Business Analytics 3rd Assignment

LINEAR REGRESSION WITH CATEGORICAL VARIABLES

16102275 Park Hyun Woo

CONTEXT

1. Summary of My Previous Analysis
2. Fix Model by Changing Categorical Variables to Dummy Variables
3. Modeling Result
4. Explain About How to Use 'zipcode' , 'lat' and 'long' variables in Linear Regression Model.
    - Without Other Resources
    - With Other Resources

SUMMARY OF
MY PREVIOUS ANALYSIS

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	

- **Because value of some renovated house is 0 . This cause some problem in Analysis .So I think 'year_built' column and 'year_renovated' column can be combined with 'recent_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.**

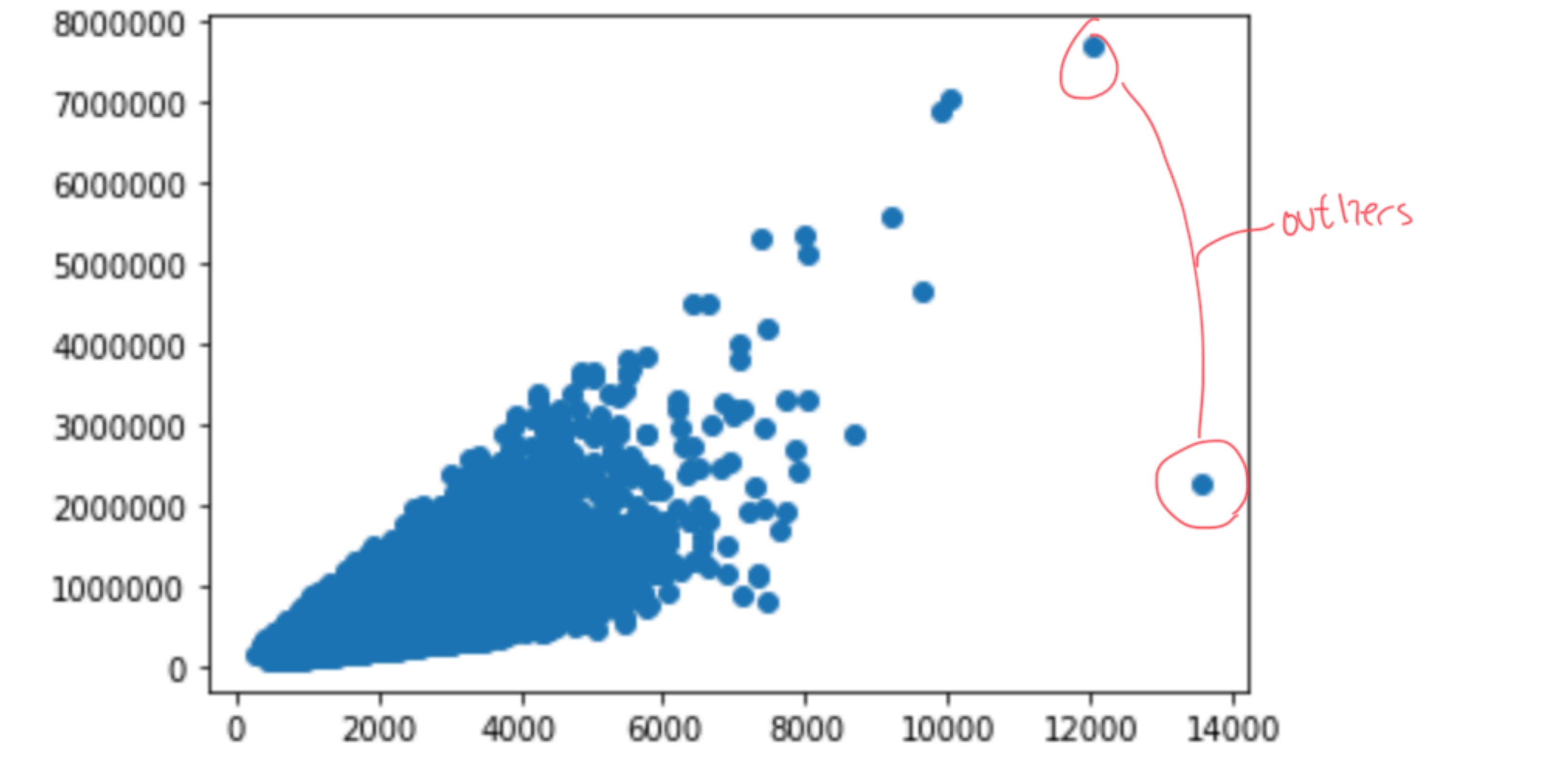
```
corr = house[['price','date','sqft_lot','recent_built','bedrooms','grade','sqft_above','bathrooms','sqft_lot','condi
corr['price']
```

```
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
```

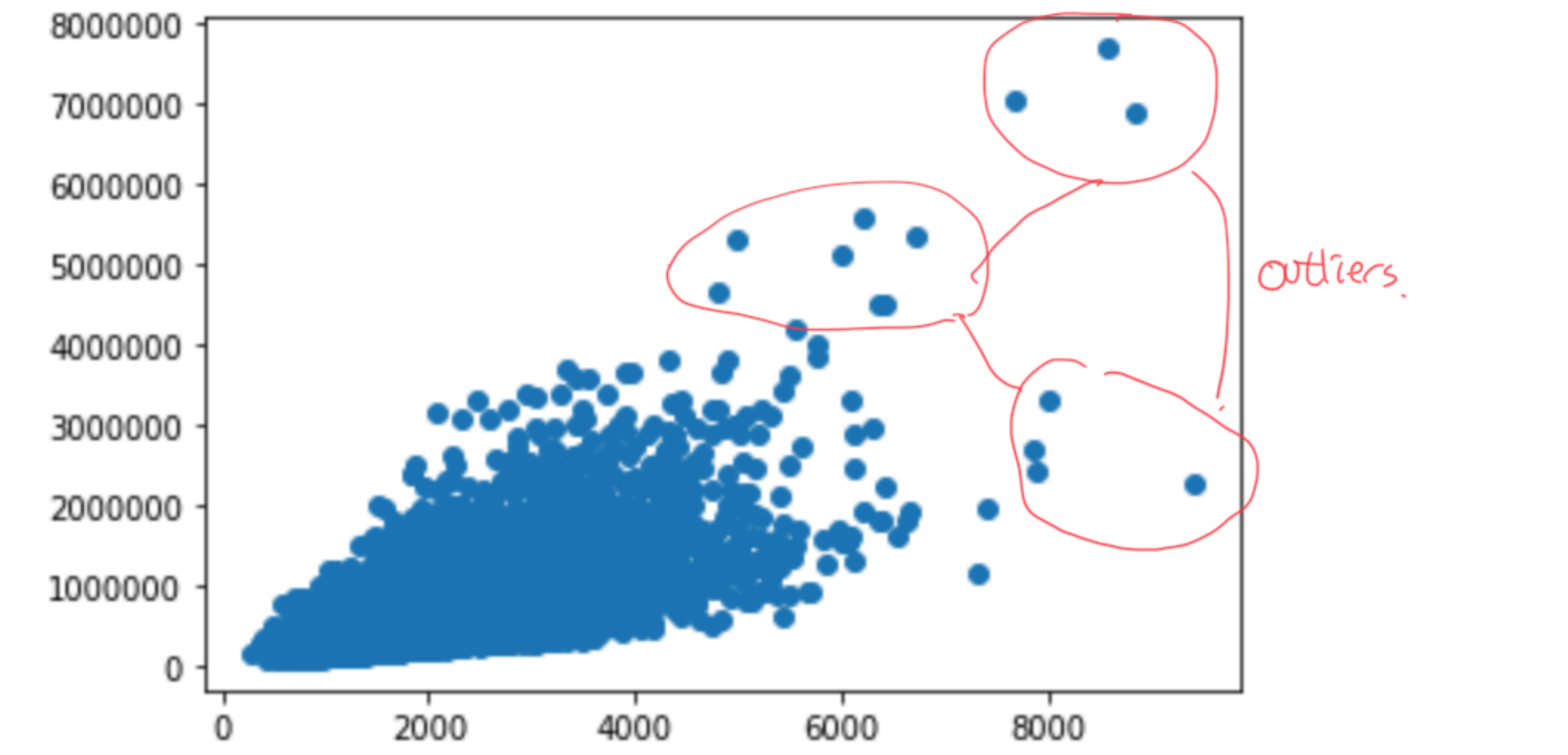
- **Above numbers represents correlation between price and other columns.**
- **'Grade', 'sqft_above', 'bathrooms','sqft_living', 'view' and 'sqft_living15' have higher correlation than other columns**

DELETING OUTLIERS

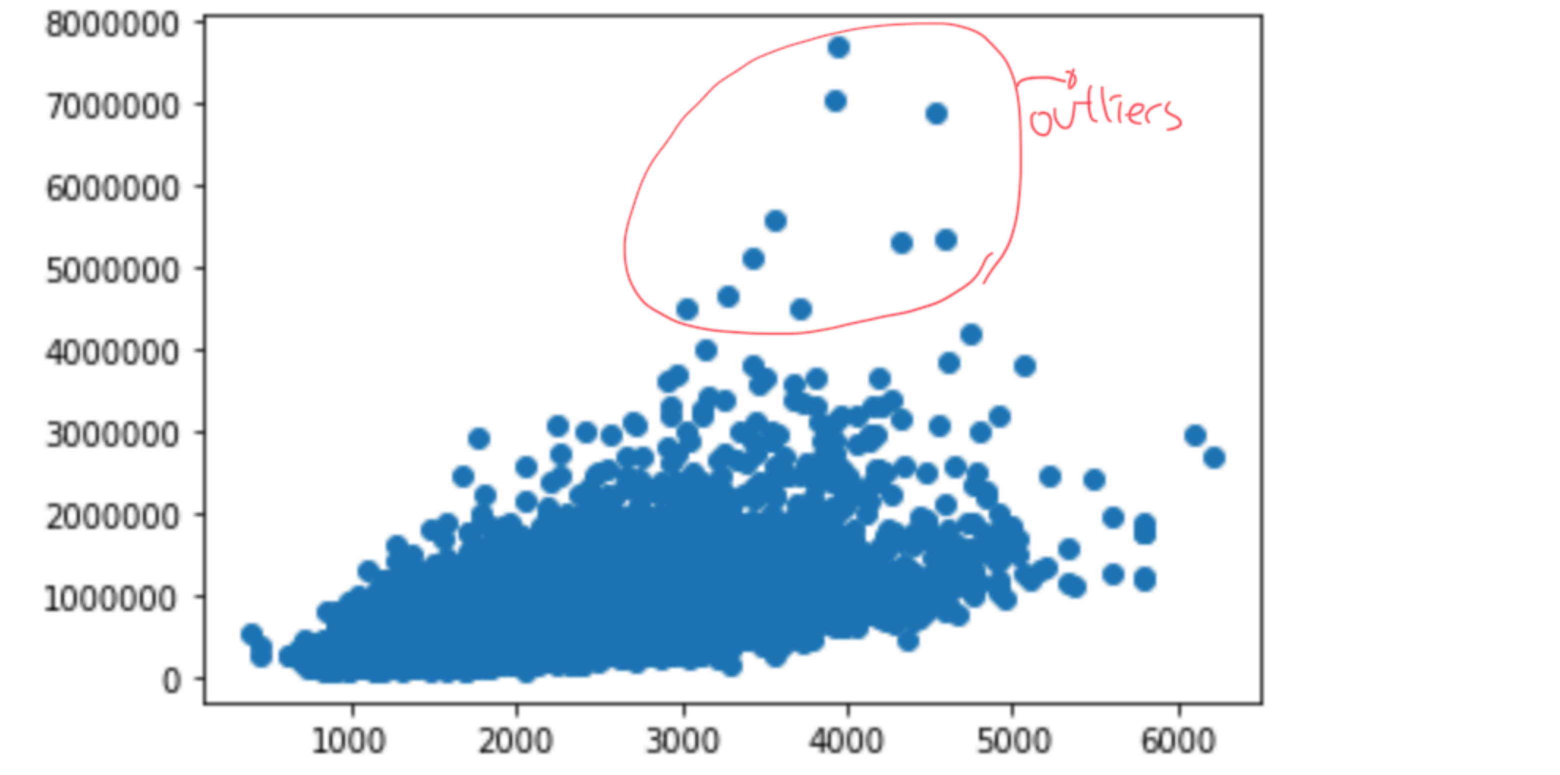
```
plt.scatter(house['sqft_living'],house['price'])
<matplotlib.collections.PathCollection at 0x7fda50dbcc50>
```




```
plt.scatter(house['sqft_above'],house['price'])
<matplotlib.collections.PathCollection at 0x7fda54537ed0>
```




```
plt.scatter(house['sqft_living15'],house['price'])
<matplotlib.collections.PathCollection at 0x7fda544a1250>
```



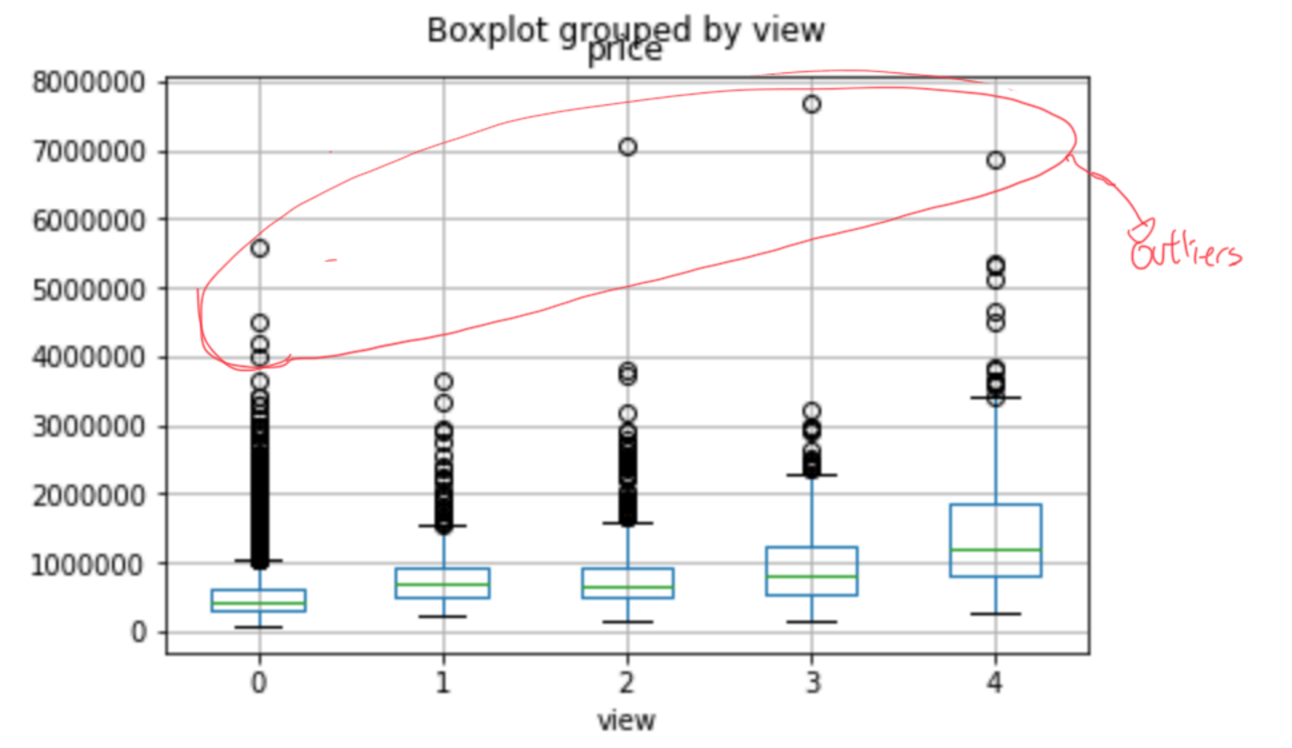
- **The reason why I delete outliers by using graph is ,when I use IQR outlier, so many datas are deleted.**

- **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

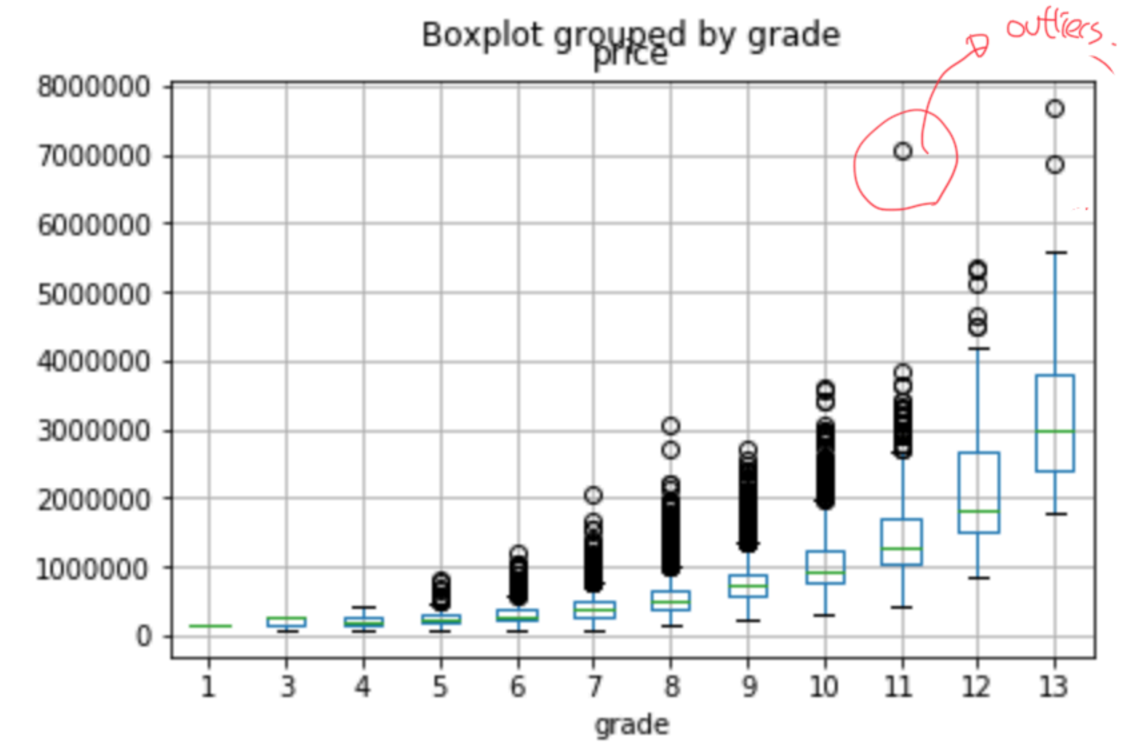
```
house.boxplot(column=['price'], by='view')
```

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

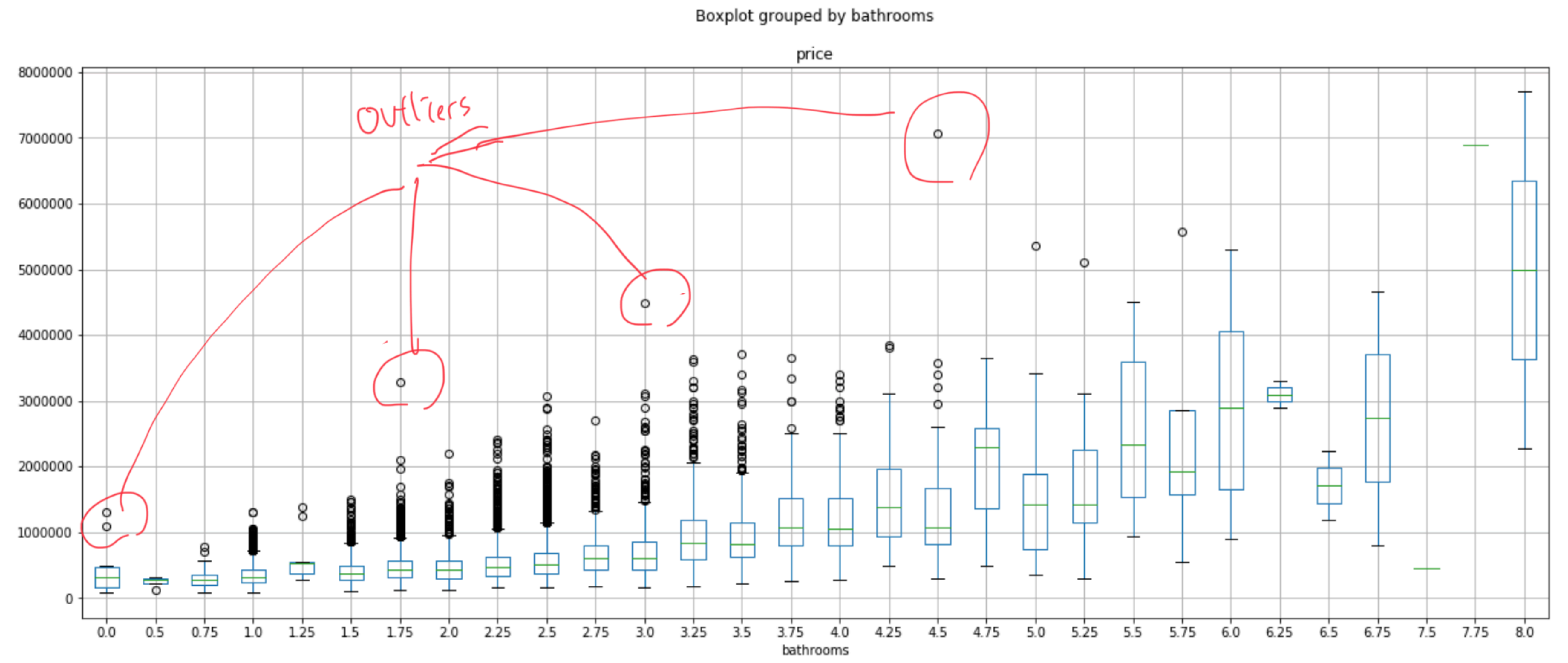



```
house.boxplot(column=['price'], by='grade')
```

<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**

IN MODELING, HAVE A COEFFICIENT PROBLEM

```python
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 sqft_living.**

MODELING BY OLS IN 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()
```

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.

- This is output of my previous model.
- That is problem what I use categorical variables 'view', 'grade' and 'bathrooms' like continuous variables.
- So I change those variables to dummy variables for linear regression model.
- 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

FIX MODEL BY CHANGING CATEGORICAL VARIABLES

CHECK MULTICOLLINEARITY

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

y, X = dmatrices('price ~sqft_above+sqft_living15+view+grade+bathrooms+condition',house, return_type = 'dataframe')
vif = pd.DataFrame()
vif["vif value"] = [variance_inflation_factor(X.values,i) for i in range(0,7)]
vif["explanatory variables"] = X.columns
vif
```

	vif value	explanatory variables
0	101.701657	Intercept
1	3.217097	sqft_above
2	2.549807	sqft_living15
3	1.110155	view
4	2.908025	grade
5	2.060369	bathrooms
6	1.036314	condition

- Because my previous model use ‘bathrooms’ variable as input variable of Linear Regression model and to compare new model with previous model, I make dummy variables not only assignment requirement (‘view’, ‘condition’ and ’grade’) but also ‘bathrooms’ variable.
- And all input variables have VIF value under 10. So we can decide the model don’t have multicollinearity

MAKE DUMMY VARIABLES

```
house['view'].value_counts()

0    19480
2      958
3      508
1      331
4      304
Name: view, dtype: int64
```

```
house['condition'].value_counts()

3    14006
4     5673
5     1700
2      172
1       30
Name: condition, dtype: int64
```

```
house['grade'].value_counts()

7     8980
8     6068
9     2615
6     2038
10    1132
11     392
5      242
12      75
4       29
13       6
3        3
1        1
Name: grade, dtype: int64
```

```
house['bathrooms'].value_counts()

2.50    5380
1.00    3852
1.75    3047
2.25    2047
2.00    1930
1.50    1446
2.75    1185
3.00     752
3.50     730
3.25     586
3.75     154
4.00     134
4.50      98
4.25      77
0.75      72
4.75      21
5.00      20
5.25      12
1.25       9
0.00       8
5.50       7
0.50       4
5.75       3
6.00       3
6.50       2
7.50       1
6.75       1
```

- **'view' and 'condition' have 5 unique values and interval is 1. So 4 dummy variables are made.**
- **'grade' looks like having 13 unique values. But there are no house which is grade = 2. So 11 dummy variables are made and interval is 1.**
- **'bathrooms' have 27 unique values and interval is 0.25. So 26 dummy variables are made**

MAKE DUMMY VARIABLES

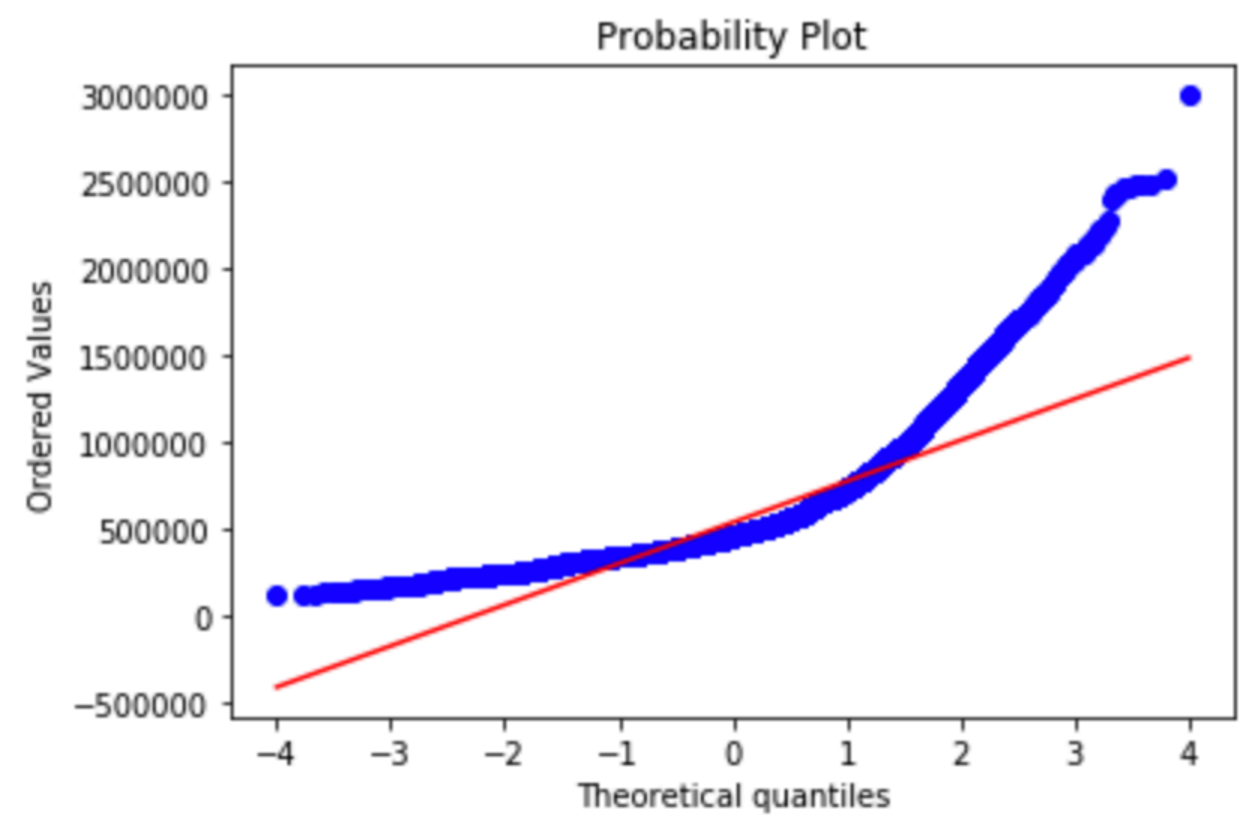
```
catvar = ['bathrooms','view','condition','grade']
for c in catvar:
    dummy = pd.get_dummies(house[c],prefix = c, drop_first = True)
    house = pd.concat((house,dummy),axis=1)
```

- By using this code, I make dummy variables.
- 'Bathrooms' 0.5 to 7.5 (interval 0.5) : 27 dummy columns are made
- 'view' 1 to 4 (interval 1) : 4 columns are made
- 'condition' 2 to 5 (interval 1 : 4 columns are made

RESIDUAL NORMALITY

Omnibus:	8594.377	**Durbin-Watson:**	1.983
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	82726.804
Skew:	1.648	**Prob(JB):**	0.00
Kurtosis:	12.008	**Cond. No.**	1.60e+06

```
import scipy.stats

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



- This model Jarque-Bera test p-value is 0. So this cannot select Null Hypothesis. And the Q-Q plot of residual shows datas are not located near normal distribution line. So we cannot decide this model's residual have normality.

MODELING RESULT

MODELING

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.615
Model:	OLS	**Adj. R-squared:**	0.614
Method:	Least Squares	**F-statistic:**	732.8
Date:	Thu, 23 Sep 2021	**Prob (F-statistic):**	0.00
Time:	22:01:19	**Log-Likelihood:**	-2.9511e+05
No. Observations:	21581	**AIC:**	5.903e+05
Df Residuals:	21533	**BIC:**	5.907e+05
Df Model:	47		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.089e+04	2.1e+05	0.147	0.883	-3.82e+05	4.43e+05
sqft_above	36.9703	3.292	11.231	0.000	30.518	43.422
sqft_living15	61.9659	3.434	18.043	0.000	55.234	68.698
bathrooms_0.5	3.376e+04	1.38e+05	0.245	0.806	-2.36e+05	3.03e+05
bathrooms_0.75	7.609e+04	9.14e+04	0.832	0.405	-1.03e+05	2.55e+05
bathrooms_1.0	1.056e+05	8.86e+04	1.193	0.233	-6.8e+04	2.79e+05
bathrooms_1.25	1.152e+05	1.13e+05	1.020	0.308	-1.06e+05	3 [illegible] click to u
bathrooms_1.5	9.745e+04	8.86e+04	1.099	0.272	-7.63e+04	2.71e+05
bathrooms_1.75	1.019e+05	8.86e+04	1.151	0.250	-7.17e+04	2.75e+05
bathrooms_2.0	1.102e+05	8.86e+04	1.243	0.214	-6.35e+04	2.84e+05
bathrooms_2.25	1.068e+05	8.86e+04	1.206	0.228	-6.68e+04	2.8e+05
bathrooms_2.5	6.197e+04	8.85e+04	0.700	0.484	-1.12e+05	2.36e+05
bathrooms_2.75	1.209e+05	8.87e+04	1.363	0.173	-5.3e+04	2.95e+05
bathrooms_3.0	1.739e+05	8.88e+04	1.958	0.050	-201.608	3.48e+05
bathrooms_3.25	2.374e+05	8.9e+04	2.668	0.008	6.3e+04	4.12e+05
bathrooms_3.5	1.911e+05	8.89e+04	2.149	0.032	1.68e+04	3.65e+05
bathrooms_3.75	3.426e+05	9.02e+04	3.797	0.000	1.66e+05	5.19e+05
bathrooms_4.0	3.553e+05	9.05e+04	3.926	0.000	1.78e+05	5.33e+05
bathrooms_4.25	4.422e+05	9.19e+04	4.811	0.000	2.62e+05	6.22e+05
bathrooms_4.5	3.354e+05	9.12e+04	3.677	0.000	1.57e+05	5.14e+05
bathrooms_4.75	8.254e+05	1e+05	8.253	0.000	6.29e+05	1.02e+06
bathrooms_5.0	4.322e+05	1.01e+05	4.298	0.000	2.35e+05	6.29e+05
bathrooms_5.25	5.355e+05	1.08e+05	4.974	0.000	3.24e+05	7.46e+05
bathrooms_5.5	4.981e+05	1.2e+05	4.145	0.000	2.63e+05	7.34e+05
bathrooms_5.75	1.239e+05	1.52e+05	0.817	0.414	-1.73e+05	4.21e+05
bathrooms_6.0	1.502e+05	1.51e+05	0.992	0.321	-1.47e+05	4.47e+05
bathrooms_6.5	4.669e+05	1.74e+05	2.684	0.007	1.26e+05	8.08e+05
bathrooms_6.75	-4.233e+05	2.29e+05	-1.850	0.064	-8.72e+05	2.51e+04
bathrooms_7.5	1.049e+05	2.28e+05	0.459	0.646	-3.43e+05	5.53e+05
view_1	1.697e+05	1.18e+04	14.417	0.000	1.47e+05	1.93e+05
view_2	1.042e+05	7115.038	14.650	0.000	9.03e+04	1.18e+05
view_3	1.84e+05	9714.090	18.939	0.000	1.65e+05	2.03e+05
view_4	4.924e+05	1.25e+04	39.479	0.000	4.68e+05	5.17e+05
condition_2	-7020.8696	4.24e+04	-0.166	0.868	-9.01e+04	7.61e+04
condition_3	-5698.2367	3.94e+04	-0.144	0.885	-8.3e+04	7.16e+04
condition_4	4.9e+04	3.95e+04	1.242	0.214	-2.84e+04	1.26e+05
condition_5	1.417e+05	3.97e+04	3.570	0.000	6.39e+04	2.2e+05
grade_3	1.227e+04	2.48e+05	0.049	0.961	-4.74e+05	4.99e+05
grade_4	-5.485e+04	2.35e+05	-0.233	0.815	-5.15e+05	4.06e+05
grade_5	-5.322e+04	2.32e+05	-0.230	0.818	-5.08e+05	4.01e+05
grade_6	2599.6369	2.32e+05	0.011	0.991	-4.52e+05	4.57e+05
grade_7	8.048e+04	2.32e+05	0.347	0.728	-3.74e+05	5.35e+05
grade_8	1.773e+05	2.32e+05	0.765	0.444	-2.77e+05	6.31e+05
grade_9	3.252e+05	2.32e+05	1.403	0.161	-1.29e+05	7.79e+05
grade_10	5.045e+05	2.32e+05	2.176	0.030	5e+04	9.59e+05
grade_11	7.319e+05	2.32e+05	3.153	0.002	2.77e+05	1.19e+06
grade_12	1.007e+06	2.33e+05	4.316	0.000	5.5e+05	1.46e+06
grade_13	1.682e+06	2.48e+05	6.783	0.000	1.2e+06	2.17e+06

Omnibus:	8594.377	**Durbin-Watson:**	1.983
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	82726.804
Skew:	1.648	**Prob(JB):**	0.00
Kurtosis:	12.008	**Cond. No.**	1.60e+06

- **F-test's p-value is 0 => Residual have homogeneity.**
- **Interestingly, the coefficient 'grade4' to 'grade13' is gradually increase ->This means that high 'grade' make high 'price'.**
- **Also the coefficient 'condition2' to 'condition5' is same with 'grade'**
- **Coefficient of 'view1' is higher than of 'view2', but of 'view2' to 'view4' is gradually increase.**
- **But coefficient of 'bathrooms' don't have trend of increasing or decreasing.**

COMPARE RESULT WITH MY PREVIOUS MODEL

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		

Previous Model

)LS Regression Results

Dep. Variable:	price	R-squared:	0.615
Model:	OLS	Adj. R-squared:	0.614
Method:	Least Squares	F-statistic:	732.8
Date:	Thu, 23 Sep 2021	Prob (F-statistic):	0.00
Time:	22:01:19	Log-Likelihood:	-2.9511e+05
No. Observations:	21581	AIC:	5.903e+05
Df Residuals:	21533	BIC:	5.907e+05
Df Model:	47		
Covariance Type:	nonrobust		

New Model

- **By using more categorical variables and changing them to dummy variables, R^2 value is increased. In other words, the explanatory power of the model is increased.**

HOW TO USE ‘ZIPCODE’, ‘LAT’ AND ‘LONG’

WITHOUT OTHER RESOURCES

```
house['zipcode'].value_counts()
```

```
98103    602
98038    590
98115    583
98052    574
98117    553
        ...
98102    103
98010    100
98024     81
98148     57
98039     44
```

```
print("Depending on zipcode value mean of standard deviation of 'lat'\n\n",lat_std.mean(),"\n")
print("standard deviation of 'lat' column \n\n" ,house['lat'].std())
```

```
Depending on zipcode value mean of standard deviation of 'lat'

 0.01589899563147026

standard deviation of 'lat' column

 0.1386231069375018
```

```
print("Depending on zipcode value mean of standard deviation of 'long'\n\n",long_std.mean(),"\n")
print("standard deviation of 'long' column \n\n" ,house['long'].std())
```

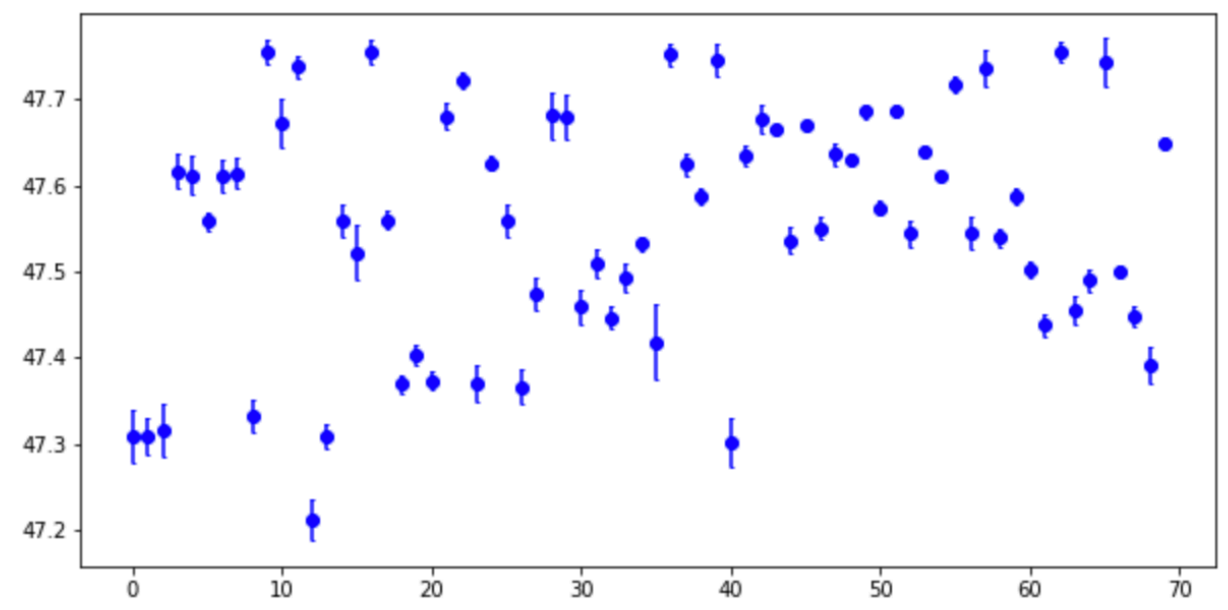
```
Depending on zipcode value mean of standard deviation of 'long'

 0.019429508395824427

standard deviation of 'long' column

 0.14085387080836465
```

- **In same 'zipcode' value , there are hundreds or lower houses.**
- **The value of standard deviation is much bigger when classify 'lat' and 'long' depending on zipcode than when handle all of data's 'lat' and 'long'. This means data of same zipcode have similar values of 'long' and 'lat'.**

WITHOUT OTHER RESOURCES

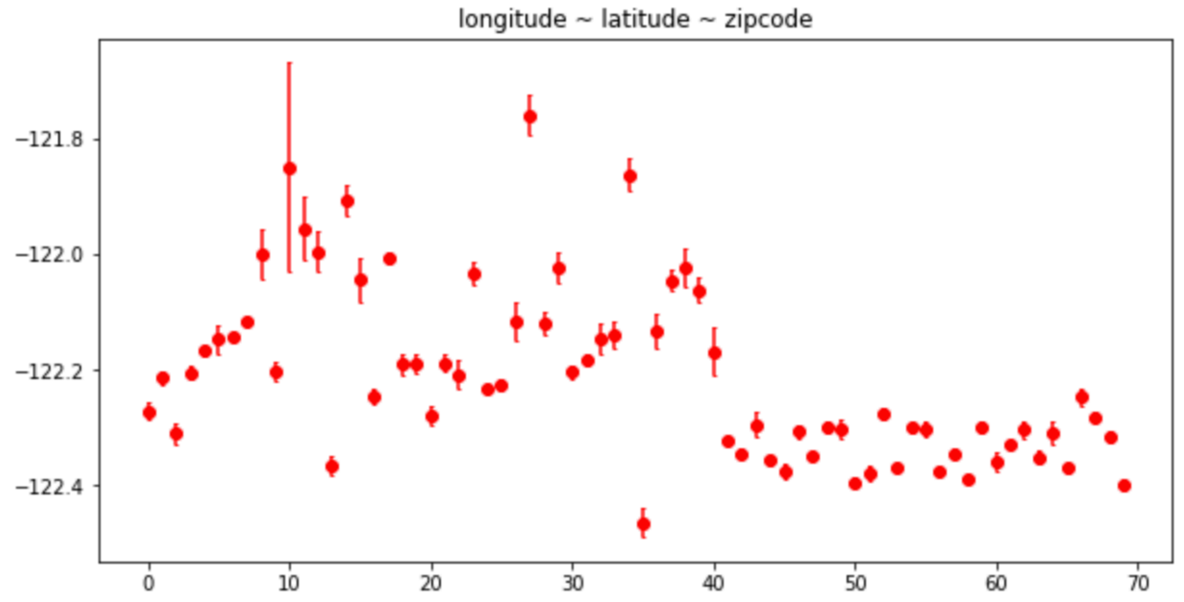
```python
plt.figure(figsize=(10,5))

lat_mean = house.groupby('zipcode')['lat'].mean()
lat_std = house.groupby('zipcode')['lat'].std()

plt.errorbar(range(len(lat_mean)),lat_mean,yerr=lat_std, fmt='o', c='b',ecolor='b',capthick=1,capsize=1)
```

```
<ErrorbarContainer object of 3 artists>
```




```python
long_mean = house.groupby('zipcode')['long'].mean()
long_std = house.groupby('zipcode')['long'].std()

plt.figure(figsize=(10,5))
plt.title("longitude ~ latitude ~ zipcode")

plt.errorbar(range(len(long_mean)),long_mean,yerr=long_std, fmt='o', c='r',ecolor='r',capthick=1,capsize=1)
```

```
<ErrorbarContainer object of 3 artists>
```



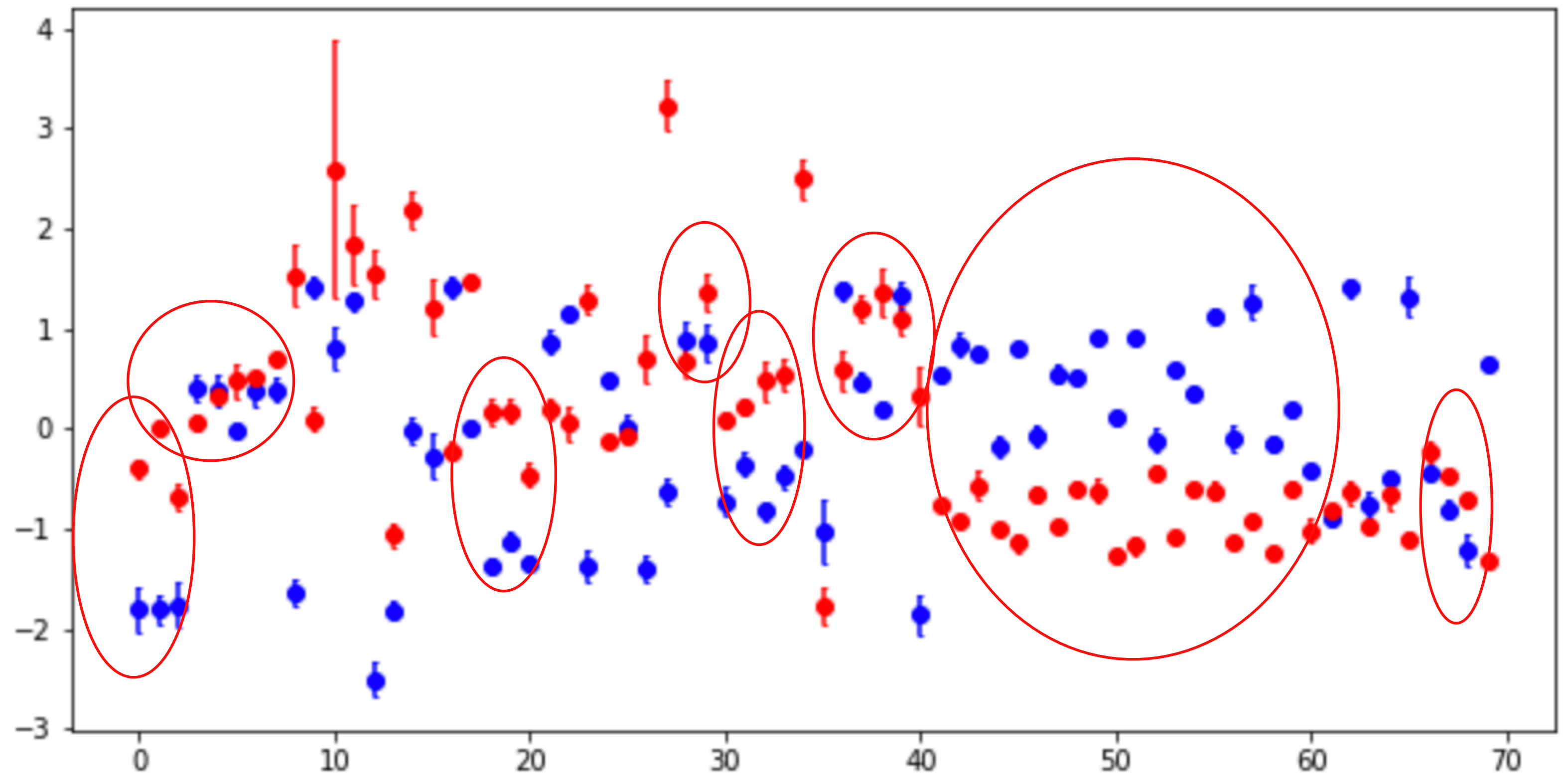
- **By using ‘long’ and ‘lat’, we can decide specific location coordinate = (‘long’, ‘lat’). And ‘zipcode’ means indicating the address of house in number.**

- **Error-bar consist dot and line. Dot is mean of y_value and line is sd of y_value. Above graphs show data describe of ‘long’ and ‘lat’ depending on value of ‘zipcode’.**

WITHOUT OTHER RESOURCES

```
# for show two graph in one graph standardization
# So i can see those graph in one graph
house['lat'] = (house['lat'] - house['lat'].mean())/house['lat'].std()
house['long'] = (house['long'] - house['long'].mean())/house['long'].std()

long_mean = house.groupby('zipcode')['long'].mean()
long_std = house.groupby('zipcode')['long'].std()
lat_mean = house.groupby('zipcode')['lat'].mean()
lat_std = house.groupby('zipcode')['lat'].std()

plt.figure(figsize=(10,5))
plt.errorbar(range(len(lat_mean)),lat_mean,yerr=lat_std, fmt='o', c='b',ecolor='b',capthick=1,capsize=1)
plt.errorbar(range(len(long_mean)),long_mean,yerr=long_std, fmt='o', c='r',ecolor='r',capthick=1,capsize=1)
```

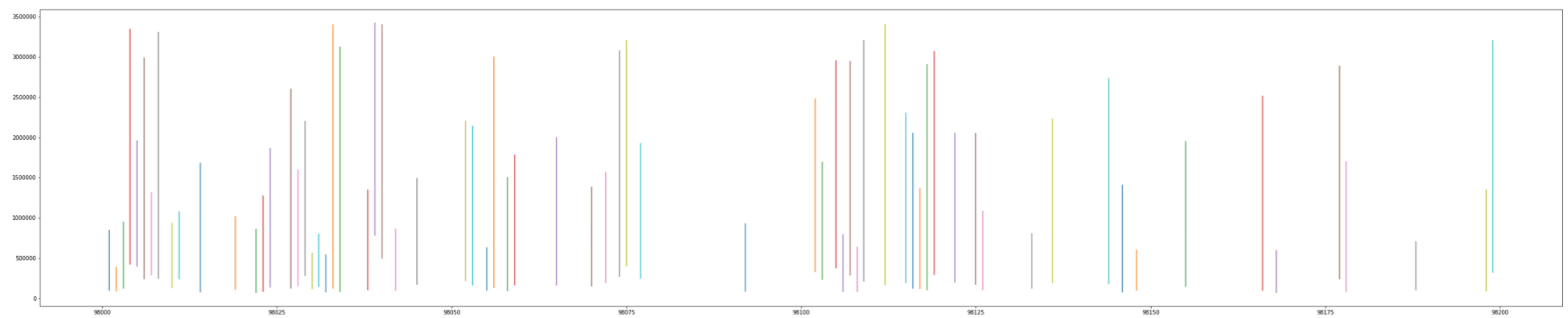
```
<ErrorbarContainer object of 3 artists>
```



- I want to show two graph in one graph with same condition. So do standardization of datas.
- When 'zipcode' are similar, many datas have similar lat or long. And when lat or long are similar, lat or long are similar. Meaning of this can see circle.
- By group (like circle)datas what have similar zipcode ,similar lat and similar long, make new column 'group_zipcode'

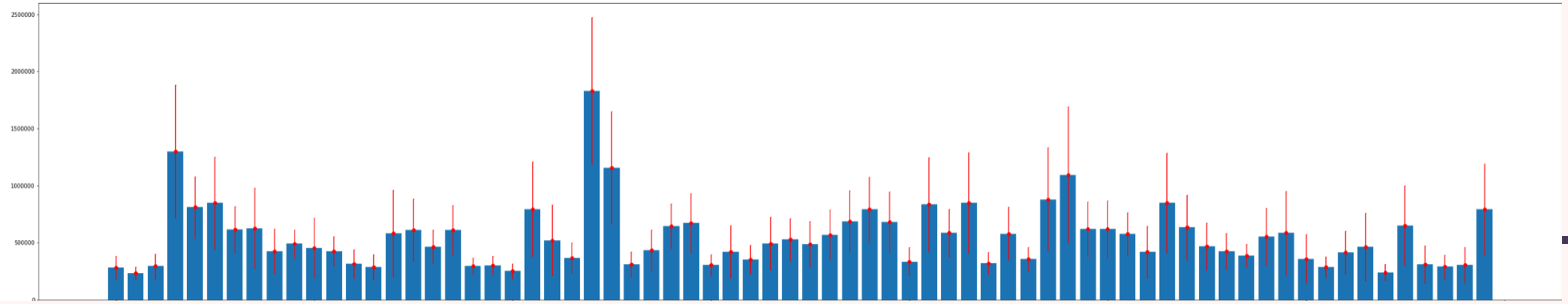
WITH OTHER RESOURCES

```python
price_min = house.groupby('zipcode')['price'].min()
price_max = house.groupby('zipcode')['price'].max()

plt.figure(figsize=(50,10))
for x in price_min.index:
    plt.plot([x,x],[price_min[x],price_max[x]])

#by this graph i can see the min_price value don't high difference between zipcode
```



```python
price_mean = house.groupby('zipcode')['price'].mean()
price_std = house.groupby('zipcode')['price'].std()

plt.figure(figsize=(50,10))
plt.bar(range(len(price_mean)),price_mean)
plt.errorbar(range(len(price_mean)),price_mean,yerr=price_std, fmt='o', c='r',ecolor='r',capthick=1,capsize=1)
```

```
<ErrorbarContainer object of 3 artists>
```

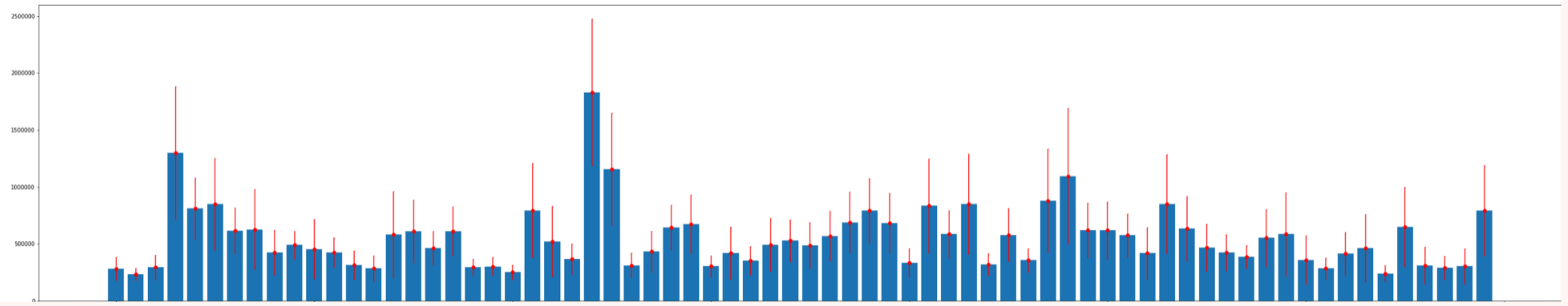


- 1st graph is collected line graphs connect minimum of price and maximum of price depending on value of 'zipcode'.
- 2nd graph is same type graph in previous silde. This shows sd and average.
- By seeing two graph together, I can think the min and max is not meaningful to group 'zipcode' by using price information. Because minimum of price is similar and maximum of price is not meaningful to average

WITH OTHER RESOURCES

```
price_mean = house.groupby('zipcode')['price'].mean()
price_std = house.groupby('zipcode')['price'].std()

plt.figure(figsize=(50,10))
plt.bar(range(len(price_mean)),price_mean)
plt.errorbar(range(len(price_mean)),price_mean,yerr=price_std, fmt='o', c='r',ecolor='r',capthick=1,capsize=1)
```

```
<ErrorbarContainer object of 3 artists>
```



```
price_mean.describe()
```

```
count    7.000000e+01
mean     5.520771e+05
std      2.734145e+05
min      2.342840e+05
25%      3.528348e+05
50%      4.919520e+05
75%      6.425136e+05
max      1.831578e+06
Name: price, dtype: float64
```

- 'lat' and 'long' is mixed with zipcode in the way I did before to 'group_zipcode' column.
- So make new column 'rank'(categorical variable). This include 'group_zipcode' in without other resource part's column,
- Average of price in each 'group_zipcode' decide value of 'rank'. So I think this 'rank' column is more meaningful when predict 'price' value

THANK YOU