

Trường Khoa học Tự nhiên - Đại học Quốc Gia Thành phố Hồ Chí Minh

Khoa Công Nghệ Thông Tin

LẬP TRÌNH KHOA HỌC DỮ LIỆU

Đồ án Cuối kỳ

Thống kê, phân tích dữ liệu về định giá bán nhà ở HN

Giáo viên hướng dẫn

Thầy Bùi Tiến Lên

Thành phố Hồ Chí Minh, tháng 1 năm 2023



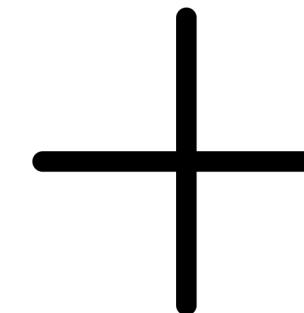
Thành viên nhóm

Mã số sinh viên	Họ và tên	Công việc
20120040	Nguyễn Quang Gia Bảo	Đặt câu hỏi + Khám phá và Tiền Xử lí dữ liệu
20120136	Huỳnh Tuấn Nam	Khám phá và Tiền Xử lí dữ liệu + Đặt câu hỏi
20120158	Trần Hoàng Anh Phi	Khám phá và Tiền xử lý dữ liệu + Mô hình hoá

Các nội dung thực hiện



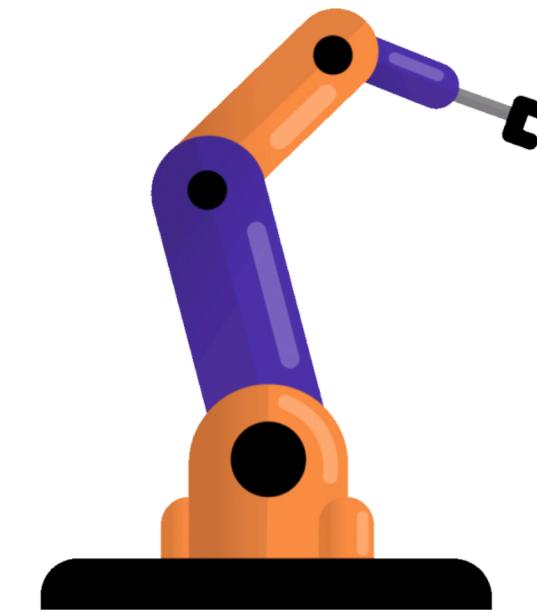
Khám phá dữ liệu



Tiền xử lý dữ liệu



Đặt câu hỏi



Mô hình hóa dữ liệu

Thông tin của dữ liệu

- Tên dữ liệu: Dữ liệu về giá nhà bán ở Hà Nội.
- Dữ liệu được lấy ở Kaggle.
- Đường dẫn của dữ liệu:

<https://www.kaggle.com/datasets/ladcva/vietnam-housing-dataset-hanoi>

- Tác giả: Le Anh Duc
- Giấy phép: CC BY-NC-SA 4.0



Tiền xử lý và Khám phá dữ liệu



Tiền xử lí và Khám phá dữ liệu

- Dataframe của bài toán được lưu trong biến *housing_df*
- Sử dụng hàm `pd.DataFrame.shape` để quan sát số chiều của Dataframe.

```
housing_df.shape
```

Kết quả trả ra: Dataframe có **82497** dòng và **13** cột

- Sử dụng hàm `pd.DataFrame.info()` để xem thông tin cơ bản của các cột.

```
housing_df.info()
```

Kết quả cho ta biết:

- Kiểu dữ liệu ở các cột đều là **object**.
- Hầu hết tất cả các cột đều bị thiếu dữ liệu.

Tiền xử lí và Khám phá dữ liệu

Kiểm tra các cột

- Các cột của bảng dữ liệu.

Sử dụng hàm **pd.DataFrame.columns** để xem tên của các cột.

```
Index(['Unnamed: 0', 'Ngày', 'Địa chỉ', 'Quận', 'Huyện', 'Loại hình nhà ở',  
       'Giấy tờ pháp lý', 'Số tầng', 'Số phòng ngủ', 'Diện tích', 'Dài',  
       'Rộng', 'Giá/m2'],  
      dtype='object')
```

- Điều chỉnh các cột để dễ dàng làm việc trong quá trình thực hiện bài toán.
 - Loại bỏ hàm Unnamed: 0.

Thông thường khi lấy dữ liệu trên Kaggle, luôn có một cột là **Unnamed: 0** để đánh số thứ tự. Vậy nên ta sẽ sử dụng hàm **drop()** của **pd.DataFrame** để loại bỏ cột đó.

```
housing_df = housing_df.drop(['Unnamed: 0'], axis = 1)
```

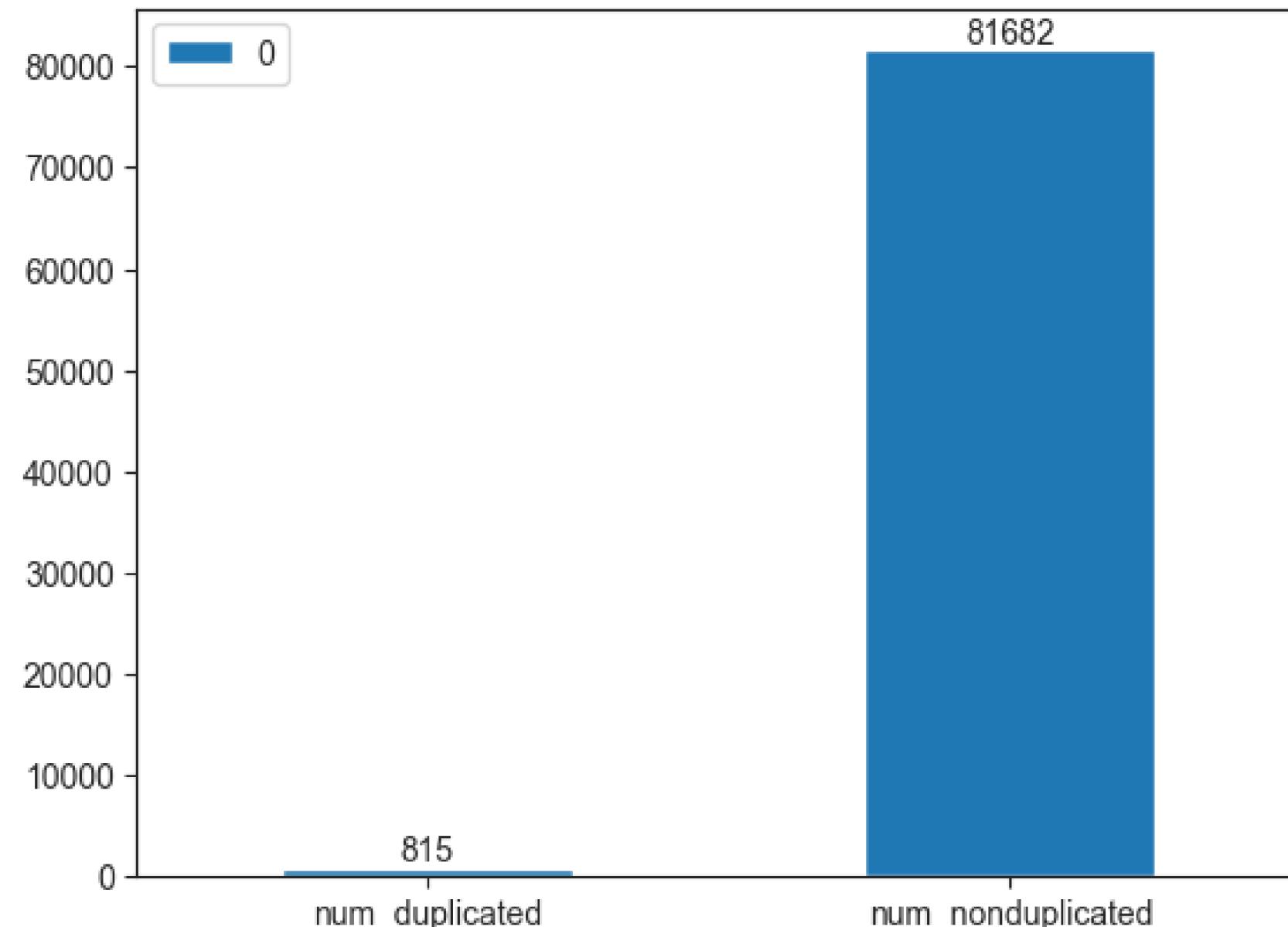
Tiền xử lí và Khám phá dữ liệu

Kiểm tra các dòng

- Kiểm tra dữ liệu bị trùng

Sử dụng hàm **pd.DataFrame.duplicated()** để tìm các dòng bị trùng nhau bằng cách trả về True/ False ở từng dòng. Và sử dụng hàm **sum()** để tính tổng các giá trị True False đó. True = 1 và False = 0.

=> Dựa vào biểu đồ bên, có thể thấy có 815 dòng nào trùng với nhau



Tiền xử lí và Khám phá dữ liệu

Kiểm tra các cột

- Điều chỉnh các cột để dễ dàng làm việc trong quá trình thực hiện bài toán.
 - Thay đổi tên của các cột sang tiếng Anh.

Các cột Tiếng Việt rất khó cho việc sử dụng indexing để truy cập vì phải chuyển đổi cách gõ phím liên tục. Để dễ dàng hơn, dựa vào tên cột Tiếng Việt, ta sẽ chuyển nghĩa tiếng Anh để dễ thực hiện hơn.

Tên cột Tiếng Việt	Tên cột Tiếng Anh	Tên cột Tiếng Việt	Tên cột Tiếng Anh	Tên cột Tiếng Việt	Tên cột Tiếng Anh
Ngày	Date	Loại hình nhà ở	Type	Diện tích	Area
Địa chỉ	Address	Giấy tờ pháp lý	Legal	Dài	Length
Quận	District	Số tầng	Number of floors	Rộng	Width
Huyện	Ward	Số phòng ngủ	Number of bedrooms	Giá/m2	Price/m2

Tiền xử lí và Khám phá dữ liệu

Kiểm tra các cột

- Kiểm tra mức độ thiếu dữ liệu của các cột.

```
missing_dictionary = {  
    "num_missing": housing_df.isna().sum(),  
    "missing_ratio (%)": housing_df.isna().sum()*100/len(housing_df)}  
missing_df = pd.DataFrame(missing_dictionary)
```

Cột **num_missing** để đếm số lượng dòng thiếu của cột.

Cột **missing_ratio** để tính % số lượng dòng thiếu so với tổng thể của cột đó.

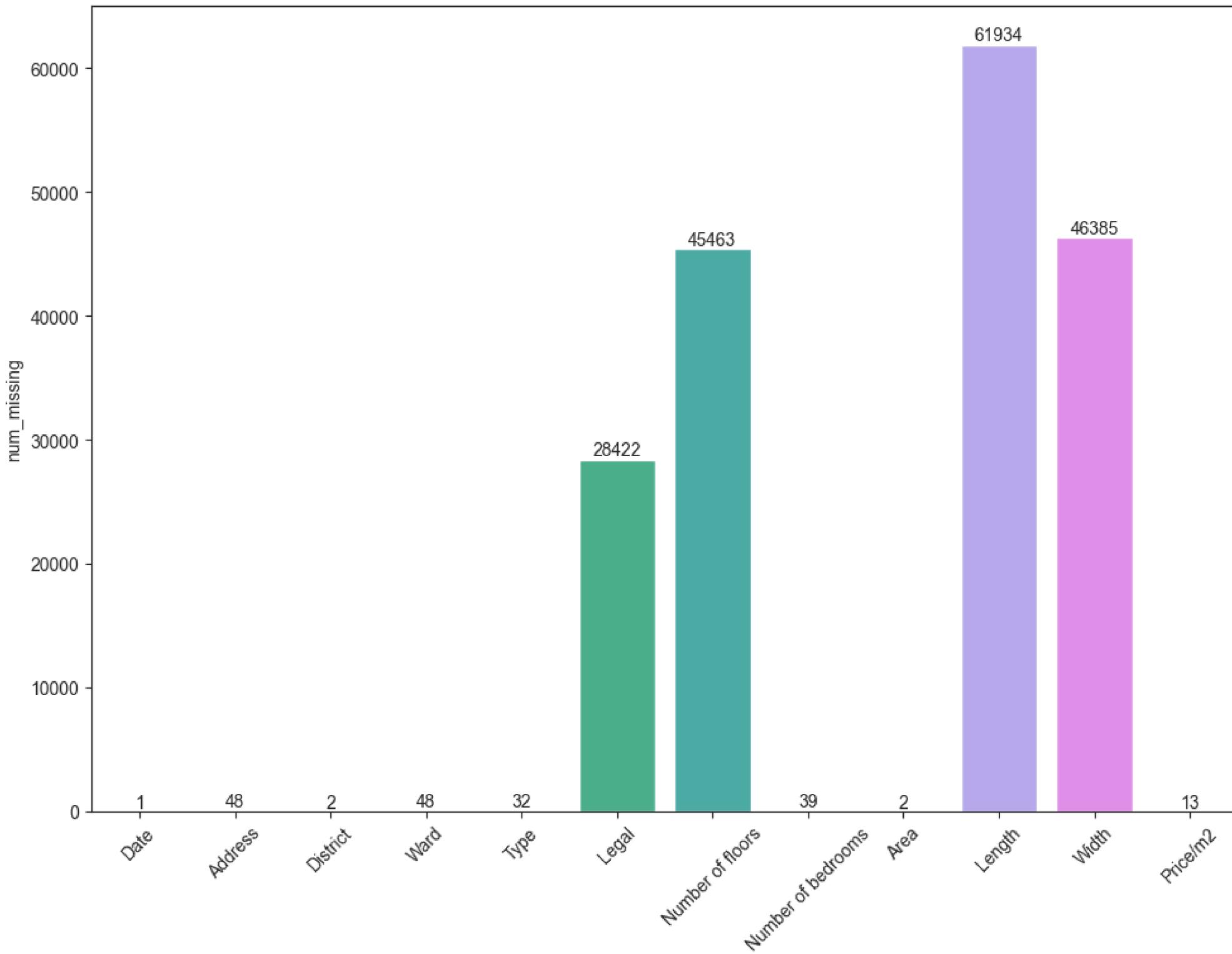
	num_missing	missing_ratio (%)
Date	1	0.001224
Address	48	0.058764
District	2	0.002449
Ward	48	0.058764
Type	32	0.039176
Legal	28422	34.795916
Number of floors	45463	55.658529
Number of bedrooms	39	0.047746
Area	2	0.002449
Length	61934	75.823315
Width	46385	56.787297
Price/m2	13	0.015915

Tiền xử lí và Khám phá dữ liệu

Kiểm tra các cột

- Kiểm tra mức độ thiếu dữ liệu của các cột.

Các cột đều có sự thiếu sót. Tuy nhiên các cột **Legal, Number of floors, Length, Width** bị thiếu rất nhiều.



Tiền xử lí và Khám phá dữ liệu

Kiểm tra các cột

- Kiểm tra kiểu dữ liệu bên trong một cột

Tạo ra một hàm **open_object_dtype** để triển khai các kiểu dữ liệu bên trong một cột

```
def open_object_dtype(series):  
    setType = list(set(series.apply(lambda x: type(x))))  
    return setType
```

	Date	Address	District	Ward	Type	Legal	Number of floors	Number of bedrooms	Area	Length	Width	Price/m2
0	<class 'float'>	<class 'float'>	<class 'float'>	<class 'float'>	<class 'float'>	<class 'float'>						
1	<class 'str'>	<class 'str'>	<class 'str'>	<class 'str'>	<class 'str'>	<class 'str'>						

Kết quả thu được, là tất cả các cột đều có 2 kiểu dữ liệu và string và float.

- string là các giá trị.
- float có thể vừa là giá trị vừa là giá trị np.nan

Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Date

Kiểu dữ liệu của cột **Date** là string và float. Cần thay đổi kiểu dữ liệu thành datetime.

Trước hết, nhận thấy cột **Date** có một dòng bị thiếu dữ liệu. Kiểm tra cho thấy:

	Date	Address	District	Ward	Type	Legal	Number of floors	Number of bedrooms	Area	Length	Width	Price/m2
81681	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Các cột của dòng này đều mang giá trị là NaN, và dòng này cũng là dòng cuối cùng của DataFrame, nên ta thực hiện việc drop một cách đơn giản.

```
housing_df = housing_df.iloc[:-1]
```

Tiền xử lí và Khám phá dữ liệu

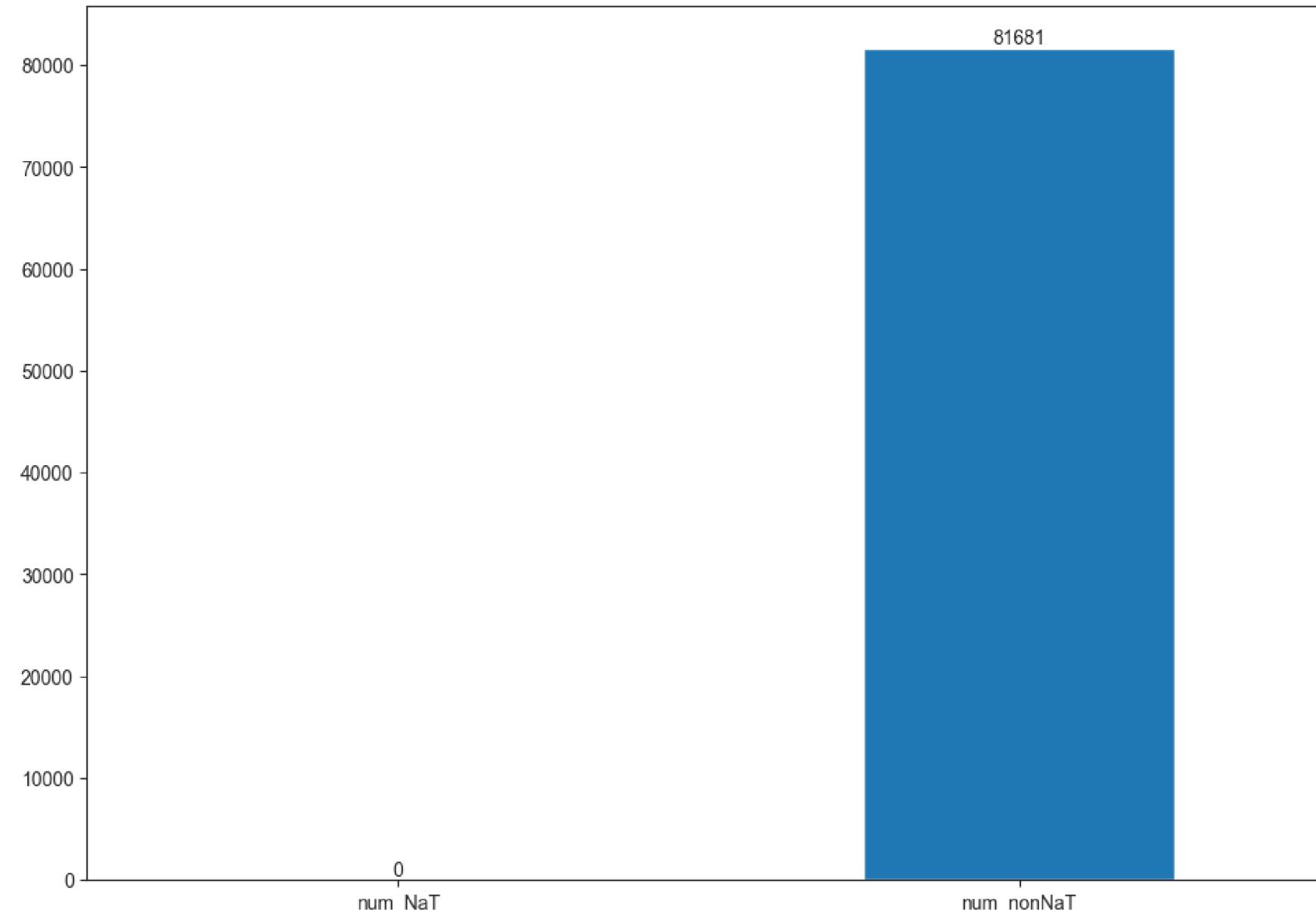
Khám phá từng cột dữ liệu

- Date

Cuối cùng, chuyển cột **Date** sang kiểu dữ liệu datetime.

```
housing_df.Date = pd.to_datetime(housing_df.Date, errors = 'coerce')
```

Kết quả thu được, là các giá trị đều được chuyển đổi thành công sang kiểu dữ liệu datetime, và không có cột nào bị lỗi.



Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Address - District - Ward

Ở cột **Address**, là một chuỗi string gồm địa chỉ, tên đường, quận, huyện, xã, và thành phố.

	Address
0	Đường Hoàng Quốc Việt, Phường Nghĩa Đô, Quận Cầu Giấy, Hà Nội
1	Đường Kim Giang, Phường Kim Giang, Quận Thanh Xuân, Hà Nội
2	phố minh khai, Phường Minh Khai, Quận Hai Bà Trưng, Hà Nội
3	Đường Võng Thị, Phường Thụy Khuê, Quận Tây Hồ, Hà Nội
4	Đường Kim Giang, Phường Kim Giang, Quận Thanh Xuân, Hà Nội

Tách chuỗi theo dấu "," để phân ra các cột.

```
adr_test = housing_df.Address.str.split(', ').apply(lambda x: [x,x,x,x] if isinstance(x, float) else x).to_frame()
```

Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Address - District - Ward

Tách chuỗi theo dấu "," để phân ra các cột.

```
adr_test = housing_df.Address.str.split(', ').apply(lambda x: [x,x,x,x] if isinstance(x, float) else x).to_frame()
```

	Address	len
0	[Đường Hoàng Quốc Việt, Phường Nghĩa Đô, Quận Cầu Giấy, Hà Nội]	4
1	[Đường Kim Giang, Phường Kim Giang, Quận Thanh Xuân, Hà Nội]	4
2	[phố minh khai, Phường Minh Khai, Quận Hai Bà Trưng, Hà Nội]	4
3	[Đường Võng Thị, Phường Thụy Khuê, Quận Tây Hồ, Hà Nội]	4
4	[Đường Kim Giang, Phường Kim Giang, Quận Thanh Xuân, Hà Nội]	4

Dự định sẽ phân thành 4 cột là **Road**, **District**, **Ward**, **City** để tùy biến cho các quá trình sau. Tuy nhiên, không phải chỉ có các giá trị giống như trên. Tồn tại list **Address** có độ dài nhiều hơn 4.

Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Address - District - Ward

	Address	len
29	[180/61/5, Đường Tây Mỗ, Phường Tây Mỗ, Quận Nam Từ Liêm, Hà Nội]	5
42	[Ngách 6A, Đường Tả Thanh Oai, Xã Tả Thanh Oai, Huyện Thanh Trì, Hà Nội]	5
44	[4, Đường Kim Giang, Phường Đại Kim, Quận Hoàng Mai, Hà Nội]	5
50	[88, Đường Phạm Văn Đồng, Phường Cổ Nhuế 1, Quận Bắc Từ Liêm, Hà Nội]	5
82	[1, Đường Nguyễn Khánh Toàn, Phường Quan Hoa, Quận Cầu Giấy, Hà Nội]	5
...
81530	[285, Đường Đội Cấn, Phường Đội Cấn, Quận Ba Đình, Hà Nội]	5
81535	[25, 25 Đường Vũ Ngọc Phan, Phường Láng Hạ, Quận Đống Đa, Hà Nội]	5
81586	[Đường Lê Trọng Tấn Hoài Đức, Hà Nội, Xã Vân Canh, Huyện Hoài Đức, Hà Nội]	5
81607	[Đường Lạc Long Quân Tây Hồ, Hà Nội, Phường Phú Thượng, Quận Tây Hồ, Hà Nội]	5
81621	[Số 7 Đường Đại lộ Thăng Long Nam Từ Liêm, Hà Nội, Phường Mễ Trì, Quận Nam Từ Liêm, Hà Nội]	5

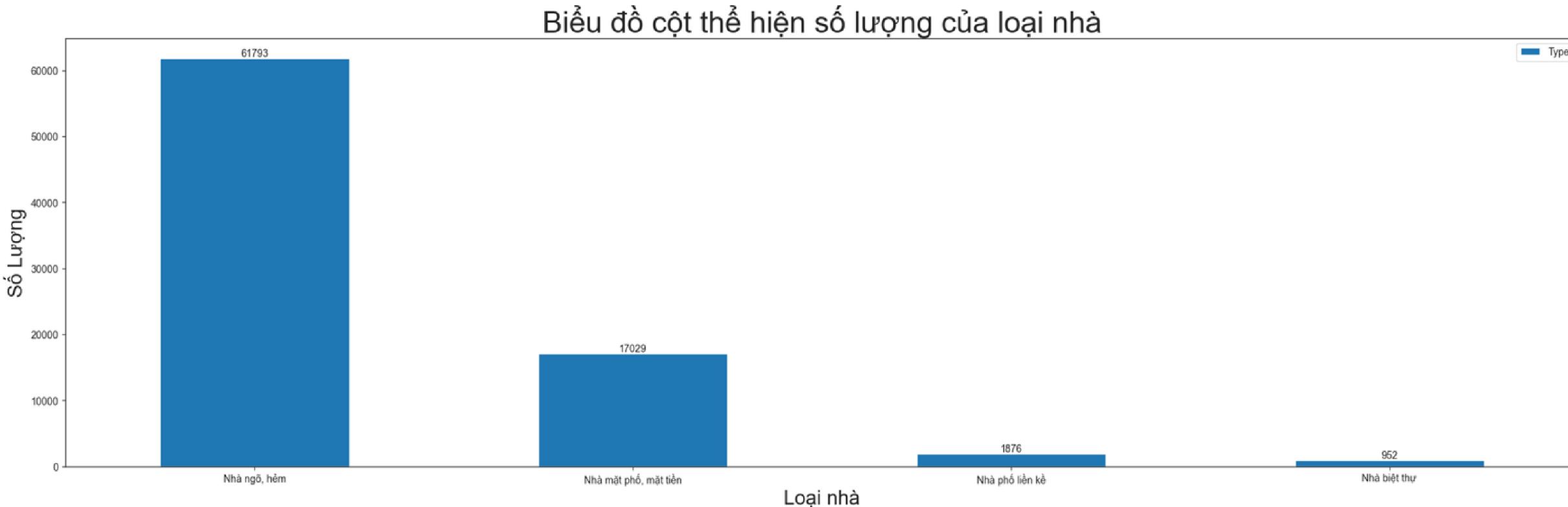
Vì thế nếu câu hỏi có liên quan đến các giá trị này, sẽ được thực hiện tiền xử lí bên trong phần **Đặt Câu Hỏi**. Ta chỉ khám phá và biết được các thông tin như thế trong cột này.

Tiền xử lí và Khám phá dữ liệu

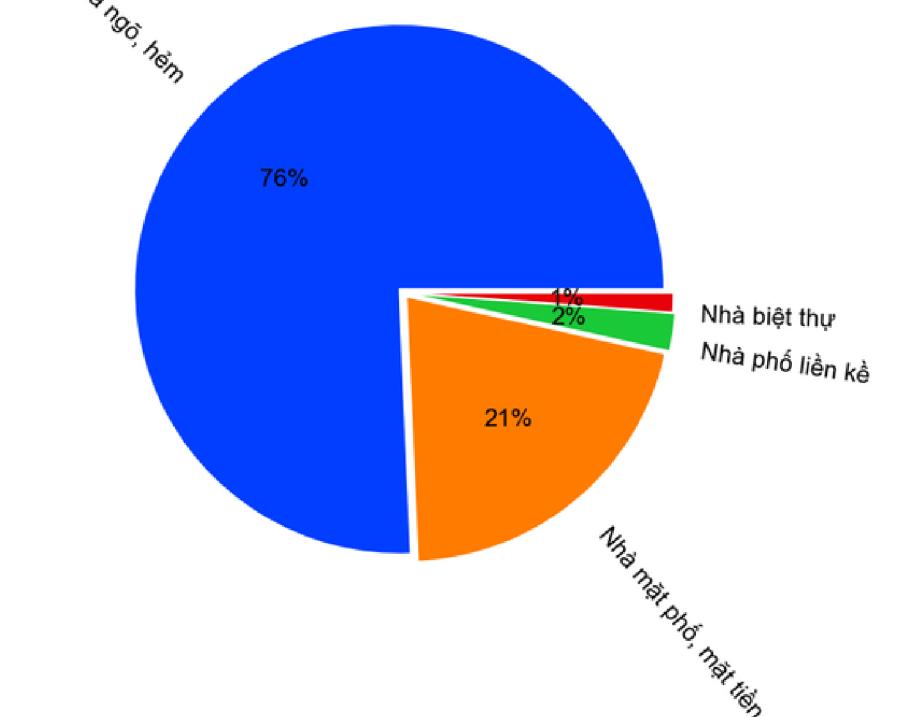
Khám phá từng cột dữ liệu

- Type

Cột này có 4 giá trị, và 1 giá trị NaN. Ở cột này có vẻ các dữ liệu đều ổn và không có dữ liệu gây nhiễu nên không cần xử lí. Và biểu đồ bên thể hiện số lượng của từng giá trị.



Biểu đồ thể hiện tỉ lệ số lượng của loại nhà

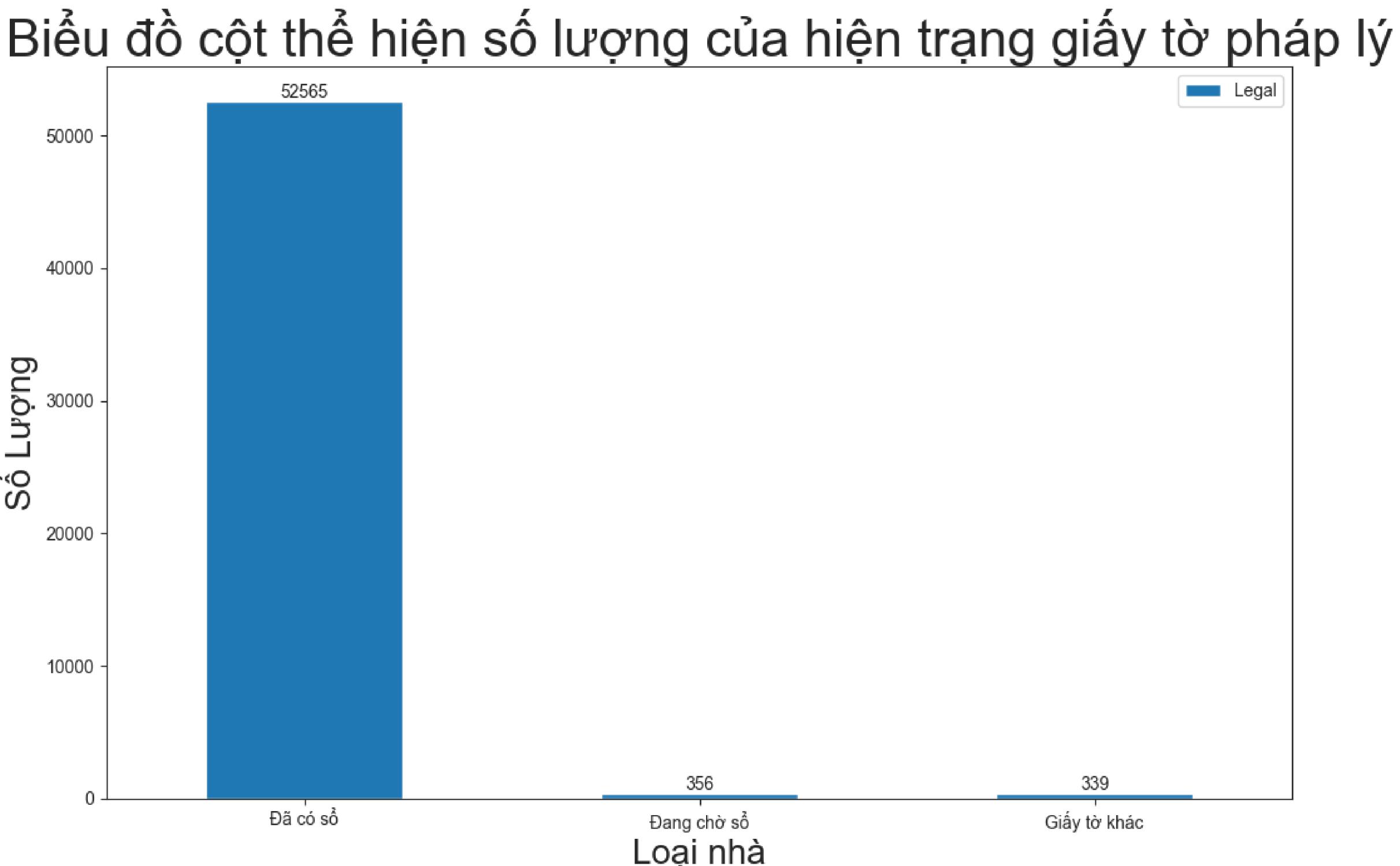


Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Legal

Cột này có 3 giá trị, và 1 giá trị NaN. Ở cột này có vẻ các dữ liệu đều ổn và không có dữ liệu gây nhiễu nên không cần xử lí. Và biểu đồ bên thể hiện số lượng của từng giá trị.



Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Number of bedrooms

```
array(['5 phòng', '3 phòng', '4 phòng', '6 phòng', 'nhiều hơn 10 phòng',
       '8 phòng', '2 phòng', '7 phòng', '9 phòng', '1 phòng', '10 phòng',
       nan], dtype=object)
```

Các giá trị của cột này dưới dạng <một số> + "phòng"

Mục tiêu:

- Loại bỏ chữ phòng ra khỏi các giá trị.
- Ta chuyển cột này trở thành cột numeric.
- Các giá trị "nhiều hơn 10 phòng" sẽ bị bỏ đi và thay bằng giá trị 11.

Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Number of bedrooms

```
def change_number_bedrooms(value):  
    if isinstance(value, str):  
        first = value.split(" ")[0]  
        if first.isnumeric():  
            return int(first)  
        else:  
            return 11  
    else:  
        return value
```

Sau khi triển khai hàm, đưa tất cả giá trị trong cột này vào bên trong hàm

```
housing_df["Number of bedrooms"] = housing_df["Number of bedrooms"].apply(change_number_bedrooms)
```

Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Number of floors

Về cơ bản, cột này cũng giống như cột **Number of bedrooms**. Tuy nhiên, có một số giá trị khó hiểu.

```
array(['4', 'nan', '6', '5', '7', '2', '3', '8', '1', '9', '50', '38', '35',  
       '10', 'Nhiều hơn 10', '45', '33', '42', '25', '73', '12', '65',  
       '55', '30', '14', '40', '52', '54', '32', '29'], dtype=object)
```

Có một số giá trị gây nhiễu. Tại sao có giá trị **Nhiều hơn 10** rồi nhưng vẫn có các giá trị lớn hơn 10. Ví dụ như các giá trị **45, 33, 50, 38, ...**

Vì tác giả không cho biết cách thu thập các dataset như thế nào, nên không thể loại bỏ các giá trị này. Vì có thể ảnh hưởng đến mô hình học máy.

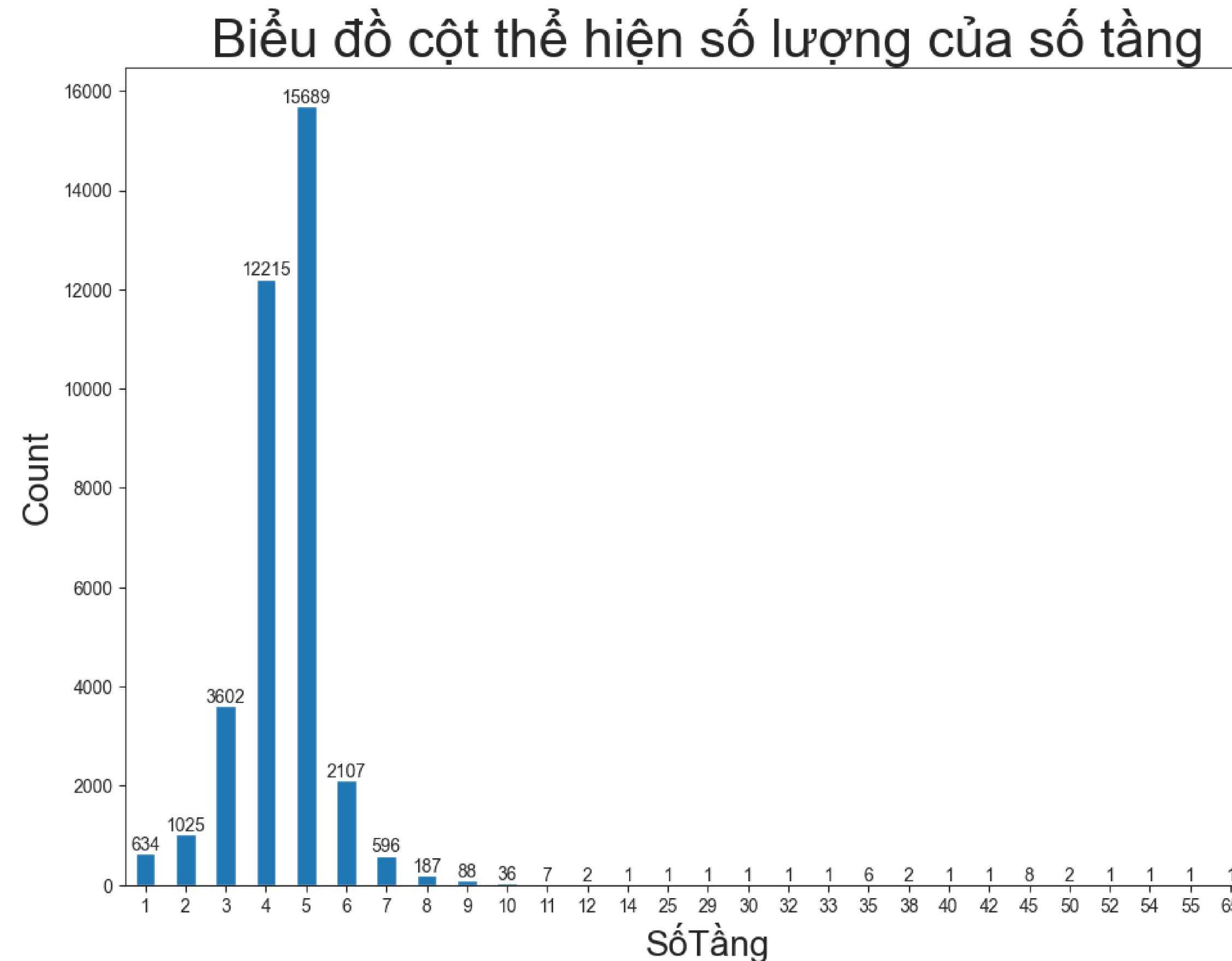
Vì giá trị **Nhiều hơn 10** và kế đến là giá trị thứ 12. Vậy nên sẽ chuyển giá trị này thành giá trị 11.

Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Number of floors

- Các cột nhỏ hơn 10 chiếm đa số trong DataFrame.
- Các giá trị nhỏ hơn 10 không có quá nhiều. Hầu như rất nhỏ so với tổng số dòng.



Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Length - Width - Area

2	10 m
3	12.75 m
4	9 m
5	12.1 m
8	12 m
...	
81540	13 m
81541	16 m
81544	14 m
81546	20 m
81547	20 m

Định dạng của Length và Width

0	46 m ²
1	37 m ²
2	40 m ²
3	51 m ²
4	36 m ²
...	
81676	38 m ²
81677	50 m ²
81678	41 m ²
81679	60 m ²
81680	45 m ²

Định dạng của Area

Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

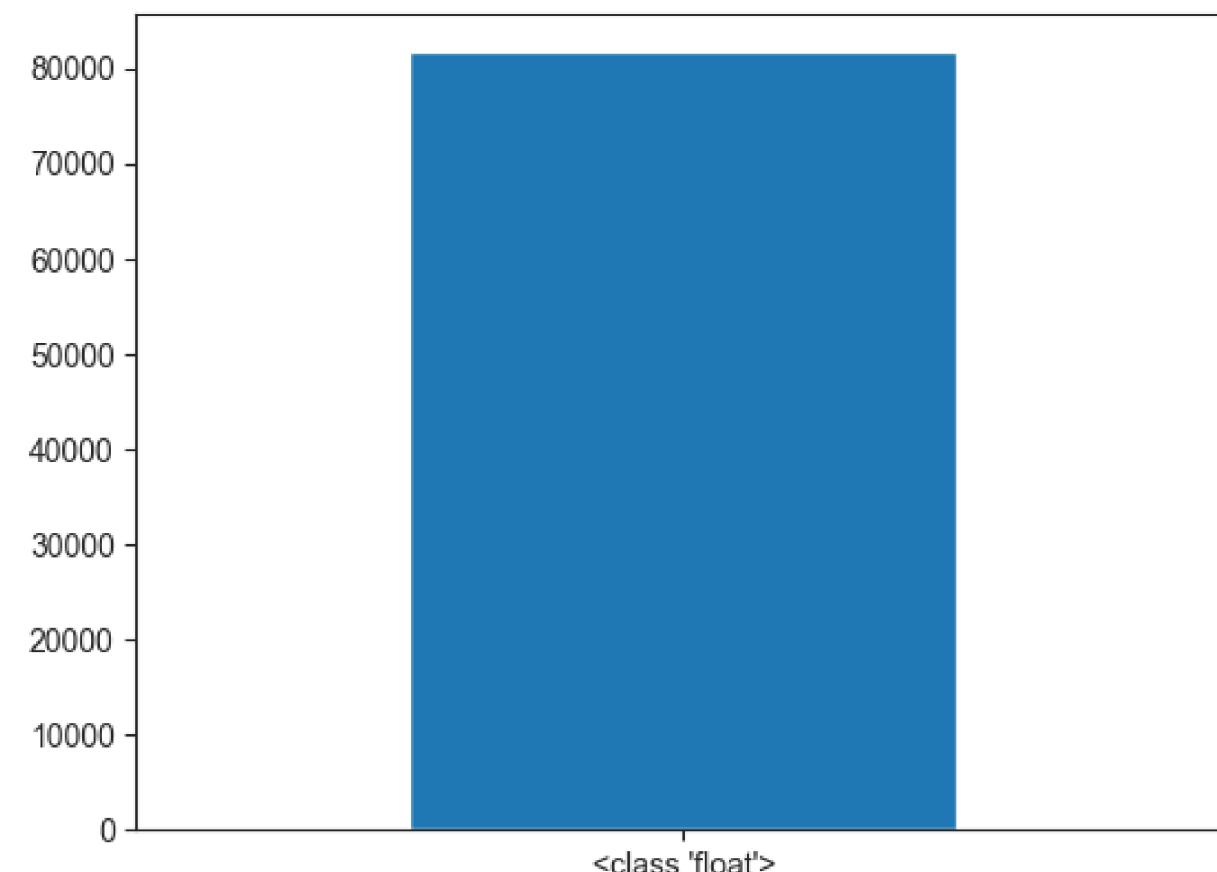
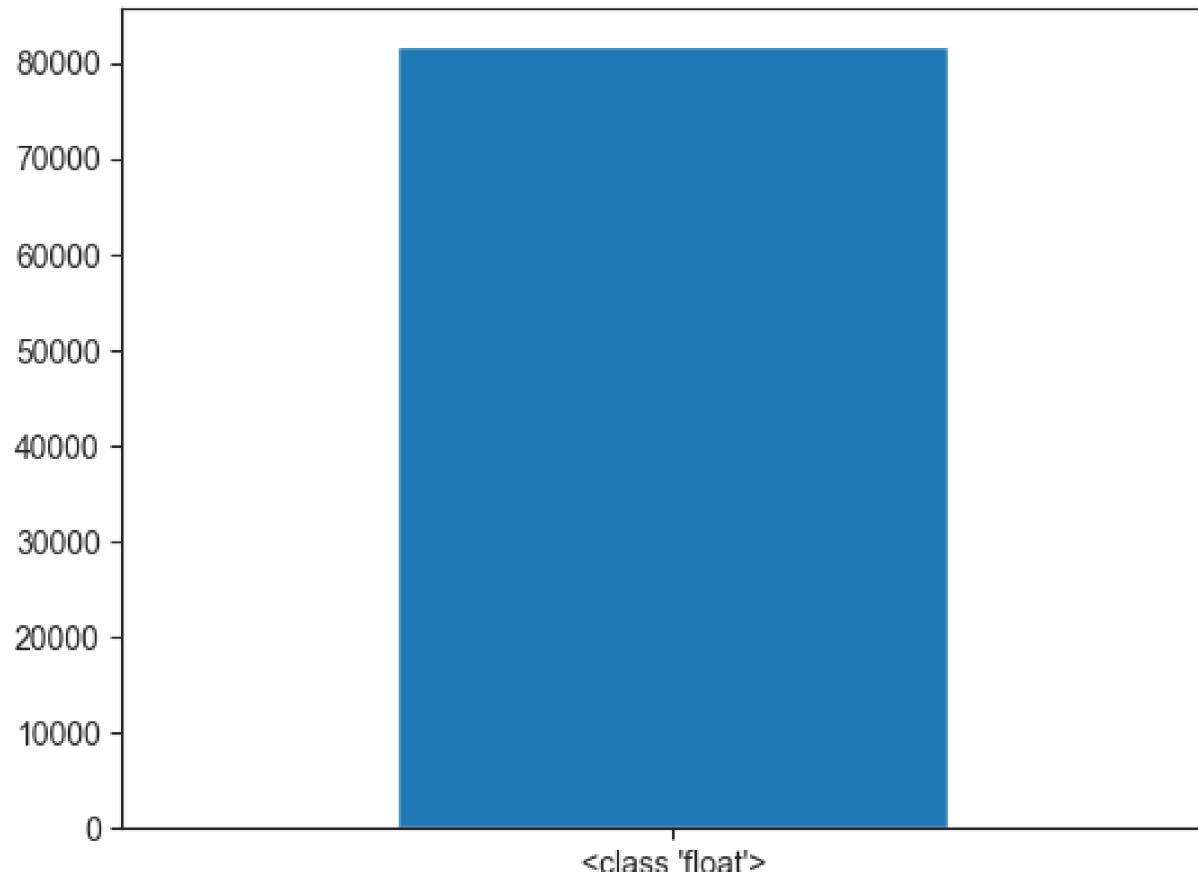
- Length - Width - Area

Mục tiêu

- Ta loại bỏ qua kí tự "m" trong cột Length và "m²" trong Area và chuyển các kí tự số sang kiểu dữ liệu numeric.

```
housing_df['Length'] = housing_df['Length'].str.replace('m', '').astype(float)
```

Cả hai cột **Width** và **Length**
đều đã được chuyển sang kiểu
float.



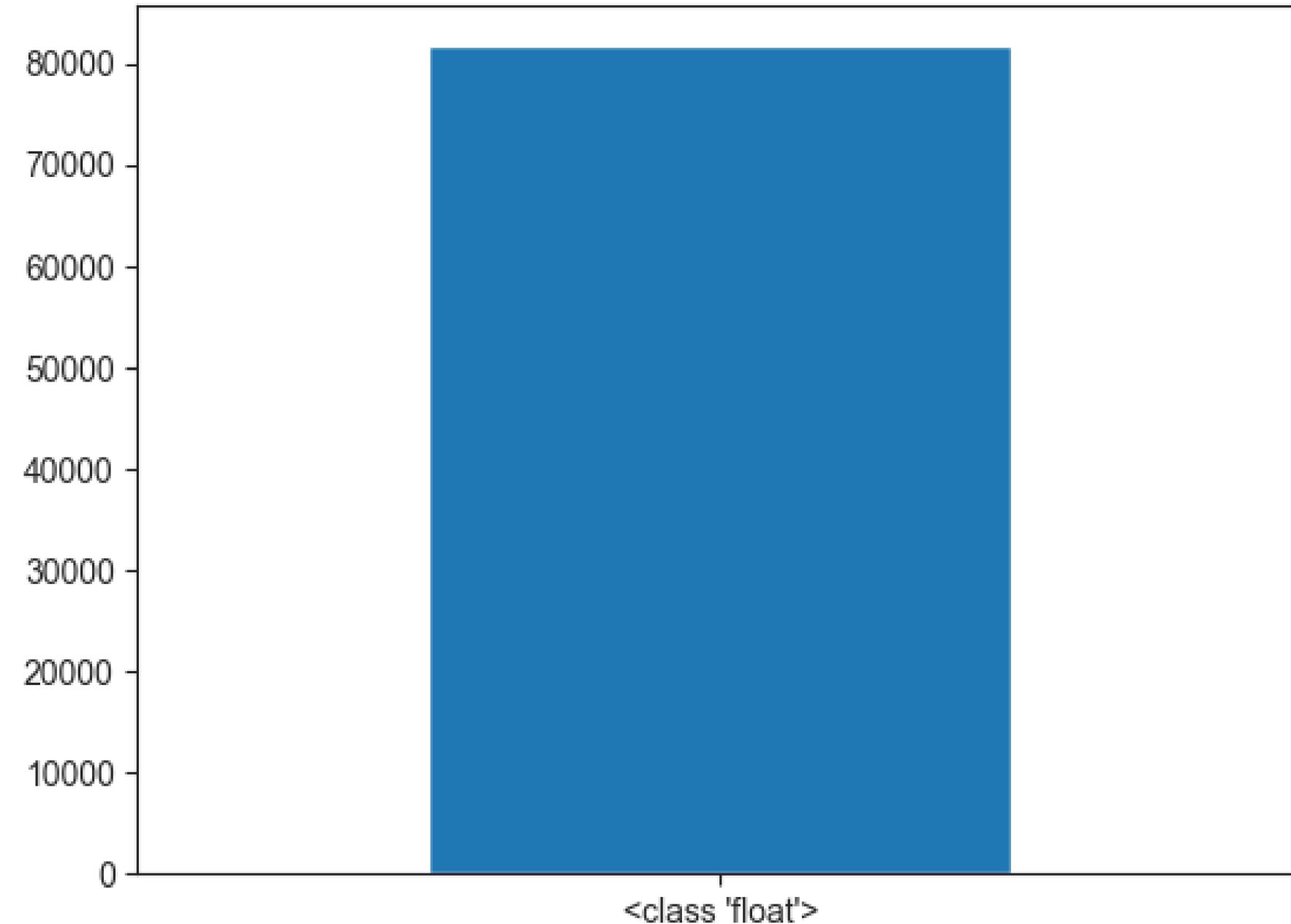
Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Length - Width - Area

```
housing_df['Area'] = housing_df['Area'].str.replace('m2', '').astype(float)
```

Các giá trị cột **Area** đều được
chuyển sang kiểu dữ liệu float



Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Price/m2

Các yếu tố và lỗi sai trong cột **Price/m2**

- Các giá trị mang đơn vị tính là **tiền tệ/m2**.
- Không giống như cột **Width** và **Length**. Sử dụng regex thì biết được rằng cột này gồm có 3 giá trị cách tính phí khác nhau.

```
unit = housing_df["Price/m2"].str.extract('[\d.,]+\\s([\w/]+)').value_counts()
```

Tiền xử lí và Khám phá dữ liệu

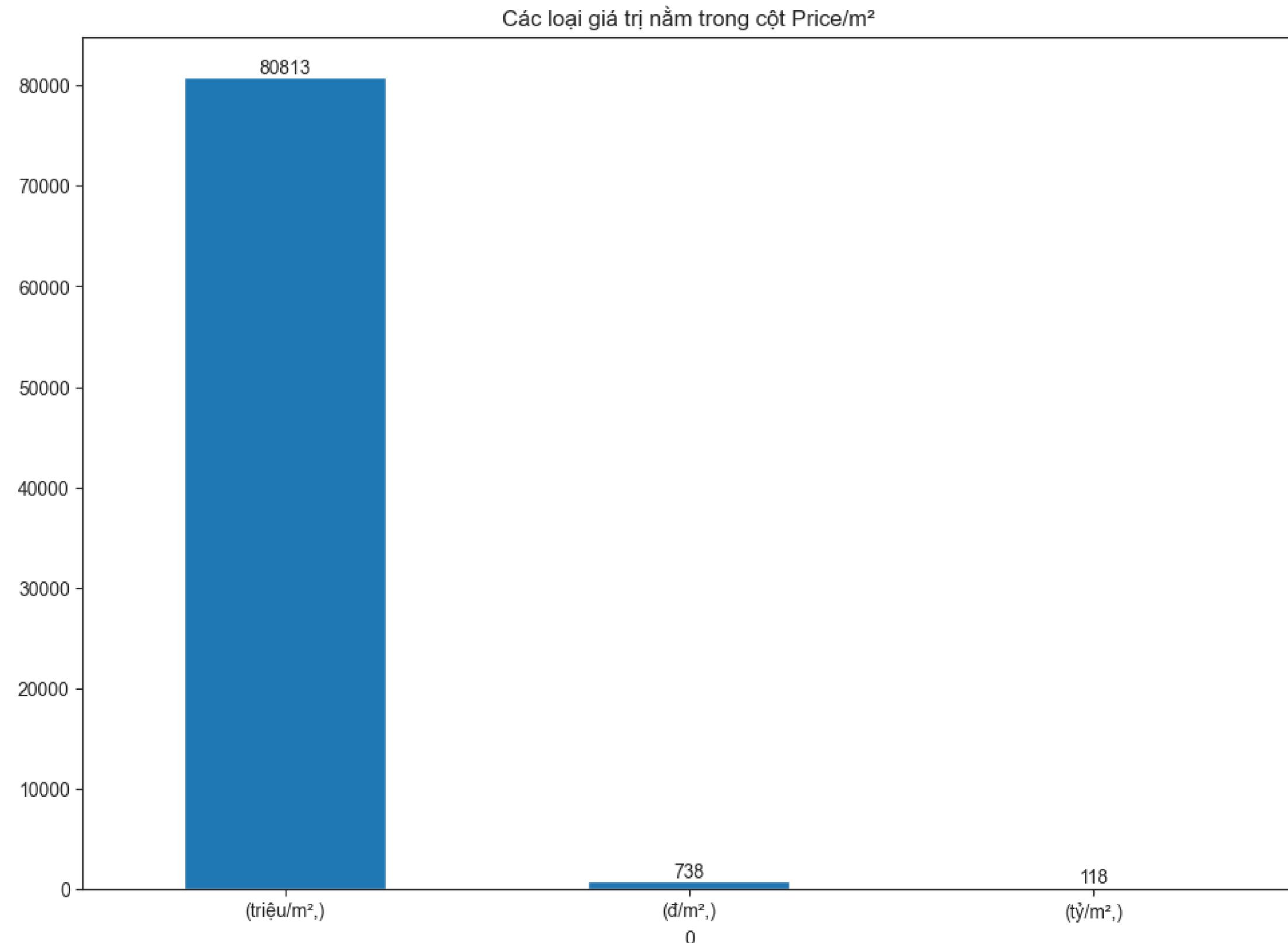
Khám phá từng cột dữ liệu

- Price/m²

Có 3 đơn vị tính tiền:

- đ/m²
- triệu/m²
- tỷ/m²

Chiếm đa số của tập dữ liệu là
đơn vị **triệu/m²**



Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Price/m²

Kiểu dữ liệu hiện tại của cột này là **string**. Vậy nên rất khó khăn trong việc tính toán, trực quan hóa ở phần **Đặt Câu Hỏi, và mô hình**. Thế nên cần phải chuyển tất cả các đơn vị này về một định dạng thống nhất.

Tuy nhiên, trước đó, vẫn còn một số giá trị bị sai định dạng để chuyển thành dạng số.

'728.000,00728 tỷ/m²',

Khác với các giá trị khác, chỉ có dấu "," để đánh dấu phần thập phân. Có một số giá trị vừa có dấu"." và dấu ",".

- Dấu "." sẽ phân cách trong lớp đơn vị.
- Dấu "," sẽ phân cách phần nguyên và phần thập phân.

Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Price/m2

Ta lần lượt bỏ đi các dấu "." và thay dấu "," thành dấu "."

```
housing_df["Price/m2"] = housing_df["Price/m2"].str.replace('.', '').str.replace(',', '.')
```

Sau đó ta chuyển các giá trị này về dạng số thực.

```
def change_currency(val):  
    if isinstance(val, str):  
        if len(val.split()) > 1:  
            return val[0]  
    return val
```

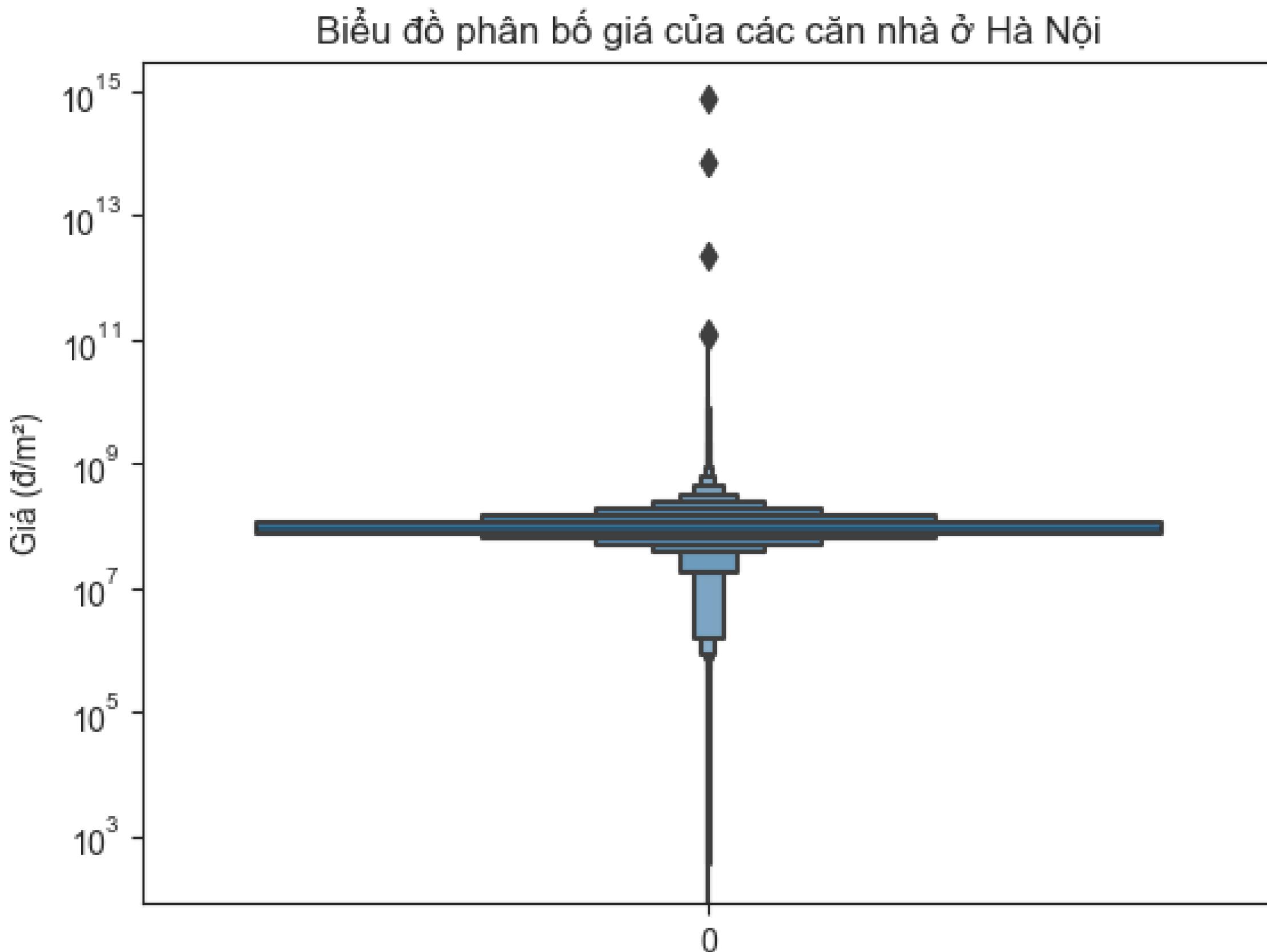
```
housing_df['Price/m2'] = housing_df['Price/m2'].replace(  
    {'đ/m2': '*1e1', 'triệu/m2': '*1e6', 'tỷ/m2': '*1e9', np.nan: 0.0},  
    regex=True).map(pd.eval)
```

Tiền xử lí và Khám phá dữ liệu

Khám phá từng cột dữ liệu

- Price/m²

Các giá trị tập trung về khoảng 10^8 đ/m² và có các khoảng outlier nằm ở các trị giá 10^{15}



Tiền xử lí và Khám phá dữ liệu

Thông tin các cột

Tên cột	Kiểu dữ liệu	Mô tả
Date	datetime	Ngày định giá bán
Address	string	Địa chỉ căn nhà
District	string	Quận
Ward	string	Huyện
Type	string	Kiểu nhà ở
Legal	string	Giấy tờ pháp lý của nhà ở
Number of floors	float	Số tầng
Number of bedrooms	float	Số phòng ngủ
Area	float	Diện tích căn nhà
Length	float	Chiều dài căn nhà
Width	float	Chiều rộng căn nhà
Price/m2	string	Số tiền mua căn nhà



Đặt câu hỏi

Đặt câu hỏi

Các câu hỏi được đặt ra:

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Câu 2: Xu hướng các căn nhà được định giá ở các thời điểm ?

Câu 3: Vị trí của các căn nhà có tầm ảnh hưởng như thế nào với giá cả?

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Trả lời câu hỏi thành công: Ta sẽ biết được chiều dài hay chiều rộng, cái nào mới là nhân tố quyết định giá cả của 1 căn nhà. Từ đó, có thể lựa chọn căn nhà có giá trị hợp lý.

Khó khăn: Dữ liệu còn rất nhiều outliers, cần phải xử lý nhiều !!!

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Các bước thực hiện câu hỏi:

Bước 1: Lấy ra các cột cần sử dụng cho câu hỏi

Bước 2: Xử lý các dữ liệu liệu NaN

Bước 3: Scale lại dữ liệu và thêm các cột mới

Bước 4: Trực quan hóa dữ liệu

Bước 5: Ta sẽ lọc dữ liệu lại và tiếp tục scale độ dài lại như sau

Bước 6: Trực quan hóa dữ liệu

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Bước 1: Lấy ra các cột cần sử dụng cho câu hỏi

	Area	Length	Width	Price/m2
0	46.0	NaN	NaN	86960000.0
1	37.0	NaN	NaN	116220000.0
2	40.0	10.00	4.0	65000000.0
3	51.0	12.75	4.0	100000000.0
4	36.0	9.00	4.0	86110000.0
...
81676	38.0	NaN	NaN	81580000.0
81677	50.0	NaN	NaN	292000000.0
81678	41.0	NaN	NaN	341460000.0
81679	60.0	NaN	NaN	101670000.0
81680	45.0	NaN	NaN	102220000.0

81681 rows × 4 columns

- Ta sẽ lấy ra các cột '**Area**', '**Length**', '**Width**', '**Price/m2**'.
- Dữ liệu có 81681 dòng

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Bước 1: Lấy ra các cột cần sử dụng cho câu hỏi

	Area	Length	Width	billion/m2
0	46.0	NaN	NaN	0.08696
1	37.0	NaN	NaN	0.11622
2	40.0	10.00	4.0	0.06500
3	51.0	12.75	4.0	0.10000
4	36.0	9.00	4.0	0.08611
...
81676	38.0	NaN	NaN	0.08158
81677	50.0	NaN	NaN	0.29200
81678	41.0	NaN	NaN	0.34146
81679	60.0	NaN	NaN	0.10167
81680	45.0	NaN	NaN	0.10222

81681 rows x 4 columns

- Nhận xét: cột '**Price/m2**' có giá trị là đồng/m² => ta sẽ scale lại thành '**billion/m2**' có giá trị là triệu/m²

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Bước 2: Xử lý các dữ liệu liệu NaN

	Area	Length	Width	billion/m2
2	40.0	10.00	4.0	0.06500
3	51.0	12.75	4.0	0.10000
4	36.0	9.00	4.0	0.08611
5	46.0	12.10	3.8	0.10435
8	75.0	12.00	6.5	0.12000
...
81540	37.0	13.00	3.0	0.04324
81541	100.0	16.00	6.0	0.11500
81544	80.0	14.00	5.0	0.26250
81546	89.0	20.00	4.0	0.25843
81547	90.0	20.00	4.0	0.06444

19627 rows x 4 columns

- Xử lý các giá trị NaN ở các cột 'Length', 'Width' bằng cách loại bỏ chúng để tiện do phân tích dữ liệu.
- dữ liệu bây giờ còn 19627 cột

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Bước 3: Scale lại dữ liệu và thêm các cột mới

Dữ liệu có đa số chiều dài chiều rộng của căn nhà, ta sẽ scale lại các độ dài đó tương ứng như sau: (được thực hiện với def scaling_range(x))

- (1000,): lớn hơn hoặc bằng 1000m
- (900, 1000): [900,1000)
- (800, 900): [800,900)
- ...
- (0,100): (0,100)
- (0,0): bằng 0

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Bước 3: Scale lại dữ liệu và thêm các cột mới

```
def scaling_range(x):  
    if x > 1000.0:  
        return (1000,)  
    elif x == (0.0):  
        return (0,0)  
    else:  
        return (x // 100 * 100, x // 100 * 100 + 100)
```

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Bước 3: Scale lại dữ liệu và thêm các cột mới

	Area	Length	Width	billion/m2	scaling_width	scaling_length
2	40.0	10.00	4.0	0.06500	(0.0, 100.0)	(0.0, 100.0)
3	51.0	12.75	4.0	0.10000	(0.0, 100.0)	(0.0, 100.0)
4	36.0	9.00	4.0	0.08611	(0.0, 100.0)	(0.0, 100.0)
5	46.0	12.10	3.8	0.10435	(0.0, 100.0)	(0.0, 100.0)
8	75.0	12.00	6.5	0.12000	(0.0, 100.0)	(0.0, 100.0)
...
81540	37.0	13.00	3.0	0.04324	(0.0, 100.0)	(0.0, 100.0)
81541	100.0	16.00	6.0	0.11500	(0.0, 100.0)	(0.0, 100.0)
81544	80.0	14.00	5.0	0.26250	(0.0, 100.0)	(0.0, 100.0)
81546	89.0	20.00	4.0	0.25843	(0.0, 100.0)	(0.0, 100.0)
81547	90.0	20.00	4.0	0.06444	(0.0, 100.0)	(0.0, 100.0)

19627 rows × 6 columns

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

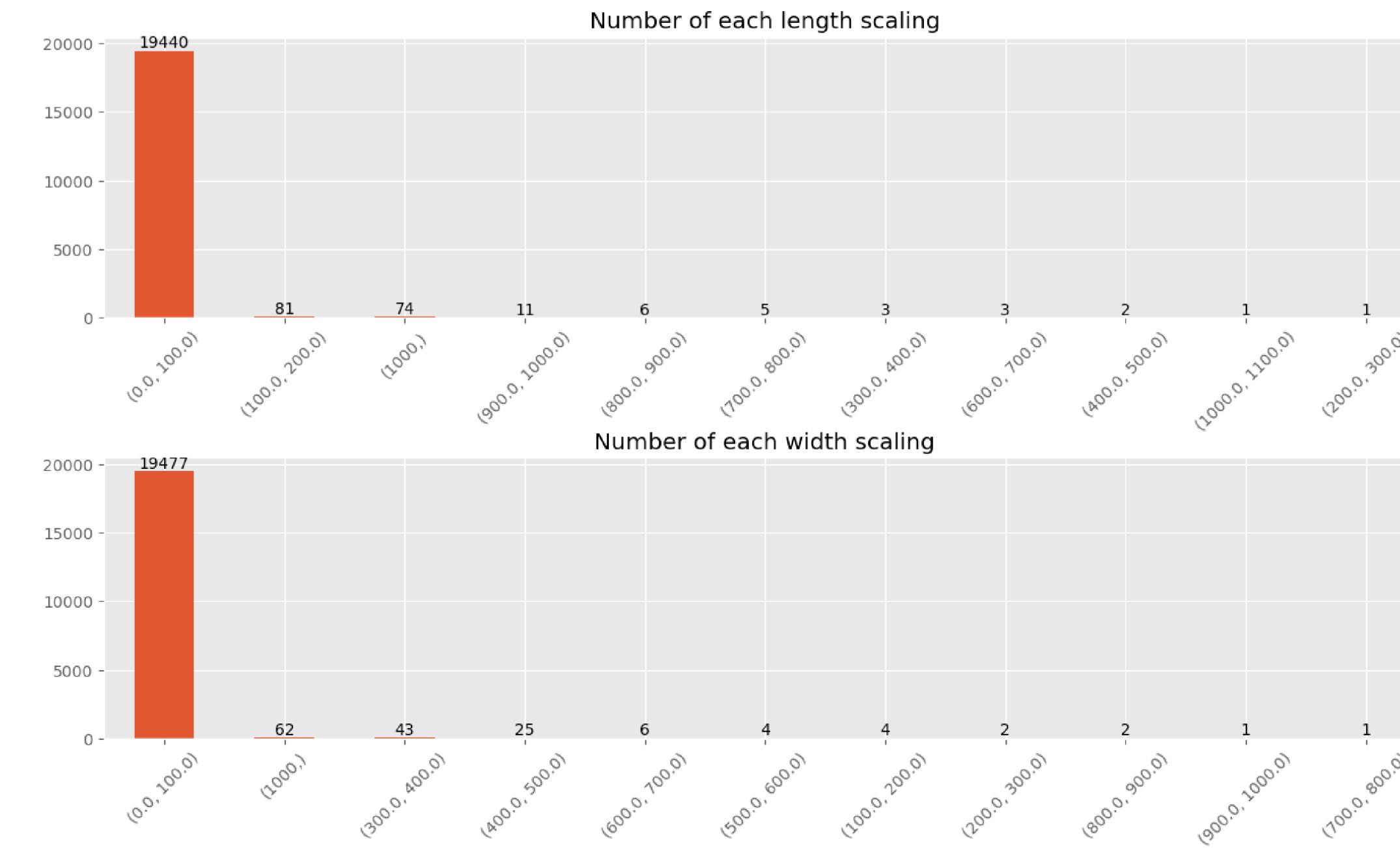
Bước 4: Trực quan hóa dữ liệu

```
length_df = ques01_df['scaling_length'].value_counts()  
width_df = ques01_df['scaling_width'].value_counts()  
plt.style.use('ggplot')  
fig, axes = plt.subplots(nrows = 2, ncols = 1, figsize=(15, 8))  
length_df.plot.bar(ax = axes[0], rot = 45)  
axes[0].set_title('Number of each length scaling')  
plt.subplots_adjust(hspace = 0.5)  
width_df.plot.bar(ax = axes[1], rot = 45)  
axes[1].set_title('Number of each width scaling')  
for i in axes[0].containers:  
    axes[0].bar_label(i,)  
for i in axes[1].containers:  
    axes[1].bar_label(i,)  
plt.show()
```

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Bước 4: Trực quan hóa dữ liệu



Ta thấy có vẻ như chiều dài và chiều rộng chiếm đa số vào từ (0,100). Ta sẽ chú ý phân tích vào các độ dài này để có dự đoán chính xác hơn !

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Bước 5: Ta sẽ lọc dữ liệu lại và tiếp tục scale độ dài lại như sau

- (90, 100): [90,100)
- (80, 90): [80,90)
- ...
- (0,10): (0,10)
- (0,0): bằng 0

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Bước 5: Ta sẽ lọc dữ liệu lại và tiếp tục scale độ dài lại như sau

```
def scaling_range(x):  
    if x > 1000.0:  
        return (1000,)  
    elif x == (0.0):  
        return (0,0)  
    else:  
        return (x // 100 * 100, x // 100 * 100 + 100)
```

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

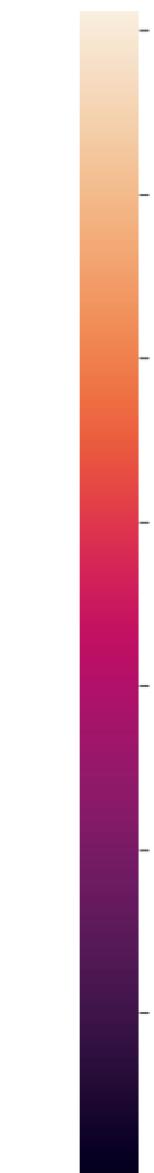
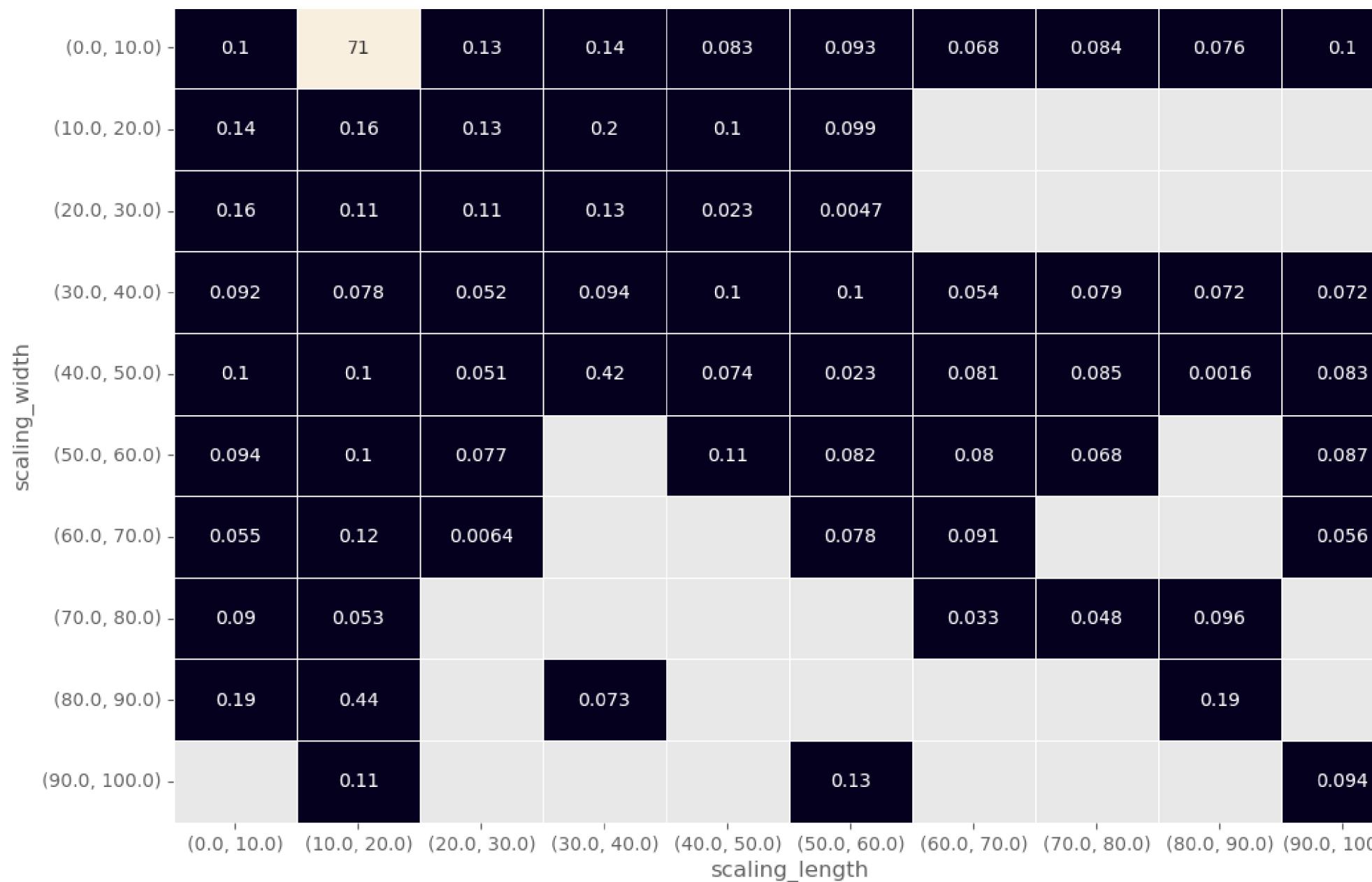
Bước 6: Trực quan hóa dữ liệu

```
heatmap = ques01_df.sort_values(by=['scaling_width', 'scaling_length'])
heatmap = ques01_df.groupby(['scaling_width', 'scaling_length'])['billion/m2'].mean()
heatmap = heatmap.unstack(level= 1)
plt.style.use('ggplot')
fig, ax = plt.subplots(figsize=(15, 8))
ax = sns.heatmap(heatmap, linewidths= 0.5, annot= True)
```

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Bước 6: Trực quan hóa dữ liệu

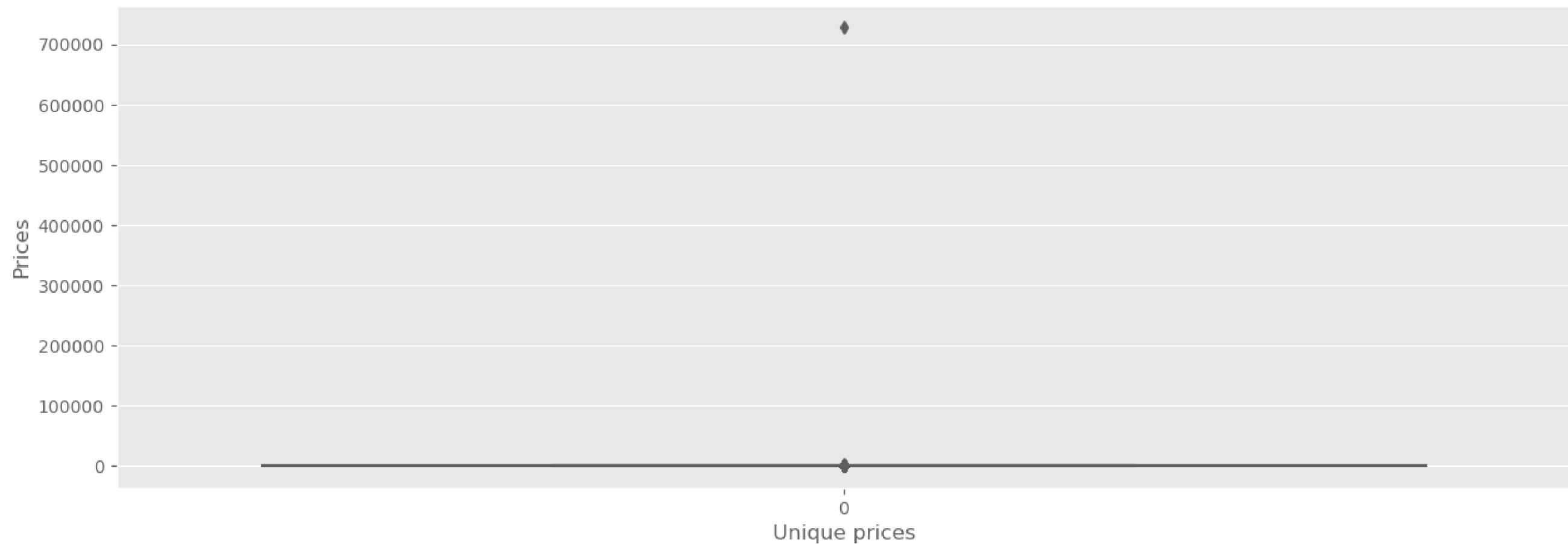


- Nhìn qua đồ thị trên, đặc biệt ở căn nhà có chiều rộng là `(0,10)` và chiều dài là `(10,20)` có trung bình tận **71 tỷ/m²**.
- Có thể dự đoán trong này có outlier, chúng ta sẽ làm rõ hơn.

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Bước 6: Trực quan hóa dữ liệu



Như dự đoán ta lại có giá trị ngoại lai (700,000 tỷ hơn cho mỗi m²). Ta sẽ loại đi giá trị ngoại lai này bằng cách giới hạn lại giá tiền chỉ rơi vào 100000 tỷ/m² và trực quan hóa lại biểu đồ

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Bước 6: Trực quan hóa dữ liệu

```
ques01_df = ques01_df.loc[ques01_df['billion/m2'] <= 100000.0]
```

```
heatmap = ques01_df.sort_values(by=['scaling_width', 'scaling_length'])
```

```
heatmap = ques01_df.groupby(['scaling_width', 'scaling_length'])['billion/m2'].mean()
```

```
heatmap = heatmap.unstack(level= 1)
```

```
plt.style.use('ggplot')
```

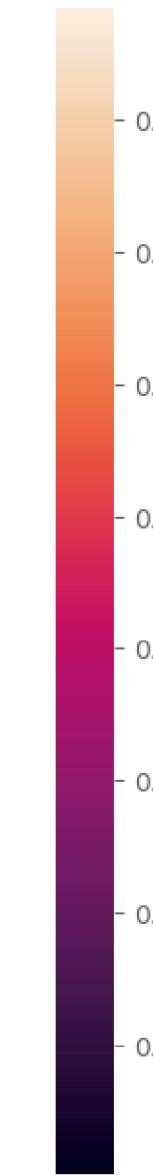
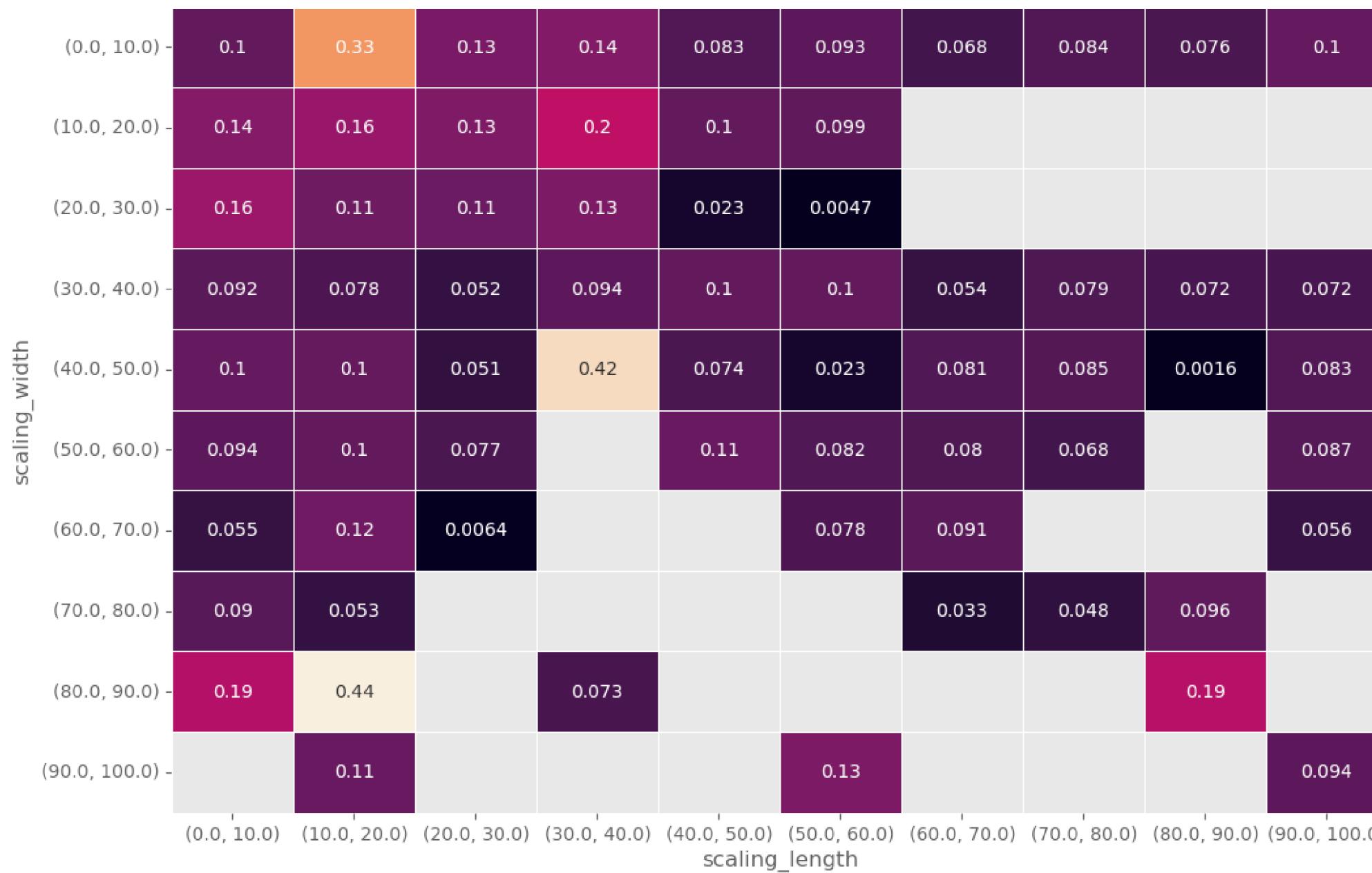
```
fig, ax = plt.subplots(figsize=(15, 8))
```

```
ax = sns.heatmap(heatmap, linewidths= 0.5, annot= True)
```

Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Bước 6: Trực quan hóa dữ liệu



Đặt câu hỏi

Câu 1: Có thể cùng 1 đơn vị diện tích mà giá cả nhà ở Hà Nội lại khác nhau do chiều dài hay chiều rộng?

Nhận xét:

- Có vẻ như với chiều rộng tầm `(0,10); (10,20); (40,50)` thì ta sẽ có đa dạng nhất về với chiều dài của nhà.
- Nhìn chung có vẻ như giá của các nhà có `chiều rộng (0,30)` với `chiều dài (0,40)` giá trung bình có vẻ mắc.
- Với các độ dài đặc biệt như nhận xét đầu thì giá trung bình có vẻ dịu đi.
- Căn có chiều rộng dài trong khoảng lần lượt như: $(0,10)(10,20)$; $(80,90)(10,20)$; $(40,50)(30,40)$ là có giá trung bình nổi trôi nhất.

Qua đây, chúng ta có thể nhận xét có thể do nhu cầu mua đất nhỏ để ở có vẻ nhiều và 3 loại căn nhà có độ dài $(0,10)(10,20)$; $(80,90)(10,20)$; $(40,50)(30,40)$ có thể vẫn là xu hướng trong tương lai

Đặt câu hỏi

Câu 2: Xu hướng các căn nhà được định giá ở các thời điểm ?

Trả lời câu hỏi thành công: Ta sẽ biết được xu hướng các căn nhà được đưa vào mục định giá/ bán qua thời gian sẽ những thế nào? Như số lượng tầng lầu của căn nhà hay số phòng của căn nhà.

Khó khăn: Dữ liệu còn chưa sạch, phải xử lý thêm để trực quan hóa.

Đặt câu hỏi

Câu 2: Xu hướng các căn nhà được định giá ở các thời điểm ?

Các bước thực hiện câu hỏi:

Bước 1: Lấy ra các cột cần sử dụng cho câu hỏi

Bước 2: Xử lý các dữ liệu mang giá trị là NaN và phân cột `Date` thành cột `Month` và `Year` vào dataframe

Bước 3: Trực quan hóa dữ liệu

Bước 4: Chuẩn bị lại dữ liệu và trực quan hóa dữ liệu

Đặt câu hỏi

Câu 2: Xu hướng các căn nhà được định giá ở các thời điểm ?

Bước 1: Lấy ra các cột cần sử dụng cho câu hỏi

	Date	Number of bedrooms	Number of floors
0	2020-08-05	5.0	4.0
1	2020-08-05	3.0	NaN
2	2020-08-05	4.0	4.0
3	2020-08-05	6.0	NaN
4	2020-08-05	4.0	NaN
...
81676	2019-08-23	3.0	NaN
81677	2019-08-07	3.0	NaN
81678	2019-08-07	4.0	NaN
81679	2019-08-05	4.0	NaN
81680	2019-08-05	4.0	NaN

81681 rows x 3 columns

- Ta sẽ lấy ra các cột 'Date', 'Number of bedrooms', 'Number of floors'.
- Dữ liệu có 81681 dòng

Đặt câu hỏi

Câu 2: Xu hướng các căn nhà được định giá ở các thời điểm ?

Bước 2: Xử lý các dữ liệu mang giá trị là `NaN` và phân cột `Date` thành cột `Month` và `Year` vào dataframe

	Date	Number of bedrooms	Number of floors	Year	Month
0	2020-08-05	5.0	4.0	2020	8
1	2020-08-05	3.0	0.0	2020	8
2	2020-08-05	4.0	4.0	2020	8
3	2020-08-05	6.0	0.0	2020	8
4	2020-08-05	4.0	0.0	2020	8
...
81676	2019-08-23	3.0	0.0	2019	8
81677	2019-08-07	3.0	0.0	2019	8
81678	2019-08-07	4.0	0.0	2019	8
81679	2019-08-05	4.0	0.0	2019	8
81680	2019-08-05	4.0	0.0	2019	8
81681 rows × 5 columns					

Đặt câu hỏi

Câu 2: Xu hướng các căn nhà được định giá ở các thời điểm ?

Bước 2: Xử lý các dữ liệu mang giá trị là NaN và phân cột `Date` thành cột `Month` và `Year` vào dataframe

Chúng ta sẽ kiểm tra xem số năm mà các căn nhà bắt đầu được định giá/ bán

Từ đó, ta sẽ lọc dataframe này thành các dataframe con tương ứng.

Số năm các căn nhà được định giá (đăng bán):

[2020, 2019]

Đặt câu hỏi

Câu 2: Xu hướng các căn nhà được định giá ở các thời điểm ?

Bước 3: Trực quan hóa dữ liệu

```
ques02_df_20 = ques02_df.loc[ques02_df['Year'] == 2020]
```

```
ques02_df_19 = ques02_df.loc[ques02_df['Year'] == 2019]
```

```
plt.style.use('ggplot')
```

```
fig, axes = plt.subplots(nrows = 2, ncols = 1, figsize=(15, 8))
```

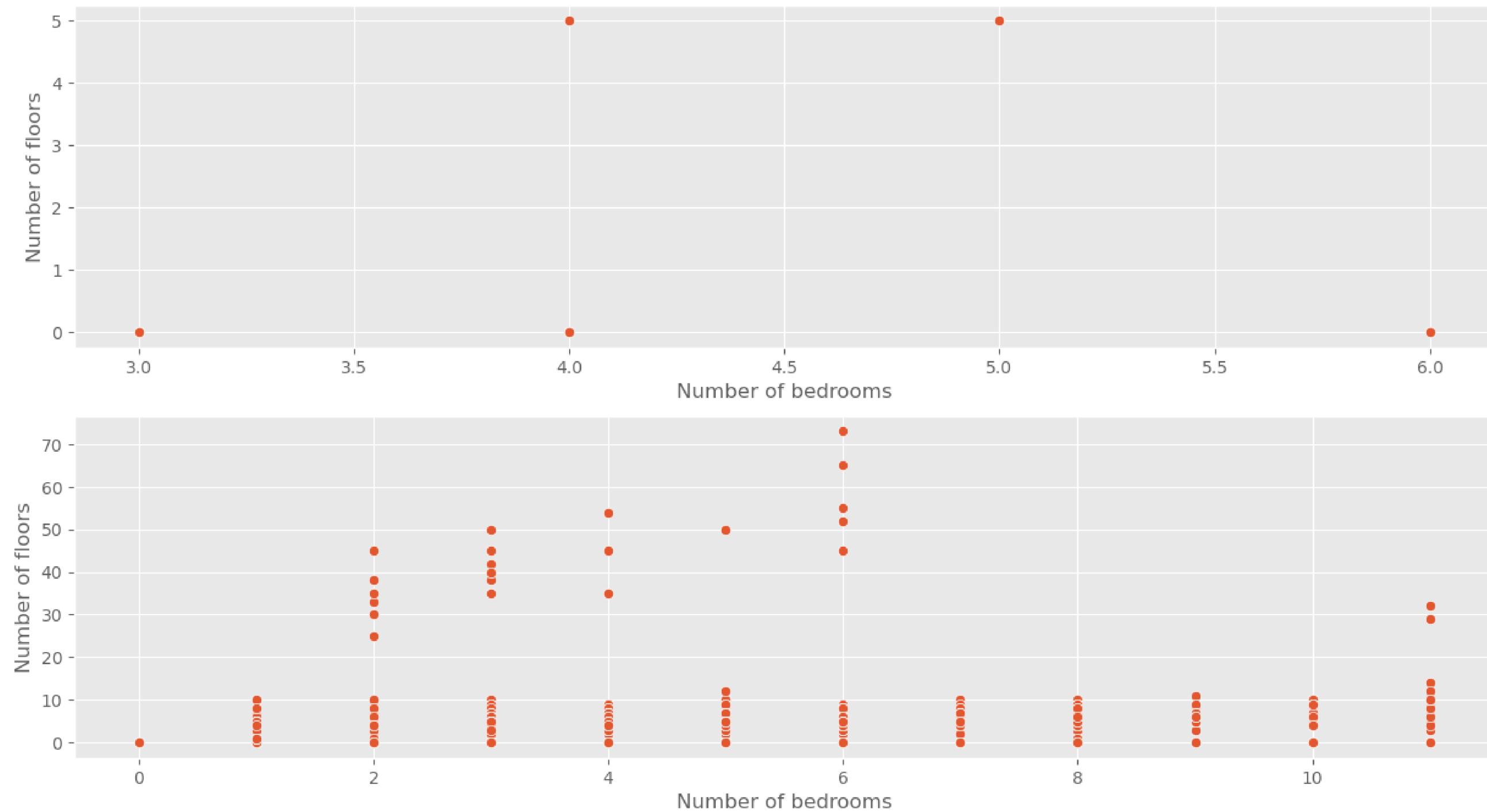
```
sns.scatterplot(data = ques02_df_19, x = 'Number of bedrooms', y = 'Number of floors', ax = axes[0])
```

```
sns.scatterplot(data = ques02_df_20, x = 'Number of bedrooms', y = 'Number of floors', ax = axes[1])
```

Đặt câu hỏi

Câu 2: Xu hướng các căn nhà được định giá ở các thời điểm ?

Bước 3: Trực quan hóa dữ liệu



Hmm chúng ta có thể thấy vào năm 2019 thì có rất ít căn nhà được định giá/ bán. Để dự đoán được chính xác hơn, chúng ta sẽ quan sát vào năm 2020, đặc biệt là vào các tháng.

Đặt câu hỏi

Câu 2: Xu hướng các căn nhà được định giá ở các thời điểm ?

Bước 4: Chuẩn bị lại dữ liệu và trực quan hóa dữ liệu

```
ques02_df_20_floor = ques02_df_20.groupby(['Month'])['Number of floors'].value_counts(sort = True)
ques02_df_20_floor = ques02_df_20_floor.unstack(level = 1).sort_index(ascending = True).transpose()
ques02_df_20_bedrooms = ques02_df_20.groupby(['Month'])['Number of bedrooms'].value_counts(sort = True)
ques02_df_20_bedrooms = ques02_df_20_bedrooms.unstack(level = 1).sort_index(ascending = True).transpose()
```

```
plt.style.use('ggplot')
fig, axes = plt.subplots(nrows = 2, ncols = 1, figsize=(30, 15))
```

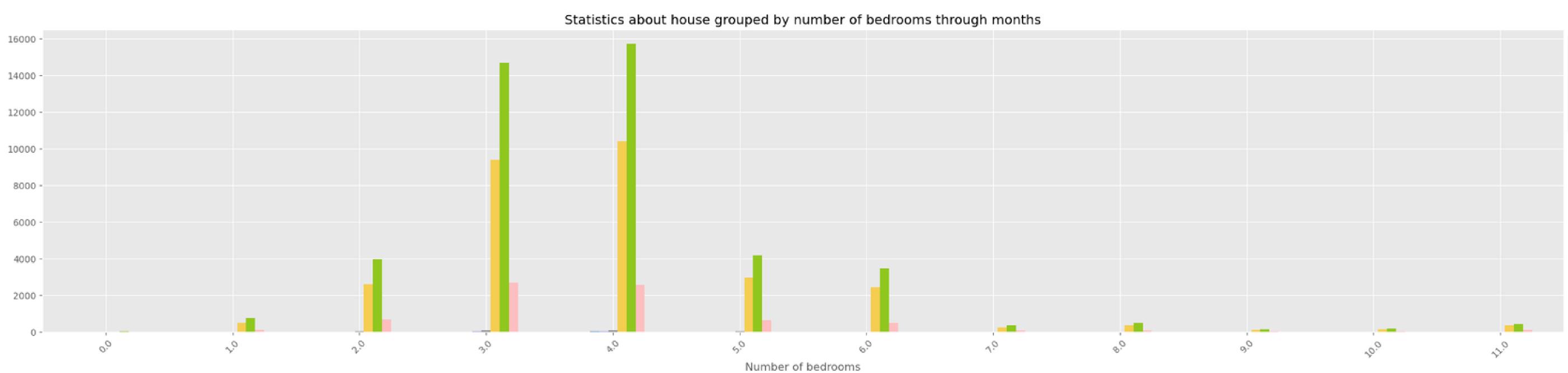
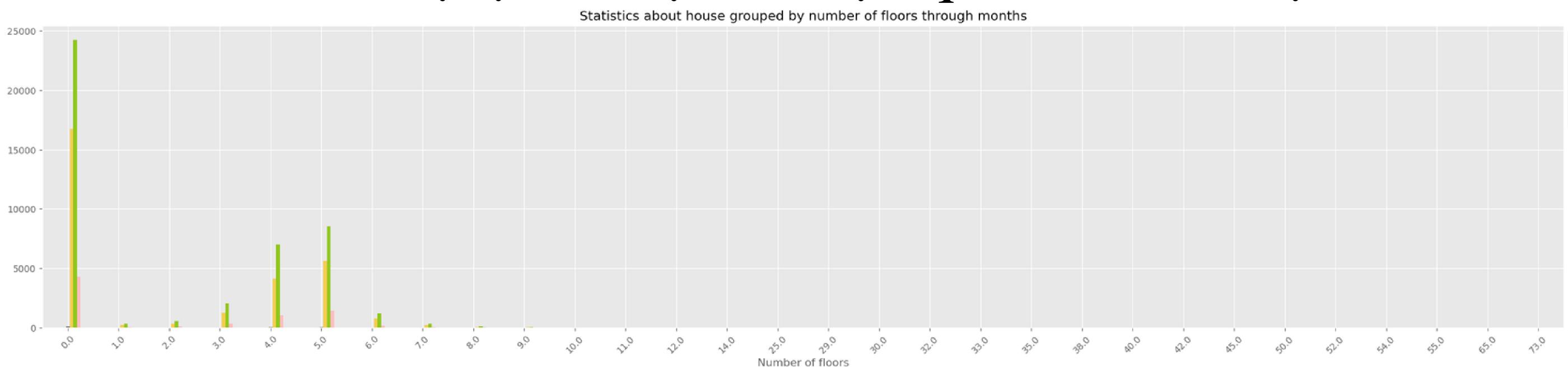
```
ques02_df_20_floor.plot.bar(ax = axes[0], rot = 45)
axes[0].legend(loc = "upper left", bbox_to_anchor = (1,1))
axes[0].set_title('Statistics about house grouped by number of floors through months')
plt.subplots_adjust(hspace = 0.5)
```

```
ques02_df_20_bedrooms.plot.bar(ax = axes[1], rot = 45)
axes[1].legend(loc = "upper left", bbox_to_anchor = (1,1))
axes[1].set_title('Statistics about house grouped by number of bedrooms through months')
```

Đặt câu hỏi

Câu 2: Xu hướng các căn nhà được định giá ở các thời điểm ?

Bước 4: Chuẩn bị lại dữ liệu và trực quan hóa dữ liệu

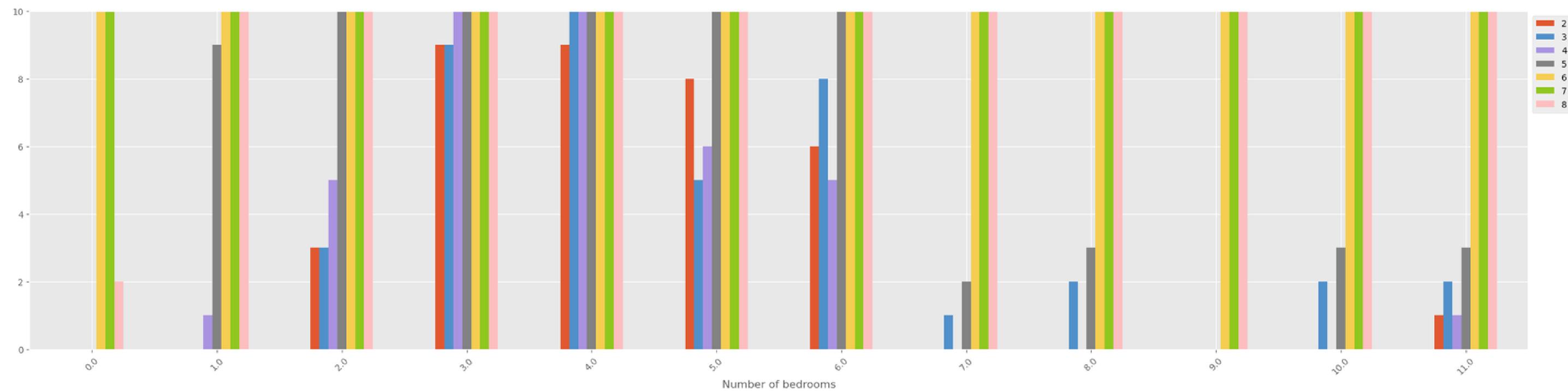
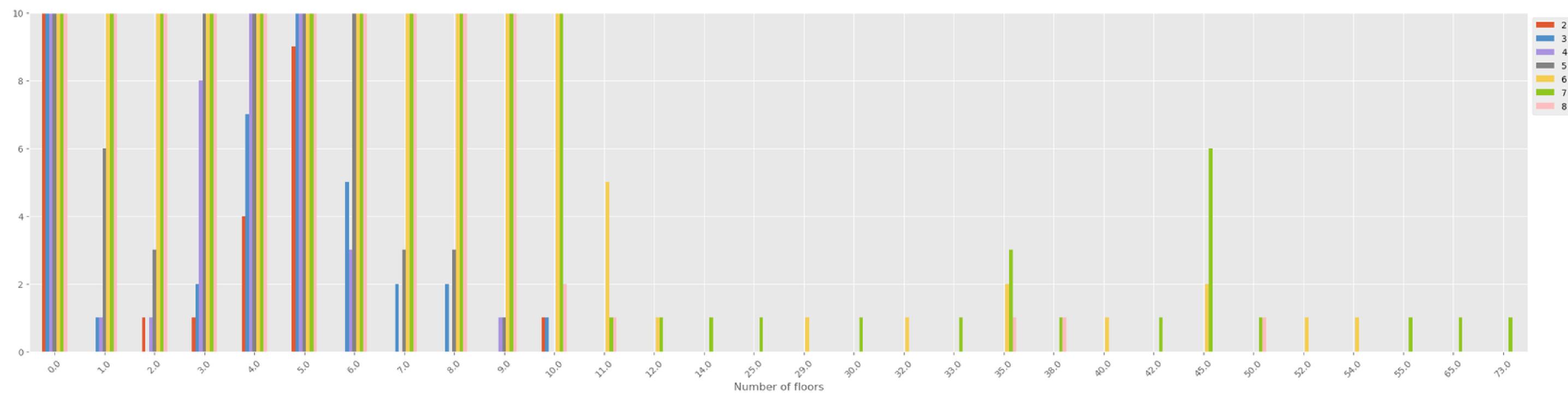


Nhận xét: Có vẻ 2 bản trên dữ liệu đã rõ ràng vào các căn nhà có số tầng là không xác định hay bằng 0, 3, 4, 5 và số phòng ngủ là 2, 3, 4, 5, 6. Nhìn kỹ vào các cột thì thấy rõ các tháng 6,7,8.

Đặt câu hỏi

Câu 2: Xu hướng các căn nhà được định giá ở các thời điểm ?

Bước 4: Chuẩn bị lại dữ liệu và trực quan hóa dữ liệu



Sau khi quan sát kỹ hơn có nhận xét: Có vẻ như vào tháng 6 7 8 bùng nổ hơn về dữ liệu, các căn nhà với số tầng và số phòng ngủ đa dạng hơn rất nhiều

Đặt câu hỏi

Câu 2: Xu hướng các căn nhà được định giá ở các thời điểm ?

Nhận xét:

- Qua biểu đồ thứ 1 (các căn nhà được định giá/ bán theo số tầng lầu), chúng ta thấy được xu hướng các căn nhà được định giá/ bán rõ ràng hơn. Cụ thể các căn nhà ko có tầng (được xem NaN) được định giá/ bán có số lượng nhiều nhất, sau đó là các căn có 3 đến 6 tầng lầu theo sau. Các số lượng này đều có điểm chung là tăng vọt ở tháng 6 7 8.
- Qua biểu đồ thứ 2 (các căn nhà được định giá/ bán theo số tầng lầu), chúng ta cũng thấy được rõ xu hướng các căn nhà được định giá/ bán rõ ràng. Chính xác các căn nhà có 2 phòng ngủ cho đến 6 phòng ngủ có số lượng nhiều nhất. Các số lượng đều chung xu hướng là tăng vọt ở tháng 6 7 8.

Vậy qua 2 biểu đồ trên có thể thấy xu hướng được xuất hiện rõ rệt ở tháng 6 7 8. Và thêm dữ liệu mới chỉ lấy được 5 ngày đầu của tháng 8 mà số lượng trên đã thể hiện được tiềm năng tăng vọt sau này. Ta cũng sẽ đoán những căn nhà có 3, 4 phòng ngủ và không xác định tầng hay 0 tầng với 4, 5 tầng sẽ là xu hướng được định giá/ bán sau này.

Đặt câu hỏi

Câu 3: vị trí của các căn nhà có tầm ảnh hưởng như thế nào với giá cả?

Trả lời câu hỏi thành công: Ta sẽ biết được vị trí căn nhà nào sẽ đắt đỏ và có thể dự đoán được tiềm năng của các vị trí căn nhà khác.

Khó khăn: Cần phải xử lý lại các giá trị ngoại lai.

Đặt câu hỏi

Câu 3: vị trí của các căn nhà có tầm ảnh hưởng như thế nào với giá cả?

Các bước thực hiện câu hỏi:

Bước 1: Lấy ra các cột cần sử dụng cho câu hỏi

Bước 2: Tiền xử lý lại các dữ liệu cần phải xử lý

Bước 3: Trực quan hóa dữ liệu

Bước 4: Ta sẽ thử quan sát thêm về tổng diện tích được định giá/bán

Đặt câu hỏi

Câu 3: vị trí của các căn nhà có tầm ảnh hưởng như thế nào với giá cả?

Bước 1: Lấy ra các cột cần sử dụng cho câu hỏi

	Address	District	Price/m2	Area
0	Đường Hoàng Quốc Việt, Phường Nghĩa Đô, Quận C...	Quận Cầu Giấy	86960000.0	46.0
1	Đường Kim Giang, Phường Kim Giang, Quận Thanh ...	Quận Thanh Xuân	116220000.0	37.0
2	phố minh khai, Phường Minh Khai, Quận Hai Bà T...	Quận Hai Bà Trưng	65000000.0	40.0
3	Đường Võng Thị, Phường Thụy Khuê, Quận Tây Hồ,...	Quận Tây Hồ	100000000.0	51.0
4	Đường Kim Giang, Phường Kim Giang, Quận Thanh ...	Quận Thanh Xuân	86110000.0	36.0
...
81676	Đường Hồ Tùng Mậu, Phường Phúc Diễn, Quận Bắc ...	Quận Bắc Từ Liêm	81580000.0	38.0
81677	Đường Trần Quốc Hoàn, Phường Quan Hoa, Quận Cầ...	Quận Cầu Giấy	292000000.0	50.0
81678	Đường Nguyễn Khánh Toàn, Phường Quan Hoa, Quận...	Quận Cầu Giấy	341460000.0	41.0
81679	Đường Quan Hoa, Phường Quan Hoa, Quận Cầu Giấy...	Quận Cầu Giấy	101670000.0	60.0
81680	Đường Hồ Tùng Mậu, Phường Mai Dịch, Quận Cầu G...	Quận Cầu Giấy	102220000.0	45.0

81681 rows x 4 columns

- Ta sẽ lấy ra các cột '**Address**', '**District**', '**Price/m2**', '**Area**'.
- Dữ liệu có 81681 dòng

Đặt câu hỏi

Câu 3: vị trí của các căn nhà có tầm ảnh hưởng như thế nào với giá cả?

Bước 2: Tiền xử lý lại các dữ liệu cần phải xử lý

	Address	District	billion/m2	Area
0	Đường Hoàng Quốc Việt, Phường Nghĩa Đô, Quận C...	Quận Cầu Giấy	0.08696	46.0
1	Đường Kim Giang, Phường Kim Giang, Quận Thanh ...	Quận Thanh Xuân	0.11622	37.0
2	phố minh khai, Phường Minh Khai, Quận Hai Bà T...	Quận Hai Bà Trưng	0.06500	40.0
3	Đường Võng Thị, Phường Thụy Khuê, Quận Tây Hồ,...	Quận Tây Hồ	0.10000	51.0
4	Đường Kim Giang, Phường Kim Giang, Quận Thanh ...	Quận Thanh Xuân	0.08611	36.0
...
81676	Đường Hồ Tùng Mậu, Phường Phúc Diễn, Quận Bắc ...	Quận Bắc Từ Liêm	0.08158	38.0
81677	Đường Trần Quốc Hoàn, Phường Quan Hoa, Quận Cầ...	Quận Cầu Giấy	0.29200	50.0
81678	Đường Nguyễn Khánh Toàn, Phường Quan Hoa, Quận...	Quận Cầu Giấy	0.34146	41.0
81679	Đường Quan Hoa, Phường Quan Hoa, Quận Cầu Giấy...	Quận Cầu Giấy	0.10167	60.0
81680	Đường Hồ Tùng Mậu, Phường Mai Dịch, Quận Cầu G...	Quận Cầu Giấy	0.10222	45.0

81680 rows x 4 columns

- Scale lại giá trị của price/m²
- Loại bỏ giá trị outlier của price như câu 1 đã gấp.

Đặt câu hỏi

Câu 3: vị trí của các căn nhà có tầm ảnh hưởng như thế nào với giá cả?

Bước 3: Trực quan hóa dữ liệu

```
mean_district = ques03_df.groupby(['District'])['billion/m2'].mean()
labels = list(mean_district.index)

plt.style.use('ggplot')
fig, ax = plt.subplots(figsize=(15, 8))
ax = squarify.plot(sizes= mean_district,
                    label= mean_district.index,
                    color= sns.color_palette('twilight', len(mean_district)),
                    alpha= 0.8,
                    text_kwargs={'fontsize': 5})

ax.axis('off')
plt.legend(loc = "upper left", bbox_to_anchor = (1,1), handles=ax.containers[0], labels= labels)
plt.show()
```

Đặt câu hỏi

Câu 3: vị trí của các căn nhà có tầm ảnh hưởng như thế nào với giá cả?

Bước 3: Trực quan hóa dữ liệu



Đặt câu hỏi

Câu 3: vị trí của các căn nhà có tầm ảnh hưởng như thế nào với giá cả?

Bước 4: Ta sẽ thử quan sát thêm về tổng diện tích được định giá/bán

```
sum_area = ques03_df.groupby(['District'])['Area'].sum().sort_values()
```

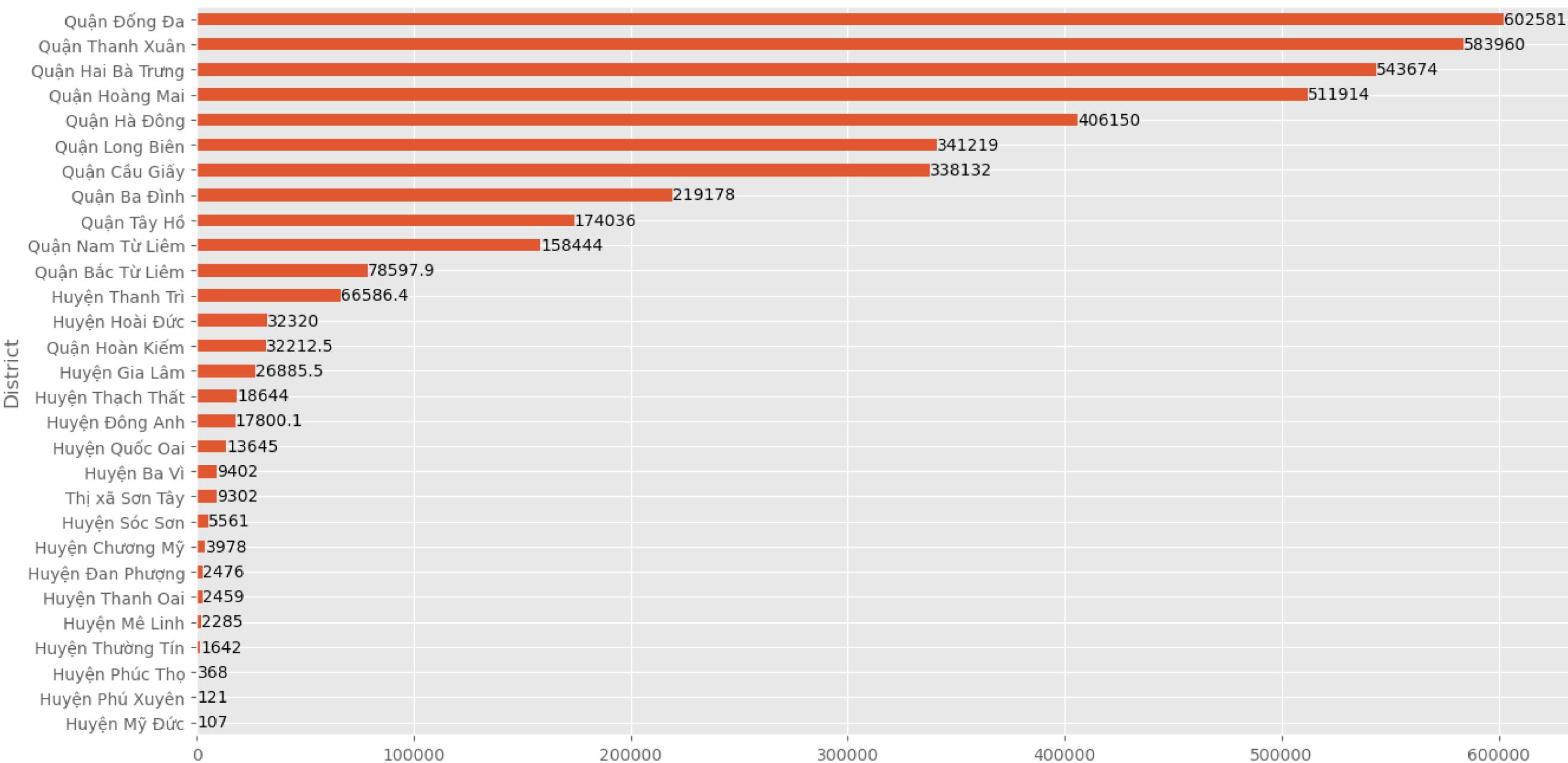
```
plt.style.use('ggplot')
fig, ax = plt.subplots(figsize=(15, 8))
sum_area.plot.barh()

for i in ax.containers:
    ax.bar_label(i)
plt.show()
```

Đặt câu hỏi

Câu 3: vị trí của các căn nhà có tầm ảnh hưởng như thế nào với giá cả?

Bước 4: Ta sẽ thử quan sát thêm về tổng diện tích được định giá/bán



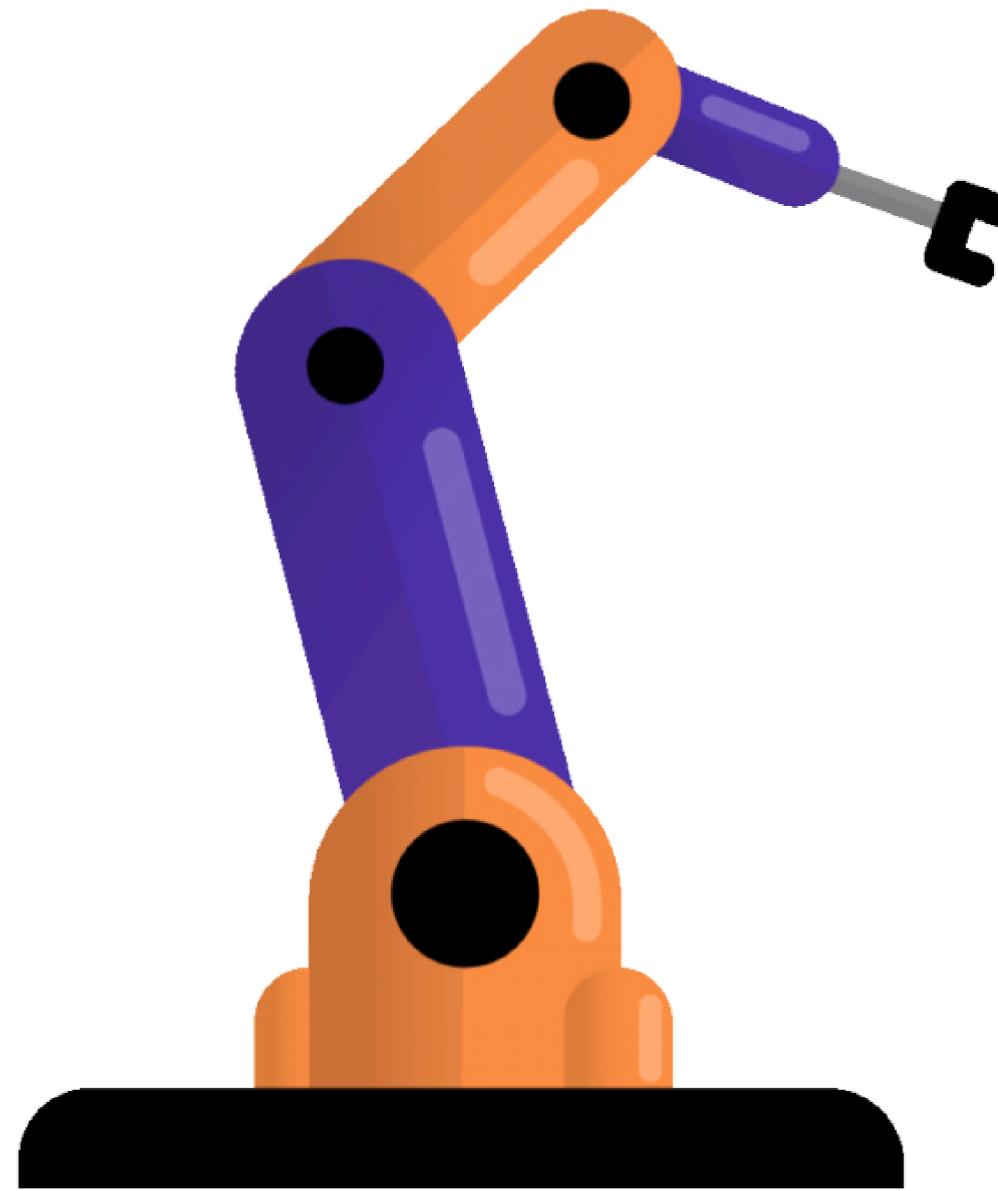
Đặt câu hỏi

Câu 3: vị trí của các căn nhà có tầm ảnh hưởng như thế nào với giá cả?

Nhận xét:

- Ở biểu đồ tree trong bước 3, chúng ta thấy đất ở Quận Hà Đông có giá cả trung bình cao nhất (vượt trội) so với các quận/huyện khác, và đứng nhì là giá đất ở Huyện Thanh Trì. Rẻ nhất chắc chắn là thuộc về Huyện Ba Vì.
- Ở biểu đồ cột ở bước 4, chúng ta thấy diện tích ở Quận Hà Đông lại đứng hạng 5 và huyện Thanh Trì lại đứng hạng 12. Số diện tích được bán ra ở Quận Đống Đa là nhiều nhất.

Qua đây, chúng ta có thể dự đoán rằng Quận Đống Đa, Hai Bà Trưng, Thanh Xuân giá đất vẫn còn rẻ và được rao bán nhiều, có thể đất ở đây sau này tiềm năng chẳng? Với diện tích được bán ra nhiều, có cơ hội phát triển với việc giá thành đất còn rẻ rất hợp lý cho các chủ đầu tư.

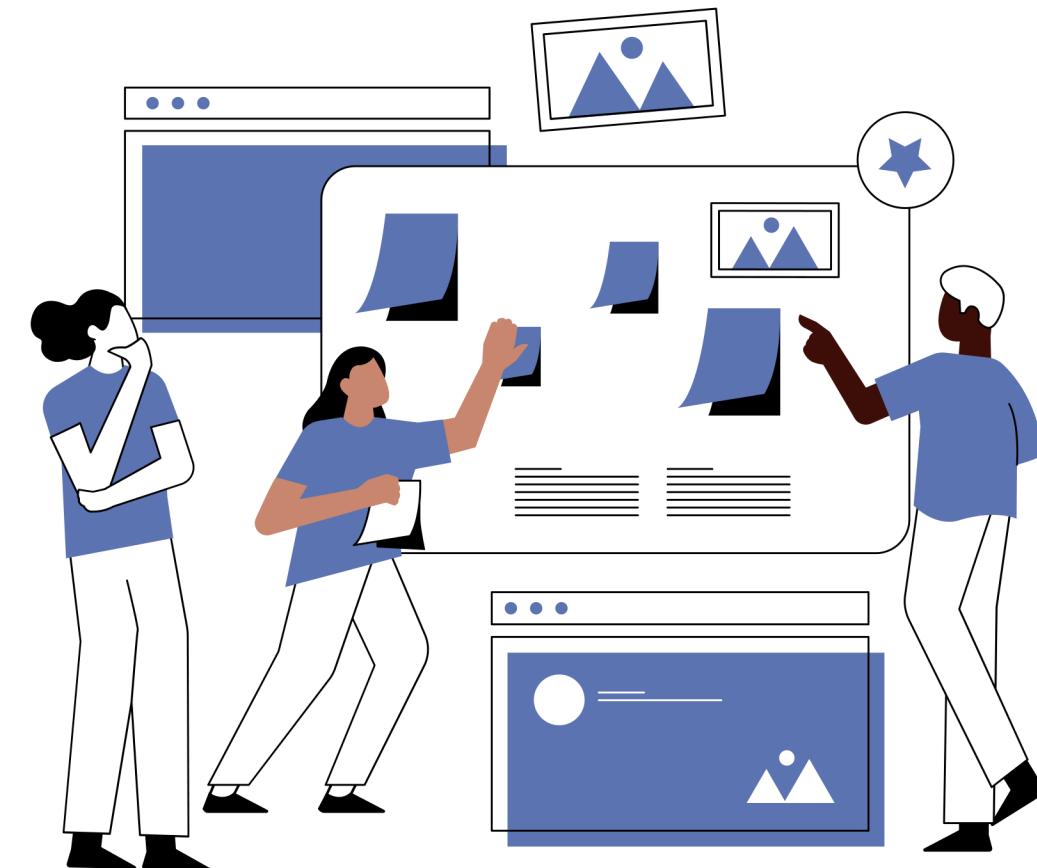


Mô hình hóa dữ liệu

Mô hình hóa dữ liệu

Xác định câu hỏi cần trả lời

- Bài toán dự đoán Price/m² dựa trên các đặc trưng của ngôi nhà:
 - Đây là một bài toán hồi quy
 - Input là các đặc trưng của một ngôi nhà: địa chỉ, kích thước, số phòng ngủ,...
 - Output là giá của căn nhà (/m²)
- Trả lời được câu hỏi này sẽ giúp cho:
 - Người bán quyết định giá bán ngôi nhà phù hợp với thị trường, đưa ra được mức giá hợp lý.
 - Người mua chọn được ngôi nhà mong muốn.



Mô hình hóa dữ liệu

Phân tích, xử lý dữ liệu để chuẩn bị cho bước mô hình hóa

- Loại bỏ các dữ liệu nan (số lượng giá trị nan > 50%)

```
house_df = house_df[house_df['Length'].notna() & house_df['Width'].notna() & house_df['Number of floors'].notna()]

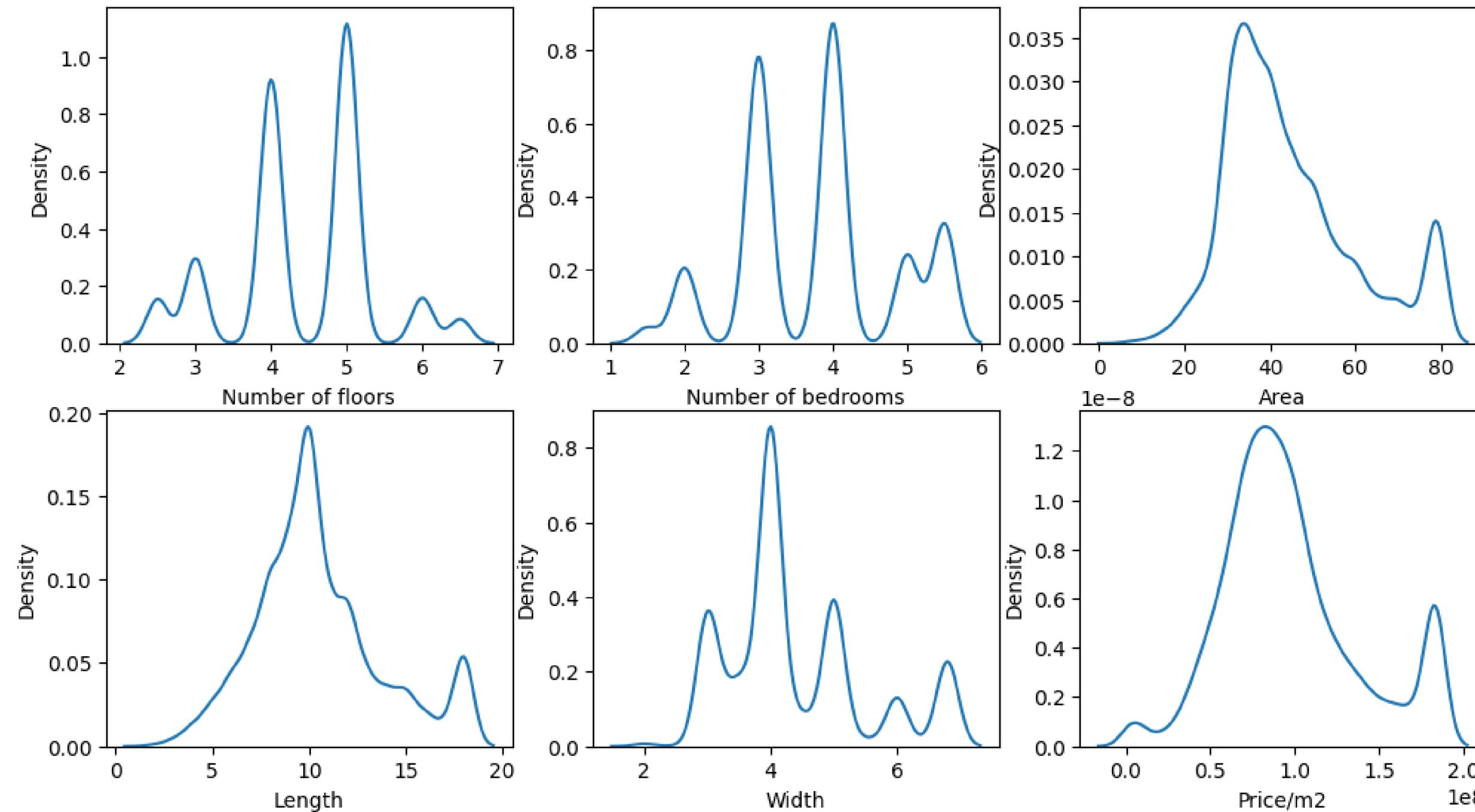
house_df.reset_index(inplace=True, drop=True)
```

- Tiếp theo, xử lý các outliers bằng Class **BoxplotOutlierClipper**:
 - Tính phân vị thứ nhất và phân vị thứ 3, từ đó tính được giá trị IQR.
 - Lower_Bound = Q1 - 1.5*IQR
 - Upper_Bound = Q3 + 1.5*IQR
 - Với các giá trị nhỏ hơn Lower_Bound hoặc lớn hơn Upper_Bound thì được xác định là outliers và được chuyển thành giá trị min (max) gần với nó.

Mô hình hóa dữ liệu

Phân tích, xử lý dữ liệu để chuẩn bị cho bước mô hình hóa

- Phân bố của các cột dữ liệu sau khi loại bỏ các giá trị outliers



Mô hình hóa dữ liệu

Phân tích, xử lý dữ liệu để chuẩn bị cho bước mô hình hóa

- Chọn các cột làm đầu vào cho mô hình.

```
cols = ['District', 'Ward', 'Type', 'Legal', 'Number of floors', 'Number of bedrooms', 'Area', 'Length', 'Width',  
        'Price/m2']  
x_df = house_df[set(cols) - {'Price/m2'}]  
y_df = house_df['Price/m2']
```

- Viết hàm `get_dataset()` nhận các tham số đầu vào gồm: 2 dataframe, các phương thức scale đối với biến đầu vào và đầu ra. Hàm này sẽ làm nhiệm vụ phân chia tập train/test và scale lại giá trị của các biến, sau đó trả ra các tập `x_train`, `x_test`, `y_train`, `y_test`.

```
def get_dataset(x_df, y_df, input_scaler, output_scaler):  
    x_train, x_test, y_train, y_test = train_test_split(x_df, y_df, test_size=0.2, random_state=5)  
    if input_scaler is not None:  
        input_scaler.fit(x_train)  
        x_train = input_scaler.transform(x_train)  
        x_test = input_scaler.transform(x_test)  
    if output_scaler is not None:  
        y_train = y_train.values.reshape(len(y_train), 1)  
        y_test = y_test.values.reshape(len(y_test), 1)  
        output_scaler.fit(y_train)  
        y_train = output_scaler.transform(y_train)  
        y_test = output_scaler.transform(y_test)  
    return x_train, x_test, y_train, y_test
```

Mô hình hóa dữ liệu

Phân tích, xử lý dữ liệu để chuẩn bị cho bước mô hình hóa

- Tạo pipeline cho bước tiền xử lý dữ liệu
 - Chọn các biến categorical và numeric để xử lý riêng:

```
cate = ['District', 'Ward', 'Type', 'Legal']  
nume = ['Number of floors', 'Number of bedrooms', 'Area', 'Length', 'Width']
```

- Scale cho biến đầu ra bằng hàm log:

```
target_transformer = FunctionTransformer(np.log, inverse_func = np.exp, validate=True ,check_inverse = True)
```

- Xử lý cho các cột categorical:

```
# Thay thế missing value bằng giá trị có tần suất xuất hiện cao
cate_transformer = Pipeline(steps= [('imputer', SimpleImputer(strategy='most_frequent')),
                                    ('OneHotEncoder', OneHotEncoder(handle_unknown='ignore'))]) # biểu diễn one hot cho các cột category
                                                               # đối với các giá trị mà model chưa từng thấy bao giờ (trong tập test) thì ta bỏ qua
```

- Scale các cột numeric bằng standard scale:

```
num_transformer = StandardScaler()
```

Mô hình hóa dữ liệu

Phân tích, xử lý dữ liệu để chuẩn bị cho bước mô hình hóa

- Tạo pipeline cho bước tiền xử lý dữ liệu
- Pipeline cho bước xử lý chung:

```
preprocessor = ColumnTransformer(transformers=[('cate_transformer', cate_transformer, cate),
                                              ('nume_transfomer', nume_transfomer, nume)])
```

Mô hình hoá dữ liệu

Mô hình hoá

- **Mô hình Random Forest Regression:**

- Đầu tiên, ta scale 2 biến đầu ra là y_train và y_test bằng hàm log.
- Sau đó, áp dụng mô hình **random forest** vào pipeline tiền xử lý đã tạo
- Fit pipeline vào tập train: (x_train, y_train).

- **Đánh giá:**

- Dựa vào bảng so sánh giữa giá trị thực tế và dự đoán ta thấy mô hình cho kết quả khá tốt trên tập test.

```
Train set: [MAE: 29289810.842, MAPE: 99.298]
Test set: [MAE: 28874672.070, MAPE: 99.388]
```

	Actual	Predict
0	121.210	107.772
1	96.490	80.601
2	103.000	80.601
3	183.435	98.441
4	90.700	80.601

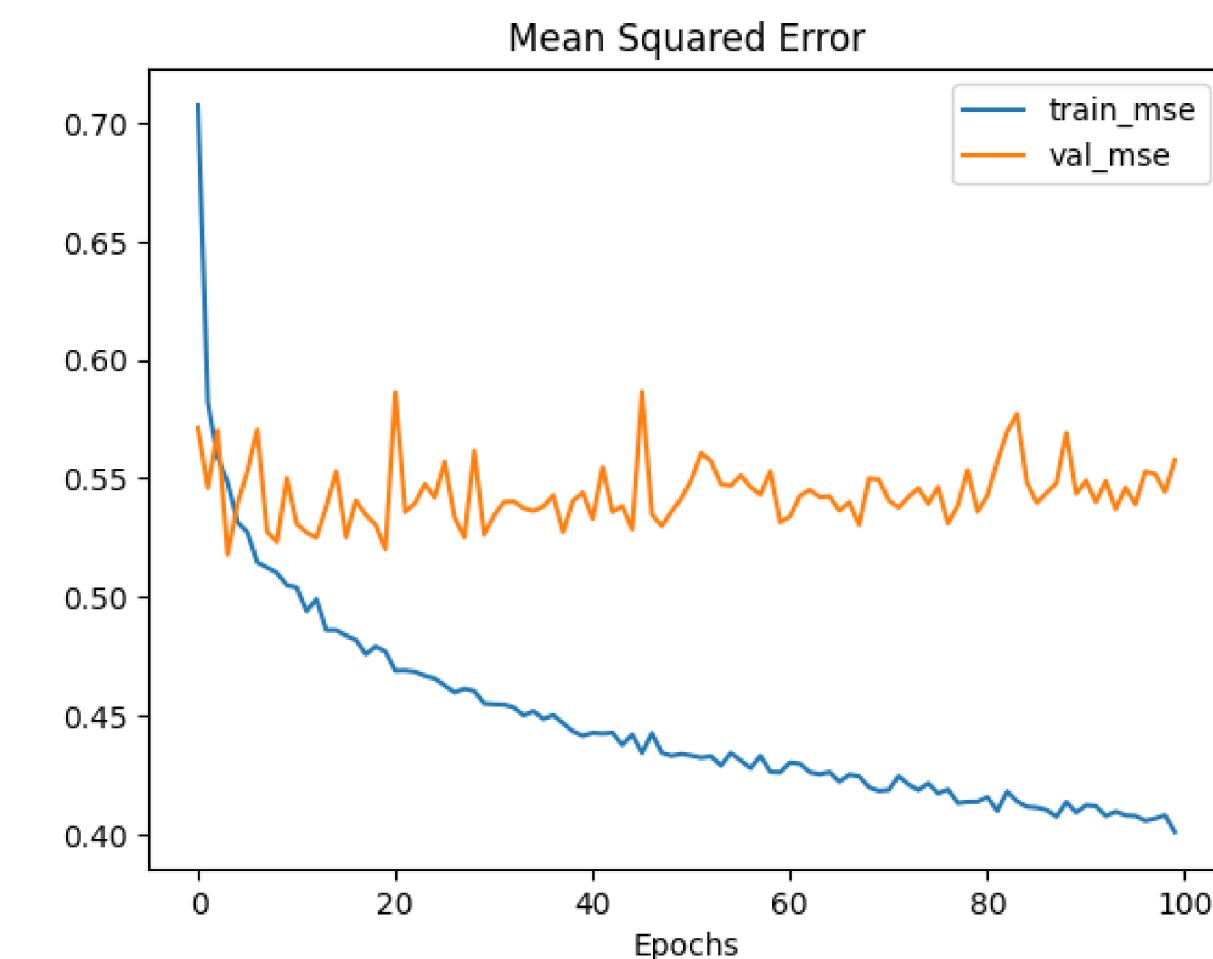
Mô hình hoá dữ liệu

Mô hình hoá

- **Mô hình MLP Regression:**

- Đầu tiên, ta scale 2 biến đầu ra là y_train và y_test bằng standard scale.
- Sử dụng mô hình multilayer perceptron với đầu vào là 266 features, 1 lớp ẩn với 10 nodes và dùng hàm kích hoạt 'relu', cách khởi tạo các trọng số là 'normal'.
- Kết hợp với hàm tối ưu là stochastic gradient(learning rate=0.01, momentum=0.9).
- Lấy tập validation là 20% tập train ban đầu

- **Độ lỗi trên tập train và val qua các epochs**



Mô hình hoá dữ liệu

Mô hình hoá

- Đánh giá:

- So sánh các giá trị **actual** và **predict** trên tập test, ta thấy kết quả tương đối tốt.
- Độ lỗi MAE và MAPE cũng tốt hơn so với mô hình Random ForestRegression.

```
Train set: [MAE: 18943899.811, MAPE: 99.456]  
Test set: [MAE: 21130070.853, MAPE: 99.471]
```

	Actual	Predict
0	121.210	92.952
1	96.490	97.923
2	103.000	82.599
3	183.435	134.626
4	90.700	94.191

Mô hình hóa dữ liệu

Mô hình hóa

- **Fine-tuning process:**

- Sử dụng kĩ thuật RandomizedSearchCV()
- Phương pháp này tìm kiếm ngẫu nhiên trên không gian tham số, kết hợp với Cross-validation và chọn ra bộ tham số tốt nhất cho mô hình.

- **Fine-tuning cho mô hình ANN**

- Không gian tham số của mô hình

```
params = {'batch_size': [10, 20, 30, 50, 80],  
          'epochs': [10, 20, 50, 80, 100],  
          'optimizer': ['adam', 'rmsprop', 'sgd', 'lbfgs']}
```

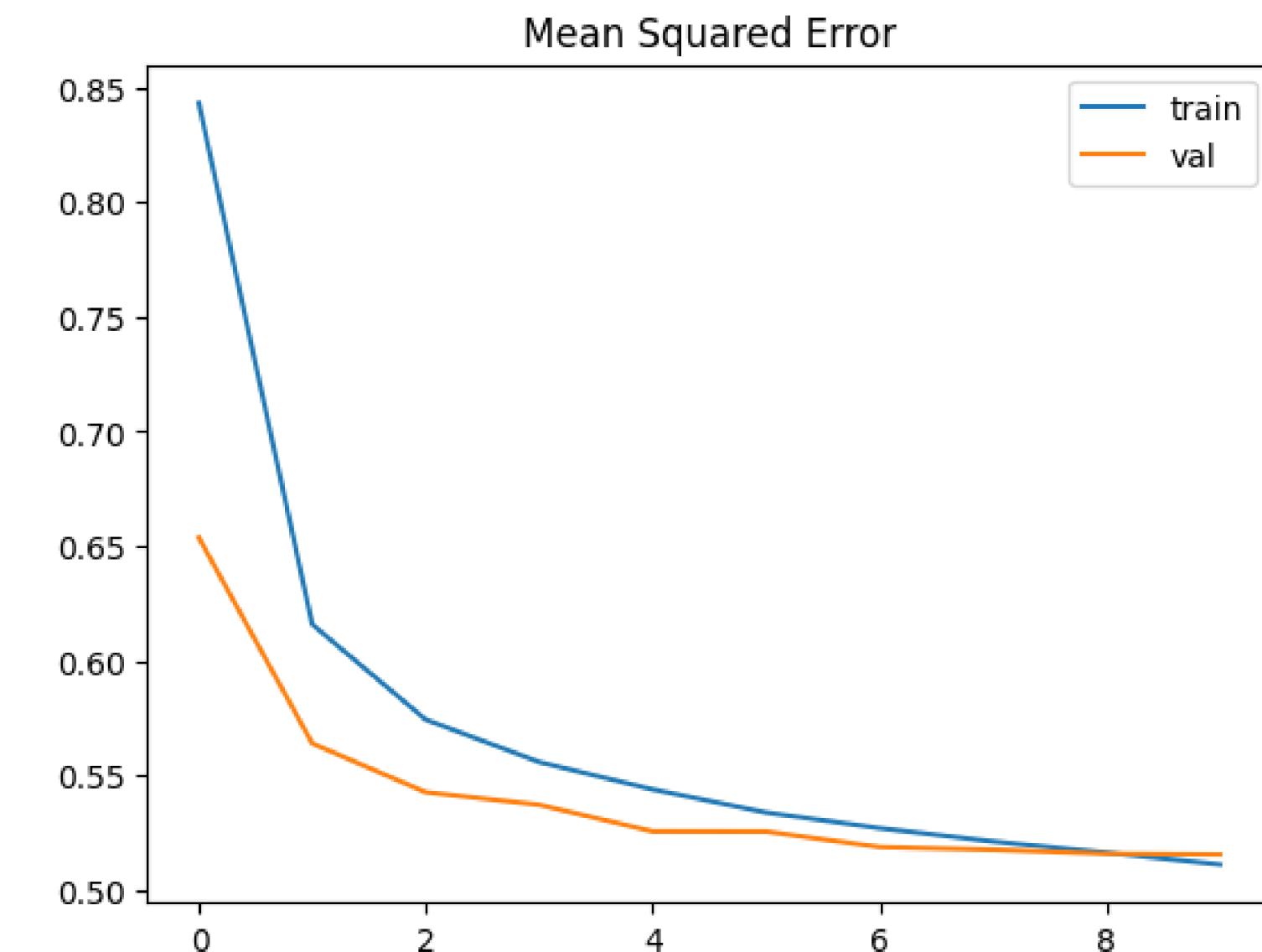
- Bộ tham số tốt nhất cho mô hình

```
ANN_random.best_params_  
[] ✓ 0.3s  
{'optimizer': 'rmsprop', 'epochs': 10, 'batch_size': 30}
```

Mô hình hoá dữ liệu

Mô hình hoá

- **Fine-tuning cho mô hình ANN**
 - Thực hiện train trên bộ tham số tốt nhất vừa tìm được
 - Độ lỗi trên tập **train** và **val** qua các epochs:



Mô hình hoá dữ liệu

Mô hình hoá

- Fine-tuning cho mô hình ANN
- Đánh giá:
 - So sánh một vài giá trị **actual** và **predict** trên tập test, ta thấy mô hình cho kết quả khá tốt.
 - Độ lỗi MAE và MAPE:

```
Train set: [MAE: 20562803.477, MAPE: 99.364]  
Test set: [MAE: 20895541.574, MAPE: 99.448]
```

	Actual	Predict
0	121.210	109.463
1	96.490	96.853
2	103.000	90.326
3	183.435	122.706
4	90.700	84.180

Mô hình hóa dữ liệu

Mô hình hóa

- Fine-tuning cho mô hình Random Forest Regression

- Không gian tham số cho mô hình Random Forest:

```
n_estimators = [200, 400, 600, 800, 1000, 1200] # số lượng cây
max_features = ['auto', 'sqrt'] # cách tính số lượng đặc trưng mỗi lần split
max_depth = [int(x) for x in np.linspace(10, 120, num = 12)] # độ sâu tối đa của cây
min_samples_split = [2, 6, 10] # số lượng mẫu tối thiểu trong một node
min_samples_leaf = [1, 3, 4] # số lượng mẫu tối thiểu trong một node lá
bootstrap = [True, False] # cách lấy mẫu (có hoàn lại hay không)
```

- Sử dụng randomized search CV với số fold là 5, n_iter = 100, độ lỗi được tính là negative mse.

```
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = params,
                                 scoring='neg_mean_absolute_error', return_train_score=True,
                                 n_iter = 100, cv = 5, verbose=2, random_state=10)

rf_random.fit(x_train, y_train)
```

Mô hình hoá dữ liệu

Mô hình hoá

- Fine-tuning cho mô hình Random Forest Regression

- Bộ tham số tốt nhất tìm được:

```
rf_random.best_params_
✓ 0.4s

{'n_estimators': 200,
 'min_samples_split': 6,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 70,
 'bootstrap': False}
```

- Thực hiện train lại trên bộ tham số tốt nhất này.

```
rf = RandomForestRegressor(n_estimators=200, min_samples_split=6, min_samples_leaf=1,
                           max_features='sqrt', max_depth=70, bootstrap=False)

rf.fit(x_train, y_train)
y_train_pred = rf.predict(x_train)
```

Mô hình hoá dữ liệu

Mô hình hoá

- Fine-tuning cho mô hình Random Forest Regression

- Đánh giá kết quả:

```
Train set: [MAE: 9552643.674, MAPE: 99.703]
Test set: [MAE: 19809681.645, MAPE: 99.462]
```

- Mô hình cho độ lỗi MAE và MAPE
đều tốt hơn mô hình ANN.

	Actual	Predict
0	121.210	110.135
1	96.490	101.187
2	103.000	81.538
3	183.435	110.835
4	90.700	87.760

