In [1]:
```
import pandas as pd
```

In [2]:
```
df=pd.read_csv("kc_house_data.csv")
```

In [4]:
```
df
```

Out[4]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	1955	0	98178
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	1951	1991	98125
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	1933	0	98028
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	1965	0	98136
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	1987	0	98074
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
21608	263000018	20140521T000000	360000.0	3	2.50	1530	1131	3.0	0	0	...	8	1530	0	2009	0	98103
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	5813	2.0	0	0	...	8	2310	0	2014	0	98146
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	1350	2.0	0	0	...	7	1020	0	2009	0	98144
21611	291310100	20150116T000000	400000.0	3	2.50	1600	2388	2.0	0	0	...	8	1600	0	2004	0	98027
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	1076	2.0	0	0	...	7	1020	0	2008	0	98144

21613 rows × 21 columns

In [5]:
```
# print first twenty rows of the dataset
print(df.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   
5   7237550310  20140512T000000  1225000.0         4       4.50         5420   
6   1321400060  20140627T000000   257500.0         3       2.25         1715   
7   2008000270  20150115T000000   291850.0         3       1.50         1060   
8   2414600126  20150415T000000   229500.0         3       1.00         1780   
9   3793500160  20150312T000000   323000.0         3       2.50         1890   
10  1736800520  20150403T000000   662500.0         3       2.50         3560   
11  9212900260  20140527T000000   468000.0         2       1.00         1160   
12   114101516  20140528T000000   310000.0         3       1.00         1430   
13  6054650070  20141007T000000   400000.0         3       1.75         1370   
14  1175000570  20150312T000000   530000.0         5       2.00         1810   
15  9297300055  20150124T000000   650000.0         4       3.00         2950   
16  1875500060  20140731T000000   395000.0         3       2.00         1890   
17  6865200140  20140529T000000   485000.0         4       1.00         1600   
18    16000397  20141205T000000   189000.0         2       1.00         1200   
19  7983200060  20150424T000000   230000.0         3       1.00         1250   

    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   
5     101930     1.0           0     0  ...     11        3890           1530   
6       6819     2.0           0     0  ...      7        1715              0   
7       9711     1.0           0     0  ...      7        1060              0   
8       7470     1.0           0     0  ...      7        1050            730   
9       6560     2.0           0     0  ...      7        1890              0   
10      9796     1.0           0     0  ...      8        1860           1700   
11      6000     1.0           0     0  ...      7         860            300   
12     19901     1.5           0     0  ...      7        1430              0   
13      9680     1.0           0     0  ...      7        1370              0   
14      4850     1.5           0     0  ...      7        1810              0   
15      5000     2.0           0     3  ...      9        1980            970   
16     14040     2.0           0     0  ...      7        1890              0   
17      4300     1.5           0     0  ...      7        1600              0   
18      9850     1.0           0     0  ...      7        1200              0   
19      9774     1.0           0     0  ...      7        1250              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   
5       2001             0    98053  47.6561 -122.005           4760   
6       1995             0    98003  47.3097 -122.327           2238   
7       1963             0    98198  47.4095 -122.315           1650   
8       1960             0    98146  47.5123 -122.337           1780   
9       2003             0    98038  47.3684 -122.031           2390   
10      1965             0    98007  47.6007 -122.145           2210   
11      1942             0    98115  47.6900 -122.292           1330   
12      1927             0    98028  47.7558 -122.229           1780   
13      1977             0    98074  47.6127 -122.045           1370   
14      1900             0    98107  47.6700 -122.394           1360   
15      1979             0    98126  47.5714 -122.375           2140   
16      1994             0    98019  47.7277 -121.962           1890   
17      1916             0    98103  47.6648 -122.343           1610   
18      1921             0    98002  47.3089 -122.210           1060   
19      1969             0    98003  47.3343 -122.306           1280   

    sqft_lot15  
0         5650  
1         7639  
2         8062  
3         5000  
4         7503  
5       101930  
6         6819  
7         9711  
8         8113  
9         7570  
10        8925  
11        6000  
12       12697  
13       10208  
14        4850  
15        4000  
16       14018  
17        4300  
18        5095  
19        8850  

[20 rows x 21 columns]
```

In [6]:
```
# print first six colums of the dataset
print(df.head(6))
```

```
           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   
5  7237550310  20140512T000000  1225000.0         4       4.50         5420   

   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   
5    101930     1.0           0     0  ...     11        3890           1530   

   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   
5      2001             0    98053  47.6561 -122.005           4760   

   sqft_lot15  
0        5650  
1        7639  
2        8062  
3        5000  
4        7503  
5      101930  

[6 rows x 21 columns]
```

In [7]:
```
# print shape of the datasest
print(df.shape)
```

```
(21613, 21)
```

In [8]:
```
df.dtypes
```

Out[8]:
```
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
```

In [9]:
```
df.describe()
```

Out[9]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_ba
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.409430	7.656873	1788.390691	291
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.650743	1.175459	828.090978	442
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	1.000000	290.000000	0
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.000000	7.000000	1190.000000	0
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.000000	7.000000	1560.000000	0
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	4.000000	8.000000	2210.000000	560
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	13.000000	9410.000000	4820

In [11]:
```
df.drop(["id","grade" ], axis=1, inplace= False)
df.describe()
```

Out[11]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_ba
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.409430	7.656873	1788.390691	291
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.650743	1.175459	828.090978	442
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	1.000000	290.000000	0
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.000000	7.000000	1190.000000	0
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.000000	7.000000	1560.000000	0
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	4.000000	8.000000	2210.000000	560
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	13.000000	9410.000000	4820

In [ ]:
```
import numpy as np
```

In [6]:
```
import seaborn as sns
```

In [9]:
```
import pandas as pd
```

In [10]:
```
df=pd.read_csv("kc_house_data.csv")
```

In [11]:
```
df['bedrooms'].value_counts().to_frame()
```

Out[11]:

	bedrooms
3	9824
4	6882
2	2760
5	1601
6	272
1	199
7	38
0	13
8	13
9	6
10	3
11	1
33	1

In [12]:
```
import numpy as np
```

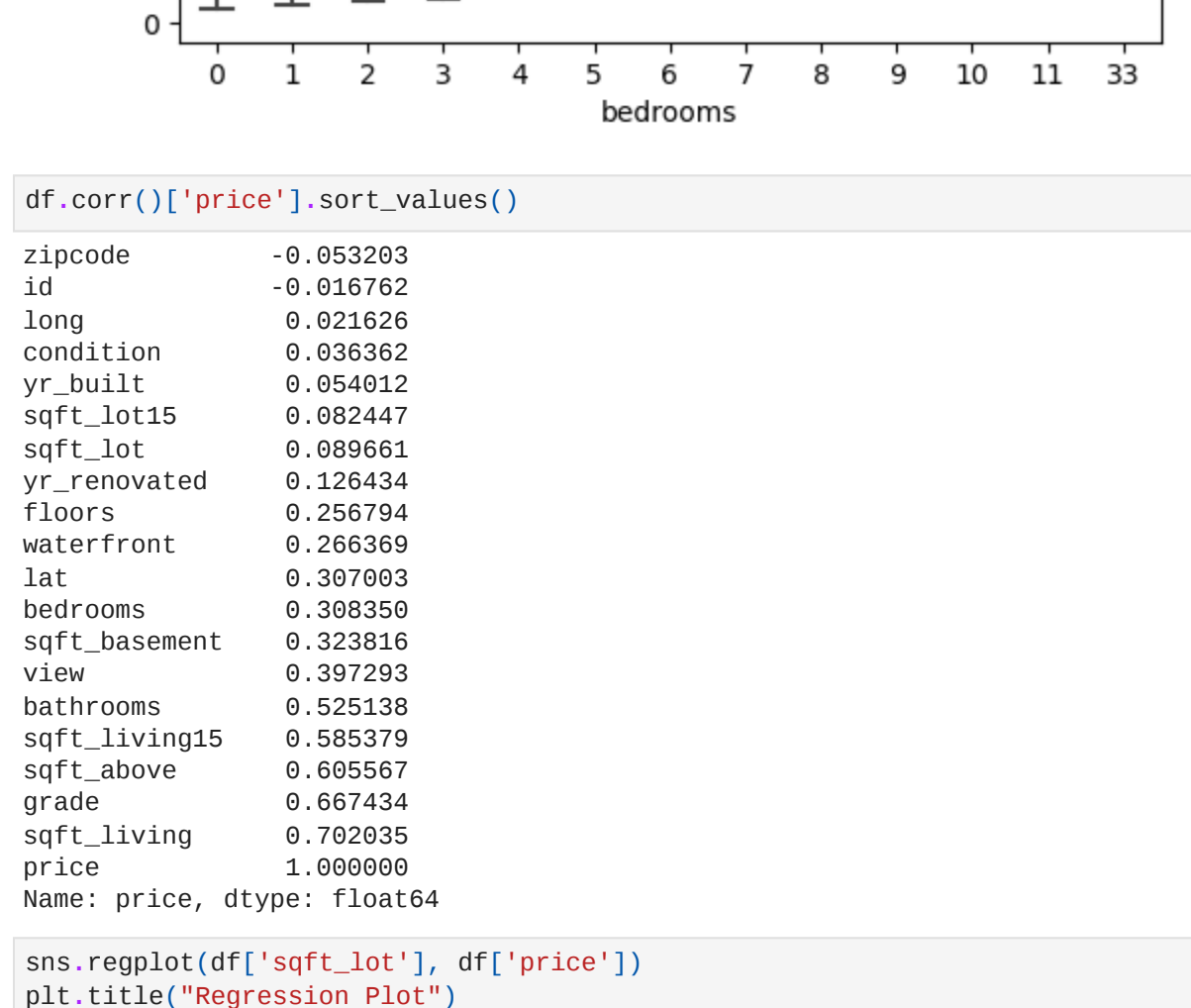
In [13]:
```
import seaborn as sns
```

In [14]:
```
import matplotlib.pyplot as plt
```

In [45]:
```
sns.boxplot(df['bedrooms'], df['sqft_living'])
plt.title("'boxplot 'bedrooms' vs 'sqft_living'")
```

```
C:\Users\Milind Vyas\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[45]:
```
Text(0.5, 1.0, "'boxplot 'bedrooms' vs 'sqft_living'")
```

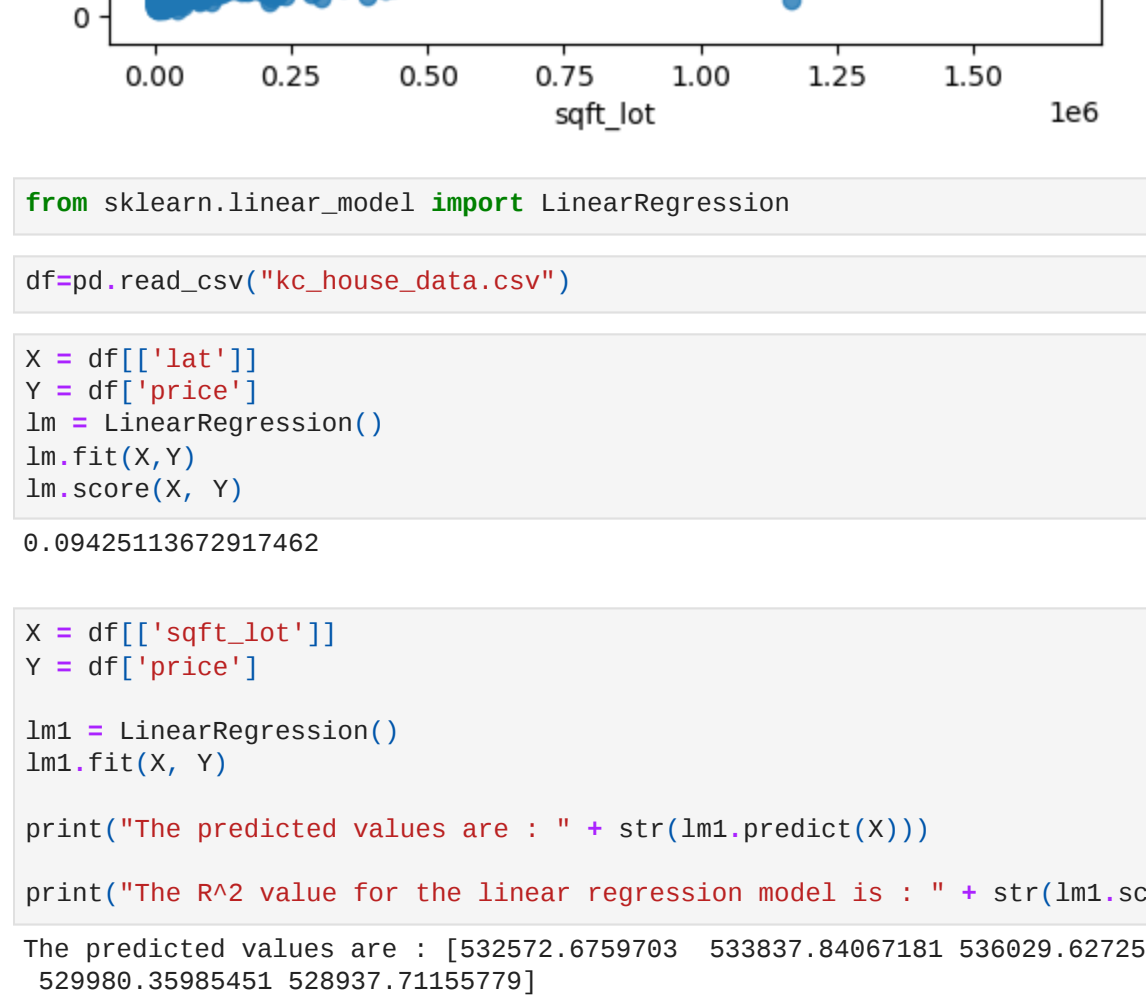


In [17]:
```
df.corr()['price'].sort_values()
```

Out[17]:
```
zipcode         -0.053203
id              -0.016762
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
```

In [46]:
```
sns.regplot(df['sqft_lot'], df['price'])
plt.title("Regression Plot")
print("We can see that it is positively correlated.")
```

```
C:\Users\Milind Vyas\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
We can see that it is positively correlated.
```



In [18]:
```
from sklearn.linear_model import LinearRegression
```

In [19]:
```
df=pd.read_csv("kc_house_data.csv")
```

In [20]:
```
X = df[['lat']]
Y = df['price']
lm = LinearRegression()
lm.fit(X,Y)
lm.score(X, Y)
```

Out[20]:
```
0.09425113672917462
```

In [21]:
```
X = df[['sqft_lot']]
Y = df['price']

lm1 = LinearRegression()
lm1.fit(X, Y)

print("The predicted values are : " + str(lm1.predict(X)))

print("The R^2 value for the linear regression model is : " + str(lm1.score(X, Y)))
```

```
The predicted values are : [532572.6759703  533837.84067181 536029.62725897 ... 529155.45975391
 529980.35985451 528937.71155779]
The R^2 value for the linear regression model is : 0.00803906699212195
```

In [23]:
```
from sklearn.preprocessing import PolynomialFeatures
```

In [24]:
```
features = ["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]
```

In [25]:
```
Y = df['price']
lm2 = LinearRegression()
lm2.fit(df[features], Y)
```

Out[25]:
```
LinearRegression()
```

In [26]:
```
print("The predicted values are mentioned as : " + str(lm2.predict(df[features])))

print("The R^2 value is : " + str(lm2.score(df[features], Y)))
```

```
The predicted values are mentioned as : [283413.07663037 662377.8434902  305956.87442548 ... 304309.00758763
 428626.84540527 304107.09789894]
The R^2 value is : 0.6577151058279327
```

In [33]:
```
from sklearn.preprocessing import StandardScaler
```

In [36]:
```
# first element in the tuple contains the name of the estimator:
'scale'

'polynomial'

'model'

# The second element in the tuple contains the model constructor

StandardScaler()

PolynomialFeatures(include_bias=False)

LinearRegression()
```

Out[36]:
```
LinearRegression()
```

In [37]:
```
Input=[('scale',StandardScaler()),('polynomial', PolynomialFeatures(include_bias=False)),('model',LinearRegression())]
```

In [39]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
print("done")
```

```
done
```

In [41]:
```
features =["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]
X = df[features]
Y = df['condition']

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
```

In [42]:
```
from sklearn.linear_model import Ridge
```

In [43]:
```
RidgeModel = Ridge(alpha = 0.1)
RidgeModel.fit(x_train, y_train)
print("The predicted values are : " + str(RidgeModel.predict(x_test)))

print("The R^2 Score value is mentioned as : " + str(RidgeModel.score(x_test, y_test)))
```

```
The predicted values are : [3.17016287 3.49847636 3.02726744 ... 3.57411953 3.65807044 3.23609822]
The R^2 Score value is mentioned as : 0.08731621572148174
```

In [47]:
```
from sklearn.preprocessing import PolynomialFeatures
```

In [48]:
```
pr = PolynomialFeatures(degree = 2)
x_train_pr = pr.fit_transform(x_train[features])
x_test_pr = pr.fit_transform(x_test[features])

RidgeModel = Ridge(alpha = 0.1)
RidgeModel.fit(x_train_pr, y_train)
yhat = RidgeModel.predict(x_test_pr)

print("The R^2 Score value for the training data is : " + str(RidgeModel.score(x_train_pr, y_train)))
print("The R^2 Score value for the testing data is : " + str(RidgeModel.score(x_test_pr, y_test)))
```

```
The R^2 Score value for the training data is : 0.11547391258349926
The R^2 Score value for the testing data is : 0.109844035373724
```

In [50]:
```
exit()
```

In [ ]:
```
```