HandlingOutliers

May 25, 2023

[238]:
```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
```

[239]:
```python
import warnings
warnings.filterwarnings('ignore')
```

[240]:
```python
#import housing dataset and display first five rows
mhDataset = pd.read_csv("Melbourne_housing_FULL.csv")
mhDataset.head()
```

[240]:
```
       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]
```

[241]:

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

[241]:

```
       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
```

[242]:

```
mhDataset.isna().sum()/len(mhDataset)*100
```

[242]:

```
Suburb            0.000000
Rooms             0.000000
Type              0.000000
Price            21.832057
Method            0.000000
SellerG           0.000000
Distance          0.002869
Bedroom2         23.573457
```

```
Bathroom         23.599277
Car              25.039447
Landsize         33.881286
BuildingArea     60.576068
CouncilArea       0.008607
Regionname        0.008607
Propertycount     0.008607
dtype: float64
```

[243]:
```
missing_values_count = mhDataset.isnull().sum()
missing_values_count
```

[243]:
```
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
```

[244]:
```
mhDataset = mhDataset.dropna()
mhDataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9244 entries, 2 to 34856
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         9244 non-null   object
 1   Rooms          9244 non-null   int64
 2   Type           9244 non-null   object
 3   Price          9244 non-null   float64
 4   Method         9244 non-null   object
 5   SellerG        9244 non-null   object
 6   Distance       9244 non-null   float64
 7   Bedroom2       9244 non-null   float64
 8   Bathroom       9244 non-null   float64
 9   Car            9244 non-null   float64
 10  Landsize       9244 non-null   float64
```

```
 11  BuildingArea   9244 non-null   float64
 12  CouncilArea    9244 non-null   object
 13  Regionname     9244 non-null   object
 14  Propertycount  9244 non-null   float64
dtypes: float64(8), int64(1), object(6)
memory usage: 1.1+ MB
```

[245]:
```python
mhDataset.isna().sum()/len(mhDataset)*100
```

[245]:
```
Suburb           0.0
Rooms            0.0
Type             0.0
Price            0.0
Method           0.0
SellerG          0.0
Distance         0.0
Bedroom2         0.0
Bathroom         0.0
Car              0.0
Landsize         0.0
BuildingArea     0.0
CouncilArea      0.0
Regionname       0.0
Propertycount    0.0
dtype: float64
```

[246]:
```python
mhDataset.isna().sum()
```

[246]:
```
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
```

[247]:
```python
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'})
```

[248]:
```
mhDataset.head()
```

[248]:
```
        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
```

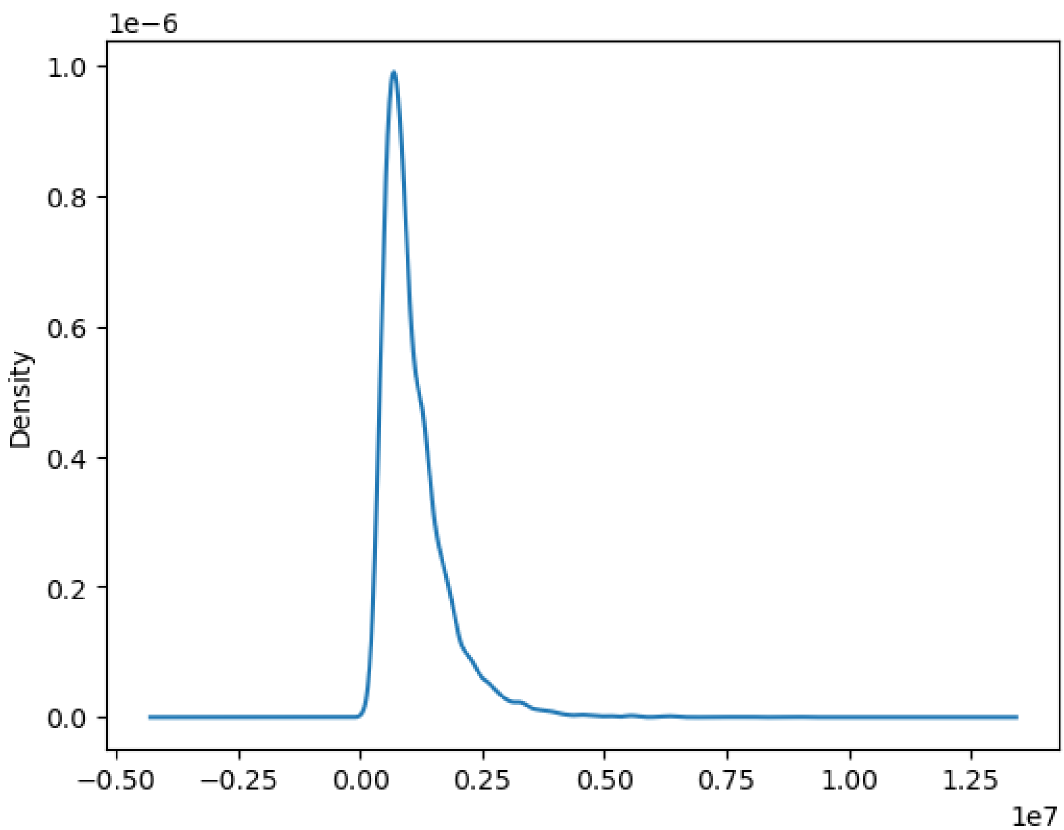
[249]:
```
mhDataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9244 entries, 2 to 34856
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         9244 non-null   object
 1   Rooms          9244 non-null   int64
 2   Type           9244 non-null   object
 3   Price          9244 non-null   float64
 4   Method         9244 non-null   object
```

```
 5   SellerG        9244 non-null   object
 6   Distance       9244 non-null   float64
 7   Bedroom2       9244 non-null   float64
 8   Bathroom       9244 non-null   float64
 9   Car            9244 non-null   float64
 10  Landsize       9244 non-null   float64
 11  BuildingArea   9244 non-null   float64
 12  CouncilArea    9244 non-null   object
 13  Regionname     9244 non-null   object
 14  Propertycount  9244 non-null   float64
dtypes: float64(8), int64(1), object(6)
memory usage: 1.1+ MB
```

[250]:
```
mhDataset['Price'].plot(kind='kde')
```

[250]:
```
<AxesSubplot:ylabel='Density'>
```



[251]:
```
total_cells = np.product(mhDataset.shape)
total_missing = missing_values_count.sum()
```

```
# percent of data that is missing
(total_missing/total_cells) * 100
```

[251]:
```
47.39362469349488
```

[252]:
```
mhDataset.describe().T
```

[252]:
```
              count          mean            std       min       25%  \
Rooms        9244.0  3.098118e+00       0.964029       1.0       2.0
Price        9244.0  1.092329e+06  679621.207086  131000.0  641000.0
Distance     9244.0  1.124115e+01       6.882570       0.0       6.4
Bedroom2     9244.0  3.077347e+00       0.966366       0.0       2.0
Bathroom     9244.0  1.652423e+00       0.724991       1.0       1.0
Car          9244.0  1.695370e+00       0.975529       0.0       1.0
Landsize     9244.0  5.288338e+02    1212.965090       0.0     210.0
BuildingArea 9244.0  1.569946e+02     480.976260       0.0     100.0
Propertycount 9244.0 7.463867e+03    4369.422310     249.0    4380.0

                   50%        75%        max
Rooms              3.0        4.0       12.0
Price         900000.0  1341250.0  9000000.0
Distance          10.3       13.9       48.1
Bedroom2           3.0        4.0       12.0
Bathroom           2.0        2.0        9.0
Car                2.0        2.0       10.0
Landsize         474.0      651.0    44500.0
BuildingArea     132.0      181.0    44515.0
Propertycount   6543.0    10331.0    21650.0
```

Looking at the table description, the values for Rooms and Bedroom2 looks almost equal. And also looking at the column names, we can eiother have Rooms column or Bedroom column. In this case, let us have "Rooms" column and delete the "Bedroom2" column

[253]:
```
mhDataset.dropna(axis=1,inplace=True)
```

[254]:
```
mhDataset = mhDataset.drop(['Bedroom2'],axis =1)
```

[255]:
```
mhDataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9244 entries, 2 to 34856
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         9244 non-null   object
 1   Rooms          9244 non-null   int64
 2   Type           9244 non-null   object
 3   Price          9244 non-null   float64
 4   Method         9244 non-null   object
```

 5   SellerG   9244 non-null   object
 6   Distance   9244 non-null   float64
 7   Bathroom   9244 non-null   float64
 8   Car   9244 non-null   float64
 9   Landsize   9244 non-null   float64
 10 BuildingArea   9244 non-null   float64
 11 CouncilArea   9244 non-null   object
 12 Regionname   9244 non-null   object
 13 Propertycount 9244 non-null   float64
dtypes: float64(7), int64(1), object(6)
memory usage: 1.1+ MB
```

Handling outliers using IQR method

The formula to exclude outliers using InterQuartile method - Dependent variable column > (Quartile1 - (1.5 * IQR)) and Dependent variable column <= (Quartile3 + (1.5 * IQR)) Lower Limit = Quartile1 - (1.5 * IQR) Upper Limit = Quartile3 + (1.5 * IQR)

[256]:
```
#Now lets handle the outliers
mhsDataset = mhDataset.copy()

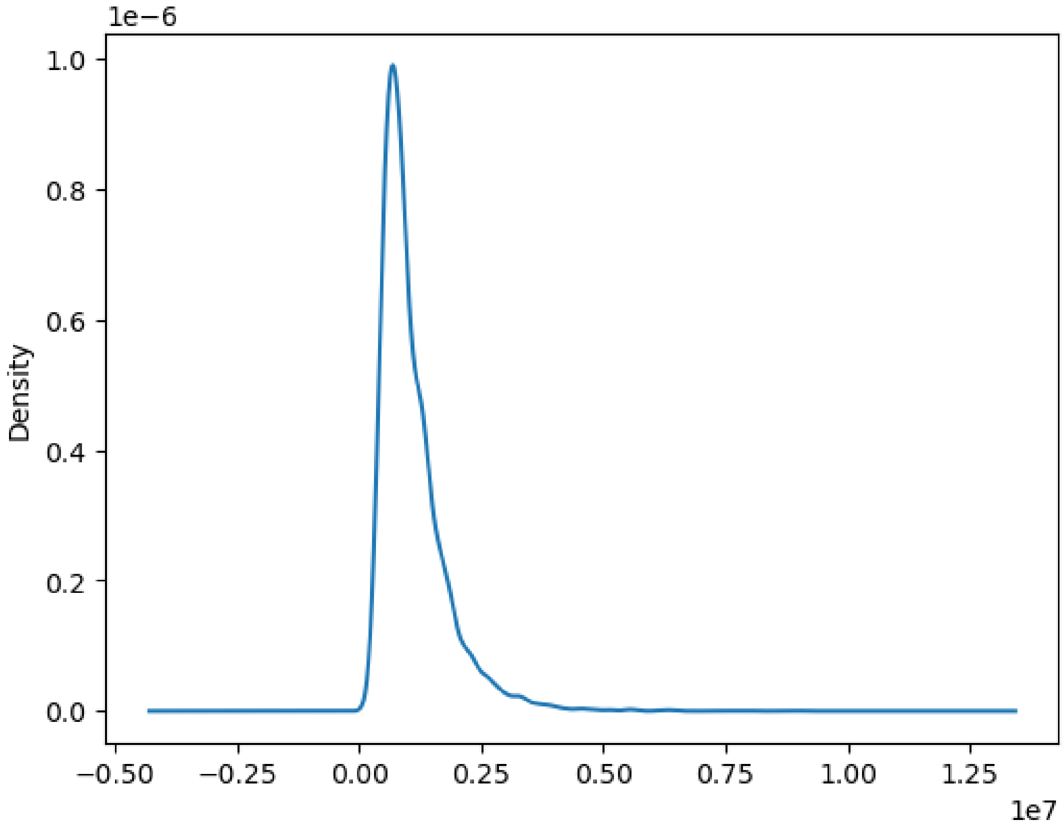
IQR = mhDataset['Price'].quantile(0.75) - mhDataset['Price'].quantile(0.25)

lower = mhDataset['Price'].quantile(0.25) - 1.5* IQR
upper = mhDataset['Price'].quantile(0.75) + 1.5* IQR

outliers = np.where(mhDataset['Price']>upper,True, np.
 ↪where(mhDataset['Price']<lower,True,False))

mhsDataset = mhDataset.loc[~(outliers)]

print("Lower Limit :",lower)
print("Uppwe Limit :",upper)
#mhsDataset = mhsDataset[~((mhsDataset['Price']<lower) &␣
 ↪(mhsDataset['Price']>upper))]
```

```
Lower Limit : -409375.0
Uppwe Limit : 2391625.0
```

[257]:
```
mhDataset['Price'].plot(kind='kde')
```

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



[258]:

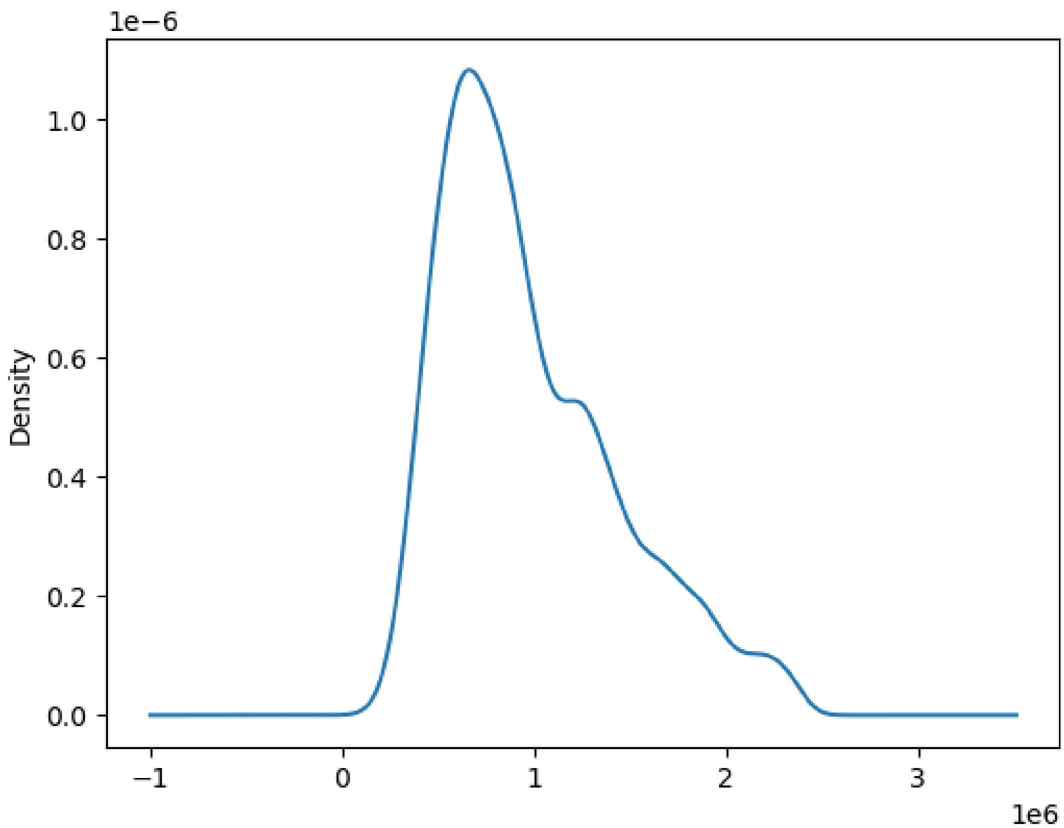
```
mhsDataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8788 entries, 2 to 34856
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         8788 non-null   object
 1   Rooms          8788 non-null   int64
 2   Type           8788 non-null   object
 3   Price          8788 non-null   float64
 4   Method         8788 non-null   object
 5   SellerG        8788 non-null   object
 6   Distance       8788 non-null   float64
 7   Bathroom       8788 non-null   float64
 8   Car            8788 non-null   float64
 9   Landsize       8788 non-null   float64
 10  BuildingArea   8788 non-null   float64
 11  CouncilArea    8788 non-null   object
 12  Regionname     8788 non-null   object
 13  Propertycount  8788 non-null   float64
```

```
dtypes: float64(7), int64(1), object(6)
memory usage: 1.0+ MB
```

[259]:
```
mhsDataset['Price'].plot(kind='kde')
```

[259]: <AxesSubplot:ylabel='Density'>



[260]:
```
mhsDataset.describe().T
```

[260]:
```
              count           mean            std       min        25%  \
Rooms        8788.0       3.039713       0.935446       1.0       2.00
Price        8788.0  985448.177856  464712.324595  131000.0  630000.00
Distance     8788.0      11.412062       6.977235       0.0       6.50
Bathroom     8788.0       1.599681       0.671289       1.0       1.00
Car          8788.0       1.665794       0.959368       0.0       1.00
Landsize     8788.0     517.113792    1237.520726       0.0     199.75
BuildingArea 8788.0     150.325378     490.774992       0.0      98.00
Propertycount 8788.0   7466.414998    4422.746339     249.0    4294.00

                  50%        75%        max
Rooms             3.0        4.0       12.0
```

```
Price          870000.0  1266000.0  2385000.0
Distance           10.5       14.0       48.1
Bathroom            2.0        2.0        9.0
Car                 2.0        2.0       10.0
Landsize          454.5      643.0    44500.0
BuildingArea      130.0      174.0    44515.0
Propertycount    6543.0    10331.0    21650.0
```

[261]:
```
independentCols = ['Rooms', 'Distance', 'Bathroom', 'Car', 'Landsize',␣
 ↪'BuildingArea', 'Propertycount']
xs =mhsDataset[independentCols]
ys=mhsDataset['Price']
```

[262]:
```
Xs_train, Xs_test, Ys_train, Ys_test = train_test_split(xs, ys, test_size=0.30,␣
 ↪random_state=100)
```

[263]:
```
linearReg = LinearRegression()
linearReg.fit(Xs_train,Ys_train)
```

[263]: LinearRegression()

[264]:
```
linearReg.score(Xs_train,Ys_train)
```

[264]: 0.3834886371405757

[265]:
```
linearReg.score(Xs_test,Ys_test)
```

[265]: 0.37114890123584865

[266]:
```
lassoReg = linear_model.Lasso(alpha=50,max_iter=100,tol=1)
lassoReg.fit(Xs_train,Ys_train)
```

[266]: Lasso(alpha=50, max_iter=100, tol=1)

[267]:
```
lassoReg.score(Xs_train,Ys_train)
```

[267]: 0.38307425605552137

[268]:
```
lassoReg.score(Xs_test,Ys_test)
```

[268]: 0.3691530493706163

[269]:
```
ridgeReg = linear_model.Ridge(alpha=100,max_iter=999,tol=1)
ridgeReg.fit(Xs_train,Ys_train)
```

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

[270]:
```
ridgeReg.score(Xs_test,Ys_test)
```

[270]: 0.37073499653704556

[271]:
```
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

knn_model = KNeighborsRegressor().fit(Xs_train, Ys_train)
predicted_values = knn_model.predict(Xs_test)
```

[272]:
```
predict_df = pd.DataFrame({"Dependent_Test" : Ys_test, "Dependent_Predicted" :
 ↪predicted_values})
predict_df.head()
```

[272]:

	Dependent_Test	Dependent_Predicted
26064	1100000.0	1309000.0
10977	1287000.0	1558700.0
14051	490000.0	497200.0
8529	767500.0	768400.0
26212	1230000.0	1832000.0

[273]:
```
predict_df = (predict_df*(np.max(mhsDataset.Price) - np.min(mhsDataset.Price)))
 ↪+ np.min(mhsDataset.Price)
```

[274]:
```
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

print("Mean Squared Error = ", mean_squared_error(predict_df.
 ↪Dependent_Predicted, predict_df.Dependent_Test))


print("R2 Score",r2_score(predict_df.Dependent_Predicted,predict_df.
 ↪Dependent_Test))
```

```
Mean Squared Error =  5.070665377882707e+23
R2 Score 0.2901374317807399
```

Handling OUtliers using Normal Distribution

The formula to handle outliers using normal distribution is Dependent variable column > mean - 3 * Standard Deviation and Dependent variable column <= mean - 3 * Standard Deviation Lower Limit = mean - 3 * Standard Deviation Upper Limit = mean + 3 * Standard Deviation

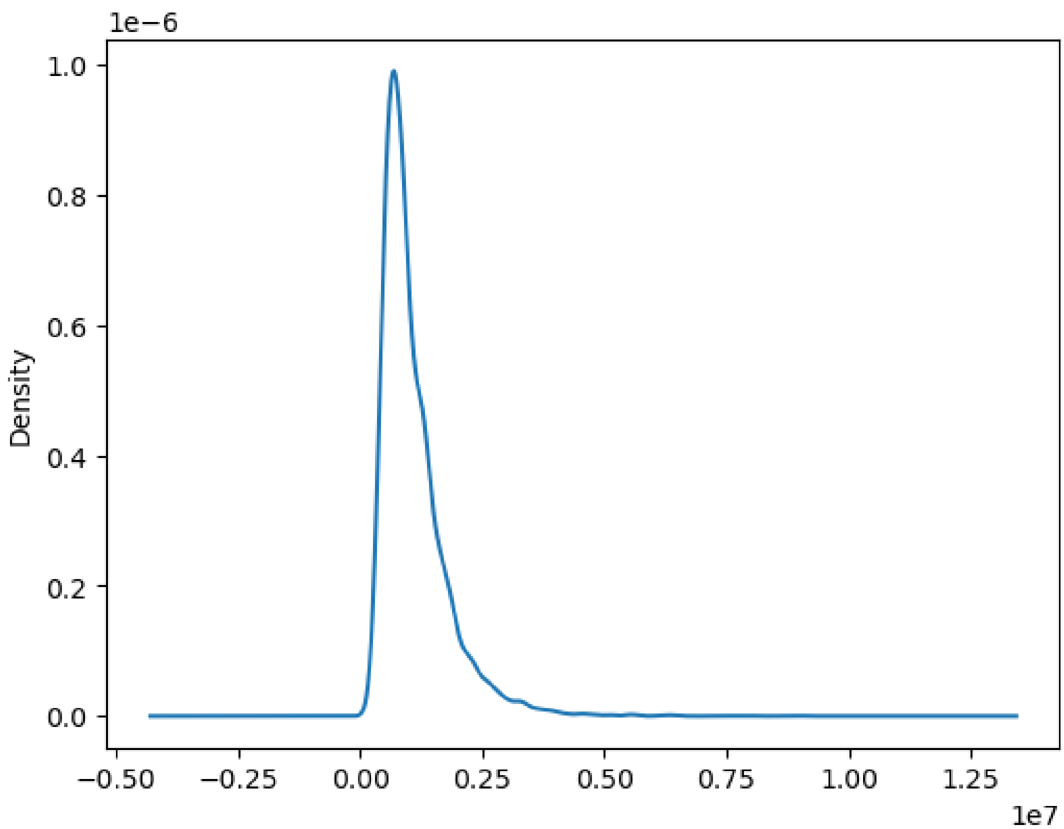
[275]:
```
#Now lets handle the outliers
mhDatasetND = mhDataset.copy()

#IQR = mhDataset['Price'].quantile(0.75) - mhDataset['Price'].quantile(0.25)

lower = mhDataset['Price'].mean() - 3 * mhDataset['Price'].std()
upper = mhDataset['Price'].mean() + 3 * mhDataset['Price'].std()
```

```
outliers = np.where(mhDataset['Price']>upper,True, np.
    ↪where(mhDataset['Price']<lower,True,False))

mhDatasetND = mhDataset.loc[~(outliers)]

print("Lower Limit :",lower)
print("Upper Limit :",upper)
#mhsDataset = mhsDataset[~((mhsDataset['Price']<lower) &␣
    ↪(mhsDataset['Price']>upper))]
```

```
Lower Limit : -946535.0324432228
Upper Limit : 3131192.210071955
```

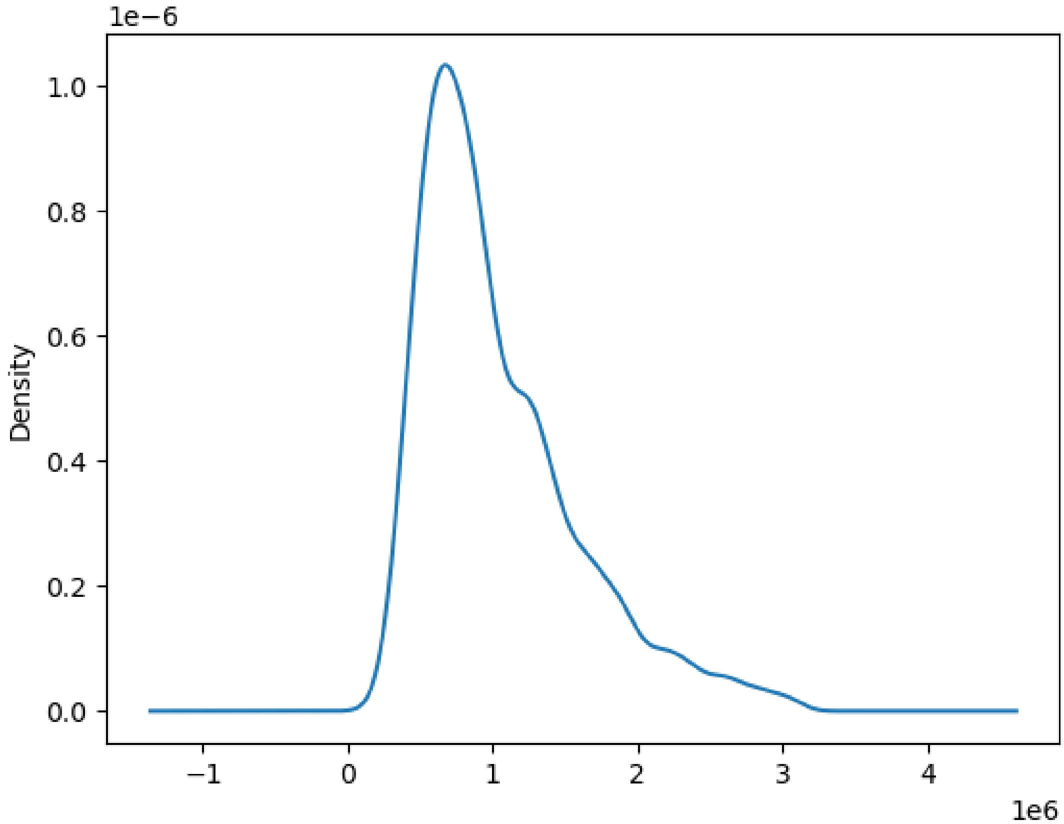
[276]:
```
mhDataset['Price'].plot(kind='kde')
```

[276]: <AxesSubplot:ylabel='Density'>



[277]:
```
mhDatasetND['Price'].plot(kind='kde')
```

[277]: <AxesSubplot:ylabel='Density'>



Looking at the above two graphs, the exclusion of outliers is evident.

[278]:
```
mhDatasetND.describe().T
```

[278]:

	count	mean	std	min	25%
Rooms	9078.0	3.075677e+00	0.952607	1.0	2.0
Price	9078.0	1.039889e+06	547912.433123	131000.0	636125.0
Distance	9078.0	1.130724e+01	6.914404	0.0	6.4
Bathroom	9078.0	1.628663e+00	0.695856	1.0	1.0
Car	9078.0	1.679445e+00	0.963541	0.0	1.0
Landsize	9078.0	5.235416e+02	1221.791081	0.0	207.0
BuildingArea	9078.0	1.541873e+02	484.691677	0.0	99.0
Propertycount	9078.0	7.465040e+03	4389.801336	249.0	4380.0

	50%	75%	max
Rooms	3.0	4.0	12.0
Price	886000.0	1310000.0	3120000.0
Distance	10.4	14.0	48.1
Bathroom	2.0	2.0	9.0
Car	2.0	2.0	10.0
Landsize	465.0	650.0	44500.0

```
BuildingArea       131.0      179.0    44515.0
Propertycount     6543.0    10331.0    21650.0
```

[279]:
```
independentCols = ['Rooms', 'Distance', 'Bathroom', 'Car', 'Landsize',␣
↪'BuildingArea', 'Propertycount']
xsn =mhDatasetND[independentCols]
ysn =mhDatasetND['Price']
```

[280]:
```
Xsn_train, Xsn_test, Ysn_train, Ysn_test = train_test_split(xsn, ysn,␣
↪test_size=0.30, random_state=100)
```

[281]:
```
linearReg = LinearRegression()
linearReg.fit(Xsn_train,Ysn_train)
```

[281]: LinearRegression()

[282]:
```
linearReg.score(Xsn_train,Ysn_train)
```

[282]: 0.4243021032287464

[283]:
```
linearReg.score(Xs_test,Ys_test)
```

[283]: 0.3577993841828496

[284]:
```
lassoReg = linear_model.Lasso(alpha=50,max_iter=100,tol=1)
lassoReg.fit(Xsn_train,Ysn_train)
```

[284]: Lasso(alpha=50, max_iter=100, tol=1)

[285]:
```
lassoReg.score(Xsn_train,Ysn_train)
```

[285]: 0.42420586222745427

[286]:
```
lassoReg.score(Xsn_test,Ysn_test)
```

[286]: 0.37889064944333106

[287]:
```
ridgeReg = linear_model.Ridge(alpha=100,max_iter=999,tol=1)
ridgeReg.fit(Xsn_train,Ysn_train)
```

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

[288]:
```
knn_model = KNeighborsRegressor().fit(Xsn_train, Ysn_train)
predicted_values = knn_model.predict(Xsn_test)
```

[289]:
```
predict_df = pd.DataFrame({"Dependent_Test" : Ysn_test, "Dependent_Predicted" :␣
↪predicted_values})
predict_df.head()
```

[289]:

	Dependent_Test	Dependent_Predicted
14949	975000.0	1053000.0
20120	1210000.0	1378300.0
34177	760000.0	866500.0
29263	910000.0	959200.0
6425	1120000.0	903800.0

[290]:
```
predict_df = (predict_df*(np.max(mhDatasetND.Price) - np.min(mhDatasetND.
↪Price))) + np.min(mhDatasetND.Price)
```

[291]:
```
print("Mean Squared Error = ", mean_squared_error(predict_df.
↪Dependent_Predicted, predict_df.Dependent_Test))


print("R2 Score",r2_score(predict_df.Dependent_Predicted,predict_df.
↪Dependent_Test))
```

```
Mean Squared Error =  1.1996829183761369e+24
R2 Score 0.2790573905413596
```

Interquartile Range Method Mean Squared Error = 5.070665377882707e+23 R2 Score 0.2901374317807399

Z-Score Method Mean Squared Error = 1.1996829183761369e+24 R2 Score 0.2790573905413596

Looking at he MSE and R2 score of both the methods, Z-Score method suits this dataset to handle outliers since there seems to be less error.

[ ]:
```

```