my_project

May 9, 2022

[1]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler,PolynomialFeatures
from sklearn.linear_model import LinearRegression
```

[19]:
```python
my_data=pd.read_csv('house_data.csv')
```

[20]:
```python
my_data.head()
```

[20]:
```
           id             date     price  bedrooms  bathrooms  sqft_living  \
0  7129300520  20141013T000000  221900.0         3       1.00         1180
1  6414100192  20141209T000000  538000.0         3       2.25         2570
2  5631500400  20150225T000000  180000.0         2       1.00          770
3  2487200875  20141209T000000  604000.0         4       3.00         1960
4  1954400510  20150218T000000  510000.0         3       2.00         1680

   sqft_lot  floors  waterfront  view  …  grade  sqft_above  sqft_basement  \
0      5650     1.0           0     0  …      7        1180              0
1      7242     2.0           0     0  …      7        2170            400
2     10000     1.0           0     0  …      6         770              0
3      5000     1.0           0     0  …      7        1050            910
4      8080     1.0           0     0  …      8        1680              0

   yr_built  yr_renovated  zipcode      lat     long  sqft_living15  \
0      1955             0    98178  47.5112 -122.257           1340
1      1951          1991    98125  47.7210 -122.319           1690
2      1933             0    98028  47.7379 -122.233           2720
3      1965             0    98136  47.5208 -122.393           1360
4      1987             0    98074  47.6168 -122.045           1800

   sqft_lot15
0        5650
1        7639
2        8062
3        5000
```

```
4        7503

[5 rows x 21 columns]
```

[21]:
```python
my_data.dtypes
```

[21]:
```
id                 int64
date              object
price            float64
bedrooms           int64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
grade              int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
zipcode            int64
lat              float64
long             float64
sqft_living15      int64
sqft_lot15         int64
dtype: object
```

[22]:
```python
my_data.describe()
```

[22]:
```
                 id         price      bedrooms     bathrooms   sqft_living  \
count  2.161300e+04  2.161300e+04  21613.000000  21613.000000  21613.000000
mean   4.580302e+09  5.400881e+05      3.370842      2.114757   2079.899736
std    2.876566e+09  3.671272e+05      0.930062      0.770163    918.440897
min    1.000102e+06  7.500000e+04      0.000000      0.000000    290.000000
25%    2.123049e+09  3.219500e+05      3.000000      1.750000   1427.000000
50%    3.904930e+09  4.500000e+05      3.000000      2.250000   1910.000000
75%    7.308900e+09  6.450000e+05      4.000000      2.500000   2550.000000
max    9.900000e+09  7.700000e+06     33.000000      8.000000  13540.000000

           sqft_lot        floors    waterfront          view     condition  \
count  2.161300e+04  21613.000000  21613.000000  21613.000000  21613.000000
mean   1.510697e+04      1.494309      0.007542      0.234303      3.409430
std    4.142051e+04      0.539989      0.086517      0.766318      0.650743
min    5.200000e+02      1.000000      0.000000      0.000000      1.000000
25%    5.040000e+03      1.000000      0.000000      0.000000      3.000000
```

```
50%    7.618000e+03      1.500000      0.000000      0.000000      3.000000
75%    1.068800e+04      2.000000      0.000000      0.000000      4.000000
max    1.651359e+06      3.500000      1.000000      4.000000      5.000000

              grade    sqft_above  sqft_basement      yr_built  yr_renovated  \
count  21613.000000  21613.000000   21613.000000  21613.000000  21613.000000
mean       7.656873   1788.390691     291.509045   1971.005136     84.402258
std        1.175459    828.090978     442.575043     29.373411    401.679240
min        1.000000    290.000000       0.000000   1900.000000      0.000000
25%        7.000000   1190.000000       0.000000   1951.000000      0.000000
50%        7.000000   1560.000000       0.000000   1975.000000      0.000000
75%        8.000000   2210.000000     560.000000   1997.000000      0.000000
max       13.000000   9410.000000    4820.000000   2015.000000   2015.000000

            zipcode           lat          long  sqft_living15     sqft_lot15
count  21613.000000  21613.000000  21613.000000   21613.000000   21613.000000
mean   98077.939805     47.560053   -122.213896    1986.552492   12768.455652
std       53.505026      0.138564      0.140828     685.391304   27304.179631
min    98001.000000     47.155900   -122.519000     399.000000     651.000000
25%    98033.000000     47.471000   -122.328000    1490.000000    5100.000000
50%    98065.000000     47.571800   -122.230000    1840.000000    7620.000000
75%    98118.000000     47.678000   -122.125000    2360.000000   10083.000000
max    98199.000000     47.777600   -121.315000    6210.000000  871200.000000
```

[26]:

```python
my_data.drop("id", axis=1, inplace=True)

my_data.reset_index(drop=True, inplace=True)
```

[27]:

```python
my_data.describe()
```

[27]:

```
              price      bedrooms     bathrooms   sqft_living      sqft_lot  \
count  2.161300e+04  21613.000000  21613.000000  21613.000000  2.161300e+04
mean   5.400881e+05      3.370842      2.114757   2079.899736  1.510697e+04
std    3.671272e+05      0.930062      0.770163    918.440897  4.142051e+04
min    7.500000e+04      0.000000      0.000000    290.000000  5.200000e+02
25%    3.219500e+05      3.000000      1.750000   1427.000000  5.040000e+03
50%    4.500000e+05      3.000000      2.250000   1910.000000  7.618000e+03
75%    6.450000e+05      4.000000      2.500000   2550.000000  1.068800e+04
max    7.700000e+06     33.000000      8.000000  13540.000000  1.651359e+06

             floors    waterfront          view     condition         grade  \
count  21613.000000  21613.000000  21613.000000  21613.000000  21613.000000
mean       1.494309      0.007542      0.234303      3.409430      7.656873
std        0.539989      0.086517      0.766318      0.650743      1.175459
min        1.000000      0.000000      0.000000      1.000000      1.000000
25%        1.000000      0.000000      0.000000      3.000000      7.000000
50%        1.500000      0.000000      0.000000      3.000000      7.000000
```

```
75%        2.000000      0.000000      0.000000      4.000000      8.000000
max        3.500000      1.000000      4.000000      5.000000     13.000000

         sqft_above  sqft_basement      yr_built  yr_renovated       zipcode  \
count  21613.000000   21613.000000  21613.000000  21613.000000  21613.000000
mean    1788.390691     291.509045   1971.005136     84.402258  98077.939805
std      828.090978     442.575043     29.373411    401.679240     53.505026
min      290.000000       0.000000   1900.000000      0.000000  98001.000000
25%     1190.000000       0.000000   1951.000000      0.000000  98033.000000
50%     1560.000000       0.000000   1975.000000      0.000000  98065.000000
75%     2210.000000     560.000000   1997.000000      0.000000  98118.000000
max     9410.000000    4820.000000   2015.000000   2015.000000  98199.000000

                lat          long  sqft_living15     sqft_lot15
count  21613.000000  21613.000000   21613.000000   21613.000000
mean      47.560053   -122.213896    1986.552492   12768.455652
std        0.138564      0.140828     685.391304   27304.179631
min       47.155900   -122.519000     399.000000     651.000000
25%       47.471000   -122.328000    1490.000000    5100.000000
50%       47.571800   -122.230000    1840.000000    7620.000000
75%       47.678000   -122.125000    2360.000000   10083.000000
max       47.777600   -121.315000    6210.000000  871200.000000
```

[28]:
```python
print("number of NaN values for the column bedrooms :", my_data['bedrooms'].
 ↪isnull().sum())
print("number of NaN values for the column bathrooms :", my_data['bathrooms'].
 ↪isnull().sum())
```

```
number of NaN values for the column bedrooms : 0
number of NaN values for the column bathrooms : 0
```

[29]:
```python
mean=my_data['bedrooms'].mean()
my_data['bedrooms'].replace(np.nan,mean, inplace=True)
```

[31]:
```python
mean=my_data['bathrooms'].mean()
my_data['bathrooms'].replace(np.nan,mean, inplace=True)
```

[34]:
```python
print("number of NaN values for the column bedrooms :", my_data['bedrooms'].
 ↪isnull().sum())
print("number of NaN values for the column bathrooms :", my_data['bathrooms'].
 ↪isnull().sum())
```

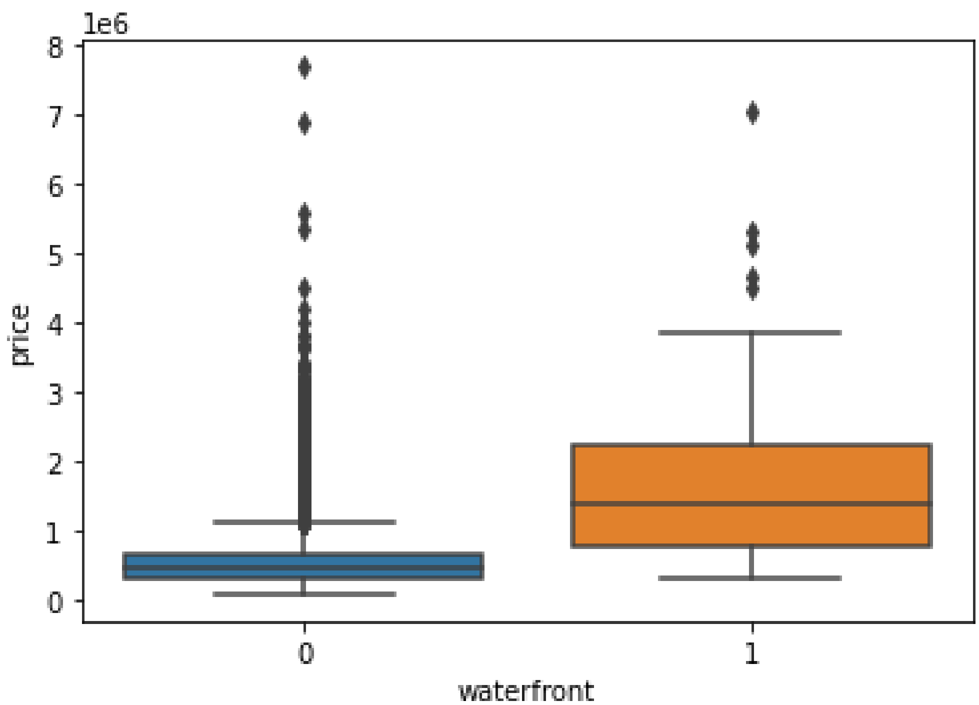
```
number of NaN values for the column bedrooms : 0
number of NaN values for the column bathrooms : 0
```

[36]:
```python
my_data['floors'].value_counts().to_frame()
```

[36]:
```
      floors
1.0    10680
2.0     8241
1.5     1910
3.0      613
2.5      161
3.5        8
```

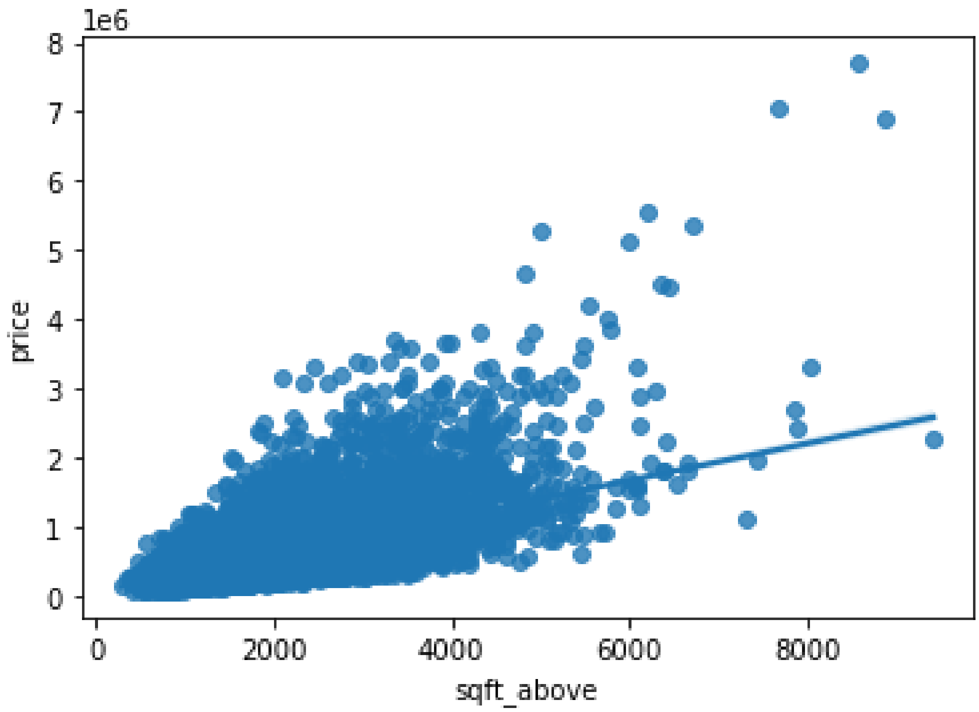
[37]:
```
sns.boxplot(x='waterfront',y='price', data=my_data)
```

[37]:
```
<AxesSubplot:xlabel='waterfront', ylabel='price'>
```



[40]:
```
sns.regplot(x="sqft_above", y="price", data=my_data)
```

[40]:
```
<AxesSubplot:xlabel='sqft_above', ylabel='price'>
```



[41]:
```
my_data.corr()
```

[41]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	\
price	1.000000	0.308350	0.525138	0.702035	0.089661	0.256794	
bedrooms	0.308350	1.000000	0.515884	0.576671	0.031703	0.175429	
bathrooms	0.525138	0.515884	1.000000	0.754665	0.087740	0.500653	
sqft_living	0.702035	0.576671	0.754665	1.000000	0.172826	0.353949	
sqft_lot	0.089661	0.031703	0.087740	0.172826	1.000000	-0.005201	
floors	0.256794	0.175429	0.500653	0.353949	-0.005201	1.000000	
waterfront	0.266369	-0.006582	0.063744	0.103818	0.021604	0.023698	
view	0.397293	0.079532	0.187737	0.284611	0.074710	0.029444	
condition	0.036362	0.028472	-0.124982	-0.058753	-0.008958	-0.263768	
grade	0.667434	0.356967	0.664983	0.762704	0.113621	0.458183	
sqft_above	0.605567	0.477600	0.685342	0.876597	0.183512	0.523885	
sqft_basement	0.323816	0.303093	0.283770	0.435043	0.015286	-0.245705	
yr_built	0.054012	0.154178	0.506019	0.318049	0.053080	0.489319	
yr_renovated	0.126434	0.018841	0.050739	0.055363	0.007644	0.006338	
zipcode	-0.053203	-0.152668	-0.203866	-0.199430	-0.129574	-0.059121	
lat	0.307003	-0.008931	0.024573	0.052529	-0.085683	0.049614	
long	0.021626	0.129473	0.223042	0.240223	0.229521	0.125419	
sqft_living15	0.585379	0.391638	0.568634	0.756420	0.144608	0.279885	
sqft_lot15	0.082447	0.029244	0.087175	0.183286	0.718557	-0.011269	

	waterfront	view	condition	grade	sqft_above	\

```
price             0.266369  0.397293   0.036362  0.667434    0.605567
bedrooms         -0.006582  0.079532   0.028472  0.356967    0.477600
bathrooms         0.063744  0.187737  -0.124982  0.664983    0.685342
sqft_living       0.103818  0.284611  -0.058753  0.762704    0.876597
sqft_lot          0.021604  0.074710  -0.008958  0.113621    0.183512
floors            0.023698  0.029444  -0.263768  0.458183    0.523885
waterfront        1.000000  0.401857   0.016653  0.082775    0.072075
view              0.401857  1.000000   0.045990  0.251321    0.167649
condition         0.016653  0.045990   1.000000 -0.144674   -0.158214
grade             0.082775  0.251321  -0.144674  1.000000    0.755923
sqft_above        0.072075  0.167649  -0.158214  0.755923    1.000000
sqft_basement     0.080588  0.276947   0.174105  0.168392   -0.051943
yr_built         -0.026161 -0.053440  -0.361417  0.446963    0.423898
yr_renovated      0.092885  0.103917  -0.060618  0.014414    0.023285
zipcode           0.030285  0.084827   0.003026 -0.184862   -0.261190
lat              -0.014274  0.006157  -0.014941  0.114084   -0.000816
long             -0.041910 -0.078400  -0.106500  0.198372    0.343803
sqft_living15     0.086463  0.280439  -0.092824  0.713202    0.731870
sqft_lot15        0.030703  0.072575  -0.003406  0.119248    0.194050

               sqft_basement  yr_built  yr_renovated   zipcode       lat  \
price               0.323816  0.054012      0.126434 -0.053203  0.307003
bedrooms            0.303093  0.154178      0.018841 -0.152668 -0.008931
bathrooms           0.283770  0.506019      0.050739 -0.203866  0.024573
sqft_living         0.435043  0.318049      0.055363 -0.199430  0.052529
sqft_lot            0.015286  0.053080      0.007644 -0.129574 -0.085683
floors             -0.245705  0.489319      0.006338 -0.059121  0.049614
waterfront          0.080588 -0.026161      0.092885  0.030285 -0.014274
view                0.276947 -0.053440      0.103917  0.084827  0.006157
condition           0.174105 -0.361417     -0.060618  0.003026 -0.014941
grade               0.168392  0.446963      0.014414 -0.184862  0.114084
sqft_above         -0.051943  0.423898      0.023285 -0.261190 -0.000816
sqft_basement       1.000000 -0.133124      0.071323  0.074845  0.110538
yr_built           -0.133124  1.000000     -0.224874 -0.346869 -0.148122
yr_renovated        0.071323 -0.224874      1.000000  0.064357  0.029398
zipcode             0.074845 -0.346869      0.064357  1.000000  0.267048
lat                 0.110538 -0.148122      0.029398  0.267048  1.000000
long               -0.144765  0.409356     -0.068372 -0.564072 -0.135512
sqft_living15       0.200355  0.326229     -0.002673 -0.279033  0.048858
sqft_lot15          0.017276  0.070958      0.007854 -0.147221 -0.086419

                   long  sqft_living15  sqft_lot15
price          0.021626       0.585379    0.082447
bedrooms       0.129473       0.391638    0.029244
bathrooms      0.223042       0.568634    0.087175
sqft_living    0.240223       0.756420    0.183286
sqft_lot       0.229521       0.144608    0.718557
```

```
floors         0.125419       0.279885   -0.011269
waterfront    -0.041910       0.086463    0.030703
view          -0.078400       0.280439    0.072575
condition     -0.106500      -0.092824   -0.003406
grade          0.198372       0.713202    0.119248
sqft_above     0.343803       0.731870    0.194050
sqft_basement -0.144765       0.200355    0.017276
yr_built       0.409356       0.326229    0.070958
yr_renovated  -0.068372      -0.002673    0.007854
zipcode       -0.564072      -0.279033   -0.147221
lat           -0.135512       0.048858   -0.086419
long           1.000000       0.334605    0.254451
sqft_living15  0.334605       1.000000    0.183192
sqft_lot15     0.254451       0.183192    1.000000
```

[44]:
```
my_data.corr()['price'].sort_values()
```

[44]:
```
zipcode         -0.053203
long             0.021626
condition        0.036362
yr_built         0.054012
sqft_lot15       0.082447
sqft_lot         0.089661
yr_renovated     0.126434
floors           0.256794
waterfront       0.266369
lat              0.307003
bedrooms         0.308350
sqft_basement    0.323816
view             0.397293
bathrooms        0.525138
sqft_living15    0.585379
sqft_above       0.605567
grade            0.667434
sqft_living      0.702035
price            1.000000
Name: price, dtype: float64
```

[50]:
```
X = my_data[['long']]
Y = my_data['price']
lm = LinearRegression()
lm.fit(X,Y)
print('The R-square is: ', lm.score(X, Y))
```

```
The R-square is:  0.00046769430149007363
```

[51]:
```
X = my_data[['sqft_living']]
Y = my_data['price']
```

```
lm = LinearRegression()
lm.fit(X,Y)
print('The R-square is: ', lm.score(X, Y))
```

```
The R-square is:  0.4928532179037931
```

[52]:
```
f =["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view"␣
↪,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]
```

[65]:
```
lm2 = LinearRegression()
lm2.fit(my_data[["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement"␣
↪,"view"␣
↪,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]],my_data['price'])
print('The R-square is: ', lm2.score(my_data[["floors", "waterfront","lat"␣
↪,"bedrooms" ,"sqft_basement" ,"view"␣
↪,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]],␣
↪my_data['price']))
```

```
The R-square is:  0.657717260844526
```

[61]:
```
Input=[('scale',StandardScaler()),('polynomial',␣
↪PolynomialFeatures(include_bias=False)),('model',LinearRegression())]
```

[62]:
```
pipe=Pipeline(Input)
pipe
```

[62]:
```
Pipeline(steps=[('scale', StandardScaler()),
                ('polynomial', PolynomialFeatures(include_bias=False)),
                ('model', LinearRegression())])
```

[67]:
```
pipe.fit(my_data[["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement"␣
↪,"view"␣
↪,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]],my_data['price'])
```

[67]:
```
Pipeline(steps=[('scale', StandardScaler()),
                ('polynomial', PolynomialFeatures(include_bias=False)),
                ('model', LinearRegression())])
```

[68]:
```
ypipe=pipe.predict(my_data[["floors", "waterfront","lat" ,"bedrooms"␣
↪,"sqft_basement" ,"view"␣
↪,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]])
ypipe[0:4]
```

[68]:
```
array([349624.25, 559245.25, 447600.25, 395416.25])
```

[72]:
```
print('The R-square is: ', pipe.score(my_data[["floors", "waterfront","lat"␣
↪,"bedrooms" ,"sqft_basement" ,"view"␣
↪,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]],␣
↪my_data['price']))
```

```
The R-square is:  0.7513461993527443
```

[74]:
```python
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
```

[75]:
```python
features =["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view"␣
↪,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]
X = my_data[features]
Y = my_data['price']

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.15,␣
↪random_state=1)


print("number of test samples:", x_test.shape[0])
print("number of training samples:",x_train.shape[0])
```

```
number of test samples: 3242
number of training samples: 18371
```

[76]:
```python
from sklearn.linear_model import Ridge
```

[77]:
```python
RigeModel=Ridge(alpha=1)
```

[78]:
```python
RigeModel.fit(x_train, y_train)
```

[78]:
```
Ridge(alpha=1)
```

[79]:
```python
yhat = RigeModel.predict(x_test)
```

[80]:
```python
print('predicted:', yhat[0:4])
print('test set :', y_test[0:4].values)
```

```
predicted: [651925.04835026 514495.27908877 794550.25245092 702535.34540042]
test set : [ 459000.  445000. 1057000.  732350.]
```

[81]:
```python
RigeModel = Ridge(alpha=10)
RigeModel.fit(x_train, y_train)
RigeModel.score(x_test, y_test)
```

[81]:
```
0.6472468277965973
```

[83]:
```python
pr=PolynomialFeatures(degree=2)
x_train_pr=pr.fit_transform(x_train[["floors", "waterfront","lat" ,"bedrooms"␣
↪,"sqft_basement" ,"view"␣
↪,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]])
x_test_pr=pr.fit_transform(x_test[["floors", "waterfront","lat" ,"bedrooms"␣
↪,"sqft_basement" ,"view"␣
↪,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]])
```

[84]:
```
RigeModel_2=Ridge(alpha=1)
```

[85]:
```
RigeModel_2.fit(x_train_pr, y_train)
```

[85]:
```
Ridge(alpha=1)
```

[87]:
```
yhat2 = RigeModel_2.predict(x_test_pr)
```

[88]:
```
print('predicted:', yhat2[0:4])
print('test set :', y_test[0:4].values)
```

```
predicted: [569251.10183516 489784.50502835 683142.56011778 688628.48375185]
test set : [ 459000.  445000. 1057000.  732350.]
```

[89]:
```
RigeModel = Ridge(alpha=10)
RigeModel.fit(x_train_pr, y_train)
RigeModel.score(x_test_pr, y_test)
```

[89]:
```
0.6998710425686812
```

[ ]:
```

```