Untitled2

December 27, 2019

[2]:

```
import numpy as np
import pandas as pd
```

[54]:

```
# Import clean data
path = 'kc_house_data.csv'
df_H = pd.read_csv(path)
df_H.head()
```

[54]:

```
           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]
```

[55]:
```
df_H.dtypes
```

[55]:
```
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
```

[56]:
```
df_H1 = df_H.copy()
df_H1.drop("id", axis = 1, inplace = True)
df_H1.describe()
```

[56]:
```
              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
```

[58]:

```
H_floors=df_H1['floors'].value_counts()
H_floors.to_frame()
```

[58]:

```
     floors
1.0   10680
2.0    8241
1.5    1910
3.0     613
2.5     161
3.5       8
```

[59]:

```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

[64]:

```
df_H1['column_name'] = df_H1['waterfront'].astype('bool')
df_H1.head()
```

[64]:

```
              date     price  bedrooms  bathrooms  sqft_living  sqft_lot  \
0  20141013T000000  221900.0         3       1.00         1180      5650
1  20141209T000000  538000.0         3       2.25         2570      7242
2  20150225T000000  180000.0         2       1.00          770     10000
3  20141209T000000  604000.0         4       3.00         1960      5000
4  20150218T000000  510000.0         3       2.00         1680      8080
```

```
   floors  waterfront  view  condition  …  sqft_above  sqft_basement  \
0     1.0           0     0          3  …        1180              0
1     2.0           0     0          3  …        2170            400
2     1.0           0     0          3  …         770              0
3     1.0           0     0          5  …        1050            910
4     1.0           0     0          3  …        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  column_name
0        5650        False
1        7639        False
2        8062        False
3        5000        False
4        7503        False

[5 rows x 21 columns]
```

[77]:

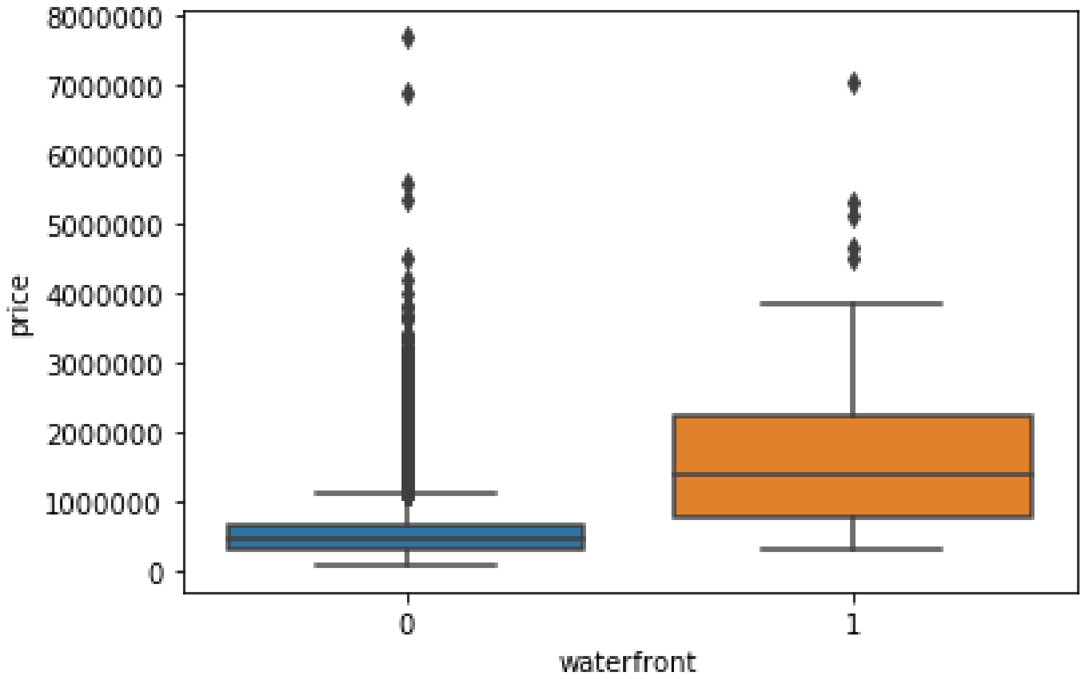
```

```

[77]:

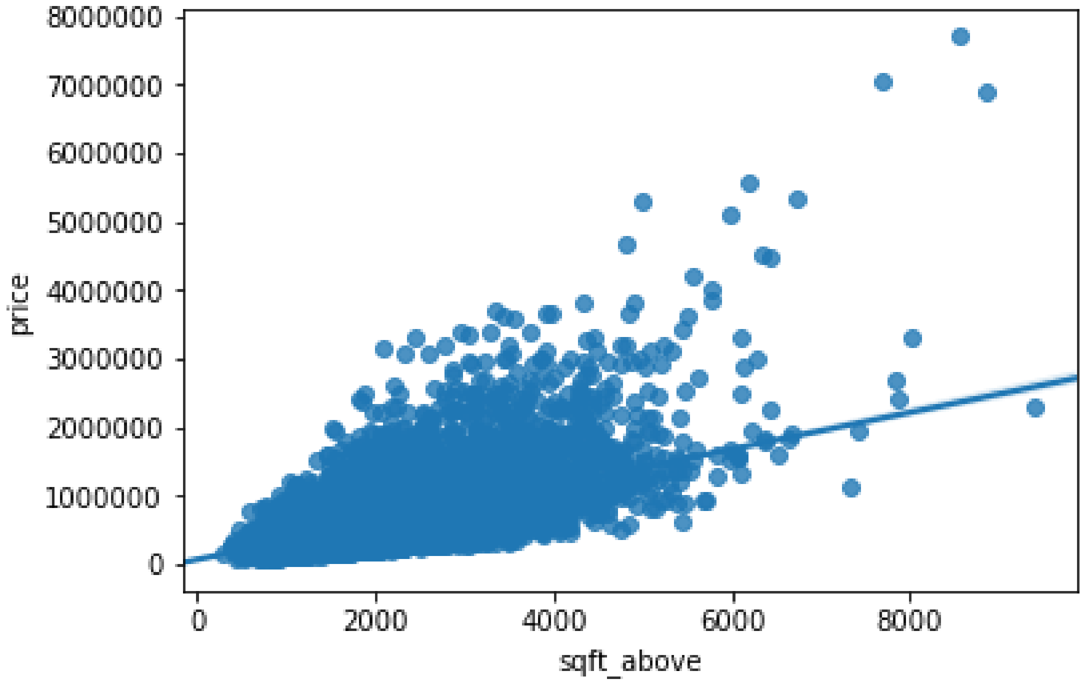
```
   waterfront     price
0           0  221900.0
1           0  538000.0
2           0  180000.0
3           0  604000.0
4           0  510000.0
```

[82]:

```python
ax = sns.boxplot(x="waterfront", y="price", data=df_H1)
```



[83]:
```
ax = sns.regplot(x="sqft_above", y="price", data=df_H1)
```



[84]:

```
from sklearn.linear_model import LinearRegression as ln
```

[89]:

```
X=df_H1[['sqft_above']]
Y=df_H1['price']

reg = ln().fit(X, Y)
reg.score(X, Y)
```

[89]:

```
0.36671175283827917
```

[90]:

```
X=df_H1[['floors','waterfront','lat','bedrooms','sqft_basement','view','bathrooms','sqft_livi
Y=df_H1['price']

reg = ln().fit(X, Y)
reg.score(X, Y)
```

[90]:

```
0.6577086983978812
```

[122]:

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

Input=[('scale',StandardScaler()), ('polynomial',␣
 ↪PolynomialFeatures(include_bias=False)), ('model',ln())]
pipe=Pipeline(Input)
pipe
pipe.fit(X,Y)
ypipe=pipe.predict(X)
from sklearn.metrics import r2_score
r_squared = r2_score(Y, ypipe)
print('The R-square value is: ', r_squared)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-
packages/sklearn/preprocessing/data.py:625: DataConversionWarning: Data with
input dtype int64, float64 were all converted to float64 by StandardScaler.
  return self.partial_fit(X, y)
/home/jupyterlab/conda/envs/python/lib/python3.6/site-
packages/sklearn/base.py:465: DataConversionWarning: Data with input dtype
int64, float64 were all converted to float64 by StandardScaler.
  return self.fit(X, y, **fit_params).transform(X)
```

```
The R-square value is:  0.7513458495077826
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-
packages/sklearn/pipeline.py:331: DataConversionWarning: Data with input dtype
int64, float64 were all converted to float64 by StandardScaler.
  Xt = transform.transform(Xt)
```

[131]:

```
from sklearn.model_selection import train_test_split

pr=PolynomialFeatures(degree=2)

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,␣
 ↪random_state=0)

x_train_pr=pr.fit_transform(x_train)
x_test_pr=pr.fit_transform(x_test)
```

[132]:

```
from sklearn.linear_model import Ridge
RigeModel=Ridge(alpha=0.1)
```

[135]:

```
RigeModel.fit(x_train_pr, y_train)
yhat =RigeModel.predict(x_test_pr)
r_squared = r2_score(y_test, yhat)
print('The R-square value is: ', r_squared)
```

```
The R-square value is:  0.7307388996885178
```

[ ]:

```

```