Method              0
SellerG             0
Distance            0
Bedroom2            0
Bathroom            0
Car                 0
Landsize            0
BuildingArea        0
CouncilArea         3
Regionname          3
Propertycount       0
dtype: int64
```

Now that Data is cleaned to an extend, but we still have null values. Since this is a voluminous dataset, we will be dropping the left null values,however this wouldn't be an issue, but would contribute to the accuracy of our prediction.

[93]:
```
mhDataset.dropna(inplace=True)
```

[94]:
```
mhDataset.isna().sum()
```

[94]:
```
Suburb           0
Rooms            0
Type             0
Price            0
Method           0
SellerG          0
Distance         0
Bedroom2         0
Bathroom         0
Car              0
Landsize         0
BuildingArea     0
CouncilArea      0
Regionname       0
```

```
Propertycount    0
dtype: int64
```

[95]:
```
#Now its time to format the data in a more presentable way. i.e to change the
↪categorical data into a numerical one by doing
#OneHotEncoding.

mhDataset = pd.get_dummies(mhDataset,drop_first=True)
```

[96]:
```
mhDataset.head()
```

[96]:
```
   Rooms      Price  Distance  Bedroom2  Bathroom  Car  Landsize  \
1      2  1480000.0       2.5       2.0       1.0  1.0     202.0
2      2  1035000.0       2.5       2.0       1.0  0.0     156.0
4      3  1465000.0       2.5       3.0       2.0  0.0     134.0
5      3   850000.0       2.5       3.0       2.0  1.0      94.0
6      4  1600000.0       2.5       3.0       1.0  2.0     120.0

   BuildingArea  Propertycount  Suburb_Aberfeldie  …  \
1      160.2564         4019.0                  0  …
2       79.0000         4019.0                  0  …
4      150.0000         4019.0                  0  …
5      160.2564         4019.0                  0  …
6      142.0000         4019.0                  0  …

   CouncilArea_Wyndham City Council  CouncilArea_Yarra City Council  \
1                                 0                               1
2                                 0                               1
4                                 0                               1
5                                 0                               1
6                                 0                               1

   CouncilArea_Yarra Ranges Shire Council  Regionname_Eastern Victoria  \
1                                       0                            0
2                                       0                            0
4                                       0                            0
5                                       0                            0
6                                       0                            0

   Regionname_Northern Metropolitan  Regionname_Northern Victoria  \
1                                 1                             0
2                                 1                             0
4                                 1                             0
5                                 1                             0
6                                 1                             0

   Regionname_South-Eastern Metropolitan  Regionname_Southern Metropolitan  \
```

```
1                                                      0                                  0
2                                                      0                                  0
4                                                      0                                  0
5                                                      0                                  0
6                                                      0                                  0

   Regionname_Western Metropolitan  Regionname_Western Victoria
1                                0                            0
2                                0                            0
4                                0                            0
5                                0                            0
6                                0                            0

[5 rows x 745 columns]
```

[97]:
```
x= mhDataset.drop('Price',axis=1)
y= mhDataset['Price']
```

[98]:
```
#Split Training and Testing data in the ratio of 70:30
# Split X and y into training and test set in 70:30 ratio

X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.30,␣
↪random_state=2)
```

We would try to find out which regression model suits best for this particular data set. Will first start with Linear Regression

Linear Regression

[27]:
```
#process to fit
linearReg = LinearRegression()
linearReg.fit(X_train,Y_train)
```

[27]:
```
LinearRegression()
```

[28]:
```
#Find out the R^2 value for Training Data
linearReg.score(X_train,Y_train)
```

[28]:
```
0.6827792395792723
```

[29]:
```
# Find out the R^2 value for Testing Data
linearReg.score(X_test,Y_test)
```

[29]:
```
0.138536831616211
```

Looking at the R^2 value of both Training and Testing data, Linear regression doesn't seem to fit well for this dataset. Since Testing score is way less than Training score, we could conclude that this is Overfitting and would require Normalisation

Lasso Regression

[46]:
```
lassoReg = linear_model.Lasso(alpha=100,max_iter=100,tol=1)
lassoReg.fit(X_train,Y_train)
```

[46]:
```
Lasso(alpha=100, max_iter=100, tol=1)
```

[47]:
```
# Find out the R^2 value for Training Data
lassoReg.score(X_train,Y_train)
```

[47]:
```
0.6594868393102264
```

[48]:
```
# Find out the R^2 value for Testing Data
lassoReg.score(X_test,Y_test)
```

[48]:
```
0.6537440212349376
```

Ridge Regression

[40]:
```
ridgeReg = linear_model.Ridge(alpha=100,max_iter=999,tol=1)
ridgeReg.fit(X_train,Y_train)
```

[40]:
```
Ridge(alpha=100, max_iter=999, tol=1)
```

[41]:
```
ridgeReg.score(X_train,Y_train)
```

[41]:
```
0.6518343038773815
```

[42]:
```
ridgeReg.score(X_test,Y_test)
```

[42]:
```
0.6587011138097523
```

In both Lasso and Ridge regression, the scores of both Testing and Training data seems to be equal.

So for this Dataset, Lasso or Ridge regression will be most preferred regression method.

KNN Regression

For KNN regression, we will take a similar dataset.The steps involved in KNN regression are as follows. 1. Load data and assign data and target values. 2. Find the nearest neighbors 3. Do cross value prediction 4. Check the mean squared error value to see if the majority voted goes well with the data and in turn gives less error. 5. Find the R2 score, to see if the model will do well for unknown data.

[103]:
```
#import dataset

#import housing dataset and display first five rows
mhsDataset = pd.read_csv("Melbourne_housing_FULL.csv")
mhsDataset.head()
```

[103]:
```
       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  BuildingArea  \
0  3/09/2016       2.5    3067.0  …       1.0  1.0     126.0           NaN
1  3/12/2016       2.5    3067.0  …       1.0  1.0     202.0           NaN
2  4/02/2016       2.5    3067.0  …       1.0  0.0     156.0          79.0
3  4/02/2016       2.5    3067.0  …       2.0  1.0       0.0           NaN
4  4/03/2017       2.5    3067.0  …       2.0  0.0     134.0         150.0

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

  Propertycount
0        4019.0
1        4019.0
2        4019.0
3        4019.0
4        4019.0

[5 rows x 21 columns]
```

[104]:
```
mhsDataset.isna().sum()/len(mhsDataset)*100
```

[104]:
```
Suburb            0.000000
Address           0.000000
Rooms             0.000000
Type              0.000000
Price            21.832057
Method            0.000000
SellerG           0.000000
Date              0.000000
Distance          0.002869
Postcode          0.002869
Bedroom2         23.573457
Bathroom         23.599277
Car              25.039447
Landsize         33.881286
BuildingArea     60.576068
YearBuilt        55.386293
CouncilArea       0.008607
Lattitude        22.882061
```

```
Longtitude       22.882061
Regionname        0.008607
Propertycount     0.008607
dtype: float64
```

In this dataset, there is more number of null values, so we will first see the null values percentage.

[105]:
```
mhsDataset = mhsDataset.dropna()
mhsDataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8887 entries, 2 to 34856
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         8887 non-null   object
 1   Address        8887 non-null   object
 2   Rooms          8887 non-null   int64
 3   Type           8887 non-null   object
 4   Price          8887 non-null   float64
 5   Method         8887 non-null   object
 6   SellerG        8887 non-null   object
 7   Date           8887 non-null   object
 8   Distance       8887 non-null   float64
 9   Postcode       8887 non-null   float64
 10  Bedroom2       8887 non-null   float64
 11  Bathroom       8887 non-null   float64
 12  Car            8887 non-null   float64
 13  Landsize       8887 non-null   float64
 14  BuildingArea   8887 non-null   float64
 15  YearBuilt      8887 non-null   float64
 16  CouncilArea    8887 non-null   object
 17  Lattitude      8887 non-null   float64
 18  Longtitude     8887 non-null   float64
 19  Regionname     8887 non-null   object
 20  Propertycount  8887 non-null   float64
dtypes: float64(12), int64(1), object(8)
memory usage: 1.5+ MB
```

[72]:
```
mhsDataset['Method'].unique()
```

[72]:
```
array(['SS', 'S', 'VB', 'SP', 'PI', 'SN', 'W', 'PN', 'SA'], dtype=object)
```

[73]:
```
mhsDataset['Type'].unique()
```

[73]:
```
array(['h', 'u', 't'], dtype=object)
```

[106]:
```
mhsDataset['Type'] = mhsDataset['Type'].replace({'h': 'House/Villa', 'u': 'Unit/
↪Duplex', 't': 'TownHouse'})
mhsDataset.head(10)


mhsDataset['Method'] = mhsDataset['Method'].replace({'SS':'Sold after auction␣
↪price not disclosed',
                                                    'S':'Property Sold',
                                                    'VB':'Vendor Bid',
                                                    'SP':'Property Sold Prior',
                                                    'PI':'Property passed in',
                                                    'SN':'Sold not disclosed',
                                                    'W':'Withdrawn Prior',
                                                    'PN':'Sold prior not␣
↪disclosed',
                                                    'SA':'Sold after auction'})
```

[107]:
```
cols_to_use =␣
↪['Suburb','Rooms','Type','Price','Method','SellerG','Distance','Bedroom2','Bathroom','Car',
↪'CouncilArea', 'Regionname', 'Propertycount']
mhsDataset = mhsDataset[cols_to_use]
mhsDataset.head()
```

[107]:
```
        Suburb  Rooms         Type      Price               Method SellerG  \
2   Abbotsford      2  House/Villa  1035000.0        Property Sold  Biggin
4   Abbotsford      3  House/Villa  1465000.0  Property Sold Prior  Biggin
6   Abbotsford      4  House/Villa  1600000.0           Vendor Bid  Nelson
11  Abbotsford      3  House/Villa  1876000.0        Property Sold  Nelson
14  Abbotsford      2  House/Villa  1636000.0        Property Sold  Nelson

    Distance  Bedroom2  Bathroom  Car  Landsize  BuildingArea  \
2        2.5       2.0       1.0  0.0     156.0          79.0
4        2.5       3.0       2.0  0.0     134.0         150.0
6        2.5       3.0       1.0  2.0     120.0         142.0
11       2.5       4.0       2.0  0.0     245.0         210.0
14       2.5       2.0       1.0  2.0     256.0         107.0

           CouncilArea             Regionname  Propertycount
2   Yarra City Council  Northern Metropolitan         4019.0
4   Yarra City Council  Northern Metropolitan         4019.0
6   Yarra City Council  Northern Metropolitan         4019.0
11  Yarra City Council  Northern Metropolitan         4019.0
14  Yarra City Council  Northern Metropolitan         4019.0
```

[108]:
```

```

[108]:

```
Suburb           0
Rooms            0
Type             0
Price            0
Method           0
SellerG          0
Distance         0
Bedroom2         0
Bathroom         0
Car              0
Landsize         0
BuildingArea     0
CouncilArea      0
Regionname       0
Propertycount    0
dtype: int64
```

[109]:

```
mhsDataset=mhsDataset.dropna()
mhsDataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8887 entries, 2 to 34856
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         8887 non-null   object
 1   Rooms          8887 non-null   int64
 2   Type           8887 non-null   object
 3   Price          8887 non-null   float64
 4   Method         8887 non-null   object
 5   SellerG        8887 non-null   object
 6   Distance       8887 non-null   float64
 7   Bedroom2       8887 non-null   float64
 8   Bathroom       8887 non-null   float64
 9   Car            8887 non-null   float64
 10  Landsize       8887 non-null   float64
 11  BuildingArea   8887 non-null   float64
 12  CouncilArea    8887 non-null   object
 13  Regionname     8887 non-null   object
 14  Propertycount  8887 non-null   float64
dtypes: float64(8), int64(1), object(6)
memory usage: 1.1+ MB
```

[110]:

```
mhsDataset.isna().sum()
```

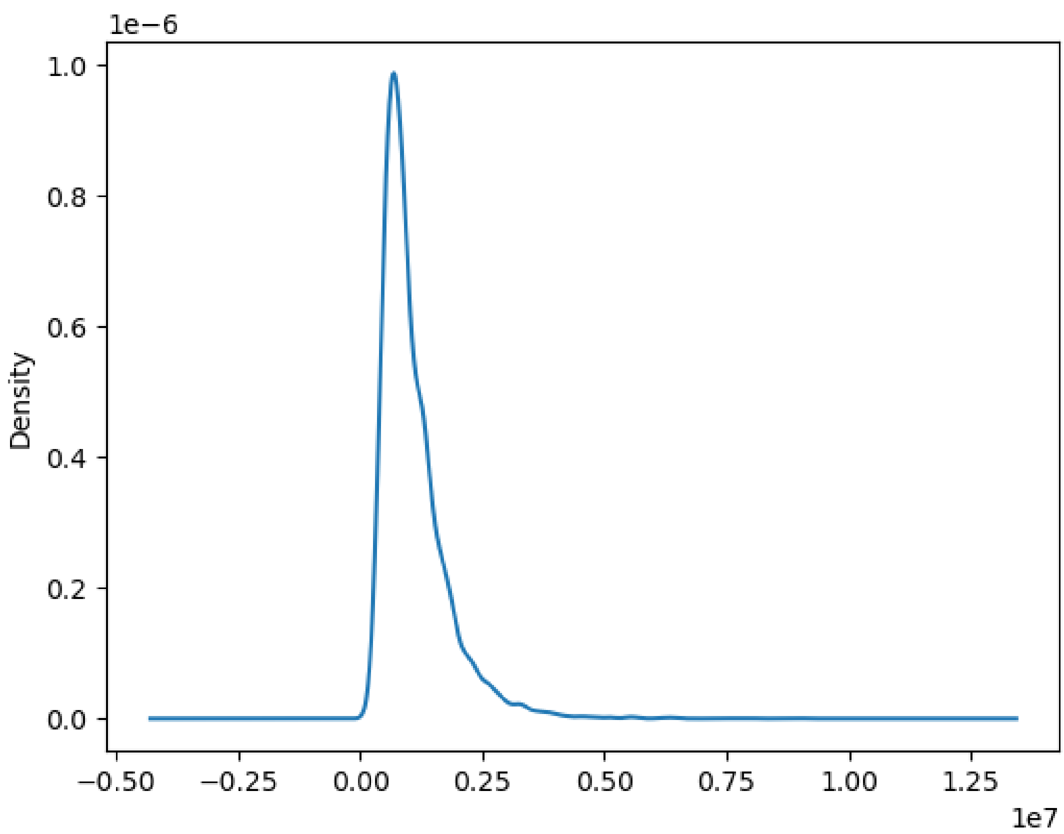
[110]:

```
Suburb           0
Rooms            0
Type             0
```

```
Price            0
Method           0
SellerG          0
Distance         0
Bedroom2         0
Bathroom         0
Car              0
Landsize         0
BuildingArea     0
CouncilArea      0
Regionname       0
Propertycount    0
dtype: int64
```

[111]:
```
mhsDataset['Price'].plot(kind='kde')
```

[111]:
```
<AxesSubplot:ylabel='Density'>
```



[112]:
```
q1,q3 = mhsDataset['Price'].quantile([0.25,0.75])
iqr = q3-q1
```

```
print(iqr)
upperLimit = q3+1.5*iqr
lowerLimit = q1-1.5*iqr
print(lowerLimit, upperLimit)
cond1 = mhsDataset.Price > lowerLimit
cond2 = mhsDataset.Price <= upperLimit
mhsDataset = mhsDataset.where(cond1 & cond2)
mhsDataset.head()
```

```
704000.0
-415000.0 2401000.0
```

[112]:

```
        Suburb  Rooms         Type      Price              Method SellerG  \
2   Abbotsford    2.0  House/Villa  1035000.0       Property Sold  Biggin
4   Abbotsford    3.0  House/Villa  1465000.0 Property Sold Prior  Biggin
6   Abbotsford    4.0  House/Villa  1600000.0          Vendor Bid  Nelson
11  Abbotsford    3.0  House/Villa  1876000.0       Property Sold  Nelson
14  Abbotsford    2.0  House/Villa  1636000.0       Property Sold  Nelson

    Distance  Bedroom2  Bathroom  Car  Landsize  BuildingArea  \
2        2.5       2.0       1.0  0.0     156.0          79.0
4        2.5       3.0       2.0  0.0     134.0         150.0
6        2.5       3.0       1.0  2.0     120.0         142.0
11       2.5       4.0       2.0  0.0     245.0         210.0
14       2.5       2.0       1.0  2.0     256.0         107.0

          CouncilArea             Regionname  Propertycount
2   Yarra City Council  Northern Metropolitan         4019.0
4   Yarra City Council  Northern Metropolitan         4019.0
6   Yarra City Council  Northern Metropolitan         4019.0
11  Yarra City Council  Northern Metropolitan         4019.0
14  Yarra City Council  Northern Metropolitan         4019.0
```

[113]:

```
mhsDataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8887 entries, 2 to 34856
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         8467 non-null   object
 1   Rooms          8467 non-null   float64
 2   Type           8467 non-null   object
 3   Price          8467 non-null   float64
 4   Method         8467 non-null   object
 5   SellerG        8467 non-null   object
 6   Distance       8467 non-null   float64
 7   Bedroom2       8467 non-null   float64
```

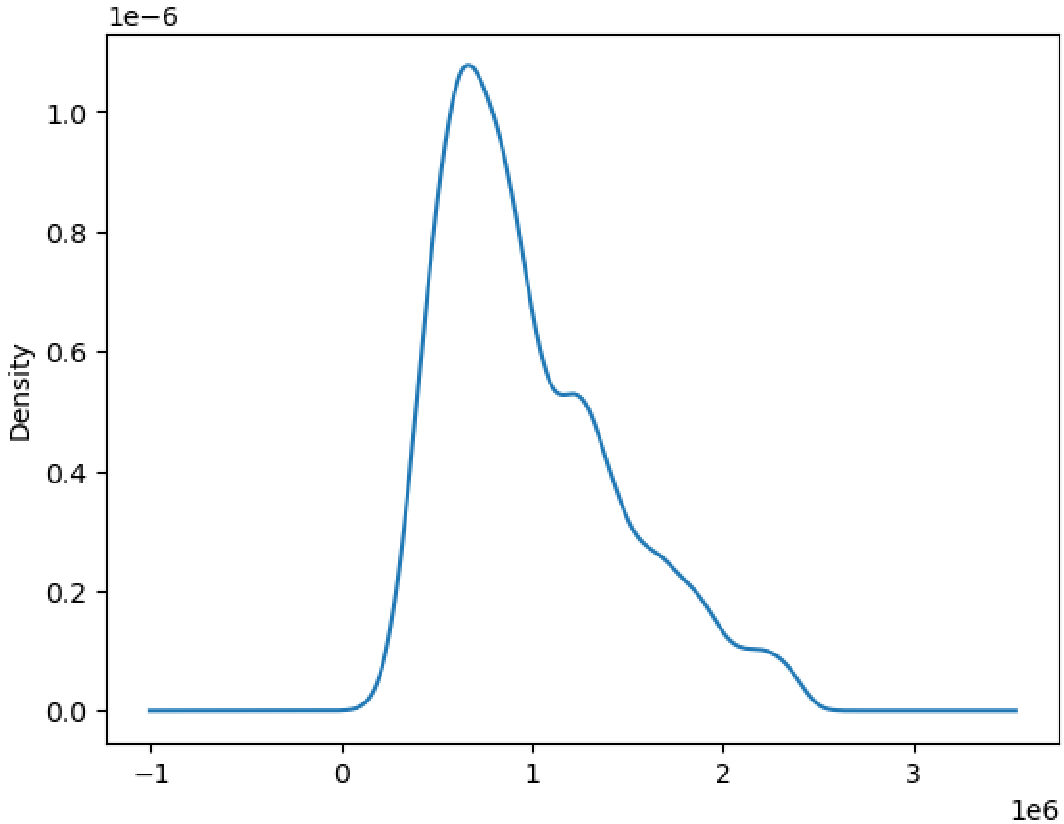
```
 8   Bathroom       8467 non-null   float64
 9   Car            8467 non-null   float64
 10  Landsize       8467 non-null   float64
 11  BuildingArea   8467 non-null   float64
 12  CouncilArea    8467 non-null   object
 13  Regionname     8467 non-null   object
 14  Propertycount  8467 non-null   float64
dtypes: float64(9), object(6)
memory usage: 1.1+ MB
```

[114]:
```
mhsDataset.describe().T
```

[114]:
```
               count           mean            std       min       25%  \
Rooms         8467.0       3.041927       0.935437       1.0       2.0
Price         8467.0  989507.486477  469584.122354  131000.0  630000.0
Distance      8467.0      11.359525       6.902434       0.0       6.5
Bedroom2      8467.0       3.022912       0.939560       0.0       2.0
Bathroom      8467.0       1.595370       0.669590       1.0       1.0
Car           8467.0       1.662690       0.958422       0.0       1.0
Landsize      8467.0     511.609425    1079.704586       0.0     203.0
BuildingArea  8467.0     142.478664      75.108814       0.0      98.0
Propertycount 8467.0    7475.671430    4425.459992     249.0    4294.0

                    50%        75%        max
Rooms               3.0        4.0       12.0
Price          870000.0  1275000.0  2400000.0
Distance           10.4       14.0       47.4
Bedroom2            3.0        4.0       12.0
Bathroom            2.0        2.0        9.0
Car                 2.0        2.0       10.0
Landsize          460.0      645.0    42800.0
BuildingArea      130.0      173.0     1561.0
Propertycount    6543.0    10331.0    21650.0
```

[115]:
```
mhsDataset['Price'].plot(kind='kde')
```

[115]:
```
<AxesSubplot:ylabel='Density'>
```



[118]:
```
correlation_matrix = mhsDataset.corr()
correlation_matrix['Price']
```

[118]:
```
Rooms            0.448966
Price            1.000000
Distance        -0.234053
Bedroom2         0.434802
Bathroom         0.378641
Car              0.160052
Landsize         0.030368
BuildingArea     0.455905
Propertycount   -0.088376
Name: Price, dtype: float64
```

[126]:
```
independentCols = ['Rooms', 'Distance', 'Bathroom', 'Car', 'Landsize',␣
 ↪'BuildingArea', 'Propertycount']
xs =mhsDataset[independentCols]
ys=mhsDataset['Price']
```

[127]:
```
sbn.heatmap(correlation_matrix,cmap="YlGnBu",linewidth = 1, annot = True)
```

[127]: <AxesSubplot:>