Predicting house price

LINEAR REGRESSION - KC_HOUSE_DATA

16102275 Park Hyun Woo

CONTEXT

1. Variable Selection & Data preprocess
2. 4 Assumptions of Linear Regression
3. Summary

VARIABLE SELECTION & DATA PREPROCESS

PURPOSE OF ANALYSIS

- **Based on information and data on housing, I want to make a modeling predict price of house by using Linear Regression.**
- **There are some explanatory variables (input variables) and target value is "price"**

INFORMATION OF VARIABLES

There are 21 variables in this dataset

I don't know information about some explanatory variables.

So I want to know mean of them

- sqft_living : the area of Residential space
- sqft_lot : the area of site
- sqft_above : the area except for the basement
- sqft_basement : the area of underground
- Waterfront : River-view
- lat : latitude
- Long : longitude
- Grade : class of the house in King Country

VARIABLE TRANSFORM

```
house.insert(1,'recent_built',0)
house['recent_built'] = house.apply(lambda x : x['yr_built']
                        if (x['yr_built']>=x['yr_renovated'])
                        else x['yr_renovated'],axis=1)
house.drop(columns= ['yr_built','yr_renovated'])
```

	id	recent_built	date	price	bedrooms	bathrooms	sqft_
0	7129300520	1955	20141013T000000	221900.0	3	1.00	
1	6414100192	1991	20141209T000000	538000.0	3	2.25	
2	5631500400	1933	20150225T000000	180000.0	2	1.00	
3	2487200875	1965	20141209T000000	604000.0	4	3.00	
4	1954400510	1987	20150218T000000	510000.0	3	2.00	
...	...	...	...	...	...	...	
21608	263000018	2009	20140521T000000	360000.0	3	2.50	
21609	6600060120	2014	20150223T000000	400000.0	4	2.50	
21610	1523300141	2009	20140623T000000	402101.0	2	0.75	
21611	291310100	2004	20150116T000000	400000.0	3	2.50	
21612	1523300157	2008	20141015T000000	325000.0	2	0.75	

- **I think 'year_built' column and 'year_renovated' column can be combined with 'recent_built. Because renovated house mean re-built house for fixing and renovating, so this is same with 'year_built'**
- **So I make new column 'recent_built' and not renovated house ('year_renovate' =0) have year_built value in 'recent_built' and renovated house have year_renovated value in 'recent_built'.**

VARIABLE SELECTION

- **Date is not numerical but object of date format. So I don't use 'Date' Column**
- **And I think latitude,longitude,zipcode is not affect to House'price**

```
zipcode describe
 count     21613.000000
mean      98077.939805
std          53.505026
min       98001.000000
25%       98033.000000
50%       98065.000000
75%       98118.000000
max       98199.000000
Name: zipcode, dtype: float64
```

```
lat describe
 count     21613.000000
mean          47.560053
std            0.138564
min           47.155900
25%           47.471000
50%           47.571800
75%           47.678000
max           47.777600
Name: lat, dtype: float64
```

```
long describe
 count     21613.000000
mean       -122.213896
std           0.140828
min        -122.519000
25%        -122.328000
50%        -122.230000
75%        -122.125000
max        -121.315000
Name: long, dtype: float64
```

There are zipcode, lat and long's describe.
The describes show there is a little difference each column's datas(location)
So I don't use these columns in Linear Regression Model

VARIABLE SELECTION

- **Before deleting outliers, I select variable first. Because if I delete outlier before variable selection, there are many datas which is deleted before use.**

```
corr = house[['price','date','sqft_lot','recent_built','bedrooms','grade','sqft_above','bathrooms','sqft_lot','condi
corr['price']
```

```
price            1.000000
sqft_lot         0.091812
recent_built     0.103392
bedrooms         0.313419
grade            0.678333
sqft_above       0.596404
bathrooms        0.519106
sqft_lot         0.091812
condition        0.041264
sqft_living      0.692901
sqft_basement    0.309669
floors           0.265588
view             0.390700
waterfront       0.230223
sqft_living15    0.599003
sqft_lot15       0.083281
Name: price, dtype: float64
```

- **Above numbers represents correlation between price and other columns.**
- **'Grade', 'sqft_above', 'bathrooms','sqft_living', 'view' and 'sqft_living15' have higher correlation than other columns**

DELETING OUTLIERS

- By using IQR, I want to delete outliers first. But I have some problem in that way I explain that after commenting IQR

```
def find_outlier(data):
    Q1 , Q3 = np.percentile(data,[25,75])
    IQR = Q3 - Q1
    Over_outlier = Q3 + 1.5*IQR
    Low_outlier = Q1 - 1.5*IQR
    location = np.where((data>Over_outlier)|(data<Low_outlier))
    result = [list(location[0]),len(list(location[0]))]
    return result
```

- Q1, Q3 each mean value of 25 percent of datas and value of 75 percent of datas
- IQR = Q3 - Q1 and Over_Outlier is Q3 + 1.5*IQR and Low_Outlier is Q1 - 1.5*IQR

I want to find outlier's index, so I make above function

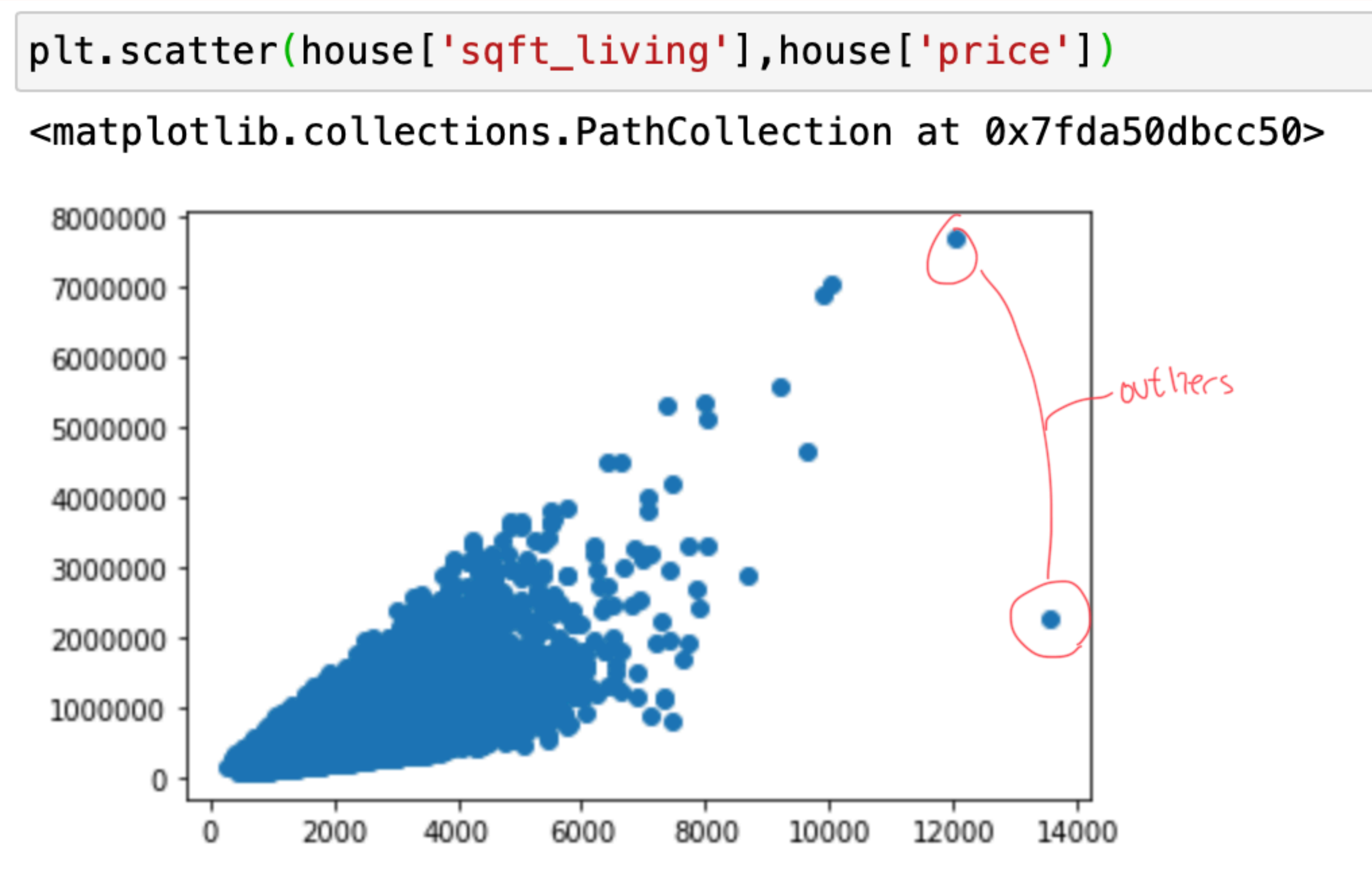
DELETING OUTLIERS

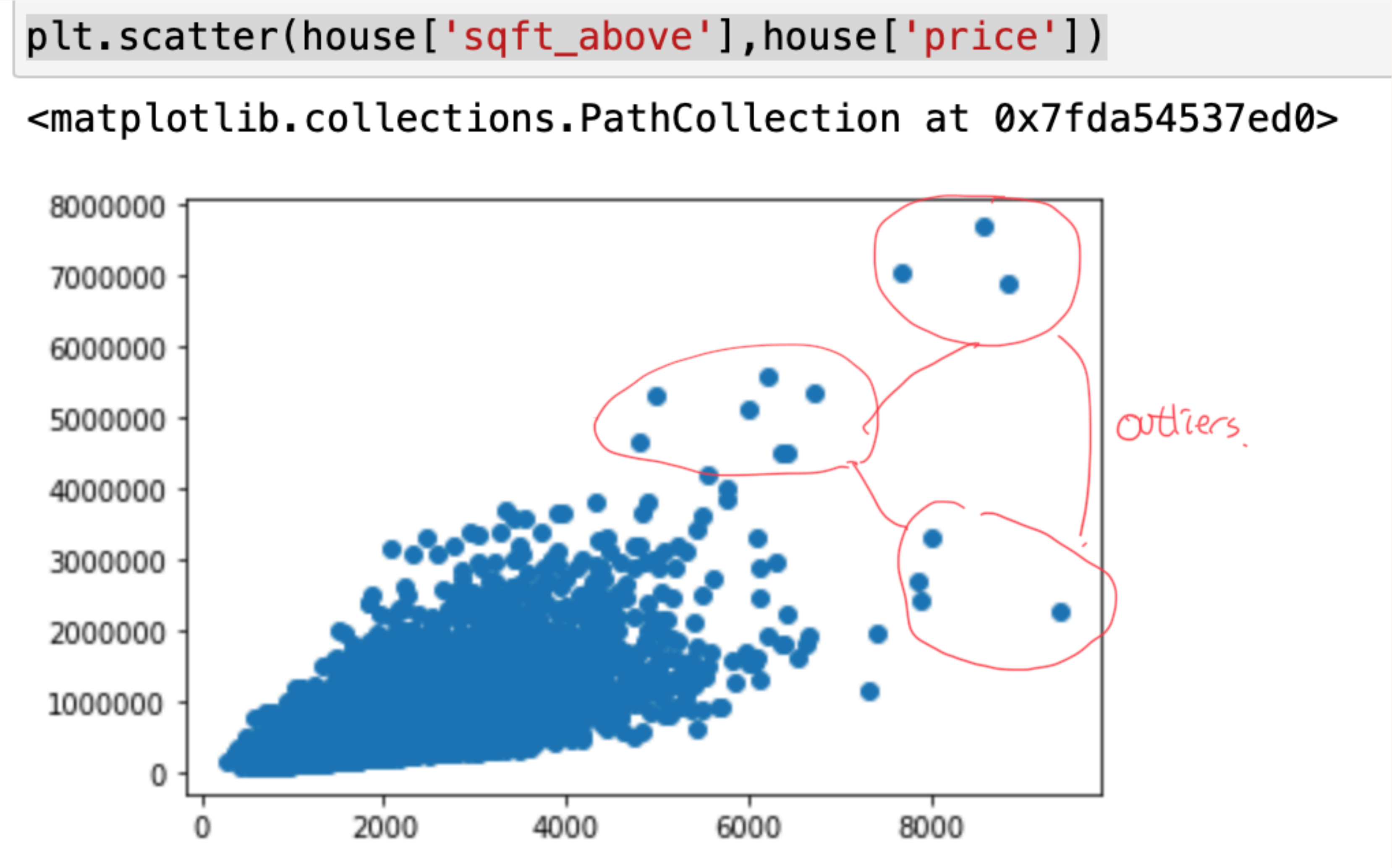
```
print("Grade column's Number of Outliers by IQR : " ,find_outlier(house['grade'])[1])
print("\nsqft_above column's Number of Outliers by IQR : " ,find_outlier(house['sqft_above'])[1])
print("\nbathrooms column's Number of Outliers by IQR : " ,find_outlier(house['bathrooms'])[1])
print("\nsqft_living column's Number of Outliers by IQR : " ,find_outlier(house['sqft_living'])[1])
print("\nview column's Number of Outliers by IQR : " ,find_outlier(house['view'])[1])
print("\nsqft_living15 column's Number of Outliers by IQR : " ,find_outlier(house['sqft_living15'])[1])
```

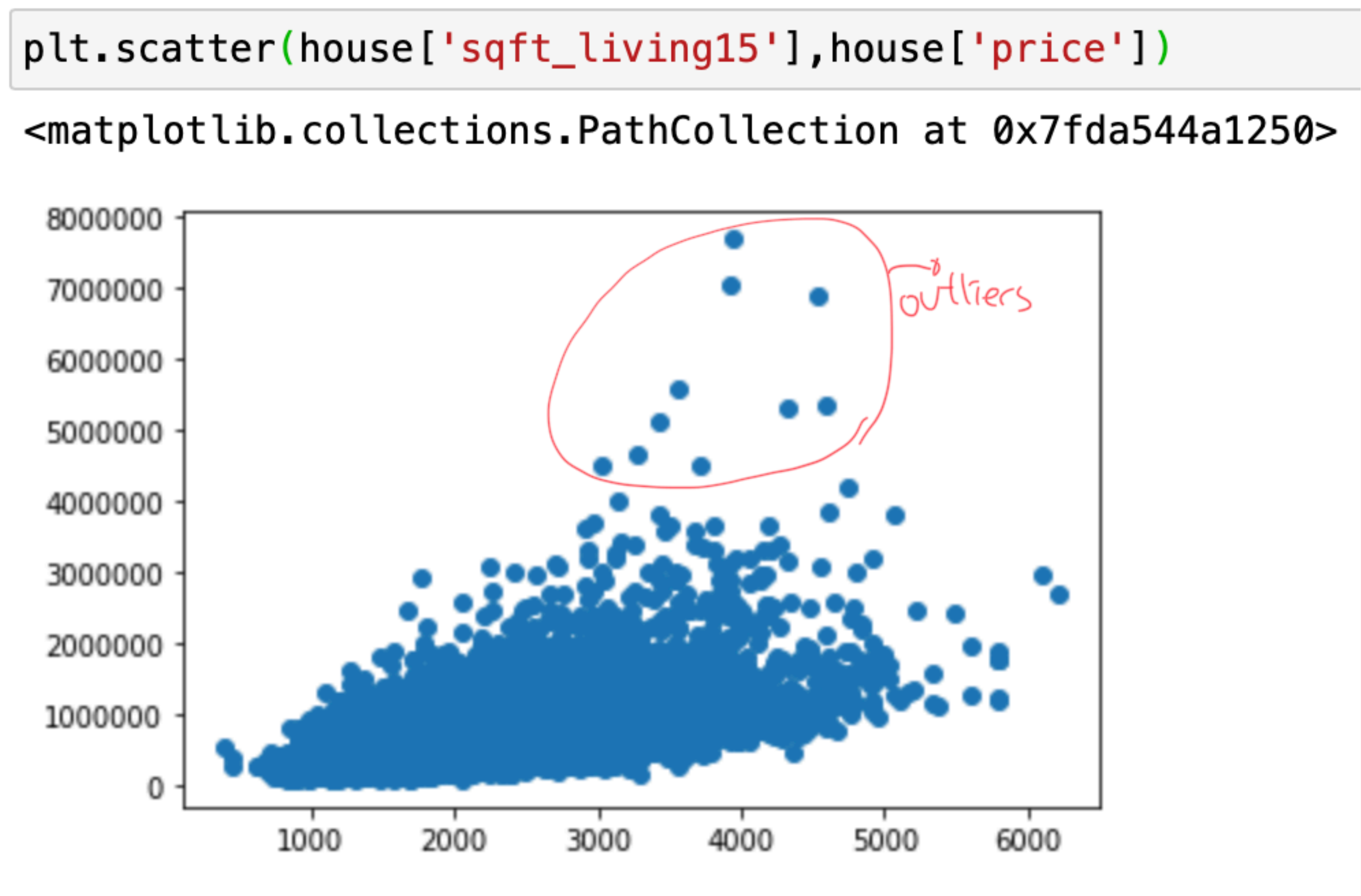
```
Grade column's Number of Outliers by IQR :  1911

sqft_above column's Number of Outliers by IQR :  611

bathrooms column's Number of Outliers by IQR :  571

sqft_living column's Number of Outliers by IQR :  572

view column's Number of Outliers by IQR :  2124

sqft_living15 column's Number of Outliers by IQR :  544
```

- **Those number show the number of Outliers (Over_Outlier + Low_Outlier)**
- **Number of our data is 21612, If I delete data by using IQR at least 3000 data are removed this is 15% of whole data. I think this is not good when modeling.**
- **So I remove outliers in a different way.**

DELETING OUTLIERS





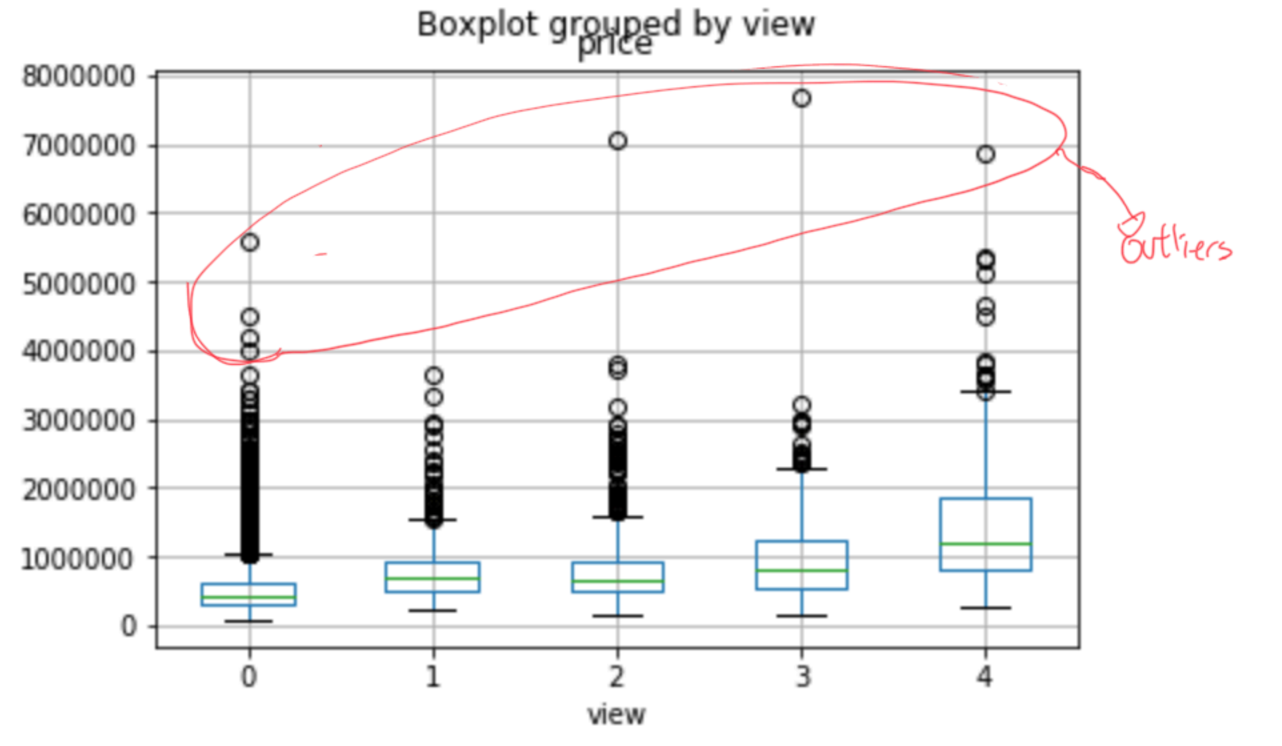


- **Those variables(sqft_living, sqft_above, sqft_living15) are continuous variable. So I make scatter plot between those columns and price. There are some outliers (when I see) that stand out. So I delete those datas**

DELETING OUTLIERS

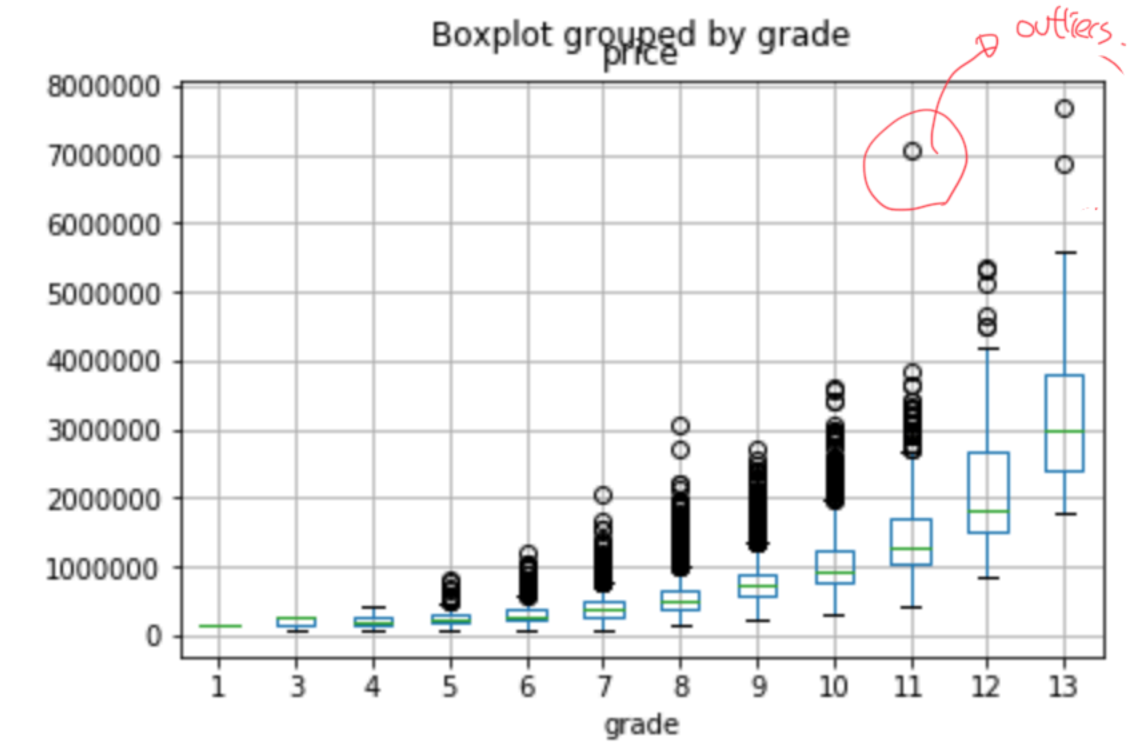
```
house.boxplot(column=['price'], by='view')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fda54466590>

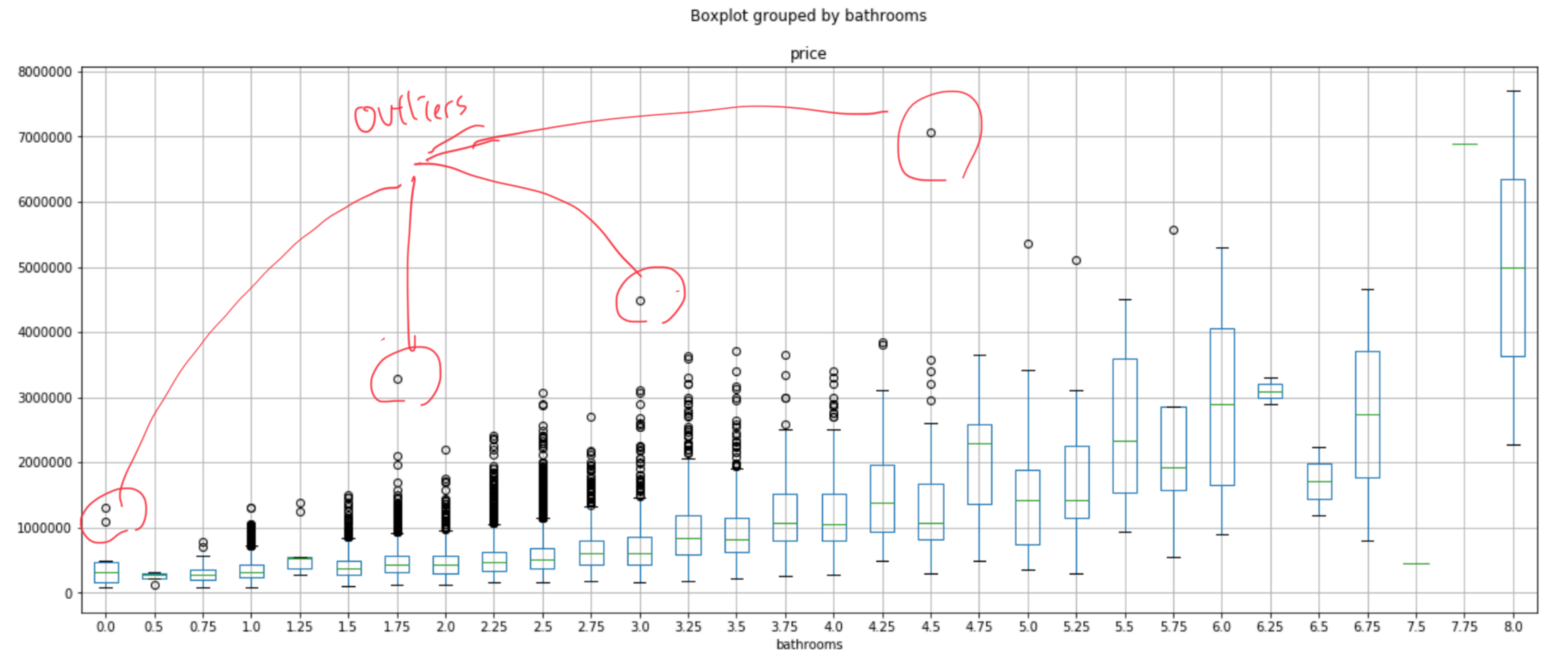



```
house.boxplot(column=['price'], by='grade')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fda5439a5d0>



<matplotlib.axes._subplots.AxesSubplot at 0x7fda544fe5d0>



- **Those variables (view, bathrooms, grade) are discrete variable. And I make box plot x is variable and y is price. So I find some outliers. So I delete them**

DELETING OUTLIERS

```
outliers which is deleted
house['price']>3500000].index),len(house[house['sqft_living']>8000].index),len(house[house['sqft_above']>7000].index
```

```
(22, 9, 9)
```

```
house.drop(house[(house['price']>3000000) & (house['bathrooms']==1.75)|(house['price']>1000000)&(house['bathrooms']=
house.drop(house[house['price']>3500000].index,inplace=True)
house.drop(house[house['sqft_living']>8000].index,inplace=True)
house.drop(house[house['sqft_above']>7000].index,inplace=True)
```

- **22+9+9 = 40 => this is number of outliers which is deleted**
- **Those code made for deleting outliers**

4 ASSUMPTIONS OF LINEAR REGRESSION

IN MODELING, HAVE A COEFFICIENT PROBLEM

```
import statsmodels.api as sm
X = house[['sqft_living','sqft_above','sqft_living15','view','grade','bathrooms']]
y = house['price']

X = sm.add_constant(X)
model = sm.OLS(y,X)
result = model.fit()
result.summary()
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.576
Model:	OLS	**Adj. R-squared:**	0.576
Method:	Least Squares	**F-statistic:**	4878.
Date:	Fri, 17 Sep 2021	**Prob (F-statistic):**	0.00
Time:	18:29:27	**Log-Likelihood:**	-2.9617e+05
No. Observations:	21581	**AIC:**	5.924e+05
Df Residuals:	21574	**BIC:**	5.924e+05
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-5.728e+05	1.22e+04	-47.035	0.000	-5.97e+05	-5.49e+05
sqft_living	178.5866	4.185	42.669	0.000	170.383	186.790
sqft_above	-47.9991	4.089	-11.739	0.000	-56.014	-39.985
sqft_living15	23.3661	3.643	6.414	0.000	16.226	30.506
view	8.44e+04	2121.413	39.783	0.000	8.02e+04	8.86e+04
grade	1.063e+05	2207.741	48.131	0.000	1.02e+05	1.11e+05
bathrooms	-2.624e+04	3071.981	-8.541	0.000	-3.23e+04	-2.02e+04

Omnibus:	10068.912	**Durbin-Watson:**	1.980
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	114415.088
Skew:	1.953	**Prob(JB):**	0.00
Kurtosis:	13.582	**Cond. No.**	2.97e+04

paste cells below

- **Except for sqft_above and bathrooms, all coefficient of values are positive. This is understandable.**
- **But coefficient of sqft_above and bathrooms are negative. This is not understandable. Because house price and sqft_above&bathroom have positive correlation, which is common sense.**
- **So After VIF calculation, I decide drop the biggest VIF value variable.**

1.TESTING MULTICOLLINEARITY

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
from patsy import dmatrices


y, X = dmatrices('price ~sqft_living+sqft_above+sqft_living15+view+grade+bathrooms',house, return_type = 'dataframe'
vif = pd.DataFrame()
vif["vif value"] = [variance_inflation_factor(X.values,i) for i in range(7)]
vif["explanatory variables"] = X.columns
vif
```

	vif value	explanatory variables
0	65.461179	Intercept
1	6.349627	sqft_living
2	5.020987	sqft_above
3	2.717812	sqft_living15
4	1.157337	view
5	2.966037	grade
6	2.452879	bathrooms

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
from patsy import dmatrices

y, X = dmatrices('price ~sqft_above+sqft_living15+view+grade+bathrooms',house,
vif = pd.DataFrame()
vif["vif value"] = [variance_inflation_factor(X.values,i) for i in range(6)]
vif["explanatory variables"] = X.columns
vif
```

	vif value	explanatory variables
0	64.127336	Intercept
1	3.231861	sqft_above
2	2.555716	sqft_living15
3	1.106683	view
4	2.938565	grade
5	2.088632	bathrooms

- **This is output of VIF calculation.**
- **Those explanatory variable's VIF value are lower than 10.**
- **But in previous slide, the problem of coefficient I delete 'sqft_living' variable.**

2.TESTING LINEARITY

	coef	std err	t	P>|t|	[0.025	0.975]
const	-6.617e+05	1.37e+04	-48.432	0.000	-6.88e+05	-6.35e+05
sqft_above	80.1591	3.704	21.643	0.000	72.900	87.419
sqft_living15	48.5349	3.979	12.197	0.000	40.735	56.335
view	1.141e+05	2342.051	48.721	0.000	1.1e+05	1.19e+05
grade	1.139e+05	2488.019	45.790	0.000	1.09e+05	1.19e+05
bathrooms	2.975e+04	3201.413	9.294	0.000	2.35e+04	3.6e+04

- Those explantory variables's t-test p-value is almost 0
- So we can decide Each explanatory variable is significant in predicting the output value(price)
- I use those all variables in this Linear Regression model.

3. ERROR'S HOMOGENEITY

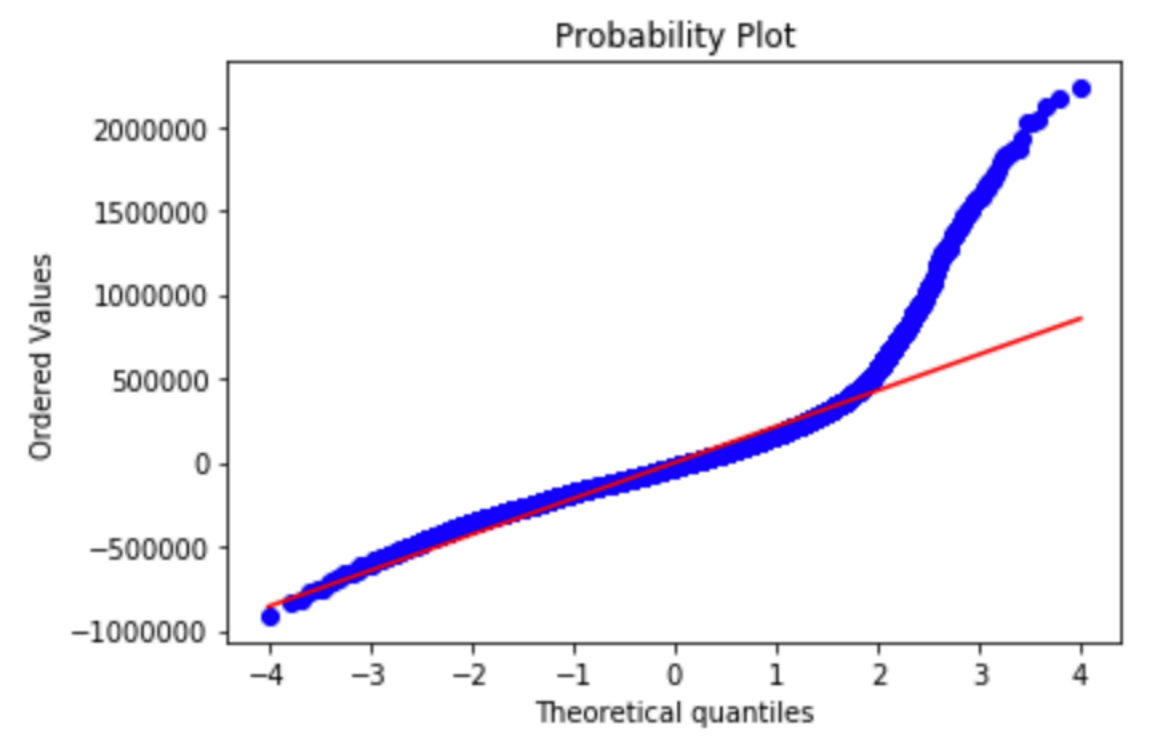
Dep. Variable:	price	**R-squared:**	0.533
Model:	OLS	**Adj. R-squared:**	0.533
Method:	Least Squares	**F-statistic:**	4940.
Date:	Fri, 17 Sep 2021	**Prob (F-statistic):**	0.00
Time:	20:16:21	**Log-Likelihood:**	-2.9937e+05
No. Observations:	21613	**AIC:**	5.987e+05
Df Residuals:	21607	**BIC:**	5.988e+05
Df Model:	5		
Covariance Type:	nonrobust		

- **Probablity for F-test (deciding error's Homogeneity) is low (almost 0). So we can decide error have homogeneity.**
- **So this model satisfy 3rd condition (Homogeneity)**

4.ERROR'S NORMALITY

Omnibus:	18289.768	Durbin-Watson:	1.972
Prob(Omnibus):	0.000	Jarque-Bera (JB):	162
Skew:	3.583	Prob(JB):	
Kurtosis:	44.864	Cond. No.	

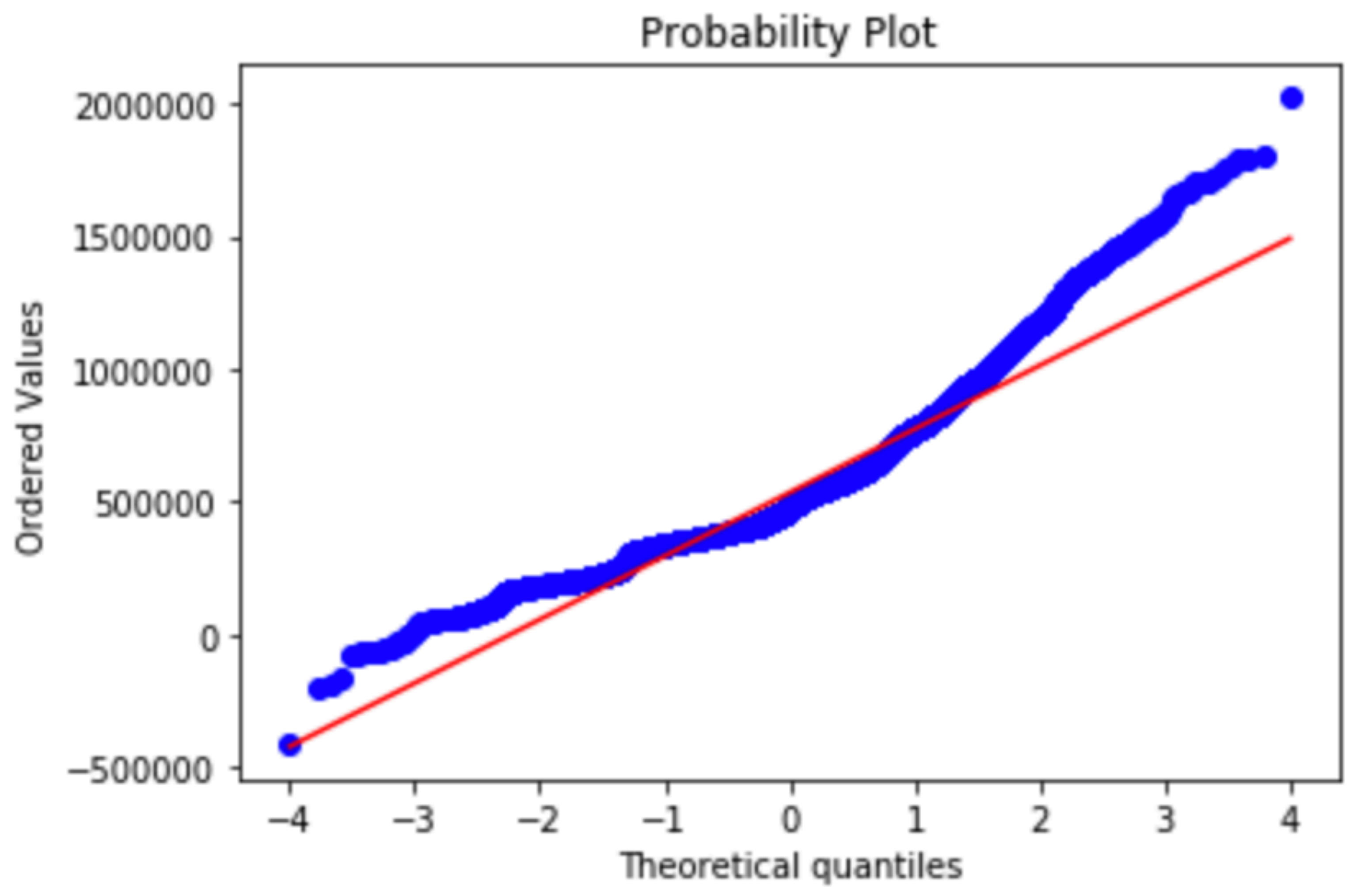
```python
# Error's Q-Q plot
import scipy.stats

scipy.stats.probplot(residual, dist=scipy.stats.norm, plot=plt)
plt.show()
```



- To decide if error have normality or not, we use Jarque-Bera
- The probability of JB-test is low(almost 0) => this mean this model's error don't follow normality. SO we can decide this model's error don't follow normality. So I draw Q-Q plot

Q-Q PLOT FOR ERROR'S NORMALITY

```python
# Error's Q-Q plot

import scipy.stats

residual = result.predict()
scipy.stats.probplot(residual, dist=scipy.stats.norm, plot=plt)
plt.show()
```



- **This Q-Q plot shows this model don't follow normality by this Q-Q plot**

MODELING BY OLS IN PYTHON

```
import statsmodels.api as sm
X = house[['sqft_above','sqft_living15','view','grade','bathrooms']]
y = house['price']

X = sm.add_constant(X)
model = sm.OLS(y,X)
result = model.fit()
result.summary()
```

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.31e+04. This might indicate that there are strong multicollinearity or other numerical problems.

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.533
Model:	OLS	**Adj. R-squared:**	0.533
Method:	Least Squares	**F-statistic:**	4940.
Date:	Fri, 17 Sep 2021	**Prob (F-statistic):**	0.00
Time:	20:16:21	**Log-Likelihood:**	-2.9937e+05
No. Observations:	21613	**AIC:**	5.987e+05
Df Residuals:	21607	**BIC:**	5.988e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-6.617e+05	1.37e+04	-48.432	0.000	-6.88e+05	-6.35e+05
sqft_above	80.1591	3.704	21.643	0.000	72.900	87.419
sqft_living15	48.5349	3.979	12.197	0.000	40.735	56.335
view	1.141e+05	2342.051	48.721	0.000	1.1e+05	1.19e+05
grade	1.139e+05	2488.019	45.790	0.000	1.09e+05	1.19e+05
bathrooms	2.975e+04	3201.413	9.294	0.000	2.35e+04	3.6e+04

Omnibus:	18289.768	**Durbin-Watson:**	1.972
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1624563.005
Skew:	3.583	**Prob(JB):**	0.00

- **Compare with previous model(After delete sqft_living column), all coefficient have positive number . It is relevant.**
- **Adjust R Squared value is 0.533**
- **This value not big.**
- **I should found problem in my analysis. The condition number is so large.**
- **So I edit some data.**

MODELING BY OLS IN PYTHON

```python
house['sqft_above'].astype('float')
house['sqft_living15'].astype('float')
house['sqft_above'] = house['sqft_above']/100
house['sqft_living15'] = house['sqft_living15']/100

reg = sm.OLS(house['price'],house[['intercept','sqft_above','sqft_living15','view','grade','bathrooms']])
result = reg.fit()
result.summary()
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.540
Model:	OLS	**Adj. R-squared:**	0.540
Method:	Least Squares	**F-statistic:**	5062.
Date:	Sat, 18 Sep 2021	**Prob (F-statistic):**	0.00
Time:	11:27:17	**Log-Likelihood:**	-2.9705e+05
No. Observations:	21581	**AIC:**	5.941e+05
Df Residuals:	21575	**BIC:**	5.942e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	-6.473e+05	1.26e+04	-51.571	0.000	-6.72e+05	-6.23e+05
sqft_above	5412.8285	345.209	15.680	0.000	4736.193	6089.464
sqft_living15	6323.6311	366.638	17.248	0.000	5604.993	7042.269
view	1.023e+05	2165.270	47.260	0.000	9.81e+04	1.07e+05
grade	1.158e+05	2287.230	50.614	0.000	1.11e+05	1.2e+05
bathrooms	2.424e+04	2952.214	8.210	0.000	1.85e+04	3e+04

Omnibus:	10534.180	**Durbin-Watson:**	1.971
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	120850.626
Skew:	2.068	**Prob(JB):**	0.00
Kurtosis:	13.830	**Cond. No.**	240.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

- **The condition number is proportion of max and min in covariance matrix**
- **If the proportion is large -> error of output is lager. So I change data by scaling for fix this problem**
- **By scaling two variables, the problem is solved.**

SUMMARY

SUMMARY

- This Linear Regression model satisfy 3 assumptions (Linearity, Error's Homogeneity and Multicollinearity) of Linear Regression. But not satisfy Error's Normality,
- Price = -6.473*10^5 + 5412.8285*sqft_above + 6323.6311*sqft_living15 + 1.023*10^5 *view + 1.158*10^5*grade + 2.424*10^4*bathrooms
- Above equation is this model's linear regression equation.
- According to coefficient, price have positive correlation with sqft_above, sqft_living + view + grade + bathrooms. Sqft_above, sqft_living15 have lower positive correlation than other 3 variables.

SUMMARY

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

X = house[['sqft_above','sqft_living15','view','grade','bathrooms']]
y = house['price']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
reg = LinearRegression()
reg.fit(X_train,y_train)
reg.score(X_test,y_test)
```

```
0.552527753927284
```

- **Accuracy of test set is 0.55**
- **This say this model represent the variance ratio of the predicted value to the variance of the acutal value is not big and low -> somewhat suitable**

THANK YOU