TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN MÔN HỌC
NHẬP MÔN MÁY HỌC

ĐỀ TÀI: HOUSE PRICE PREDICTION

Giảng viên hướng dẫn: Vũ Ngọc Thanh Sang

Nhóm sinh viên thực hiện: Lương Ngọc Minh Khuê - 3120410256

Nguyễn Lê Đăng Khoa - 3120410247

Huỳnh Lê Thanh Nga - 3120410339

Chung Phát Tài - 3120410455

Ngày 15, tháng 12, năm 2022

LỜI MỞ ĐẦU

Thời gian gần đây, chắc hẳn mọi người đã được nghe nhiều về thuật ngữ "Machine Learning" (học máy hoặc máy học) - vốn được sử dụng rất nhiều trong việc phát hiện lừa đảo trực tuyến, cung cấp các đề xuất mua hàng trực tuyến trên Netflix hay Amazon, trong công nghệ nhận diện khuôn mặt, xe hơi tự lái, hay ứng dụng trong các Trí tuệ nhân tạo (AI) tiên tiến nhất thế giới hiện nay. Nói rõ hơn máy học là một thuật ngữ đề cập đến các chương trình máy tính có khả năng học hỏi về cách hoàn thành các nhiệm vụ, đồng thời cải thiện hiệu suất theo thời gian.

Học kỳ này, nhóm 4 chúng em đã được tiếp cận gần hơn với machine learning qua học phần ‘Nhập môn máy học’. Để hiểu rõ hơn và nắm chắc các kiến thức của học phần này, chúng em xin lựa chọn đề tài House Price Prediction (dự đoán giá nhà) để thực hiện đồ án giữa kỳ.

MỤC LỤC

PHẦN 1
GIỚI THIỆU CHUNG

I. Định nghĩa bài toán

1.1. Vấn đề cần giải quyết

Dự đoán giá bán của một căn nhà từ các đặc điểm có được từ căn nhà đó bằng các mô hình máy học.

1.2. Định nghĩa bài toán cho người không cùng chuyên môn

Từ một tập dữ liệu có sẵn (đã được thu thập trong một khoảng thời gian), ta sẽ thống kê những điểm tương đồng dẫn đến giá tăng cũng như lý do khiến giá của nó giảm. Từ đó tổng hợp lại để khi xét một trường hợp mới, ta có thể dự đoán từ gần chính xác đến chính xác giá của căn nhà.

1.3. Định nghĩa bài toán cho người cùng chuyên môn

Từ một tập dữ liệu có sẵn (đã được thu thập trong một khoảng thời gian), ta trải qua các bước xử lý dữ liệu, huấn luyện mô hình, cải thiện mô hình, lựa chọn mô hình tối ưu nhất để đạt được độ chính xác cao.

1.4. Giải thuyết của bài toán:

Số liệu nhận được từ quá trình khảo sát có tính xác thực cao, những người ra giá bán nhà trong quá trình thực hiện khảo sát trả lời thật.

1.5. Các bài toán tương tự trong thực tế:

Bài toán dự đoán giá đất, bài toán dự đoán giá nhà cho thuê, bài toán dự đoán.

II. Sự cần thiết của project

2.1. Động lực để giải quyết bài toán

Ngày nay, đầu tư là một hoạt động kinh doanh mà hầu kết mọi người đều quan tâm trong thời đại toàn cầu hóa. Bên cạnh một số đối tượng được các nhà đầu từ hướng đến như vàng, chứng khoán thì bất động sản cũng có một vị trí khá quan trọng đối với một số nhà đầu tư. Tuy nhiên, trong một số hoàn cảnh giá nhà chịu tác động của các yếu tố bên ngoài như marketing, lừa đảo gây nên hiện tượng thua lỗ ở nhiều người. Hệ thống dự đoán giá nhà ra đời đáp ứng nhu cầu trên.

2.2. Giải pháp mang lại lợi ích gì

Hệ thống dự đoán giá nhà giúp người có nhu cầu mua bất động sản có thể đưa ra những lựa chọn hợp lý để có thể lựa chọn cho mình căn nhà phù hợp. Đồng thời cũng

giúp cho các nhà đầu tư dự đoán được mặt bằng chung giá nhà ở những vùng khác nhau để có thể đầu tư hợp lý.

2.3. Giải pháp được sử dụng thế nào

Giải pháp được sử dụng trong hoàn cảnh một người đang có nhu cầu mua một hoặc nhiều căn nhà mới. Sau khi tổng hợp đầy đủ các đặc điểm của căn nhà chẳng hạn như căn nhà có bao nhiêu phòng ngủ, bao nhiêu phòng tắm, diện tích sinh hoạt bao nhiêu, … Do không biết chính xác giá cả của căn nhà đó so với thị trường, họ phải suy ra giá cả của căn nhà đó dựa trên các đặc điểm của nó. Do đó, mô hình giúp cho việc ước tính giá cả của những căn nhà trong trường hợp này là cần thiết đối với họ.

III. Giải pháp thủ công cho bài toán

3.1. Giải pháp hiện tại của bài toán

Giải pháp hiện tại của bài toán chỉ dừng ở mức hỗ trợ ước tính giá nhà thông qua đó giúp cho những người có nhu cầu mua nhà có thể dự đoán hướng đi đúng đắn trongg việc đầu tư bất động sản cũng như tìm kiếm cho mình một căn nhà phù hợp với giá cả hợp lý so với thực tế

3.2. Các chuyên gia trong lĩnh vực giải quyết như thế nào

Tìm kiếm thông tin về bất động sản được giao dịch trong thời gian gần nhất để so sánh với bất động sản mục tiêu về các mặt, các yếu tố ảnh hưởng đến giá trị như: Tình trạng vật chất, vị trí, tính trạng pháp lý, điều kiện môi trường. Lựa chọn từ 3 đến 6 bất động sản để so sánh để tìm ra bất động sản phù hợp nhất. Xác định các yếu tố khác nhau giữa bất động sản chứng cớ với bất động sản mục tiêu để dựa trên những yếu tố đó để điều chỉnh giá của bất động sản mục tiêu. Nếu bất động sản mục tiêu có các yếu tố được đánh giá tốt hơn thì tiến hành tăng giá trị giao dịch của bất động mục tiêu lên và ngược lại.

Ngoài ra còn có các phương pháp khác như phương pháp thu nhập, phương pháp giá thành, phương pháp lợi nhuận, phương pháp thặng dư, ….

3.3. Giải pháp công nghệ thông tin cho bài toán

Giải pháp công nghệ thông tin cho bài toán là nhờ đến sự trợ giúp của máy học cụ thể là áp dụng thuật toán hồi quy để dự đoán giá đầu ra dựa trên các giá trị đầu vào (các đặc điểm của căn nhà) và mối quan hệ giữa giá và các đặc điểm đó (đặc điểm nào gây tác động đến giá). Từ đó giúp ước tính ra giá tương đối của những căn nhà cần dự đoán.

IV. Chuẩn bị dữ liệu

4.1. Mô tả dữ liệu

Dữ liệu gồm có 18 đặc trưng bao gồm:

Date	Ngày mà căn nhà được bán
Price	Giá bán của căn nhà
Bedrooms	Số lượng phòng ngủ của căn nhà
Bathrooms	Số lượng phòng tắm của căn nhà
Sqft_living	Diện tích dùng để ở của căn nhà (đv: SquareFeet)
Sqft_lot	Tổng diện tích của căn nhà (đv: SquareFeet)
Floors	Số tầng của căn nhà
Waterfront	Bờ sông trước nhà
View	Góc nhìn từ nhà, khung cảnh đẹp…
Condition	Tình trạng của căn nhà
Sqft_above	Diện tích trên không của căn nhà (đv: SquareFeet)
Sqft_basement	Diện tích của tầng hầm (đv: SquareFeet)
Yr_built	Năm xây nhà
Yr_renovated	Năm mà căn nhà được cải tạo
Street	Con đường mà căn nhà nằm trên
City	Thành phố mà căn nhà ở
Statezip	Mã bưu chính của căn nhà
Country	Quốc gia của căn nhà

Ý nghĩa của từng đặc trưng

```
[ ] df.shape

    (4600, 18)
```

Kích thước dữ liệu là 4600 dòng, 18 cột

```
[ ]  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           4600 non-null   object 
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64  
 5   sqft_lot       4600 non-null   int64  
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64  
 8   view           4600 non-null   int64  
 9   condition      4600 non-null   int64  
 10  sqft_above     4600 non-null   int64  
 11  sqft_basement  4600 non-null   int64  
 12  yr_built       4600 non-null   int64  
 13  yr_renovated   4600 non-null   int64  
 14  street         4600 non-null   object 
 15  city           4600 non-null   object 
 16  statezip       4600 non-null   object 
 17  country        4600 non-null   object 
dtypes: float64(4), int64(9), object(5)
memory usage: 647.0+ KB
```

Số lượng dữ liệu không null và kiểu dữ liệu của các đặc trưng

4.2. Trong hoàn cảnh nào thì có thể thu thập được dũ liệu

Trong hoàn cảnh giá bán của các căn nhà được khai báo minh bạch, đồng thời giá của những căn nhà trong khu vực đó dao động ổn định trong một thời gian.

4.3. Trong hoàn cảnh nào thì không thể thu thập được dữ liệu

Trong hoàn cảnh lừa đảo đẩy giá nhà lên cao so với thực tế, không thống nhất về định giá của một cân nhà, một số bất động sản giống nhau nhưng định giá khác nhau nên thiếu minh bạch. Ngoài ra khoảng cách giữa thời gian diễn ra khảo sát so với thời gian

tiến hành dự đoán giá nhà cũng khá quan trọng. Vì giá bất động sản biến đổi không ngừng sau hàng năm đặc biệt là trong các khoảng thời gian diễn ra các sự kiện như dịch bệnh, suy thoái kinh tế, world cup, ... nên nếu khoảng cách giữa hai mốc thời gian quá lớn thì kết quả thu được sẽ không chính xác vì lúc này giá nhà thực tế đã thay đổi so với giá nhà ở thời gian tiến hành khảo sát.

V. Xử lý dữ liệu

5.1. Thống kê dữ liệu

```
df.describe()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated
count	4.600000e+03	4600.000000	4600.000000	4600.000000	4.600000e+03	4600.000000	4600.000000	4600.000000	4600.000000	4600.000000	4600.000000	4600.000000	4600.000000
mean	5.519630e+05	3.400870	2.160815	2139.346957	1.485252e+04	1.512065	0.007174	0.240652	3.451739	1827.265435	312.081522	1970.786304	808.608261
std	5.638347e+05	0.908848	0.783781	963.206916	3.588444e+04	0.538288	0.084404	0.778405	0.677230	862.168977	464.137228	29.731848	979.414536
min	0.000000e+00	0.000000	0.000000	370.000000	6.380000e+02	1.000000	0.000000	0.000000	1.000000	370.000000	0.000000	1900.000000	0.000000
25%	3.228750e+05	3.000000	1.750000	1460.000000	5.000750e+03	1.000000	0.000000	0.000000	3.000000	1190.000000	0.000000	1951.000000	0.000000
50%	4.609435e+05	3.000000	2.250000	1980.000000	7.683000e+03	1.500000	0.000000	0.000000	3.000000	1590.000000	0.000000	1976.000000	0.000000
75%	6.549625e+05	4.000000	2.500000	2620.000000	1.100125e+04	2.000000	0.000000	0.000000	4.000000	2300.000000	610.000000	1997.000000	1999.000000
max	2.659000e+07	9.000000	8.000000	13540.000000	1.074218e+06	3.500000	1.000000	4.000000	5.000000	9410.000000	4820.000000	2014.000000	2014.000000

Số lượng của các đặc trưng, giá trị trung bình, độ lệch chuẩn, min, max,
25% - The *25% percentile*, *50% - The 50% percentile*, *75% - The 75% percentile.*

5.2. Có bao nhiêu dữ liệu số (numeric) có bao nhiêu dữ liệu chữ (category).

```
df.dtypes
```

```
date             object
price           float64
bedrooms        float64
bathrooms       float64
sqft_living       int64
sqft_lot          int64
floors          float64
waterfront        int64
view              int64
condition         int64
sqft_above        int64
sqft_basement     int64
yr_built          int64
yr_renovated      int64
street           object
city             object
statezip         object
country          object
dtype: object
```

Ta có 13 dữ liệu dạng numeric và 5 dữ liệu dạng category

5.3. Xác định và xử lý dữ liệu bị thiếu, dữ liệu bị nhiễu

```
[ ]  df.isnull().sum()

date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

Ta gọi hàm kiểm tra các dữ liệu có bị thiếu hay không và thiếu bao nhiêu. Nhìn vào output, ta thấy được rằng tập dữ liệu không bị thiếu ở bất kỳ đặc trưng nào cả.

```
[10]  df[df==0].count()

date                0
price              49
bedrooms            2
bathrooms           2
sqft_living         0
sqft_lot            0
floors              0
waterfront       4567
view             4140
condition           0
sqft_above          0
sqft_basement    2745
yr_built            0
yr_renovated     2735
street              0
city                0
statezip            0
country             0
dtype: int64
```

Ngoài việc không điền dữ liệu dẫn đến thiếu thì một số tập dữ liệu người ta thay thế số 0 vào, vì vậy chúng ta cũng sẽ tiến hành kiểm tra luôn các giá trị bằng 0 ở từng đặc trưng.

Và điều cần quan tâm là có tận 49 căn nhà có giá tiền bằng 0, 2 căn không có phòng ngủ và 2 căn không có phòng tắm. Tầm nhìn, bờ sông, diện tích tầng hầm và năm tái tạo thì có thể chấp nhận bằng 0 được.

```
df[df["price"]==0].agg([min, max, 'mean', 'median'])
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view
min	2014-05-05 00:00:00	0.0	1.000000	1.00000	720.000000	3500.000000	1.0	0.000000	0.000000
max	2014-07-08 00:00:00	0.0	6.000000	6.25000	8020.000000	188200.000000	3.0	1.000000	4.000000
mean	NaN	0.0	3.979592	2.69898	2787.142857	16453.306122	1.5	0.061224	0.795918
median	NaN	0.0	4.000000	2.50000	2600.000000	9000.000000	1.5	0.000000	0.000000

condition	sqft_above	sqft_basement	yr_built	yr_renovated	street	city	statezip	country
2.000000	720.000000	0.000000	1920.000000	0.000000	101-127 247th Ave SE	Auburn	WA 98001	USA
5.000000	8020.000000	1950.000000	2013.000000	2009.000000	9243 NE 20th St	Woodinville	WA 98199	USA
3.673469	2295.714286	491.428571	1969.918367	812.714286	NaN	NaN	NaN	NaN
3.000000	1990.000000	0.000000	1962.000000	0.000000	NaN	NaN	NaN	NaN

Kiểm tra thông tin theo nhóm các đặc trưng có giá bằng 0

Vậy những căn nhà có giá là 0 có trung vị số phòng ngủ là 4, trung vị số phòng tắm là 2,5, trung vị diện tích sống là 2600 sqft… Chúng ta sẽ tạo một tập gồm các yếu tố phù hợp để lấy giá trị trung bình của giá tiền, sau đó thay vào các căn nhà có giá bằng 0.

```
df1 = df[(df.bedrooms == 4) & (df.bathrooms > 1) & (df.bathrooms < 4) & (df.sqft_living > 2500)
      & (df.sqft_living < 3000) & (df.floors < 3) & (df.yr_built < 1970)]
df1.price.mean()
```

```
735475.0370705189
```

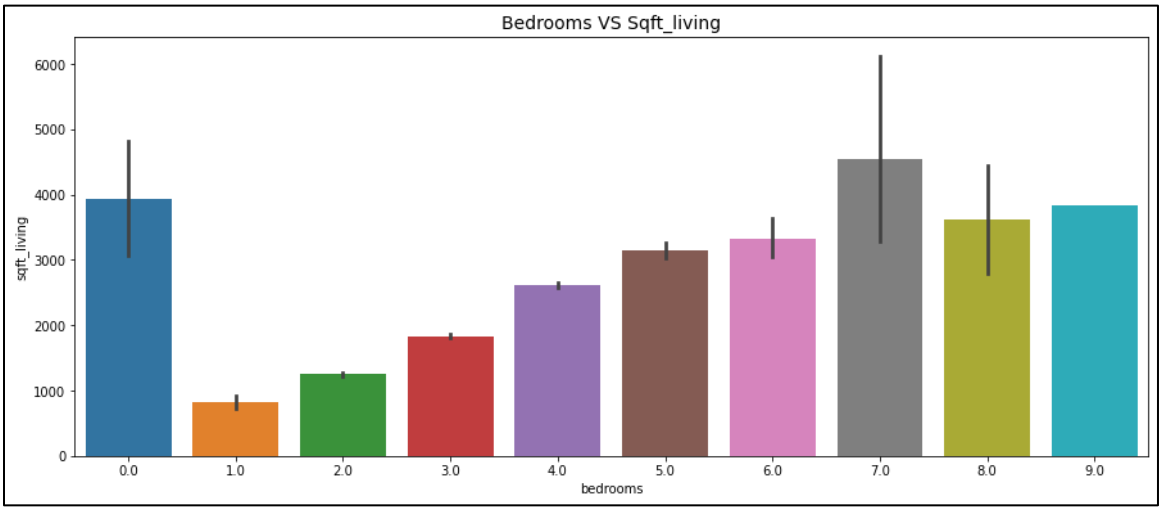
```
df['price'].replace(to_replace = 0, value = 735000, inplace = True)
len(df[(df['price'] == 0)])
```

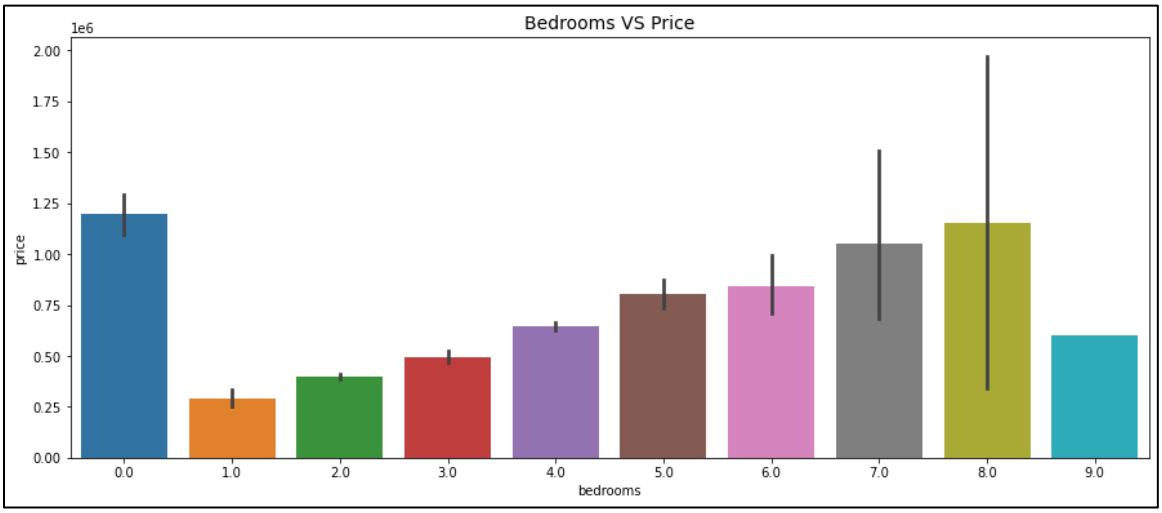
```
0
```

Giải quyết xong các căn nhà có giá bằng 0, ta tiến tới giải quyết các căn nhà sở hữu 0 phòng ngủ hoặc 0 phòng tắm.

```
plt.figure(figsize=(15,6))
ax = sns.barplot(x=df['bedrooms'], y=df['sqft_living'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=0)
ax.set_title('Bedrooms VS Sqft_living', fontsize=14)

plt.figure(figsize=(15,6))
ax = sns.barplot(x=df['bedrooms'], y=df['price'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=0)
ax.set_title('Bedrooms VS Price', fontsize=14)
```





Kiểm tra lược đồ cột giữa price với bedroom và sqft_living với bedroom ta thấy rằng có vài sự bất ổn là vài căn nhà có 6 phòng ngủ trở lên còn ít tiền hơn căn không có phòng ngủ, vài căn 8 phòng ngủ còn có diện tích nhỏ hơn căn 5,6 phòng ngủ. Ta sẽ kiểm tra số lượng phòng ngủ.

```
bybedroom = df.groupby(['bedrooms']).price.agg([len, min, max])
bybedroom
```

bedrooms	len	min	max
0.0	2	1095000.0	1295648.0
1.0	38	80000.0	735000.0
2.0	566	7800.0	1695000.0
3.0	2032	83300.0	26590000.0
4.0	1531	84350.0	4489000.0
5.0	353	185000.0	7062500.0
6.0	61	175000.0	3100000.0
7.0	14	280000.0	3200000.0
8.0	2	340000.0	1970000.0
9.0	1	599999.0	599999.0

Số lượng căn nhà có 0, 6, 7, 8, 9 là không nhiều và cũng có nhiều bất ổn nên ta quyết định bỏ bớt đi.

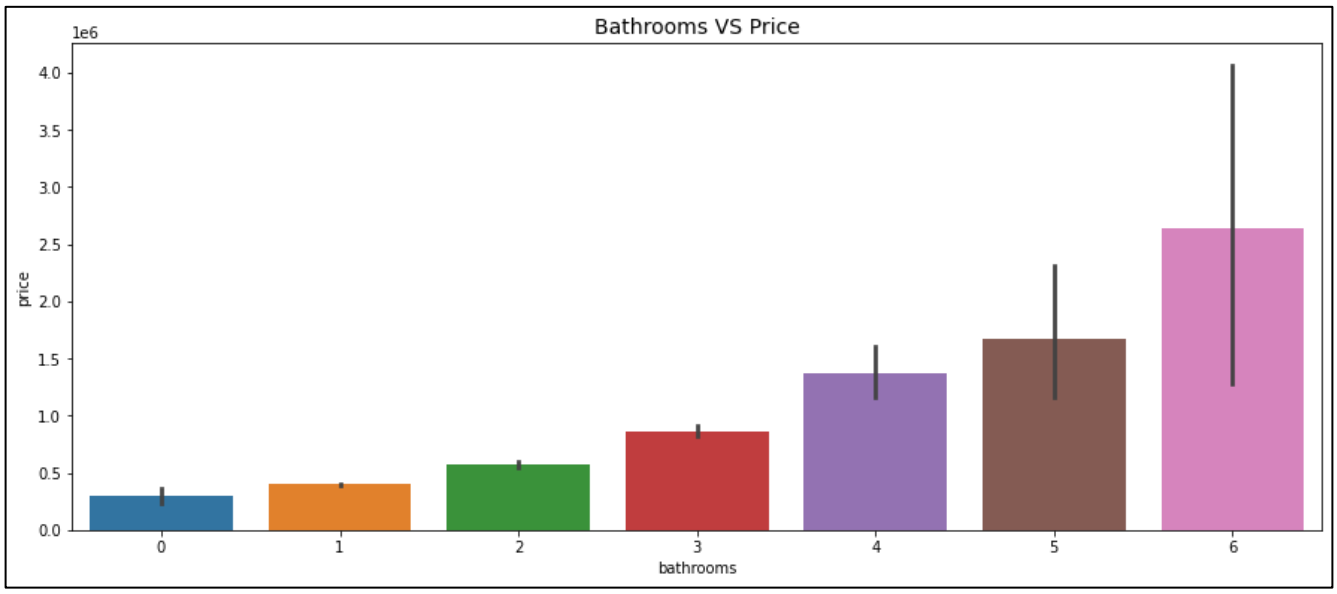
```
df = df.loc[(df.bedrooms>0) & (df.bedrooms<6)]
df.bedrooms.value_counts()
```

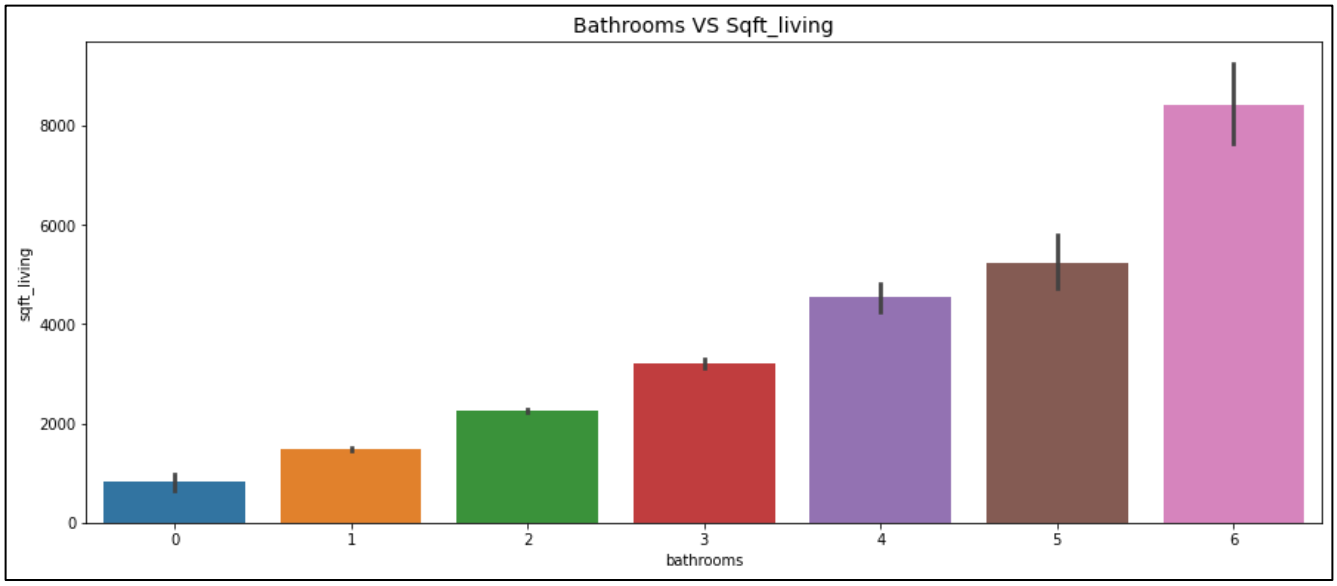
```
3.0    2032
4.0    1531
2.0     566
5.0     353
1.0      38
Name: bedrooms, dtype: int64
```

Đây là tổng các căn nhà có cùng số lượng phòng ngủ sau khi đã lược bỏ những điều theo ta mong muốn. Giờ ta sẽ làm tương tự với phòng tắm.

```
plt.figure(figsize=(15,6))
ax = sns.barplot(x=df['bathrooms'], y=df['price'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=0)
ax.set_title('Bathrooms VS Price', fontsize=14)

plt.figure(figsize=(15,6))
ax = sns.barplot(x=df['bathrooms'], y=df['sqft_living'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=0)
ax.set_title('Bathrooms VS Sqft_living', fontsize=14)
```

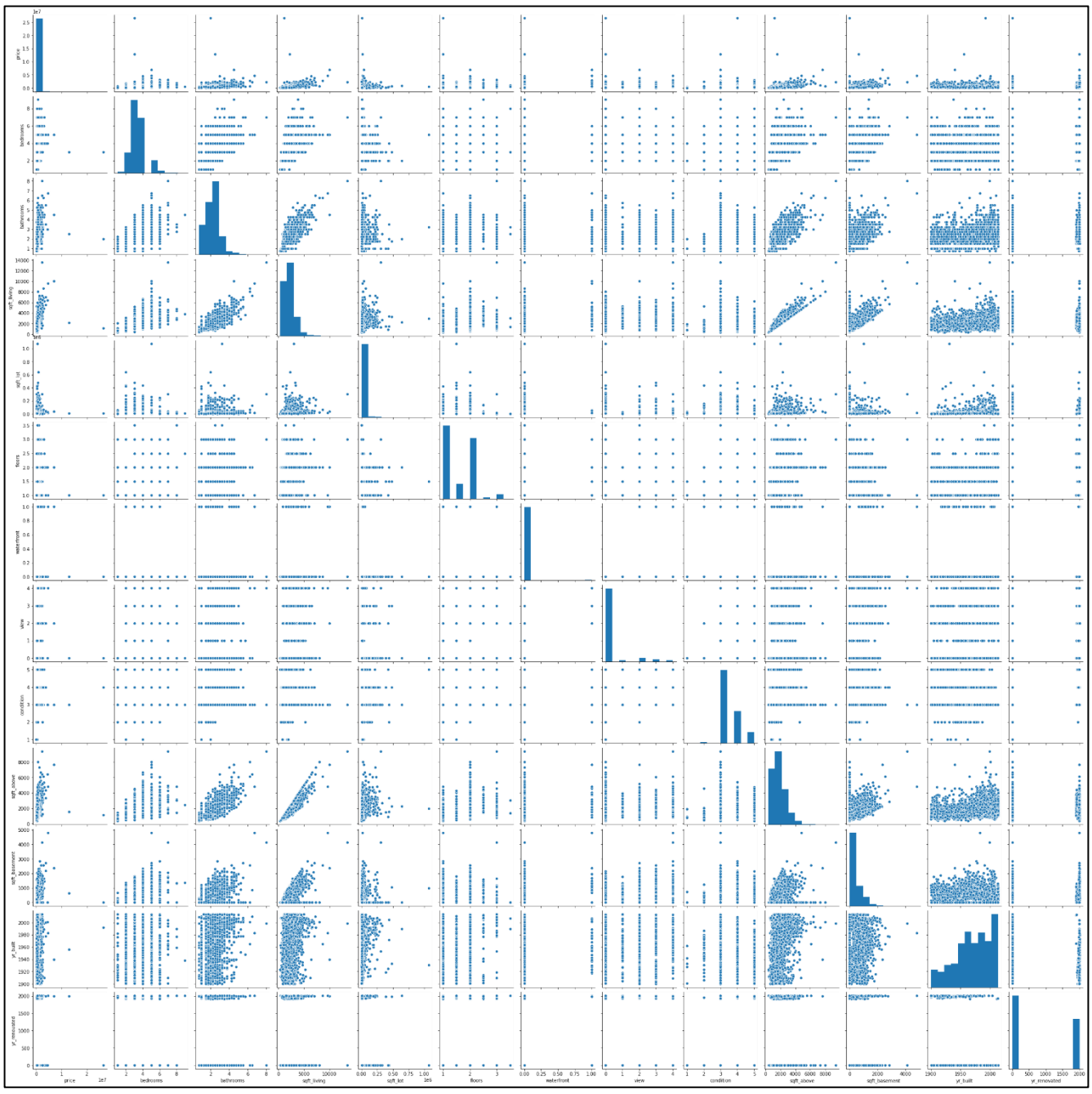




Kiểm tra thông số và thấy 2 căn có phòng tắm bằng 0 ban nãy đã tình cờ bị lược bỏ trong quá trình bỏ bớt phòng ngủ, vậy ta sẽ qua các bước tiếp theo.

5.4. Xác định và xử lý dữ liệu ngoại lai

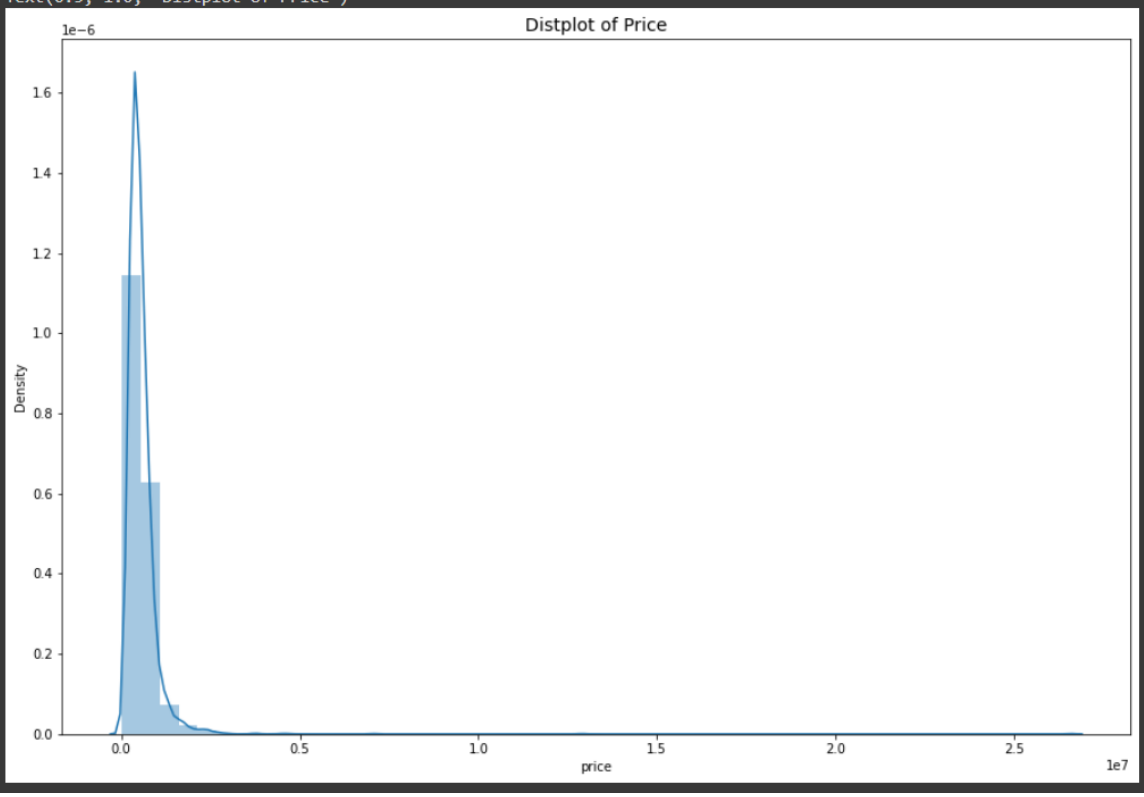
```
ax = sns.pairplot(df)
```



Ta vẽ biểu đồ pairplot để quan sát dữ liệu. Từ biểu đồ ta thấy sự phân tán giữa các cột khác nhau. Đồng thời cũng thấy có nhiều ngoài lệ cần chỉnh sửa. Ta sẽ đi sâu hơn vào nó.

```
plt.figure(figsize=(15,10))
ax = sns.distplot(df['price'], kde=True)
ax.set_title('Distplot of Price', fontsize=14)
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function
  warnings.warn(msg, FutureWarning)
Text(0.5, 1.0, 'Distplot of Price')
```

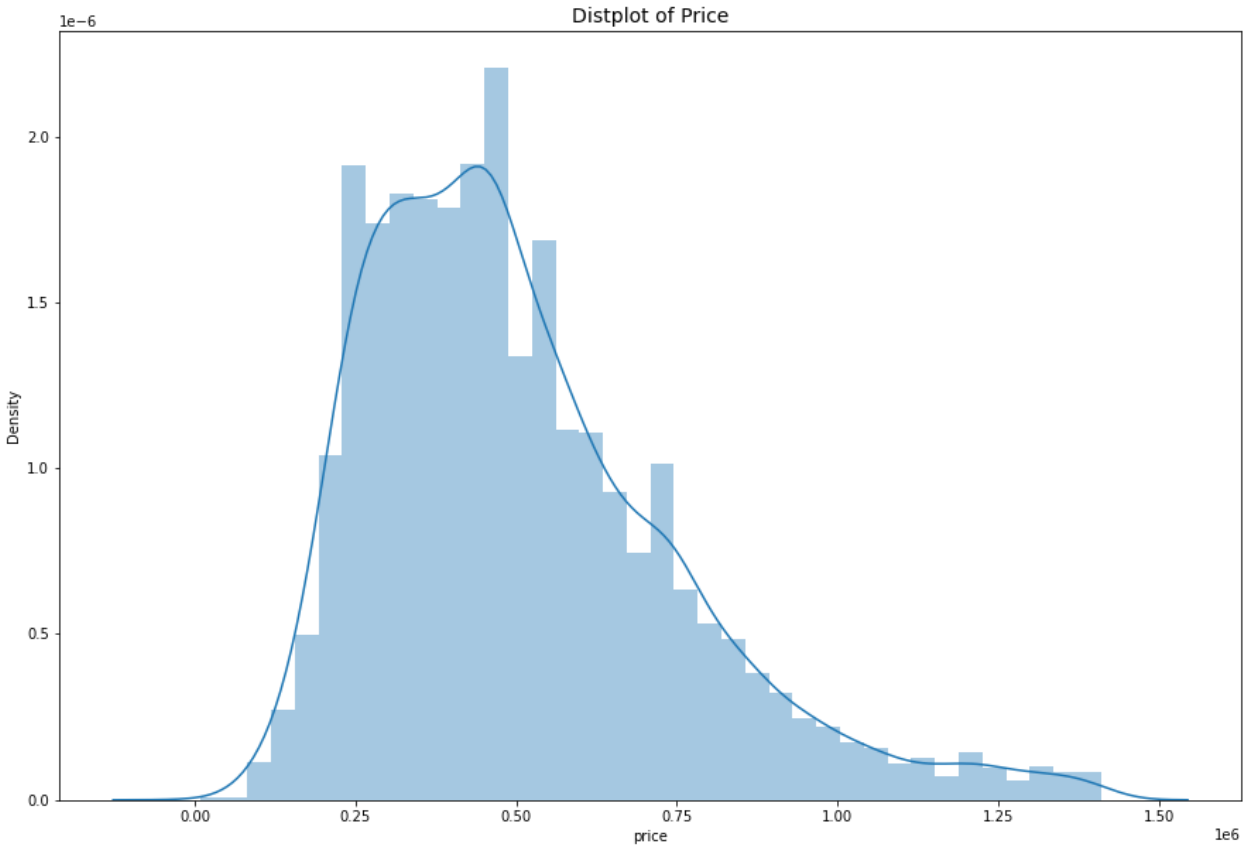


Đầu tiên ta xem biểu đồ phân phối chuẩn của giá nhà, giá nhà có độ phân phối lệch rất cao. Ta có thể điều chỉnh việc này bằng cách loại bỏ hoặc có thể thay thế bằng một giá trị khác.

```
df['price'] = df['price'].replace([df['price'][np.abs(stats.zscore(df['price'])) > 3]],np.median(df['price']))

plt.figure(figsize=(15,10))
ax = sns.distplot(df['price'], kde=True)
ax.set_title('Distplot of Price', fontsize=14)
```

Ta lựa chọn cách thay thế các điểm z thành giá trị trung vị.



Biểu đồ đã được cải thiện đáng kể, giờ ta sẽ trực quan hóa nhanh một số dữ liệu.

```
plt.figure(figsize=(15,6))
ax = sns.scatterplot(data=df, x="sqft_living", y="price")
ax.set_title('Sqft_living VS Price', fontsize=14)
```

```
Text(0.5, 1.0, 'Sqft_living VS Price')
```

Có nhiều dữ liệu nhỏ hơn 6000 nên ta sẽ xử lý dữ liệu không đồng đều này.

```
df['sqft_living'] = np.where((df.sqft_living >6000 ), 6000, df.sqft_living)
```

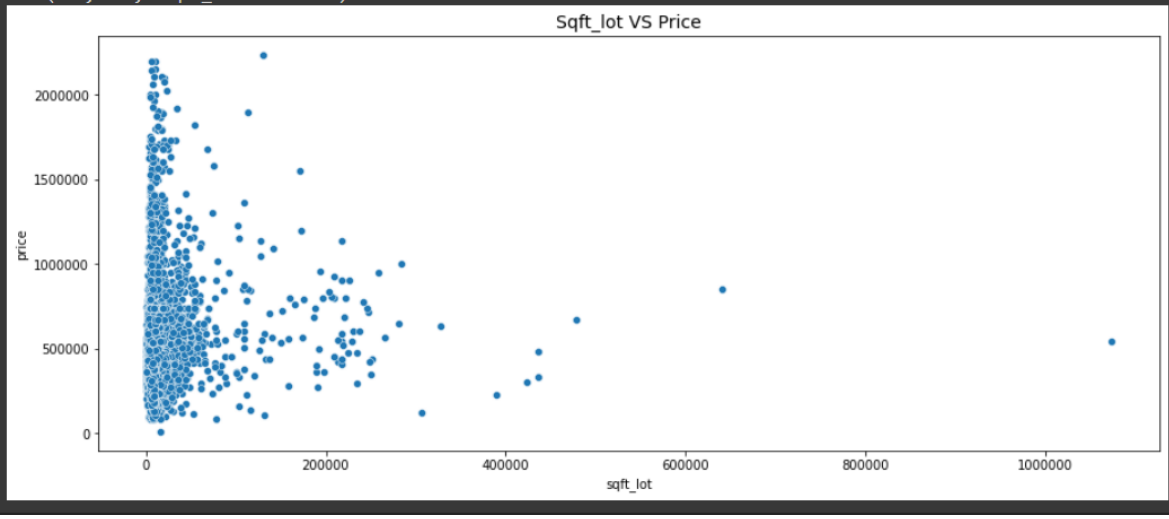
```
[ ]  plt.figure(figsize=(15,6))
     ax = sns.scatterplot(data=df, x="sqft_living", y="price")
     ax.set_title('Sqft_living VS Price', fontsize=14)
```

```
Text(0.5, 1.0, 'Sqft_living VS Price')
```

Ta đã điều chỉnh các giá trị vượt ngưỡng 6000 xuống còn 6000. Tương tự làm với các cột khác.

```
plt.figure(figsize=(15,6))
ax = sns.scatterplot(data=df, x="sqft_lot", y="price")
ax.ticklabel_format(style='plain')
ax.set_title('Sqft_lot VS Price', fontsize=14)
```

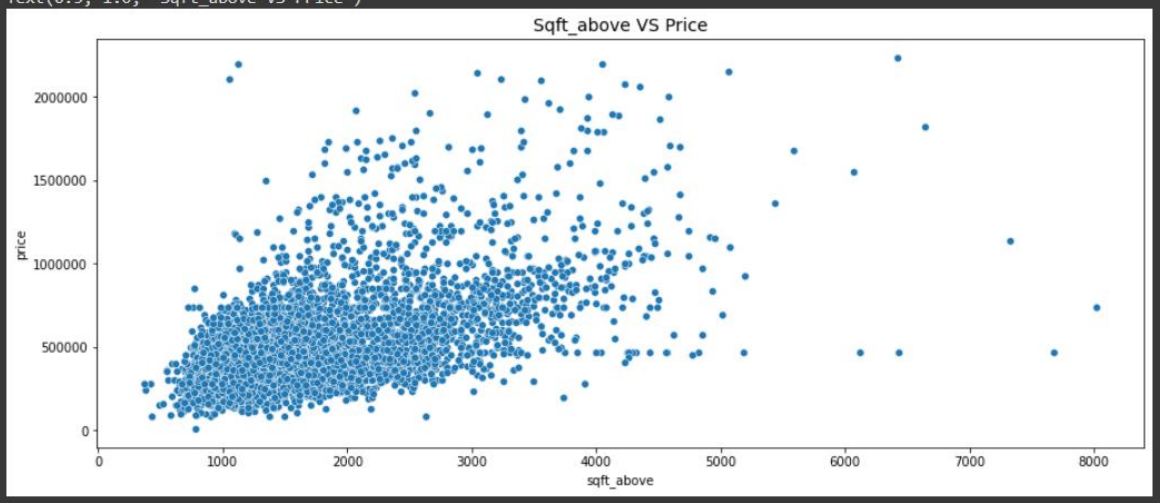
```
Text(0.5, 1.0, 'Sqft_lot VS Price')
```



Quá nhiều xáo trộn trong cột này và như đã kiểm tra ở trên thì nó không tương quan nhiều với cột giá. Vì vậy, ta sẽ để yên nó.

```
plt.figure(figsize=(15,6))
ax = sns.scatterplot(data=df, x="sqft_above", y="price")
ax.ticklabel_format(style='plain')
ax.set_title('Sqft_above VS Price', fontsize=14)
```

```
Text(0.5, 1.0, 'Sqft_above VS Price')
```

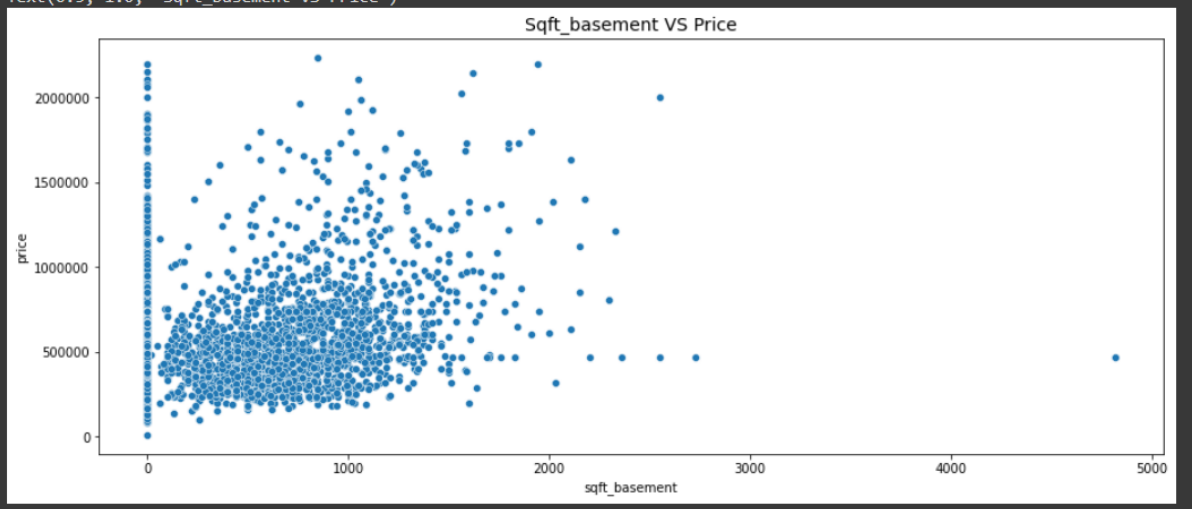



```
[ ]  df['sqft_above'] = np.where((df.sqft_above >5000 ), 5000, df.sqft_above)
```

Chỉnh sửa cột sqft_above.

```
plt.figure(figsize=(15,6))
ax = sns.scatterplot(data=df, x="sqft_basement", y="price")
ax.ticklabel_format(style='plain')
ax.set_title('Sqft_basement VS Price', fontsize=14)
```

```
Text(0.5, 1.0, 'Sqft_basement VS Price')
```




```
[ ]  df['sqft_basement'] = np.where((df.sqft_basement >2000 ), 2000, df.sqft_basement)
```

Chỉnh sửa cột sqft_basement.

5.5. Xác định và xử lý nhiễu

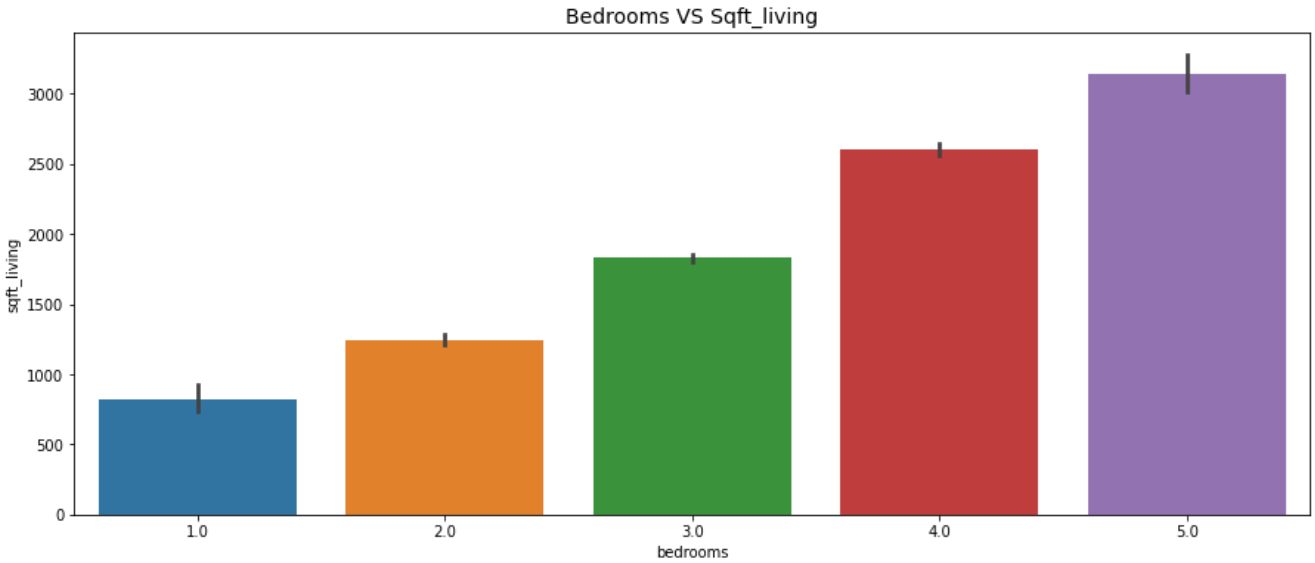
Như từ đầu đã xem, ta thấy một số đặc trưng như phòng ngủ, phòng tắm,… mang kiểu dữ liệu float nên ta sẽ chỉnh sửa chúng thành kiểu int.

```
df.bedrooms= df.bedrooms.astype(int)
df.bathrooms = df.bathrooms.astype(int)
df.floors = df.floors.astype(int)
```

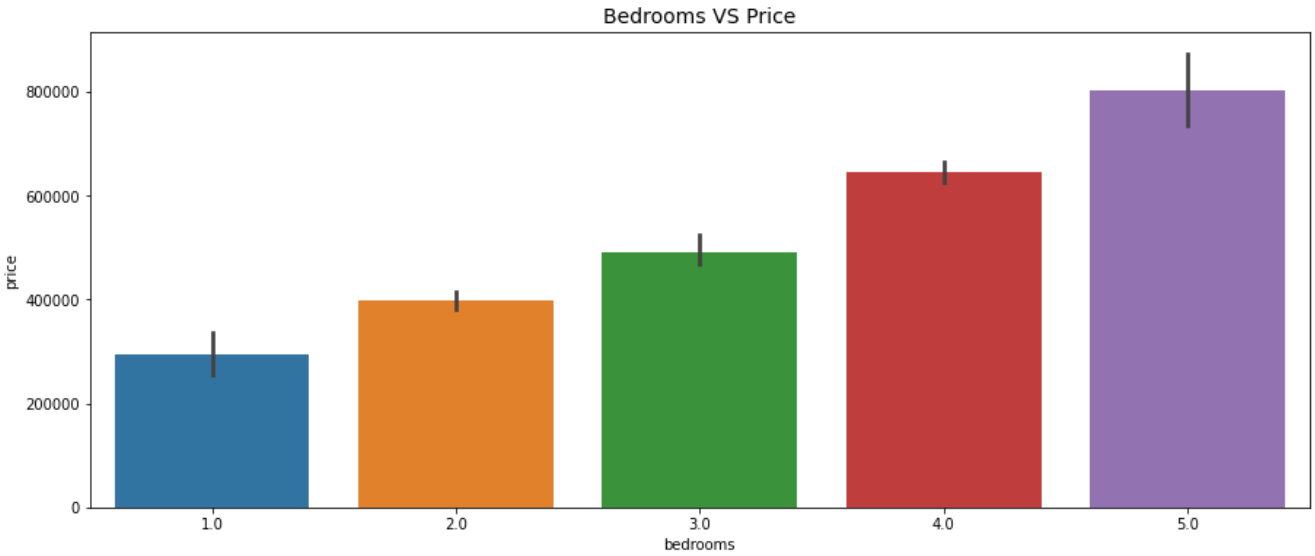
Ta lần lượt xem sự tương quan giữa các cột với cột giá nhà và cột diện tích sống.

```
plt.figure(figsize=(15,6))
ax = sns.barplot(x=df['bathrooms'], y=df['price'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=0)
ax.set_title('Bathrooms VS Price', fontsize=14)

plt.figure(figsize=(15,6))
ax = sns.barplot(x=df['bathrooms'], y=df['sqft_living'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=0)
ax.set_title('Bathrooms VS Sqft_living', fontsize=14)
```



Ta thấy chiều cao của các cột ‘sqft_living’ tăng dần theo chiều tăng của ‘bedrooms’. Điều đó thể hiện rằng ngôi nhà có số phòng ngủ càng nhiều thì diện tích dùng để của nó càng lớn.



Tương tự như vậy, ta thấy chiều cao của các cột 'price' tăng dần theo chiều tăng của 'bedrooms'. Điều đó thể hiện rằng ngôi nhà có số phòng ngủ càng nhiều thì giá bán của nó càng cao.

Từ đó rút ra được kết luận là những ngôi nhà có số phòng ngủ càng nhiều thì diện tích dùng để ở của nó càng lớn và giá bán của nó cũng càng cao.

Do sửa thành kiểu dữ liệu int nên xuất hiện thêm cột 0 phòng, nhìn chung giá của các căn 0 phòng cũng ngang xấp xỉ với các căn 1 phòng nên ta sẽ xem số lượng nhà có 0 phòng là bao nhiêu, nếu thấp ta sẽ lựa chọn việc thay cột 0 bằng 1.

```
df.bathrooms.value_counts()
```

```
2    2282
1    1659
3     482
4      66
0      17
5      10
6       4
Name: bathrooms, dtype: int64
```

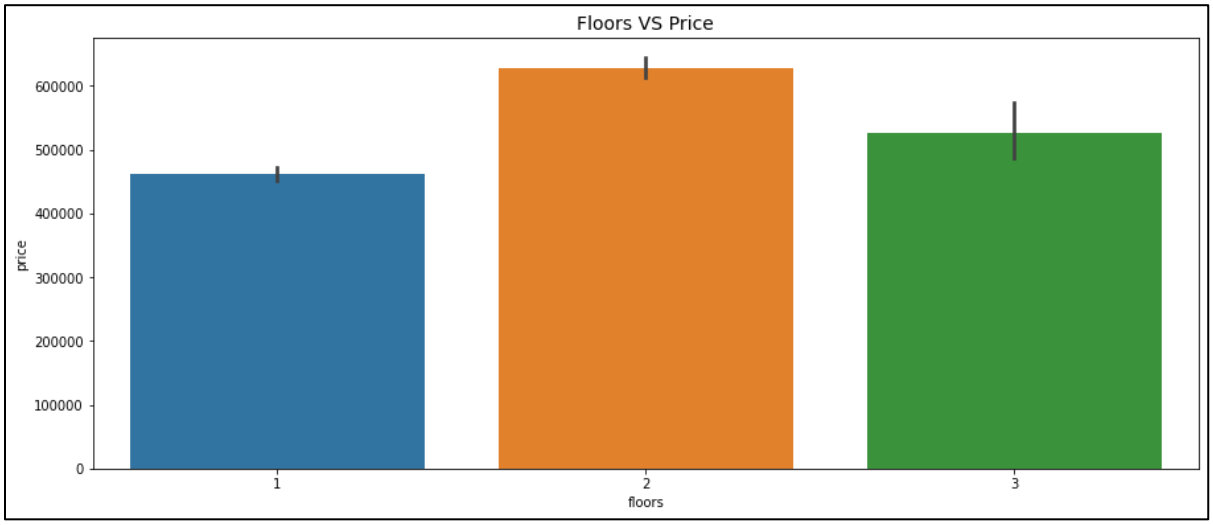
Số lượng cũng không nhiều nên ta quyết định thay đổi.

```
df['bathrooms'].replace(to_replace = 0, value = 1, inplace = True)
```

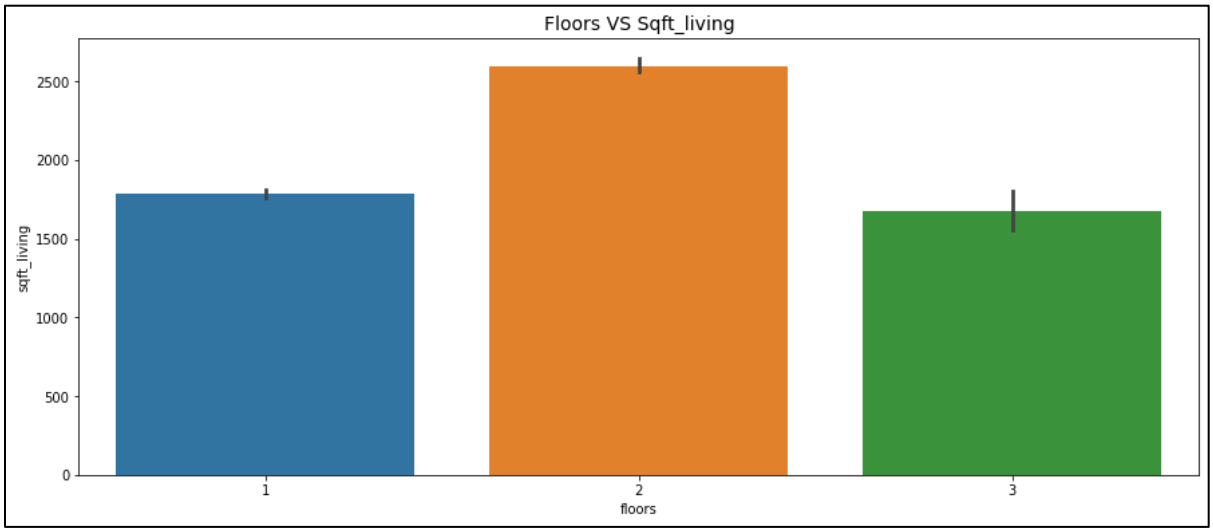
Ta chuyển sang xem thử số lượng lầu của các căn nhà như nào thì thấy cũng ổn định nên sẽ không chỉnh sửa gì hết.

```
plt.figure(figsize=(15,6))
ax = sns.barplot(x=df['floors'], y=df['price'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=0)
ax.set_title('Floors VS Price', fontsize=14)

plt.figure(figsize=(15,6))
ax = sns.barplot(x=df['floors'], y=df['sqft_living'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=0)
ax.set_title('Floors VS Sqft_living', fontsize=14)
```



Ta thấy được giá nhà của những căn nhà có 2 lầu là cao nhất.

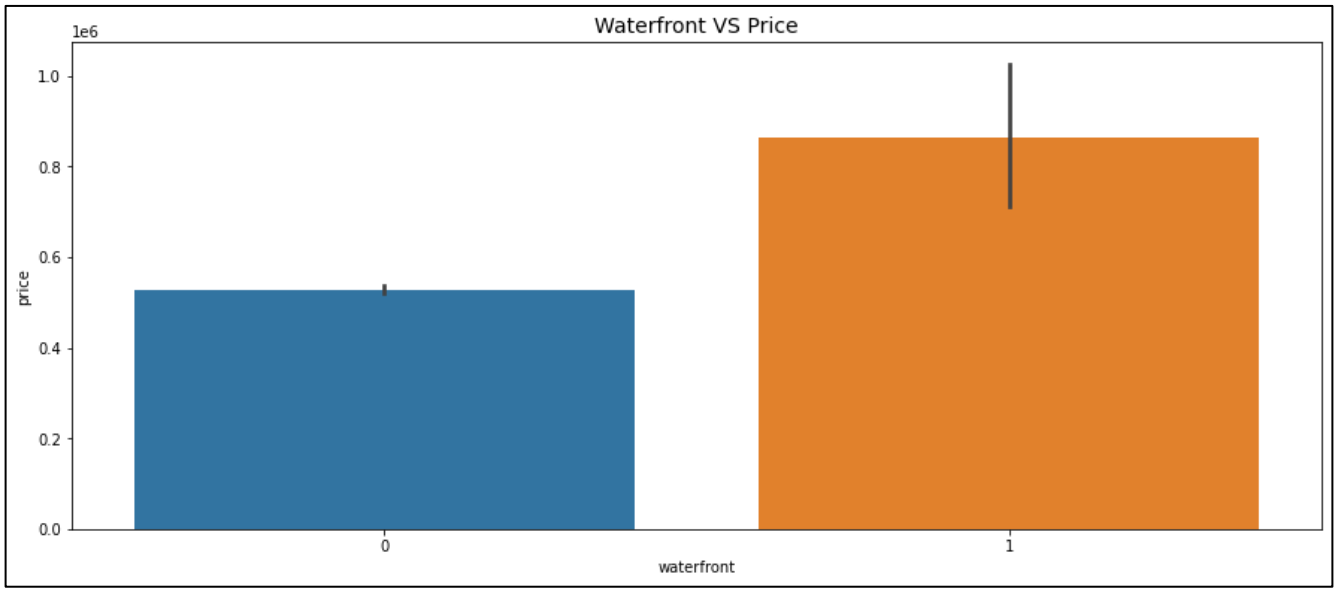


Ta cũng thấy được do giá nhà của những căn nhà có 2 lầu là cao nhất nên diện tích dùng để ở của nó cũng lớn nhất.

Tiếp đây ta sẽ xem các đặc trưng khác với giá nhà. Đầu tiên là ‘waterfront’.

```
plt.figure(figsize=(15,6))
ax = sns.barplot(x=df['waterfront'], y=df['price'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=0)
ax.set_title('Waterfront VS Price', fontsize=14)

waterfront = df.groupby(['waterfront']).price.agg([len , min, max])
waterfront
```

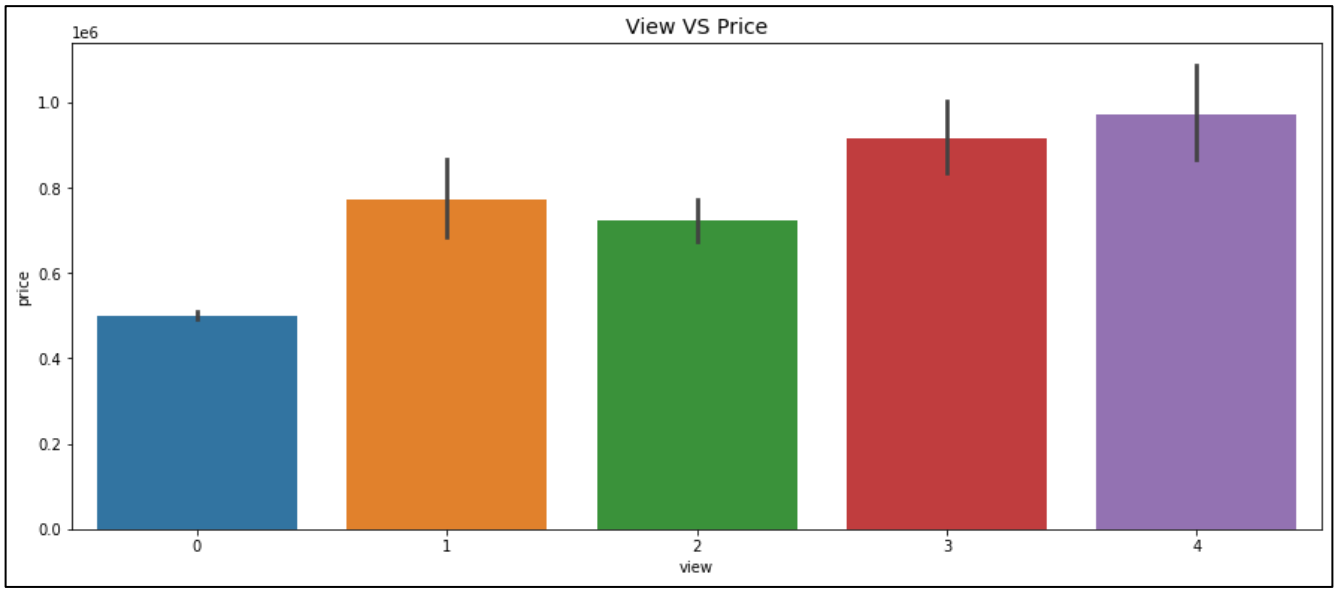


waterfront	len	min	max
0	4488	7800.0	2238888.0
1	32	385000.0	2200000.0

Thông số không có gì đáng quan trọng, ta chuyển sang ‘view’

```
plt.figure(figsize=(15,6))
ax = sns.barplot(x=df['view'], y=df['price'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=0)
ax.set_title('View VS Price', fontsize=14)

view = df.groupby(['view']).price.agg([len , min, max])
view
```

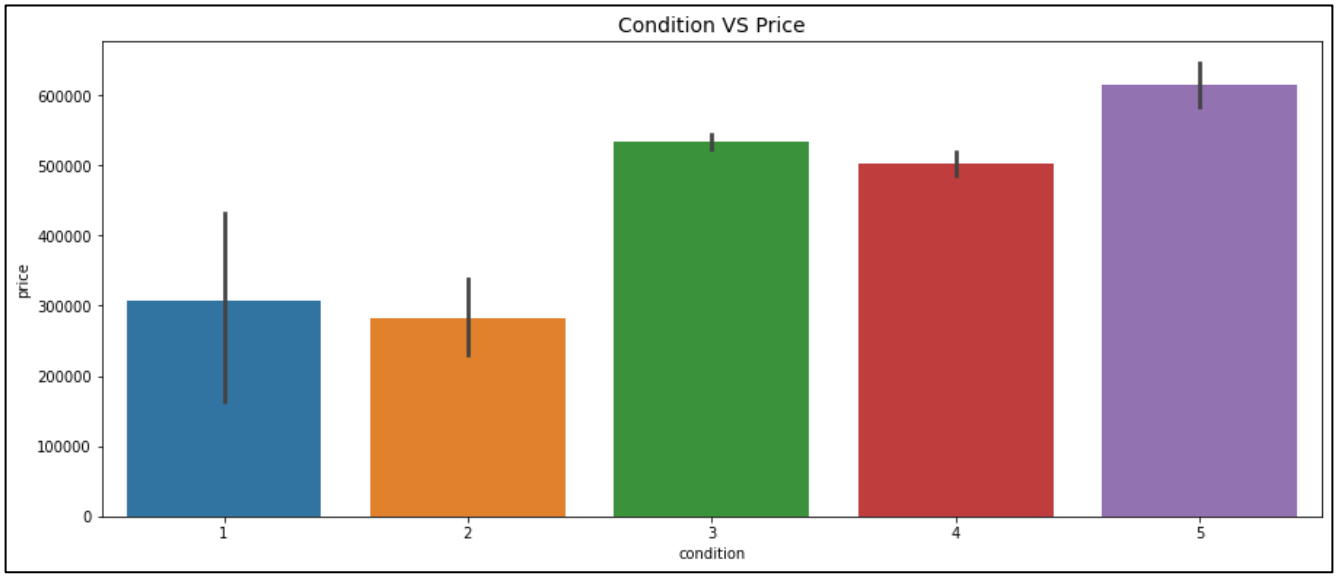


view	len	min	max
0	4086	7800.000000	2238888.0
1	66	180785.714286	1965221.0
2	197	175000.000000	1925000.0
3	108	107500.000000	2027000.0
4	63	361000.000000	2200000.0

Kế tiếp là ‘condition’.

```
plt.figure(figsize=(15,6))
ax = sns.barplot(x=df['condition'], y=df['price'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=0)
ax.set_title('Condition VS Price', fontsize=14)

condition = df.groupby(['condition']).price.agg([len , min, max])
condition
```



condition	len	min	max
1	6	7800.0	550000.0
2	31	80000.0	735000.0
3	2832	83000.0	2238888.0
4	1228	83300.0	2110000.0
5	423	83300.0	2147500.0

Số lượng căn có điều kiện là 1 khá thấp nên ta sẽ chỉnh sửa.

```
df['condition'] = np.where((df.condition == 1 ), 2, df.condition)
```

5.6. Mã hóa dữ liệu:

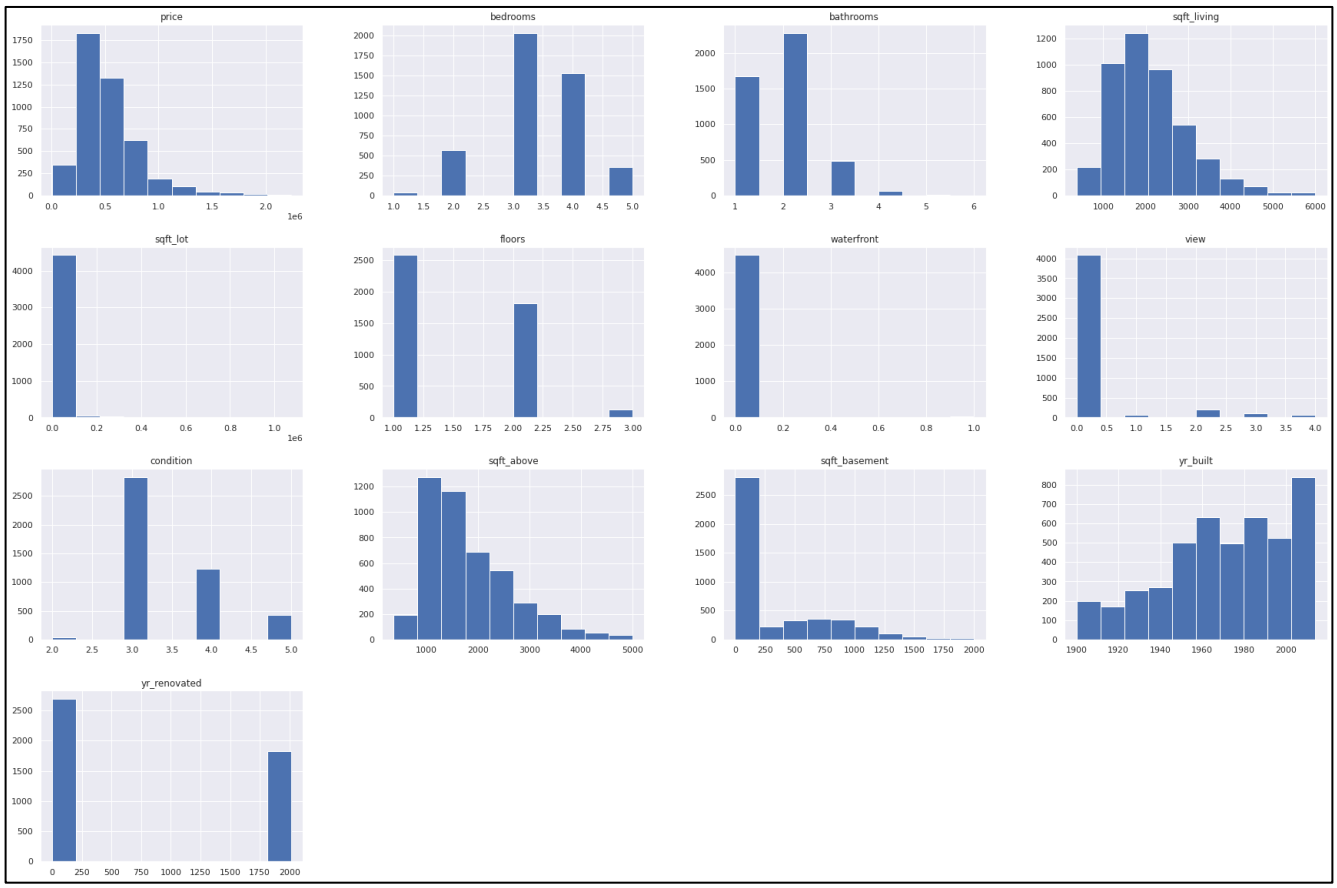
```
df = pd.get_dummies(df, columns=['statezip'], prefix = ['statezip'])
df.shape
```

```
(4520, 84)
```

Do ‘statezip’ có kiểu dữ liệu là object nên ta sẽ mã hóa nó theo theo phương pháp one hot encoding mỗi giá trị statezip khác nhau sẽ thành một cột, mỗi cột chứa 1 trong 2 giá trị là 0 và 1, ngôi nhà thuộc statezip nào thì cột đó sẽ bật lên 1 còn lại là 0.

5.7. Phân bố của từng đặc trưng như thế nào?

```
sns.set()
plt.rcParams['figure.figsize'] = [30,20]
df.hist()
plt.show()
```

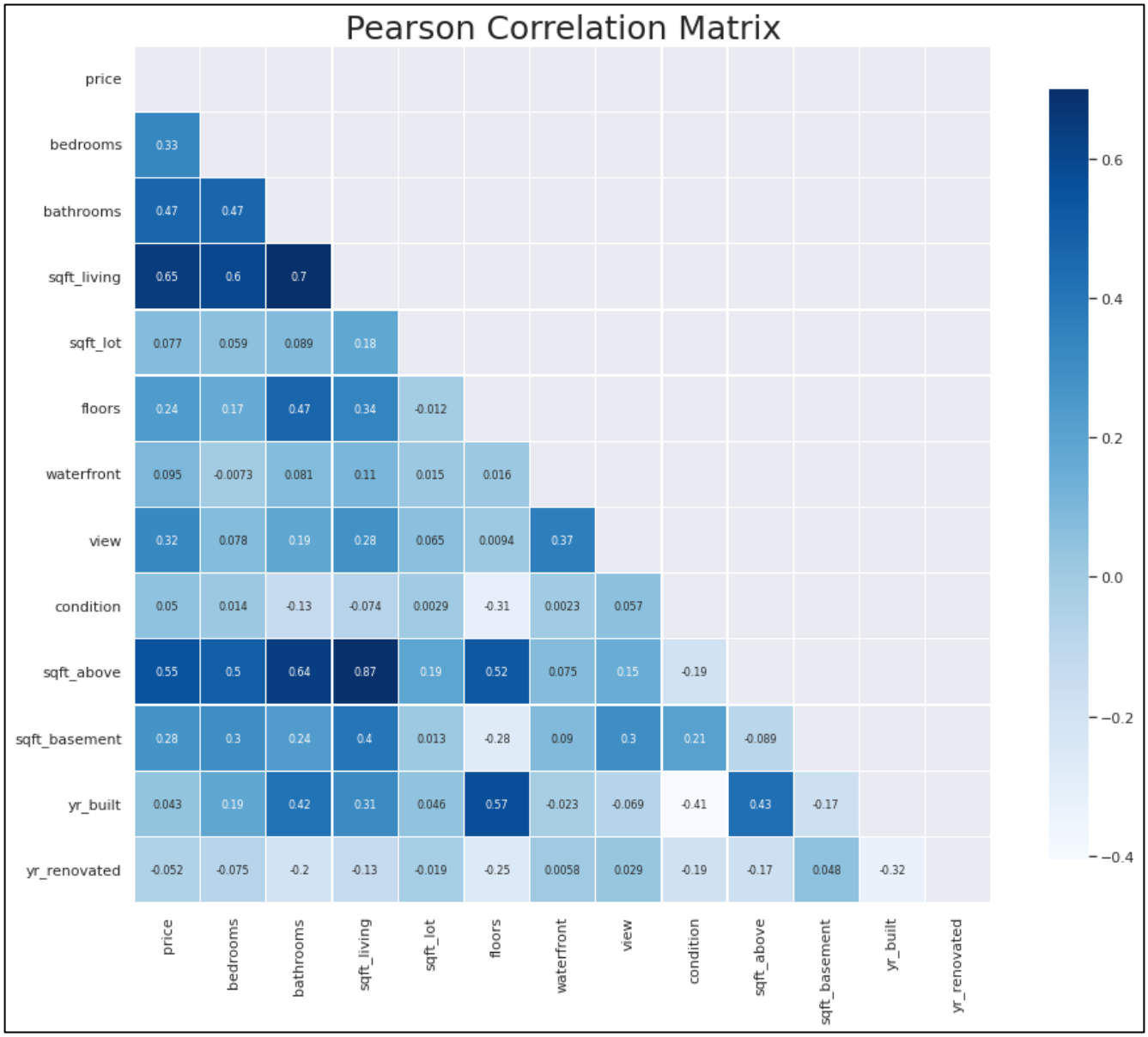


5.8. Mối tương quan của từng cặp đặc trưng

```
mask = np.zeros_like(df.corr(), dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

f, ax = plt.subplots(figsize=(16, 12))
plt.title('Pearson Correlation Matrix',fontsize=25)

sns.heatmap(df.corr(),linewidths=0.25,vmax=0.7,square=True,cmap="Blues",
            linecolor='w',annot=True,annot_kws={"size":8},mask=mask,cbar_kws={"shrink": .9});
```



VI. Đặc trưng

Chọn lọc đặc trưng

```
df.drop(["date",'yr_built','yr_renovated','sqft_lot'], axis=1, inplace = True)
```

Sau khi nhìn biểu đồ phân tán (pairplot), ta lược bỏ các đặc trưng có tương quan ít với giá nhà như ‘yr_built’, ‘yr_renovated’, ‘sqft_lot’ và các giá trị không ảnh hướng tới giá nhà như ‘date’.

```
X1 = df.drop(['street', 'city','statezip','country'], axis=1)
```

Kiểm tra đa cộng tuyến của các biến liên tục.

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant

X_vif = add_constant(X1)

pd.Series([variance_inflation_factor(X_vif.values, i)
               for i in range(X_vif.shape[1])],
              index=X_vif.columns)
```

```
const            58.116728
price             1.841407
bedrooms          1.656049
bathrooms         2.280215
sqft_living     425.726824
floors            1.767343
waterfront        1.174893
view              1.365457
condition         1.157972
sqft_above      358.194975
sqft_basement   101.452651
dtype: float64
```

Bởi vì điểm của sqft_living và sqft_above quá cao nên ta sẽ lựa chọn bỏ một trong 2 cái. Nhìn vào biểu đồ tương quan thì so với cột đặc trưng ‘price’, sqft_above có thấp hơn nên ta sẽ lựa chọn bỏ bớt cột sqft_above và xem lại số điểm.

```
X_vif = X_vif.drop(['sqft_above'],axis = 1)
pd.Series([variance_inflation_factor(X_vif.values, i)
                for i in range(X_vif.shape[1])],
               index=X_vif.columns)
```

```
const            58.721606
price             1.832595
bedrooms          1.654392
bathrooms         2.279669
sqft_living       3.567241
floors            1.763631
waterfront        1.162540
view              1.364388
condition         1.160958
sqft_basement     1.672734
dtype: float64
```

Điểm số đã trở nên tốt hơn, ta sẽ bỏ các đặc trưng có điểm tương quan thấp dựa vào biểu đồ thể hiện sự tương quan (heatmap).

```
df.drop(['waterfront','condition','sqft_above'],axis=1, inplace=True)
```

Ta kiểm tra lại một lần nữa mối quan hệ giữa các biến bị loại bỏ với ‘price’.

```
X1 = df.drop(['price', 'bedrooms', 'bathrooms', 'sqft_living', 'floors', 'view',
       'sqft_basement'],axis = 1)
y = df["price"]
```

```
import scipy.stats as stats
for i in X1.columns:
    print(stats.f_oneway(X1[i],y))
```

```
F_onewayResult(statistic=14629.99634464501, pvalue=0.0)
F_onewayResult(statistic=14629.996747470805, pvalue=0.0)
F_onewayResult(statistic=14629.996613195557, pvalue=0.0)
F_onewayResult(statistic=14629.996259197133, pvalue=0.0)
F_onewayResult(statistic=14629.99680850502, pvalue=0.0)
F_onewayResult(statistic=14629.995868578222, pvalue=0.0)
F_onewayResult(statistic=14629.99669864343, pvalue=0.0)
F_onewayResult(statistic=14629.996564368168, pvalue=0.0)
F_onewayResult(statistic=14629.997052641867, pvalue=0.0)
F_onewayResult(statistic=14629.996771884482, pvalue=0.0)
F_onewayResult(statistic=14629.996893952923, pvalue=0.0)
F_onewayResult(statistic=14629.996649816063, pvalue=0.0)
F_onewayResult(statistic=14629.996820711845, pvalue=0.0)
F_onewayResult(statistic=14629.99601506031, pvalue=0.0)
F_onewayResult(statistic=14629.997040435017, pvalue=0.0)
F_onewayResult(statistic=14629.996039473988, pvalue=0.0)
F_onewayResult(statistic=14629.99635685187, pvalue=0.0)
F_onewayResult(statistic=14629.996100508179, pvalue=0.0)
F_onewayResult(statistic=14629.996674229747, pvalue=0.0)
F_onewayResult(statistic=14629.996356851874, pvalue=0.0)
F_onewayResult(statistic=14629.996918366587, pvalue=0.0)
F_onewayResult(statistic=14629.99607609452, pvalue=0.0)
F_onewayResult(statistic=14629.995954026097, pvalue=0.0)
F_onewayResult(statistic=14629.995990646628, pvalue=0.0)
F_onewayResult(statistic=14629.997028228168, pvalue=0.0)
F_onewayResult(statistic=14629.996161542393, pvalue=0.0)
F_onewayResult(statistic=14629.99594181926, pvalue=0.0)
F_onewayResult(statistic=14629.99655216133, pvalue=0.0)
F_onewayResult(statistic=14629.9970892624, pvalue=0.0)
F_onewayResult(statistic=14629.997138089766, pvalue=0.0)
F_onewayResult(statistic=14629.997077055534, pvalue=0.0)
F_onewayResult(statistic=14629.995526786675, pvalue=0.0)
F_onewayResult(statistic=14629.995990646623, pvalue=0.0)
F_onewayResult(statistic=14629.996820711847, pvalue=0.0)
F_onewayResult(statistic=14629.996185956077, pvalue=0.0)
F_onewayResult(statistic=14629.997003814497, pvalue=0.0)
F_onewayResult(statistic=14629.996015060318, pvalue=0.0)
F_onewayResult(statistic=14629.995868578222, pvalue=0.0)
F_onewayResult(statistic=14629.996308024489, pvalue=0.0)
F_onewayResult(statistic=14629.997150296604, pvalue=0.0)
F_onewayResult(statistic=14629.996808505015, pvalue=0.0)
F_onewayResult(statistic=14629.996393472367, pvalue=0.0)
F_onewayResult(statistic=14629.995966232942, pvalue=0.0)
F_onewayResult(statistic=14629.996161542393, pvalue=0.0)
F_onewayResult(statistic=14629.996539954485, pvalue=0.0)
F_onewayResult(statistic=14629.996161542393, pvalue=0.0)
```

Ta thấy không có pvalue nào có giá trị lớn hơn 0,05 chứng tỏ không có biến nào có mối quan hệ chặt chẽ với y nên trong trường hợp này, không có biến nào có tác động lớn tới y bị lược bỏ.

*Xử lý các biến có kiểu dữ liệu là object:

- Xử lý ‘country’ và ‘street’:

```
df['country'].nunique()
```

```
1
```

Ta thấy tất cả các dòng trong bộ dữ liệu đều có chung duy nhất một quốc gia. Trong hồi quy, biến phụ thuộc phải dựa vào sự biến thiên của các biến độc lập để đưa ra kết quả dự đoán. Do đó, trong trường hợp này, ‘country’ sẽ không bao giờ ảnh hưởng đến ‘price’. Ta sẽ loại bỏ đặc trưng này.

```
df.shape
```

```
(4520, 11)
```

```
print(df['street'].nunique())
```

```
4446
```

Ta thấy ‘street’ có đến 4446 giá trị khác nhau trong khi kích thước của dữ liệu chỉ có 4520 dòng. Vì có quá nhiều giá trị như vậy nên biến phụ thuộc cũng sẽ không thể điều chỉnh để cho kết quả được. Do đó ‘street’ cũng sẽ không ảnh hưởng đến ‘price’ nên ta cũng sẽ loại bỏ đặc trưng này.

- Xử lý ‘city’ và ‘statezip’:

Ta sẽ lựa chọn 1 trong 2 đặc trưng xem đặc trưng nào liên hệ với giá nhiều nhất.

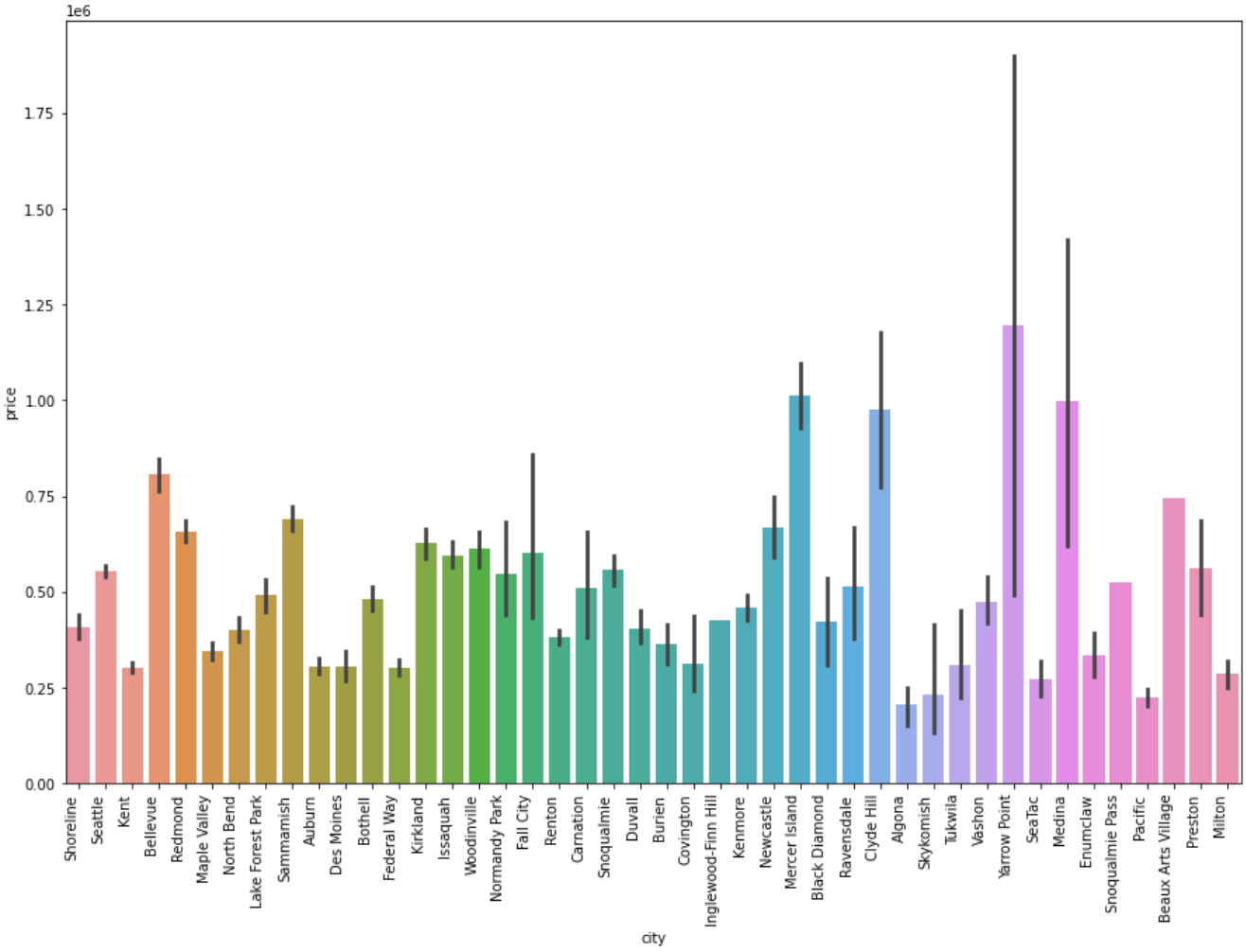
```
print(df['city'].nunique())
print(df['statezip'].nunique())
```

```
44
77
```

```
city = df.groupby(['city']).price.agg([len, min, max])
pd.set_option('display.max_rows',70)
city
```

city	len	min	max
Algona	5	100000.000000	262000.0
Auburn	173	87500.000000	900000.0
Beaux Arts Village	1	745000.000000	745000.0
Bellevue	280	248000.000000	2150000.0
Black Diamond	9	224000.000000	735000.0
Bothell	33	347000.000000	749995.0
Burien	71	100000.000000	1035000.0
Carnation	22	80000.000000	1680000.0
Clyde Hill	11	465000.000000	1388000.0
Covington	43	83300.000000	2199900.0
Des Moines	58	140000.000000	950000.0
Duvall	42	117833.333333	955000.0
Enumclaw	28	107500.000000	735000.0
Fall City	10	275000.000000	1550000.0
Federal Way	144	120750.000000	819000.0
Inglewood-Finn Hill	1	425000.000000	425000.0
Issaquah	183	195000.000000	2238888.0
Kenmore	65	238750.000000	1120000.0
Kent	183	100000.000000	735000.0
Kirkland	187	90000.000000	1710000.0
Lake Forest Park	34	260000.000000	790000.0
Maple Valley	96	108333.333333	735000.0
Medina	11	188000.000000	2100000.0
Mercer Island	82	435000.000000	2027000.0
Milton	2	250000.000000	320000.0
Newcastle	33	339900.000000	1200000.0
Normandy Park	18	192000.000000	1309500.0
North Bend	50	240000.000000	845000.0
Pacific	6	174000.000000	260000.0
Preston	2	439900.000000	685000.0
Ravensdale	7	225000.000000	792500.0
Redmond	232	170000.000000	1700000.0
Renton	288	100000.000000	1135250.0
Sammamish	174	237333.333333	2000000.0
SeaTac	29	110700.000000	735000.0
Seattle	1536	90000.000000	2147500.0
Shoreline	121	176225.000000	1735000.0
Skykomish	3	134000.000000	415000.0
Snoqualmie	70	235000.000000	1149000.0
Snoqualmie Pass	1	525000.000000	525000.0
Tukwila	29	7800.000000	2110000.0
Vashon	29	160000.000000	849900.0
Woodinville	114	250000.000000	1820000.0
Yarrow Point	4	84350.000000	1901000.0

```
plt.figure(figsize=(15,10))
ax = sns.barplot(x="city", y="price", data=df)
ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha="right");
```

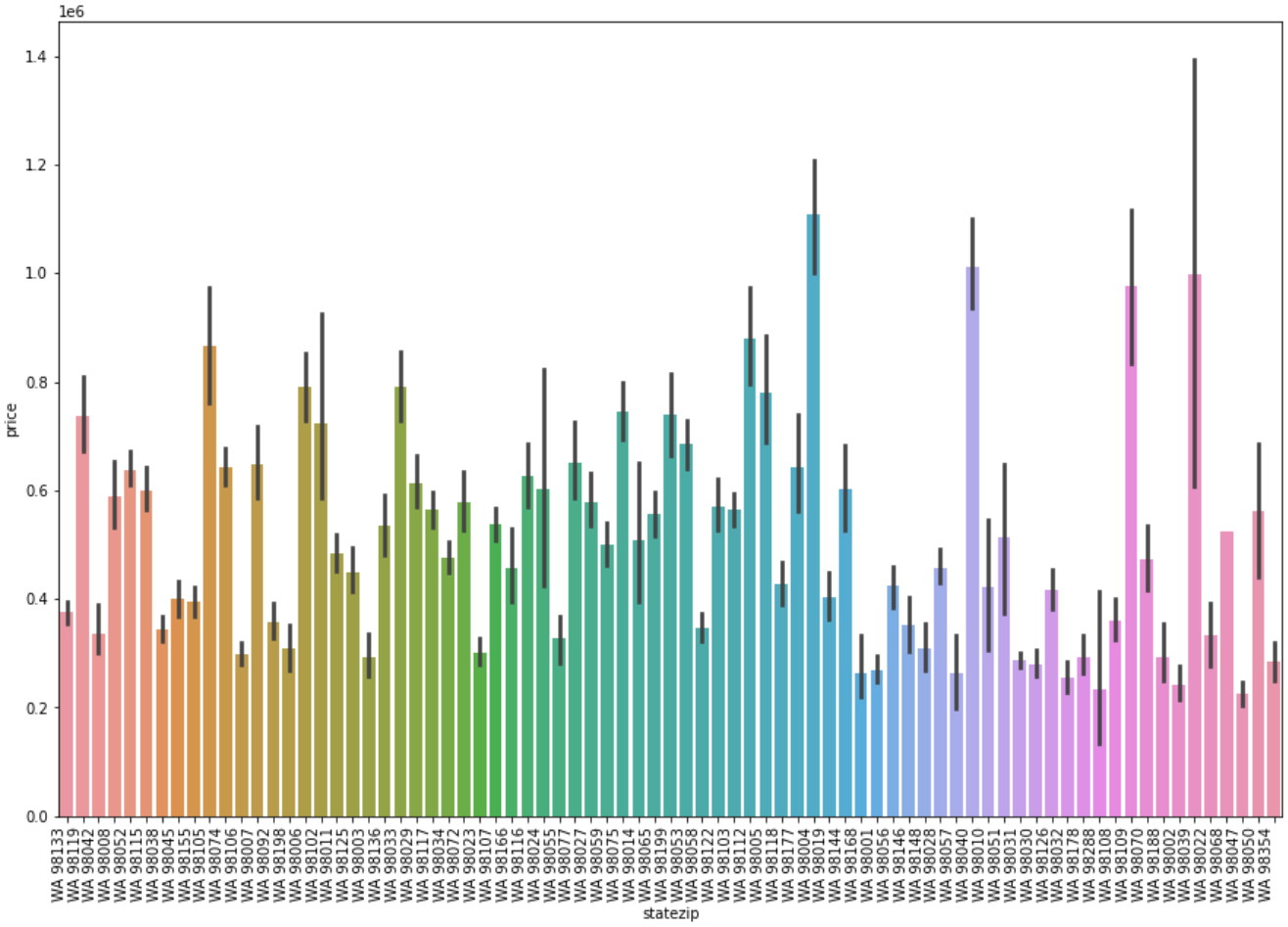



```
statezip = df.groupby(['statezip']).price.agg([len, min, max])
pd.set_option('display.max_rows',70)
statezip
```

statezip	len	min	max
WA 98001	67	100000.000000	735000.0
WA 98002	34	87500.000000	735000.0
WA 98003	45	156000.000000	819000.0
WA 98004	74	84350.000000	2150000.0
WA 98005	29	497333.333333	1900000.0
...	...	...	...
WA 98188	23	110700.000000	735000.0
WA 98198	56	140000.000000	950000.0
WA 98199	67	193000.000000	1655000.0
WA 98288	3	134000.000000	415000.0
WA 98354	2	250000.000000	320000.0

77 rows × 3 columns

```
plt.figure(figsize=(15,10))
ax = sns.barplot(x="statezip", y="price", data=df)
ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha="right");
```



Lựa chọn zip vì nó có variation cao hơn.

```
df.drop(['street','city','country'],axis=1, inplace=True)
```

VII. Huấn luyện mô hình

7.1. Chia dữ liệu thành các tập train, validation, test (k-fold, leave-one-out crossvalidation)

```
from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y= train_test_split(X,y, test_size=0.2, random_state=8)
```

7.2. Lựa chọn thuật toán để giải quyết bài toán

```
[ ]  baseline_models = ['Linear_Reg.','Random_Forest_Reg.','Lassio_Reg.','XGB_Reg.']
```

7.3. Các yếu tố có thể ảnh hưởng đến mô hình:

Các yếu tố có ảnh hưởng đến độ chính xác của mô hình hồi quy là các giá trị ngoại lai, các giá trị gây nhiễu và hiện tượng đa cộng tuyến giữa các biến. Các trường hợp này đều đã được chúng em giải quyết trong quá trình xử lý dữ liệu.

7.4. Huấn luyện mô hình

```
from sklearn.linear_model import LinearRegression
mlrm = LinearRegression()

mlrm.fit(train_X,train_y)
print("Training accuracy: ", mlrm.score(train_X, train_y))
print("Testing accuracy: ", mlrm.score(test_X, test_y))
scores.append(mlrm.score(test_X, test_y))
```

```
Training accuracy:  0.6804396363928333
Testing accuracy:  0.7207589106121428
```

```
from sklearn.ensemble import RandomForestRegressor
rf_model = RandomForestRegressor()
rf_model.fit(train_X,train_y)
print(rf_model.score(train_X, train_y))
print(rf_model.score(test_X, test_y))
```

```
0.9470908990473492
0.6544626745225748
```

```
from sklearn.linear_model import Lasso
lasso_reg= Lasso(max_iter=10000)
lasso_reg.fit(train_X, train_y)
print("Training accuracy: ", lasso_reg.score(train_X, train_y))
print("Testing accuracy: ", lasso_reg.score(test_X, test_y))
scores.append(lasso_reg.score(test_X,test_y))
```

```
Training accuracy:  0.6812316545399182
Testing accuracy:  0.7194029112467424
```

```
from xgboost import XGBRegressor
xgb_reg = XGBRegressor(max_depth= 3, n_estimators= 300, reg_lambda= 100)
xgb_reg.fit(train_X, train_y)
print("Training accuracy: ", xgb_reg.score(train_X, train_y))
print("Testing accuracy: ", xgb_reg.score(test_X, test_y))
scores.append(xgb_reg.score(test_X, test_y))
```

```
[13:00:08] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:lir
Training accuracy:  0.6830804136187244
Testing accuracy:  0.6595795288650853
```

VIII. Cải thiện mô hình

Tinh chỉnh tham số trong không gian tham số:

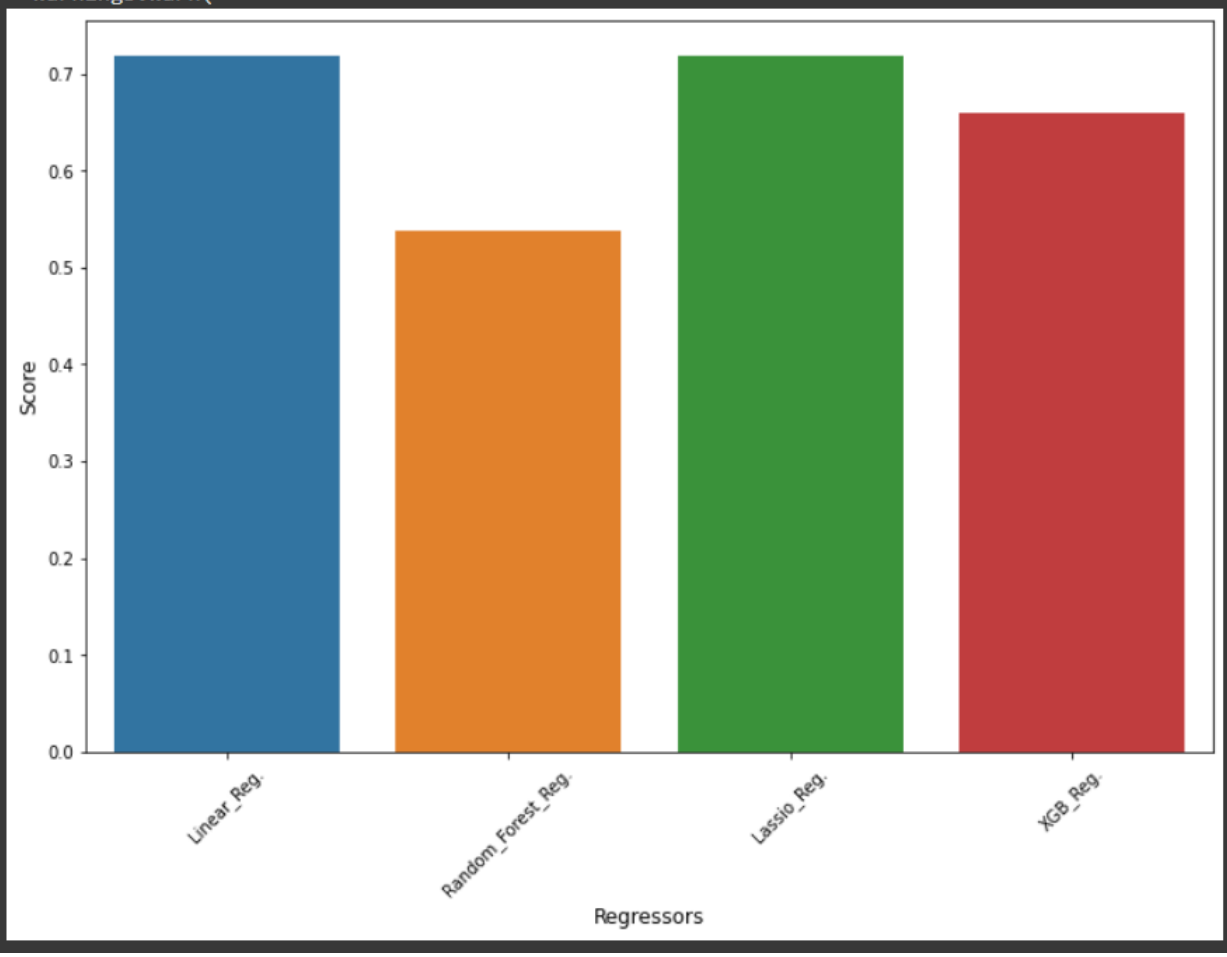
Do mô hình Random Forest thi huấn luyện theo phương pháp chia tập train-test có độ chính xác cho tập train xấp xỉ 94% nhưng độ chính xác đối với tập test chỉ đạt cấp xỉ 64% nên có xảy ra hiện tượng Overfitting đối với mô hình này. Do đó chúng em dùng GridSearchCV để điều chỉnh lại tham số cho mô hình.

IX. Lựa chọn mô hình

9.1. So sánh hiệu suất

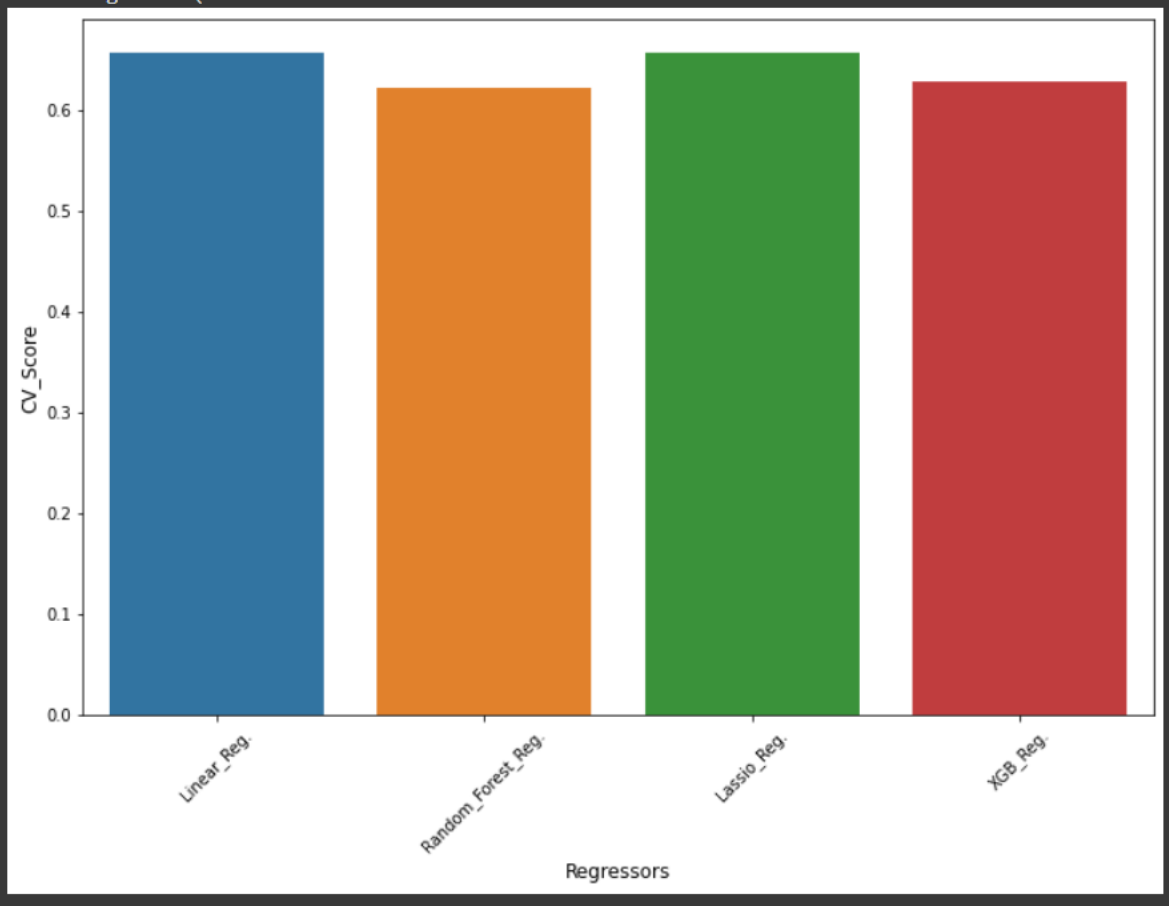
```
plt.figure(figsize = (12,8))
sns.barplot(final_score['Regressors'],final_score['Score'])
plt.xlabel('Regressors', fontsize = 12)
plt.ylabel('Score', fontsize = 12)
plt.xticks(rotation=45)
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the follow
  warnings.warn(
```



```
[ ]  plt.figure(figsize = (12,8))
     sns.barplot(final_score['Regressors'],final_score['CVScore'])
     plt.xlabel('Regressors', fontsize = 12)
     plt.ylabel('CV_Score', fontsize = 12)
     plt.xticks(rotation=45)
     plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following
  warnings.warn(
```



Sau khi kiểm tra độ chính xác bằng phương pháp chia tập train-test và kiểm tra bằng cross- validation, chúng em rút ra được kết luận là mô hình Lasso Regression có độ chính xác cao nhất, đứng thứ hai là mô hình Linear Regression, đứng thứ ba là mô hình XGB-Regression và cuối cùng là mô hình Random Forsest Regression.

```
from sklearn.metrics import r2_score
y_pred= lasso_reg.predict(test_X)
print(r2_score(test_y, y_pred))
```

```
0.7207793059332985
```

Chỉ số R- square của mô hình Lasso đạt được là xấp xỉ 0.72.

LỜI KẾT THÚC

Kết thúc bài tiểu luận, nhóm em đúc kết được thêm khá nhiều kiến thức mới mẻ. Do thời gian học ở trường không nhiều nên đồ án được yêu cầu ở mức đơn giản không quá khó, độ phức tạp ở mức thấp. Nhưng không vì vậy mà ta xem nhẹ đề tài, trong thời gian làm bài và tìm tòi cách giải quyết, chúng em học hỏi thêm được nhiều kiến thức mới lạ, nhiều hướng để giải quyết cho một vấn đề chung. Chính vì vậy chúng em càng cần phải khám phá nhiều hơn, không ngừng học hỏi để thúc đẩy bản thân phát triển. Và cũng vì ngành công nghệ thông tin nói chung và máy học nói riêng là thứ không ngừng đổi mới, cập nhật xu hướng, nên để bản thân học hành là tự giúp chính mình không bị xã hội đào thải khỏi hướng đi tương lai mà chúng em đang chọn.

Một lần nữa xin cám ơn giảng viên Vũ Ngọc Thanh Sang đã cùng chúng em trên hành trình chập chững học học phần Nhập môn máy học. Bản thân chúng em sẽ học hành chăm chỉ để không phụ công giảng dạy của thầy. Tuy nhiên trong qua trình nghiên cứu và thực hiện đồ án, kiến thức chuyên ngành vẫn còn hạn chế nên chúng em có thể có nhiều thiếu sót khi tìm hiểu và trình bày về bài toán. Rất mong nhận được sự thông cảm và góp ý của thầy để phần bài làm của chúng em được hoàn thiện hơn.