BÁO CÁO ĐỒ ÁN

Dự đoán giá bán chung cư ở TP.HCM

Đồ án cuối kì – Nhập môn KDHL – Nhóm 10

chotot.vn

THÔNG TIN THÀNH VIÊN

20120128 - Nguyễn Thị Cẩm Lai – Nhóm trưởng

20120037 - Trần Thị Minh Anh – Thành viên

20120166 - Nguyễn Dương Tuấn Phương – Thành viên

20120547 - Võ Thành Phong – Thành viên

Nội dung thuyết trình

01

Thu thập dữ liệu

02

Khám phá dữ liệu

03

Đặt và trả lời câu hỏi

04

Mô hình hóa dữ liệu

I. Thu thập dữ liệu

1. Chủ đề: Thị trường giá bán căn hộ chung cư tại TP.HCM.
2. Nguồn: Trên trang web chotot.com.
3. Thư viện hỗ trợ thu thập dữ liệu:
   - Selenium.
   - BeautifulSoup.

Ứng với từng chung cư, thu thập các trường dữ liệu (nếu có) hiển thị trên trang web.

1,135

Dòng dữ liệu

14

Cột dữ liệu

Địa chỉ
Diện tích
Tên dự án
Giá bán
Giá bán / m2
Tình trạng BĐS
Giấy tờ pháp lý

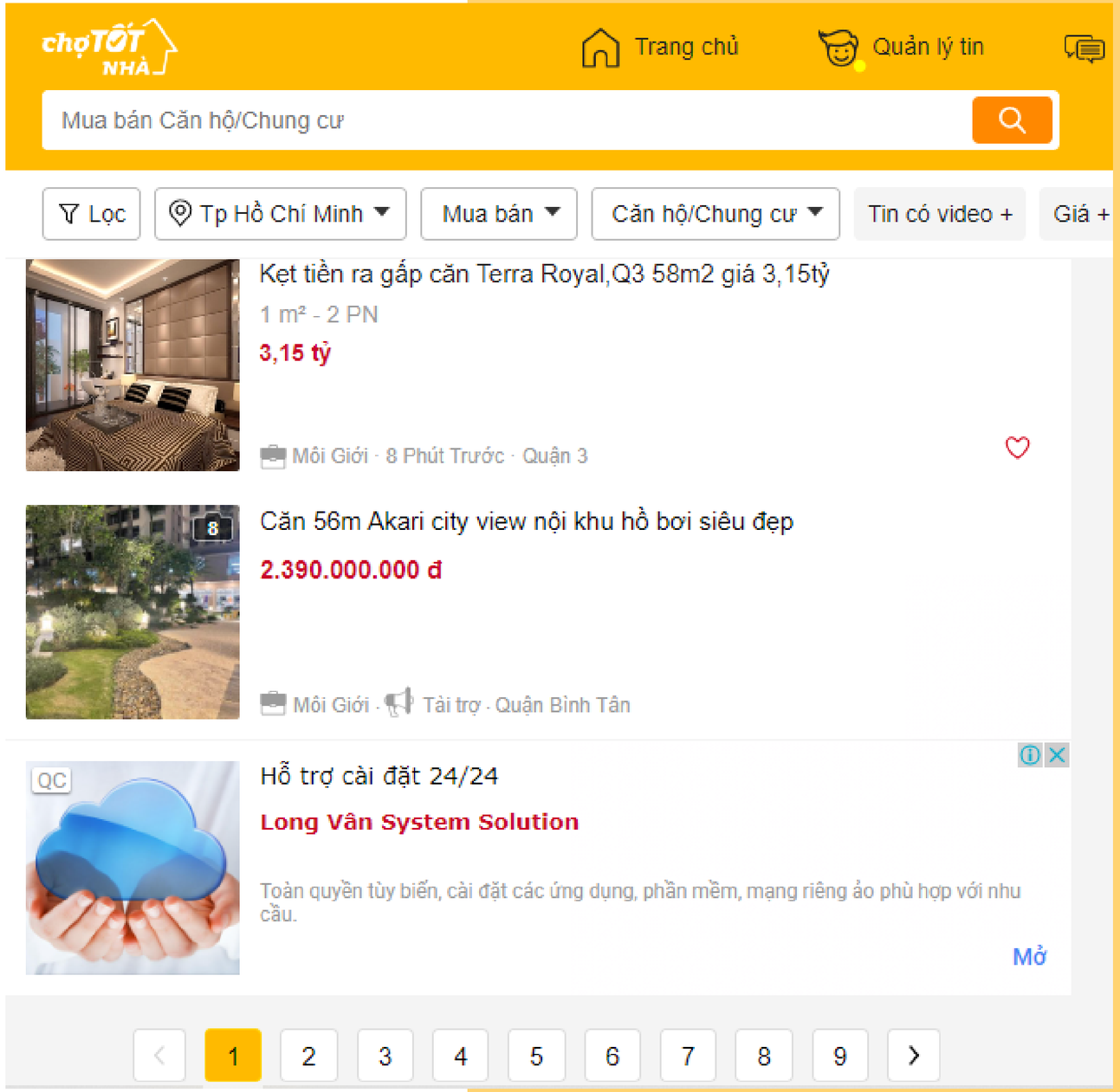
Số phòng ngủ
Số phòng toilet
Số tầng
Hướng ban công
Hướng cửa chính
Tình trạng nội thất
Đặc điểm

Cách thu thập

- Dùng **Selenium**: truy cập và chuyển trang
- Dùng **BeautifulSoup**: trích xuất HTML
- Truy cập vào từng trang -> Link các chung cư
- Truy cập vào link các chung cư -> Lấy dữ liệu
- Chia ra từng nhóm để truy cập, lưu lại file csv.

  => Gộp lại thanh file csv cuối cùng.

  => giảm rủi ro, linh hoạt thời gian chạy code.



II. Khám phá dữ liệu

Giới thiệu: Ở phần này chúng ta sẽ thực hiện khám phá dữ liệu bằng cách sử dụng thống kê mô tả để hiểu dữ liệu tốt hơn, tức là để xác định các vấn đề về dữ liệu

NỘI DUNG THỰC HIỆN

- Đọc dữ liệu và tính số dòng và cột.
- Mỗi dòng có ý nghĩa gì? Có vấn đề các dòng có ý nghĩa khác nhau không?
- Dữ liệu có các dòng bị lặp không?
- Tỉ lệ giá trị thiếu và thống kê mô tả của từng cột.
- Kiểu dữ liệu của mỗi cột.
- Xem xét tập giá trị của các thuộc tính phân loại.
- Xem xét sự phân bố giá trị của các cột dữ liệu dạng số.
- Xem xét sự phân bố giá trị của các cột dữ liệu không phải dạng số.

Một số vấn đề phát hiện từ bộ dữ liệu cần tiền xử lý

❑ Tỉ lệ giá trị thiếu của từng thuộc tính

```
missing_ratio = house_df.isnull().sum()
missing_ratio = missing_ratio / num_rows
missing_ratio
```

```
DiaChi              0.000000
GiaBan              0.000000
DuAn                0.000000
DienTich            0.000000
TinhTrangBDS        0.000881
Gia/m2              0.000000
PhongNgu            0.000000
PhongVeSinh         0.166520
SoTang              0.751542
TinhTrangGiayTo     0.496916
TinhTrangNoiThat    0.536564
HuongBanCong        0.748899
HuongCua            0.796476
DacDiem             0.824670
dtype: float64
```

Nhận xét:

- Có thể thấy, dữ liệu có rất nhiều thuộc tính bị thiếu và tỉ lệ thiếu khá lớn.
- Nguyên nhân là do người đăng thông tin lên web có các thông tin cho ngôi nhà khác nhau, và trang web không có một form bắt buộc cho các bài được đăng nên việc muốn cung cấp bao nhiêu thông tin là phụ thuộc vào người đăng.

Tuy nhiên:

Ở bước khám phá, chúng ta chưa cần vội để tìm cách tiền xử lý vấn đề này, vì nếu cách xử lý không phù hợp sẽ ảnh hưởng đến mô hình học máy.

Một số vấn đề phát hiện từ bộ dữ liệu cần tiền xử lý

- Về mặt kiểu dữ liệu, các thuộc tính của tập dữ liệu có nhiều cột đang ở định dạng chưa phù hợp.

```
DiaChi                 object
GiaBan                 object
DuAn                   object
DienTich               object
TinhTrangBDS           object
Gia/m2                 object
PhongNgu               object
PhongVeSinh            object
SoTang                float64
TinhTrangGiayTo        object
TinhTrangNoiThat       object
HuongBanCong           object
HuongCua               object
DacDiem                object
dtype: object
```

Nhận xét:

- Các cột: GiaBan, DienTich, Gia/m2, PhongNgu, PhongVeSinh nên được đưa về kiểu dữ liệu numerical để tiếp tục xử lý.

Cách xử lý:

- Bước 1: Kiểm tra xem các giá trị trong cùng một thuộc tính liệu chúng có cùng một đơn vị hay không.
- Bước 2: Nếu ở bước 1, các giá trị không cùng đơn vị với nhau thì ta sẽ tiến hành xử lý đưa về cùng đơn vị.
- Bước 3: Xóa phần đơn vị, chỉ để lại phần giá trị số.

Một số vấn đề phát hiện từ bộ dữ liệu cần tiền xử lý

- Cột địa chỉ chứa các giá trị quá cụ thể dẫn đến sự riêng biệt, không có ý nghĩa cho việc trực quan hay phân tích.

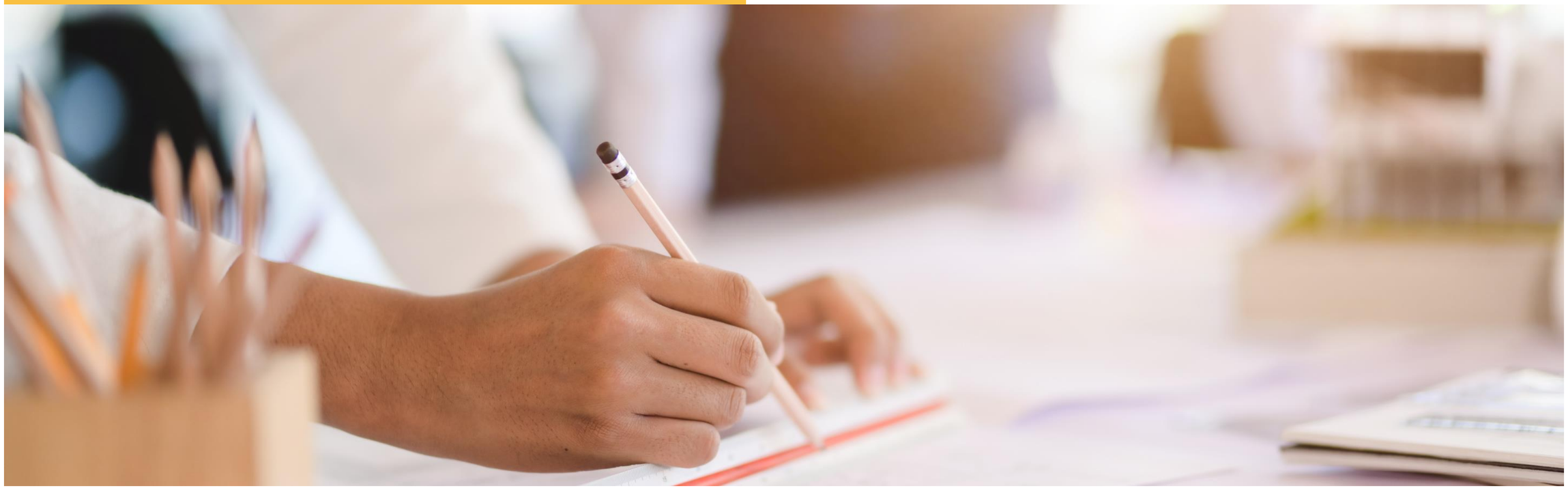
DiaChi
Đường Huy Cận, Phường Phước Long B (Quận 9 cũ), Thành phố Thủ Đức, Tp Hồ Chí Minh
số 88, số 88 Đường N1, Phường Sơn Kỳ, Quận Tân Phú, Tp Hồ Chí Minh
Võ Văn Kiệt, Phường An Lạc, Quận Bình Tân, Tp Hồ Chí Minh

Cách xử lý: Ta sẽ lấy địa chỉ một cách tổng quát là quận/huyện/thành phố trực thuộc và thêm vào dataframe cột 'Quan'.

DiaChi	Quan
Đường Huy Cận, Phường Phước Long B (Quận 9 cũ), Thành phố Thủ Đức, Tp Hồ Chí Minh	Thành phố Thủ Đức
số 88, số 88 Đường N1, Phường Sơn Kỳ, Quận Tân Phú, Tp Hồ Chí Minh	Quận Tân Phú
Võ Văn Kiệt, Phường An Lạc, Quận Bình Tân, Tp Hồ Chí Minh	Quận Bình Tân

THU THẬP KẾT QUẢ

Sau khi đã khám phá và đan xen thực hiện tiền xử lý dữ liệu, chúng ta sẽ tiến hành lưu dữ liệu có được vào file .csv nhằm phục vụ cho các phần sau của đồ án (Đặt các câu hỏi có ý nghĩa cần trả lời, Mô hình hóa).



III. Đặt và trả lời câu hỏi

1. Nguồn cảm hứng cho các câu hỏi
2. Các câu hỏi được đặt ra
3. Trả lời một số câu hỏi

Nguồn cảm hứng

CHỦ ĐỀ

Giá bán chung cư HCM

Tham khảo từ các bài báo trên mạng

- Nhu cầu về nơi ở là một nhu cầu vô cùng cần thiết
- Thị trường nhà ở đang tăng trưởng mạnh mẽ
- Có rất nhiều yếu tố quyết định đến giá cả

CÂU HỎI ĐƯỢC ĐẶT RA

- **Câu 1:** Số phòng ngủ phổ biến của các căn hộ chung cư được rao bán là bao nhiêu?
- **Câu 2:** Các căn hộ chung cư được rao bán có diện tích bao nhiêu là phổ biến nhất?
- **Câu 3:** Thống kê theo tính pháp lí, giấy tờ của dự án đầu tư của các chung cư? Thông thường các căn hộ có giá cao sẽ có tình trạng pháp lí, giấy tờ như thế nào?
- **Câu 4:** Phân bố căn hộ dựa trên vị trí và vị trí khu vực nào trong TPHCM sẽ có giá bán căn hộ chung cư cao nhất?
- **Câu 5:** Phân bố căn hộ dựa trên tình trạng nội thất và loại tình trạng nội thất nào của căn hộ được rao bán (bàn giao thô, nội thất cao cấp, nội thất cơ bản,...) sẽ được ưa chuộng nhất?
- **Câu 6:** Giá cả của một căn hộ chung cư có bị ảnh hưởng bởi phong thủy cụ thể là hướng ban công hay hướng cửa không?

CÂU HỎI

Câu 1: Số phòng ngủ phổ biến của các căn hộ chung cư được rao bán là bao nhiêu?

Ý nghĩa khi trả lời được câu hỏi:

- Các căn hộ chung cư được thiết kế rất đa dạng tùy nhu cầu sử dụng.
- Biết được đâu sẽ là loại hình căn hộ có số phòng ngủ phổ biến nhất? Nó sẽ phù hợp cho các gia đình như thế nào?
- Từ đây chúng ta sẽ nắm bắt được nhu cầu thị trường nhà ở và có thể đầu tư sinh lời từ lĩnh vực này.

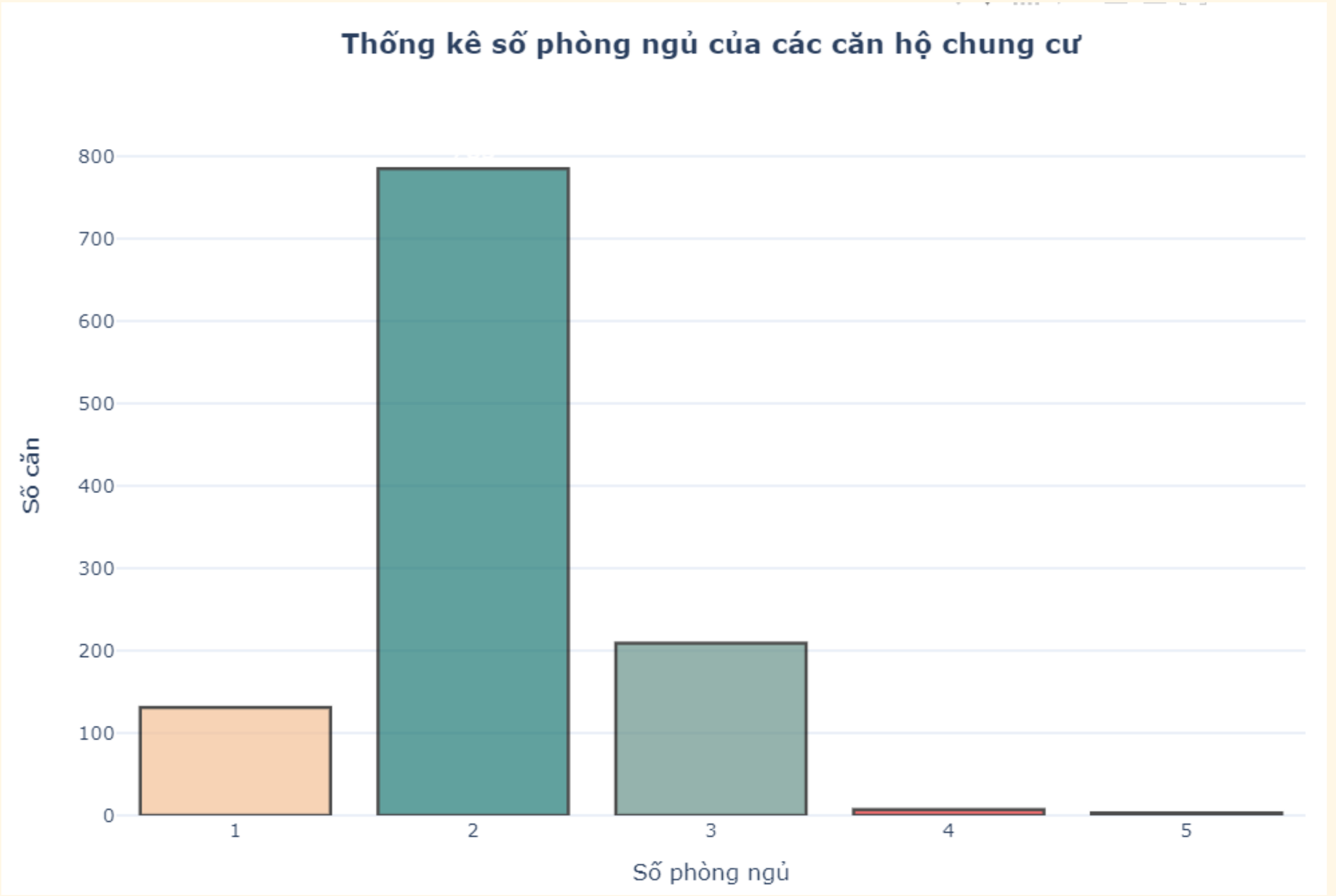
CÂU HỎI

Câu 1: Số phòng ngủ phổ biến của các căn hộ chung cư được rao bán là bao nhiêu?

Để phân tích dữ liệu trả lời cho câu hỏi này, chúng ta sẽ làm như sau:

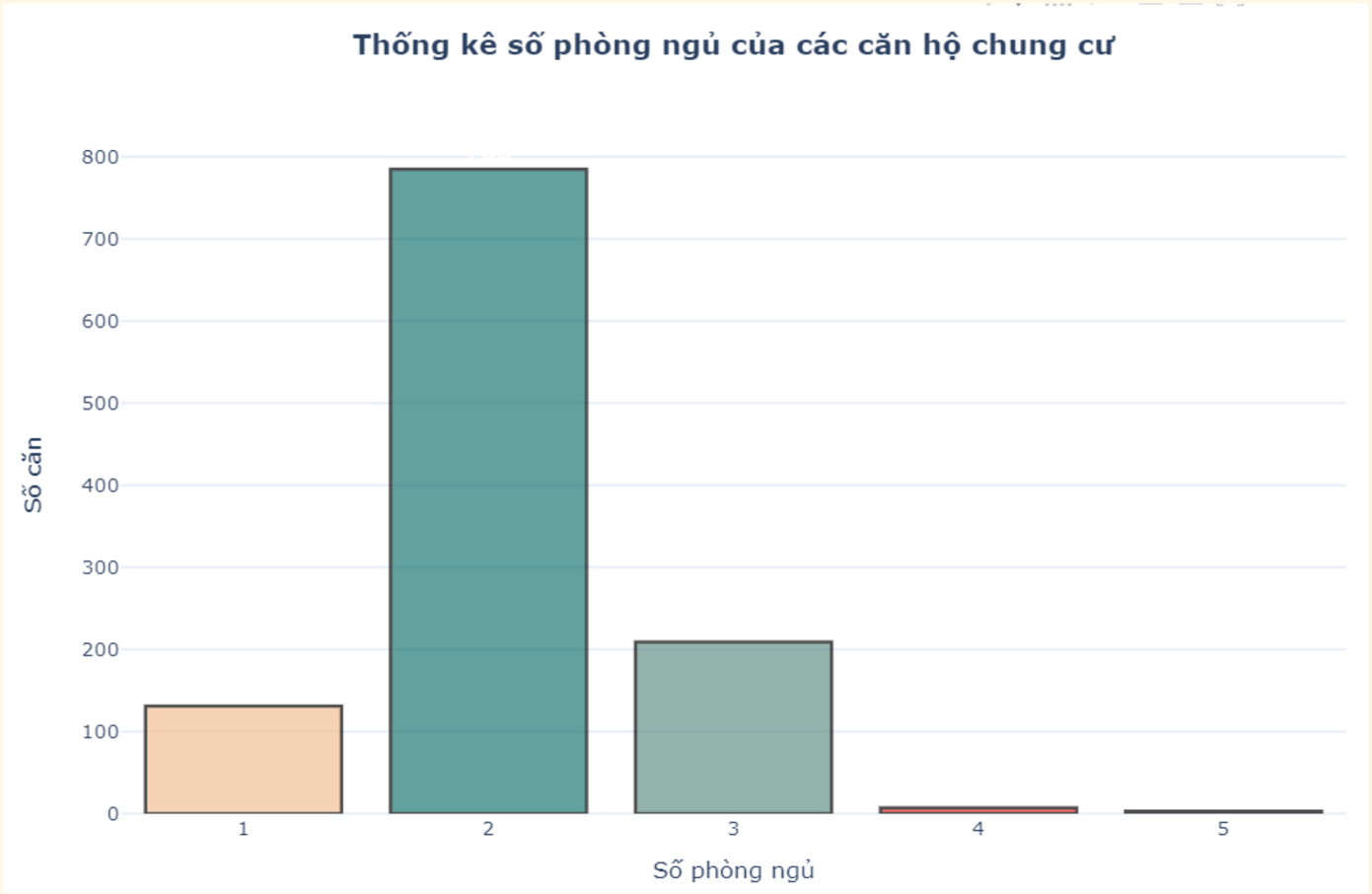
- Sử dụng phương thức “values_counts” để đếm số lượng các căn nhà có cùng số "PhongNgu" và gán vào biến “data”, với:
  - “data.index”: số phòng ngủ
  - “data.values”: số căn

- Tiến hành trực quan hóa bằng barplot (thư viện plotly).


Thống kê số phòng ngủ của các căn hộ chung cư
800
700
600
500
400
300
200
100
0
Số căn
1
2
3
4
5
Số phòng ngủ

CÂU HỎI

Câu 1: Số phòng ngủ phổ biến của các căn hộ chung cư được rao bán là bao nhiêu?



Nhận xét: Đa số các căn hộ được rao bán có số phòng ngủ là 2. Với số lượng phòng ngủ là 2 thì sẽ phù hợp nhất cho các gia đình có từ 2 đến 4 thành viên ở. Và nó cũng phù hợp với xu hướng hình thái của các gia đình hiện nay là gia đình hạt nhân (2 thế hệ).

CÂU HỎI

Câu 4: Phân bố căn hộ dựa trên vị trí và vị trí khu vực nào trong TPHCM sẽ có giá bán căn hộ chung cư cao nhất?

Ý nghĩa khi trả lời được câu hỏi:

- Vị trí dự án luôn là một yếu tố then chốt để đảm bảo giá trị của một bất động sản.
- Khu vực nào trong TPHCM sẽ là vị trí “đắc địa” nhất khi lựa chọn mua chung cư.
- Xác định được những khu vực tiềm năng phát triển để có thể đầu tư sinh lời.

CÂU HỎI

Câu 4: Phân bố căn hộ dựa trên vị trí và vị trí khu vực nào trong TPHCM sẽ có giá bán căn hộ chung cư cao nhất?

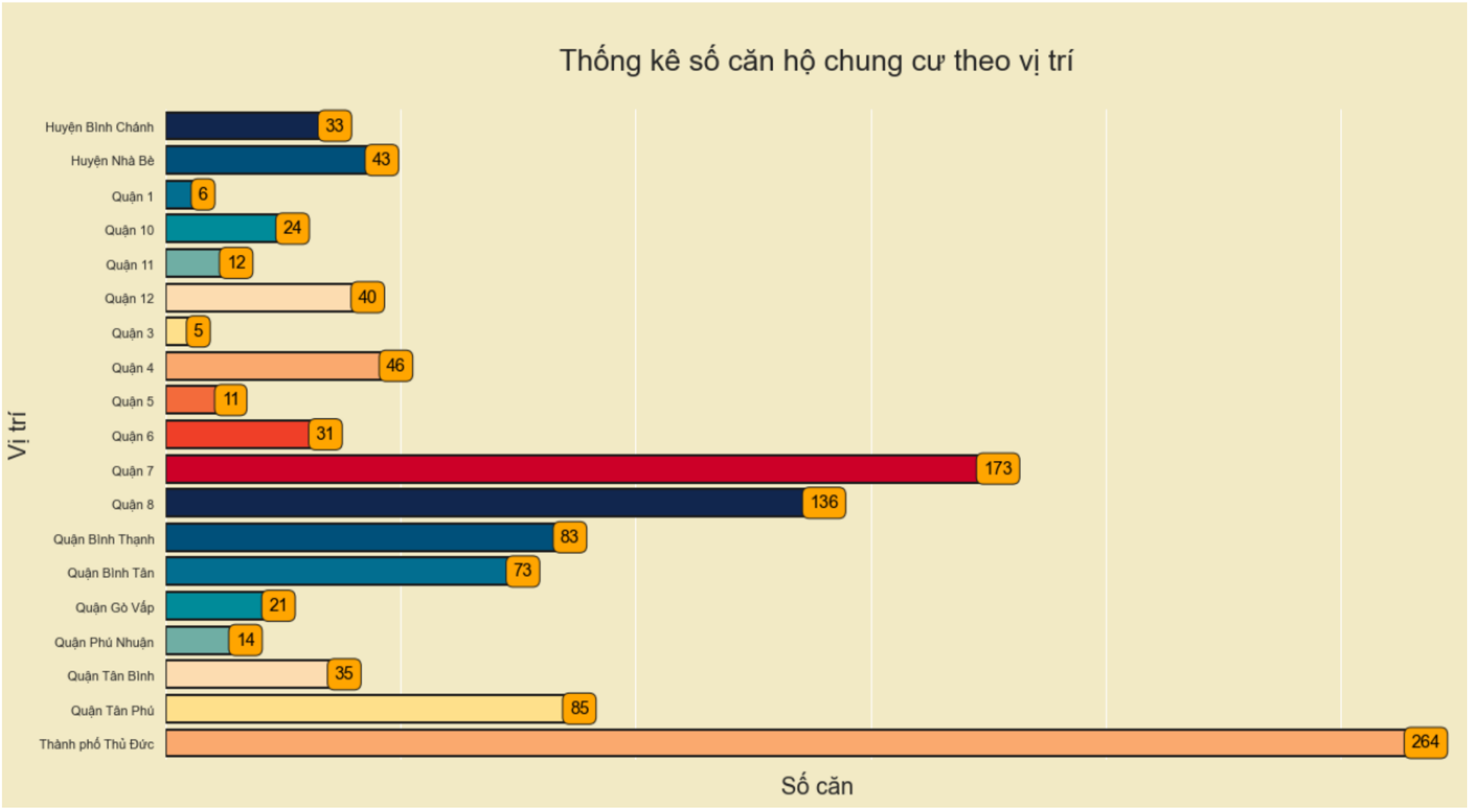
Phân tích dữ liệu để trả lời câu hỏi

Đầu tiên ta sẽ coi xem các căn hộ chung cư được rao bán sẽ đến từ các khu vực nào và có số lượng là bao nhiêu

Sử dụng phương thức “values_counts” để đếm số lượng của các giá trị có trong các cột "Quan" và gán vào các biến “data” với:

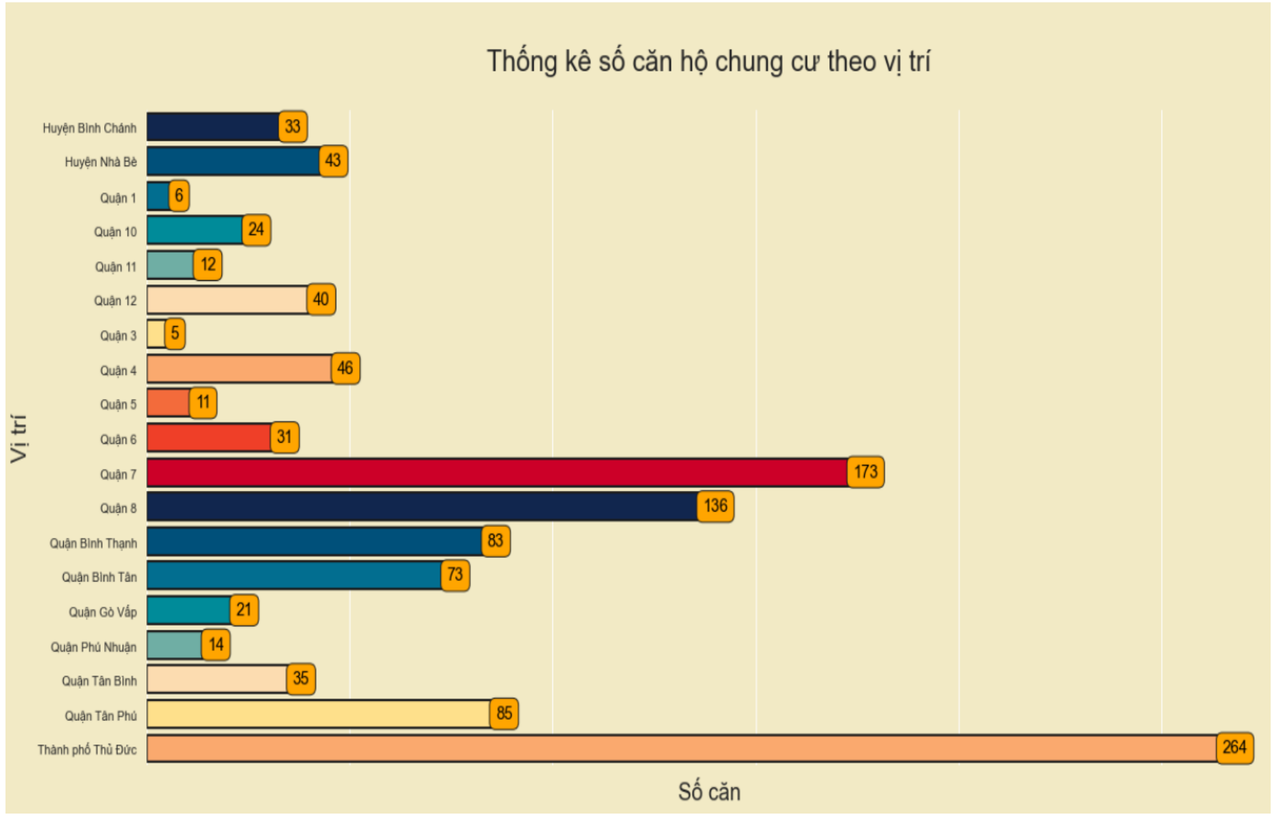
- “data.index”: thông tin quận/thành phố
- “data.values”: số căn chung cư ứng với index

Tiến hành trực quan hóa bằng barplot (thư viện plotly).


Thống kê số căn hộ chung cư theo vị trí
Huyện Bình Chánh
33
Huyện Nhà Bè
43
Quận 1
6
Quận 10
24
Quận 11
12
Quận 12
40
Quận 3
5
Quận 4
46
Quận 5
11
Quận 6
31
Quận 7
173
Quận 8
136
Quận Bình Thạnh
83
Quận Bình Tân
73
Quận Gò Vấp
21
Quận Phú Nhuận
14
Quận Tân Bình
35
Quận Tân Phú
85
Thành phố Thủ Đức
264
Vị trí
Số căn

CÂU HỎI

Câu 4: Phân bố căn hộ dựa trên vị trí và vị trí khu vực nào trong TPHCM sẽ có giá bán căn hộ chung cư cao nhất?



Nhận xét:

- Các căn hộ chung cư được rao bán nhiều nhất đến từ các khu vực: TP Thủ Đức, Quận 7 và Quận 8.
- Quận 1 tuy được xem là nơi có kinh tế phát triển nhất, nhưng lĩnh vực mua bán chung cư lại không được tập trung nhiều ở khu vực này.

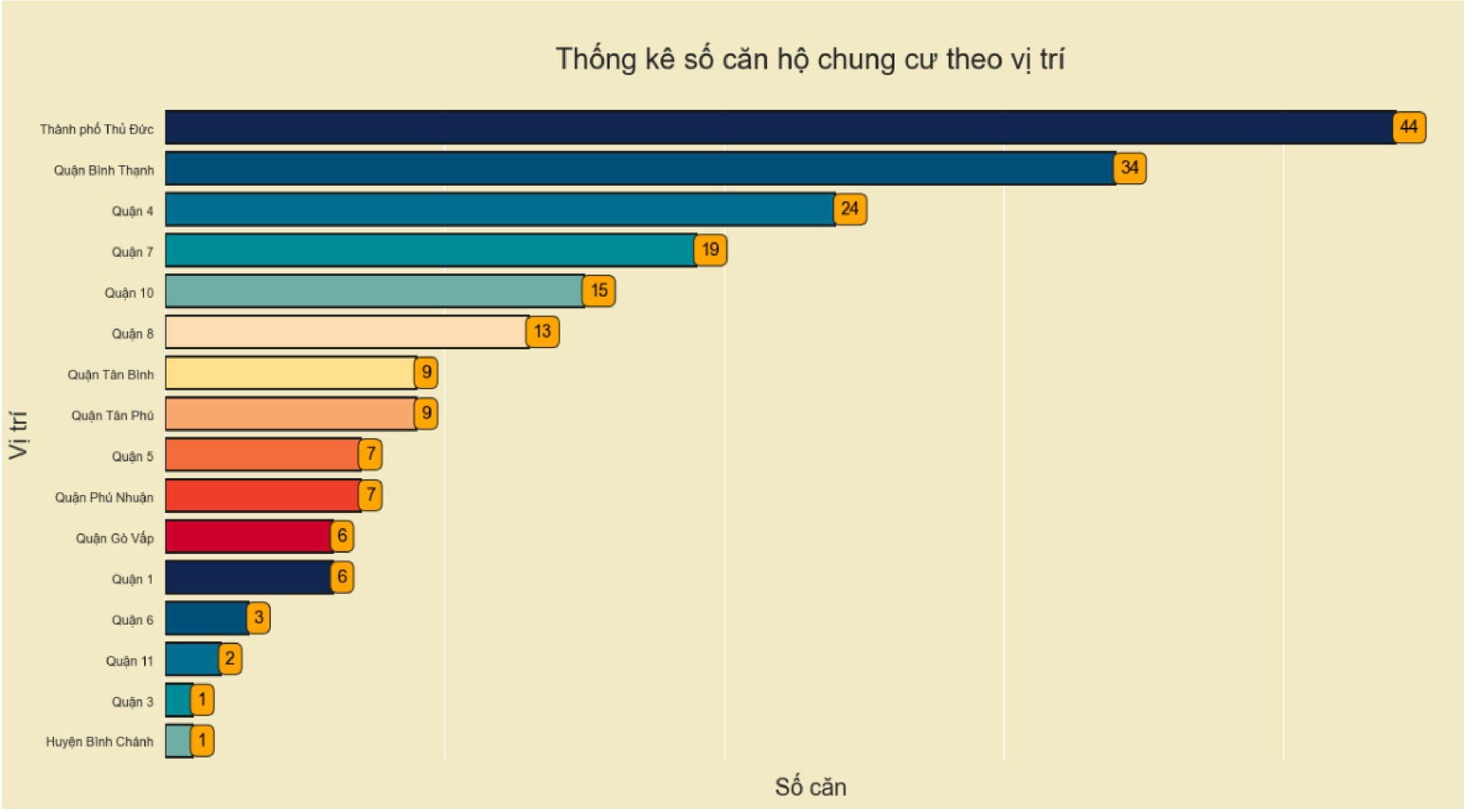
CÂU HỎI

Câu 4: Phân bố căn hộ dựa trên vị trí và vị trí khu vực nào trong TPHCM sẽ có giá bán căn hộ chung cư cao nhất?

Tiếp theo, ta sẽ lấy **200 căn hộ** có giá **cao nhất** trong dữ liệu hiện có (tương đương khoảng **20% dữ liệu**) và phân tích xem các căn hộ này sẽ thuộc về khu vực nào trong TPHCM.

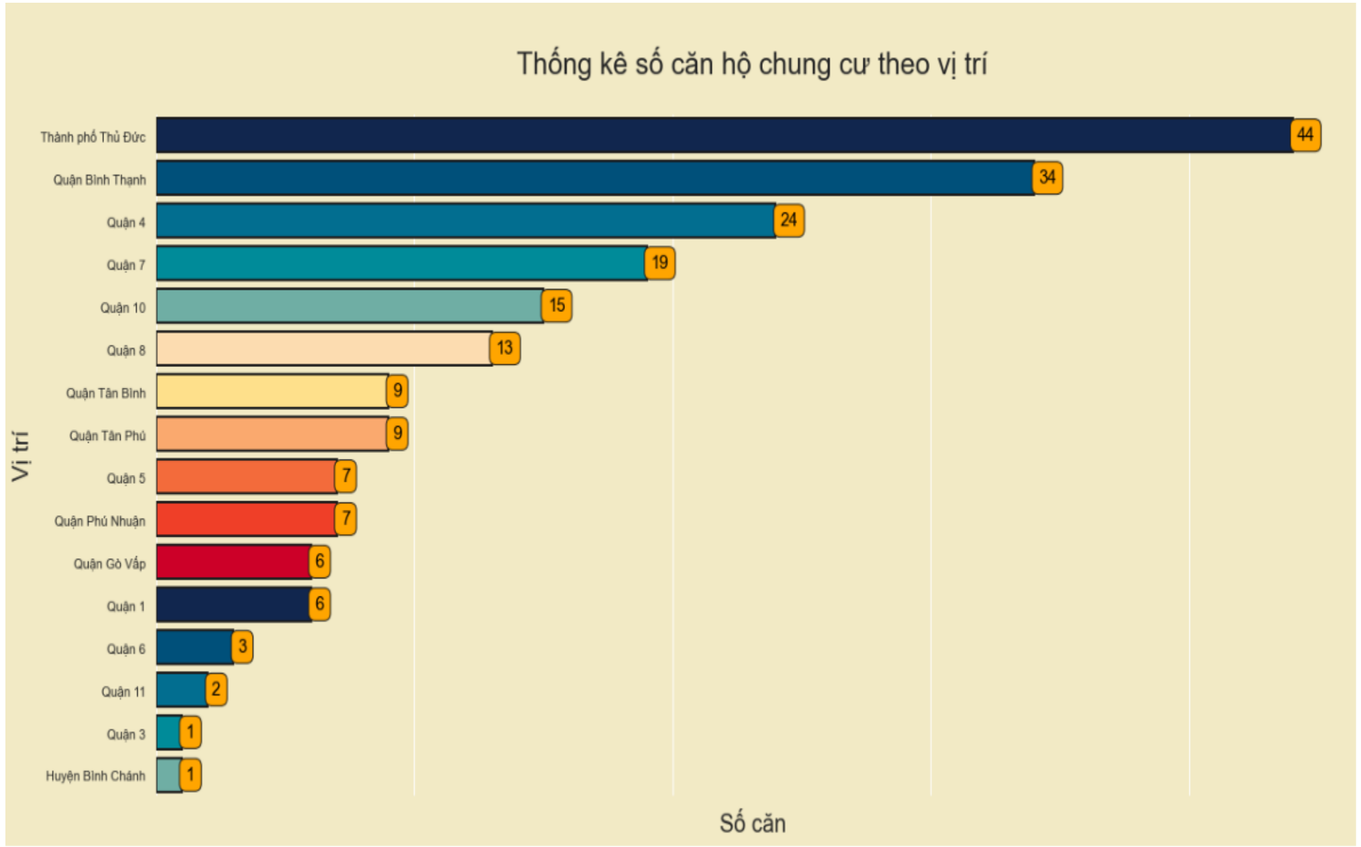
- Sử dụng phương thức “values_counts” để đếm số lượng của các giá trị cũng như lấy các giá trị có trong biến thông qua “index” và lấy số lượng của các giá trị thông qua “values”.

- Sau khi có dữ liệu ta sẽ tiến hành vẽ biểu đồ tượng tự như trên.


Thống kê số căn hộ chung cư theo vị trí
Thành phố Thủ Đức
Quận Bình Thạnh
Quận 4
Quận 7
Quận 10
Quận 8
Quận Tân Bình
Quận Tân Phú
Quận 5
Quận Phú Nhuận
Quận Gò Vấp
Quận 1
Quận 6
Quận 11
Quận 3
Huyện Bình Chánh
44
34
24
19
15
13
9
9
7
7
6
6
3
2
1
1
Vị trí
Số căn

CÂU HỎI

Câu 4: Phân bố căn hộ dựa trên vị trí và vị trí khu vực nào trong TPHCM sẽ có giá bán căn hộ chung cư cao nhất?



Nhận xét:

Các căn hộ có giá cao tập trung nhiều ở các khu vực: TP Thủ Đức, Quận Bình Thạnh và Quận 4. Đây cũng là những khu vực đang vô cùng phát triển của TPHCM.

IV. Mô hình hóa dữ liệu

- Bài toán: Dự đoán giá bán căn hộ chung cư ở TP.HCM.
- Sử dụng thuật toán Hồi quy tuyến tính.
- 2 bước:
  - Tiền xử lí dữ liệu.
  - Xây dựng mô hình học máy.

IV. Mô hình hóa dữ liệu

Bài toán: Dự đoán giá bán căn hộ chung cư ở TP.HCM.

* Giới thiệu chung:

- Trong học máy, học có giám sát là một nhóm các thuật toán phổ biến trong lĩnh vực này và một trong những vấn đề quan trọng của học có giám sát là hồi quy (regression). Hồi quy là các bài toán liên quan đến việc dự đoán đầu ra có giá trị liên tục (predicting continous valued output).

- Và trong bài toán mà nhóm đề ra thì từ những cột thuộc tính đầu vào như **diện tích căn hộ, số phòng ngủ, số phòng vệ sinh,....** Nhóm sẽ **dự đoán cột mục tiêu là giá bán của chung cư** bằng thuật toán hồi quy tuyến tính (linear regression).

IV. Mô hình hóa dữ liệu

A. Tiền xử lí dữ liệu

1. Loại những thuộc tính không có ý nghĩa cho việc mô hình hóa.
2. Chuyển đổi các cột không phải dạng số về dạng số.
3. Tính Correlations và tiếp tục chọn những thuộc tính thật sự có giá trị.
4. Xử lý các giá trị NaN.

IV.A. Tiền xử lí dữ liệu

1. **Loại những thuộc tính không có ý nghĩa cho việc MHH**

IV.A. Tiền xử lí dữ liệu

1. Loại những thuộc tính không có ý nghĩa cho việc MHH

- Dựa vào ý nghĩa liên quan tới bài toán, loại bỏ:

DiaChi DuAn Gia/m2 DacDiem

- Giữ lại:

GiaBan DienTich TinhTrangBDS PhongNgu PhongVeSinh

TinhTrangGiayTo TinhTrangNoiThat HuongBanCong

HuongCua Quan

Diễn giải:

- Cột ‘DiaChi’, ‘DuAn’ có các giá trị quá riêng biệt.
- Cột ‘Gia/m2’ tính bằng cách 'GiaBan'/'DienTich’.
- Cột ‘DacDiem’: là nan hoặc 'căn góc’.

⇒ Không có ý nghĩa cho việc phân tích

⇒ Loại bỏ

	GiaBan	DienTich	TinhTrangBDS	PhongNgu	PhongVeSinh	TinhTrangGiayTo	TinhTrangNoiThat	HuongBanCong	HuongCua	Quan
0	2.35	66.0	Chưa bàn giao	2.0	2.0	Đã có sổ	Nội thất cao cấp	Tây Bắc	Đông Bắc	Thành phố Thủ Đức
1	3.42	71.0	Đã bàn giao	2.0	2.0	Đang chờ sổ	Nội thất cao cấp	NaN	NaN	Quận Tân Phú

IV.A. Tiền xử lí dữ liệu

2. Chuyển đổi sang các cột dạng số

IV.A. Tiền xử lí dữ liệu

2. Chuyển đổi sang các cột dạng số

- Đảm bảo sao cho các giá trị vẫn có ý nghĩa phân loại.
- Dữ liệu đầu vào cho các mô hình học máy đều phải ở dang số. Mỗi số đại diện cho một giá trị riêng biệt ban đầu trong miền giá trị của thuộc tính đó.

Sử dụng:

- import OrdinalEncoder của thư viện **sklearn**.
- hàm fit_transform: fit to data, then transform it.

```
from sklearn.preprocessing import OrdinalEncoder
enc = OrdinalEncoder()
new_df[['TinhTrangBDS', 'TinhTrangGiayTo', 'TinhTrangNoiThat', 'HuongBanCong',
       'HuongCua', 'Quan']]=enc.fit_transform(np.array(new_df[['TinhTrangBDS', 'TinhTrangGiayTo',
                                                               'TinhTrangNoiThat', 'HuongBanCong',
                                                               'HuongCua', 'Quan']]))
```

	GiaBan	DienTich	TinhTrangBDS	PhongNgu	PhongVeSinh	TinhTrangGiayTo	TinhTrangNoiThat	HuongBanCong	HuongCua	Quan
0	2.35	66.0	0.0	2.0	2.0	2.0	2.0	3.0	6.0	18.0
1	3.42	71.0	1.0	2.0	2.0	1.0	2.0	NaN	NaN	17.0
2	2.40	56.0	1.0	2.0	NaN	NaN	NaN	NaN	NaN	13.0

IV.A. Tiền xử lí dữ liệu

3. Tính Correlations và tiếp tục chọn các thuộc tính thật sự có giá trị.

IV.A. Tiền xử lí dữ liệu

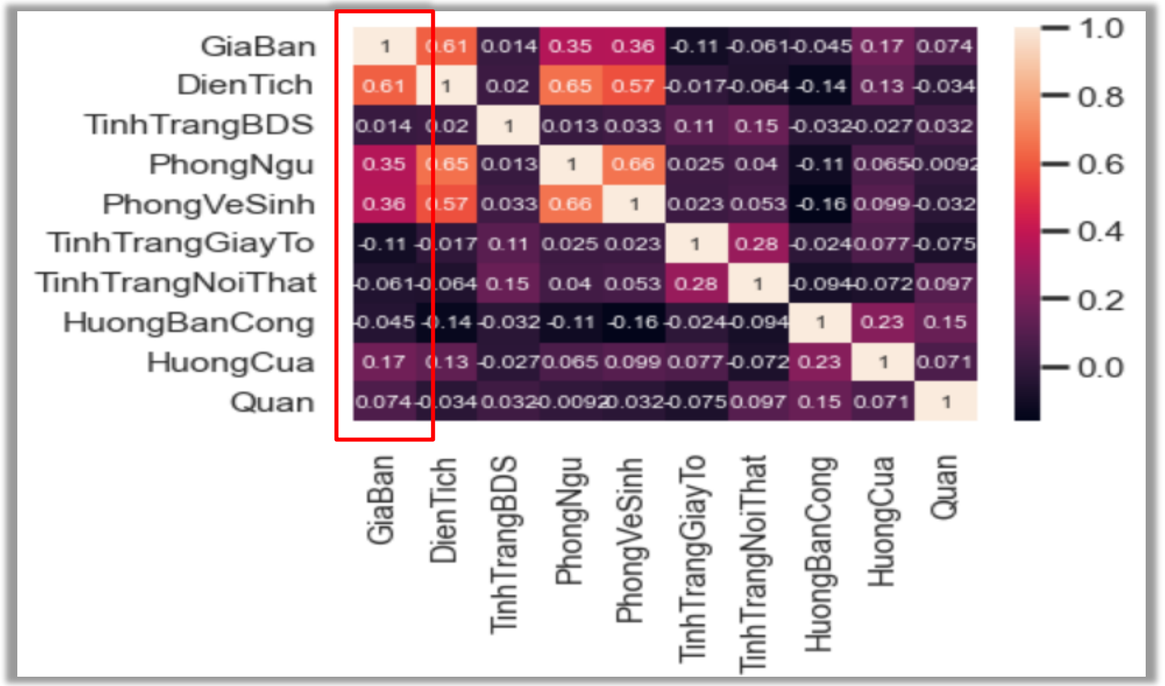
3. Tính Correlations và tiếp tục chọn các thuộc tính thật sự có giá trị.

💡 **Sử dụng:**

- Correlation: đề cập mức độ liên quan của 2 biến dựa trên mối quan hệ tuyến tính của chúng.
- Nhóm tạo một dataframe chứa các correlations của từng cột trong bộ dữ liệu.

⇒ Nhận xét mức độ tương quan giữa các biến.

⇒ Trực quan bằng heatmap.



	GiaBan	DienTich	PhongNgu	PhongVeSinh
0	2.35	66.0	2.0	2.0
1	3.42	71.0	2.0	2.0
2	2.40	56.0	2.0	NaN
3	7.50	107.0	3.0	NaN
4	3.10	75.0	2.0	2.0

→ Các thuộc tính được giữ lại: DienTich, PhongNgu, PhongVeSinh.

→ Với cột biến mục tiêu: GiaBan.

IV.A. Tiền xử lí dữ liệu

4. Xử lí các giá trị NaN

IV.A. Tiền xử lí dữ liệu

4. Xử lí các giá trị NaN:

- Vấn đề: mỗi cột có số lượng giá trị thiếu khá nhiều, do đó việc bỏ đi các dòng chứa giá trị NaN có thể gây ảnh hưởng lớn đến tính chính xác khi tiến hành học máy.
- Giải pháp: thay thế NaN bằng các giá trị đặc biệt của cột dữ liệu (như median, mean, most, ...)

💡 Sử dụng lớp impute của thư viện sklearn để fill những giá trị NaN.

```
#Quan sát số lượng null ở các thuộc tính.
new_df.isnull().sum()
```

```
GiaBan           0
DienTich         0
PhongNgu         0
PhongVeSinh    189
dtype: int64
```

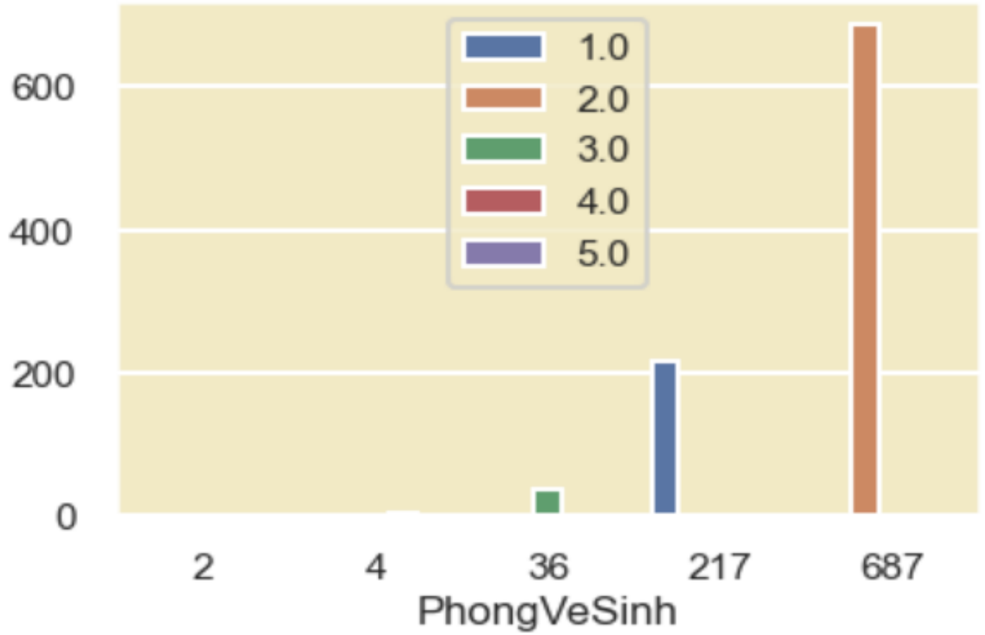
→ Chỉ cần xử lý giá trị NaN cho cột PhongVeSinh

IV.A. Tiền xử lí dữ liệu

4. Xử lí các giá trị NaN:

🗺️ **Thống kê phân bố của 'PhongVeSinh'.**

```
veSinh=new_df['PhongVeSinh'].value_counts()
sns.barplot(veSinh,veSinh.values,veSinh.index);
```



Nhóm sẽ sử dụng giá trị **most_frequent** để thay thế các giá trị NaN cho thuộc tính 'PhongVeSinh'.

😊 Bây giờ thì đầu vào đã được tiền xử lý xong để phù hợp hơn cho việc áp dụng mô hình học máy.

IV. Mô hình hóa dữ liệu

B. Xây dựng mô hình học máy

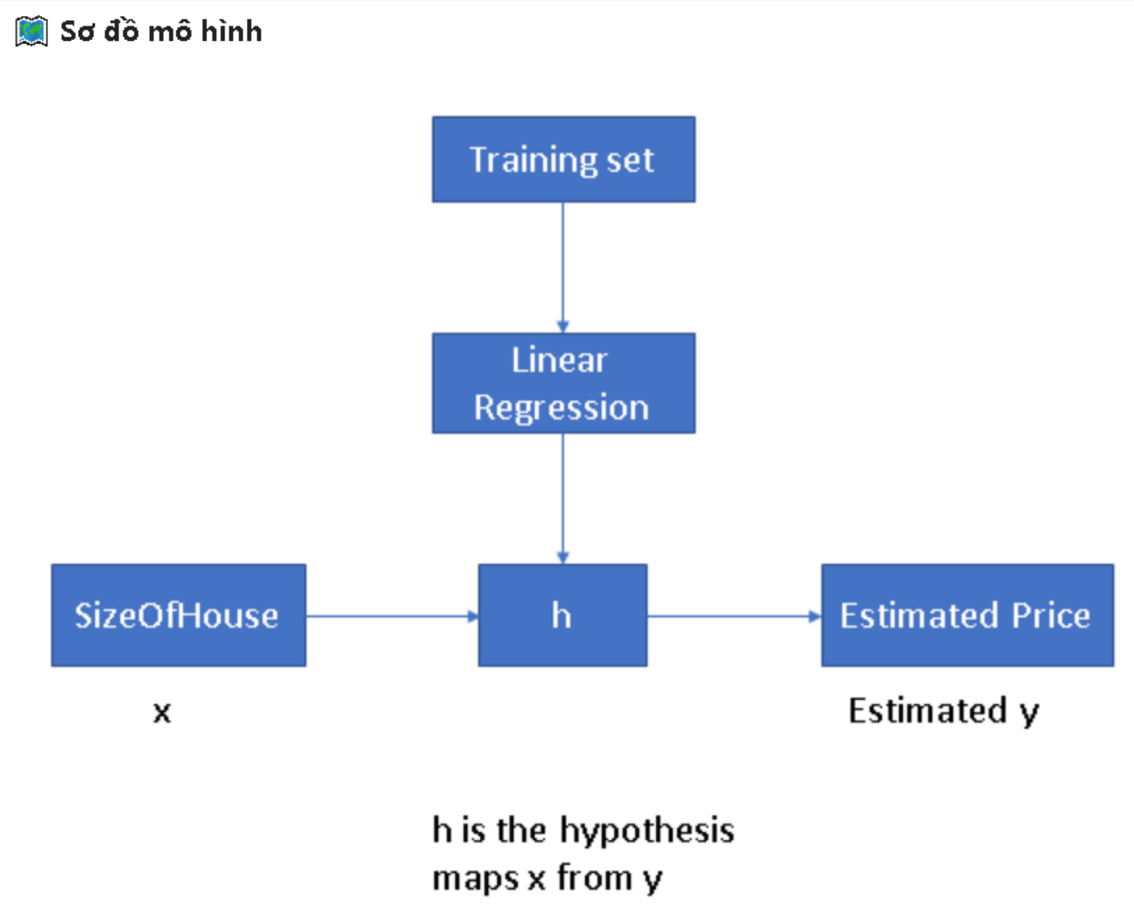
1. Hồi quy tuyến tính đơn biến (Univariable Linear Regression).
2. Hồi quy tuyến tính đa biến (Linear Regression with Multi Variables).
3. Đánh giá mô hình (Evaluation).
4. Hồi quy đa thức (Polynomial Regression) đơn giản với đơn biến.
5. Tổng kết.

IV.B. Xây dựng mô hình học máy

1. Hồi quy tuyến tính đơn biến

IV.B. Xây dựng mô hình học máy

1. Hồi quy tuyến tính đơn biến



- Mục tiêu của hồi quy đơn biến: dự đoán giá trị của một biến phụ thuộc dựa vào một biến độc lập (thuộc tính).

Một vài ký hiệu

- X: tập thuộc tính đầu vào.
- y: tập giá trị đầu ra.
- $(x^{(i)}, y^{(i)})$: mẫu huấn luyện thứ i trong bộ dữ liệu

IV.B. Xây dựng mô hình học máy

1. Hồi quy tuyến tính đơn biến

🗺 Cost Function

- Ta có hypothesis là h_θ(x): $\theta_0 + \theta_1$x với θ_i (i=0,1) là các tham số (parameters) của công thức hồi quy và θ_0 còn đươc gọi là hệ số tự do.
- Tìm các θ_i để tạo ra đường thẳng phù hợp với dữ liệu (tạo ra h_θ(x) càng gần y càng tốt).
- Để đánh giá xem hypothesis đã tốt hay chưa chúng ta xây dựng một hàm lỗi bình phương sum square error để tính toán độ sai lệch giữa giá trị dự đoán và giá trị thực tế.

$$J(\theta_0, \theta_1) = J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Lúc này ta gọi $J(\theta)$ là 'Cost Function' với ý nghĩa là hàm chi phí đánh giá hiệu suất của hypothesis từ các tham số θ.

Do càng gần y càng tốt nên giá trị của Cost Function càng nhỏ sẽ càng tốt.

IV.B. Xây dựng mô hình học máy

1. Hồi quy tuyến tính đơn biến

🗺 Cost Function

$$J(\theta_0, \theta_1) = J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Lúc này ta gọi $J(\theta)$ là 'Cost Function' với ý nghĩa là hàm chi phí đánh giá hiệu suất của hypothesis từ các tham số θ.

🤔 Vậy làm thế nào để chọn được theta tốt cho hypothesis:

- Đầu tiên, khởi tạo theta với $\theta_0 = \theta_1 = 0$ hoặc giá trị random.
- Phần lớn trường hợp Cost Function ban đầu sẽ có giá trị rất lớn do đó để giảm giá trị hàm chi phí thì cần có các giải pháp để tìm ra vector tham số theta phù hợp nhất cho hypothesis.

$\Rightarrow$ Trong bài làm của nhóm sẽ sử dụng 2 cách:

- Thuật toán Gradient Descent
- Phương pháp Normal Equation.

IV.B. Xây dựng mô hình học máy

1. Hồi quy tuyến tính đơn biến

🗺 Gradient Descent

- Với hồi quy tuyền tính thì mục tiêu là cực tiểu hàm chi phí

$$J(\theta_0, \theta_1) = J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

với $h_\theta(x)$ là đường thẳng tuyến tính.

- Các tham số θ_j sẽ là những giá trị mà chúng ta phải thay đổi để có thể tối ưu hóa Cost Function. Và trong thuật toán Gradient Descent cụ thể làm như sau:

Trong mỗi lần lặp cập nhật một cách đồng thời các tham số θ_j theo công thức

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

Trong đó: alpha là 'learning rate' giúp việc học được tối ưu hơn.

IV.B. Xây dựng mô hình học máy

1. Hồi quy tuyến tính đơn biến

🗺 Vectorization

- Ký hiệu:
  - θ: là vector tham số $[\theta_0, \theta_1]$
  - X là ma trận tất cả các mẫu đầu vào: $[[1, x^1] [1, x^2] [...] [1, x^m]]$ với m là kích thước bộ dữ liệu.
- 1 là được thêm vào mỗi đầu vào được gọi là bias.
- Hypothesis: h_θ=X.θ

IV.B. Xây dựng mô hình học máy

1. Hồi quy tuyến tính đơn biến

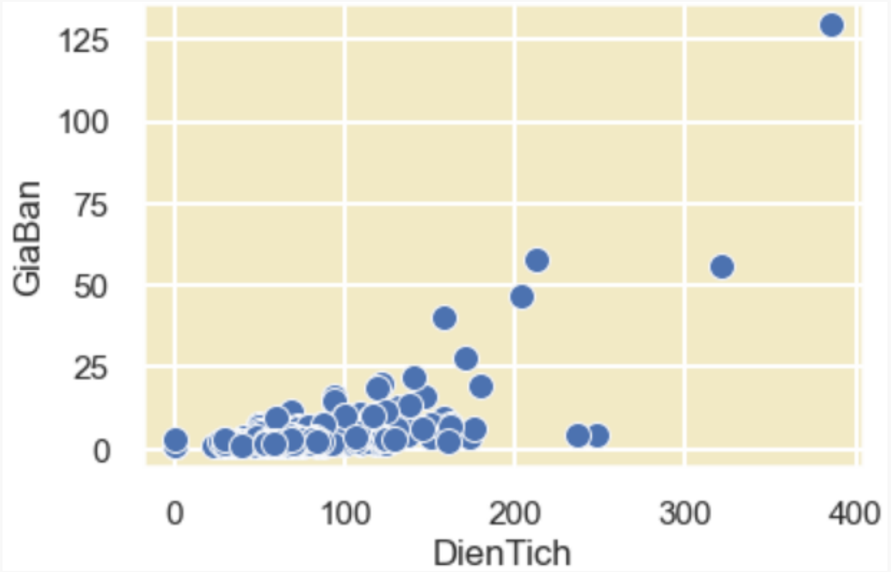
🛠 Cài đặt

Trực quan mối quan hệ giữa 2 biến ‘GiaBan’ và ‘DienTich’

👉 Chưa thấy rõ mối quan hệ có thể do chênh lệch giá trị giữa 2 thuộc tính. Có thể cần feature scaling về sau.

👉 Loại các outlier có diện tích lớn hơn 200.

👉 Việc tiếp theo là lấy ra cột GiaBan và DienTich để tạo thành bộ dữ liệu cho hồi quy tuyến tính đơn biến.




```
data=new_df[:,[0,1]]
data
```

```
array([[ 2.35, 66.  ],
       [ 3.42, 71.  ],
       [ 2.4 , 56.  ],
       ...,
       [ 2.2 , 83.5 ],
       [ 3.35, 69.  ],
       [ 1.68, 58.  ]])
```

```
X=data[:,1]
y=data[:,0]
```

```
#add bias unit.
X = np.stack([np.ones(len(X)), X], axis=1)
```

```
print(len(X), len(y))
```

```
1129 1129
```

IV.B. Xây dựng mô hình học máy

1. Hồi quy tuyến tính đơn biến

🛠 Cài đặt

👉 Bước tiếp theo là **Phân tách bộ dữ liệu thành 2 tập training set và test set.**

👉 Sử dụng thư viện sklearn để hỗ trợ việc split data này.

- Kích thước mỗi tập như sau:
  - Size of Training set = 80% * (Size of Dataset).
  - Size of Test set = 20% * (Size of Dataset).

```
print(len(X_train),len(y_train))
```

```
903 903
```

```
from sklearn.model_selection import train_test_split
```

```
np.random.seed(13)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

```
print(len(X_test),len(y_test))
```

```
226 226
```

IV.B. Xây dựng mô hình học máy

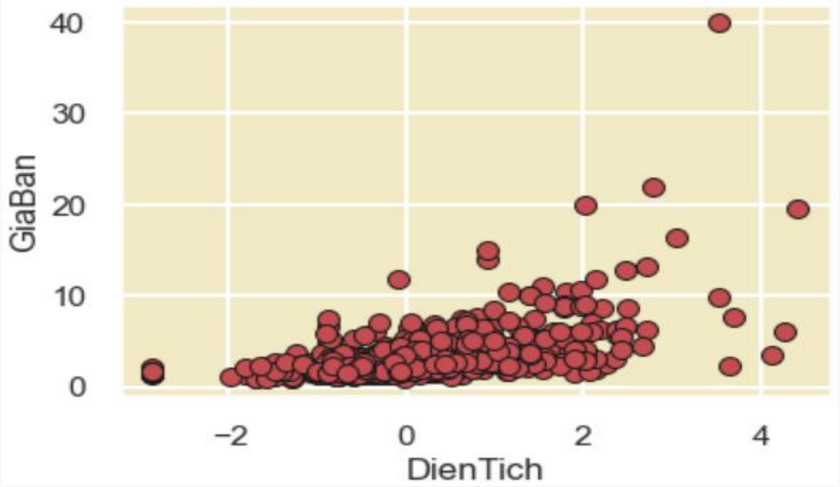
1. Hồi quy tuyến tính đơn biến

🗺 Feature Scaling

Khi khoảng giá trị giữa 2 thuộc tính quá cách xa nhau thì việc mô hình hóa cũng như trực quan mối quan hệ có thể gặp khó khăn, do đó phải thực hiện kĩ thuật 'Feature Scaling' hay việt hóa là 'Co giãn thuộc tính'.
Có 3 phương pháp feature scaling chính là: Standardisation(Chính quy hóa), Normalisation(Tiêu chuẩn hóa), MinMax Scaler.
Trong bài này nhóm chọn phương pháp **Standardisation** để scaling khoảng giá trị của thuộc tính về khoảng gần hơn với giá trị của tập y là 'GiaBan'. Và chỉ thực hiện scaling trên cột 'DienTich' mà không thực hiện trên cột bias unit.

```
X_train[:5,:]
```

```
array([[ 1.        , -1.25373557],
       [ 1.        ,  0.67157469],
       [ 1.        ,  1.28343176],
       [ 1.        , -0.59292993],
       [ 1.        ,  1.12026987]])
```



IV.B. Xây dựng mô hình học máy

1. Hồi quy tuyến tính đơn biến

🛠 **Implementation Cost Function**

```
def computeCostFunction(X, y, theta):
    m = y.size  # number of training examples

    # init the cost function's value is equal to 0.
    J = 0

    # compute cost function J.
    J=(1/(2*m))*np.sum(np.square(np.dot(X,theta.reshape(-1,1))-y.reshape(-1,1)),axis=0)

    return J
```

🛠 **Implement Gradient Descent**

```
def gradientDescent(X, y, theta, alpha, num_iters):
    m = y.shape[0]  # number of training examples

    theta = theta.copy()

    J_records = [] # to track value of J throw each iteration.

    for i in range(num_iters):
        temp=theta.copy()
        for j in range(len(theta)):
            theta[j]=temp[j]-alpha*(1/m)*np.sum((np.dot(X,temp.reshape(-1,1))-y.reshape(-1,1)).squeeze()*X[:,j])

        # save the cost J in every iteration
        J_records.append(computeCostFunction(X, y, theta))

    return theta, J_records
```

IV.B. Xây dựng mô hình học máy

1. Hồi quy tuyến tính đơn biến

🛠 Sau khi cài đặt các hàm, tiến hành chạy thuật toán gradient descent.

```
# initialize fitting parameters
theta = np.random.rand(2)

# some gradient descent settings
iterations = 1000
alpha = 0.01 #learning rate

theta, J_history = gradientDescent(X_train ,y_train, theta, alpha, iterations)
```

Khởi tạo random các tham số theta

Khởi tạo số lần lặp.

Khởi tạo alpha

🛠 Kết quả thu được sau khi chạy thuật toán.

```
theta

array([3.07176983, 1.31468626])
```

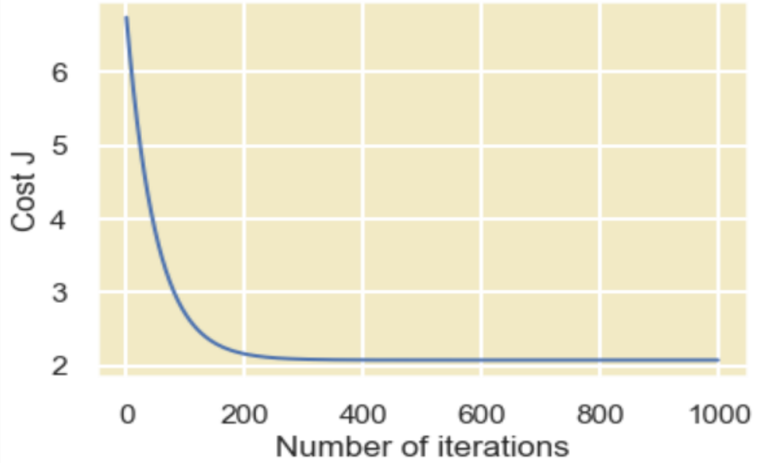
θ_1

θ_0

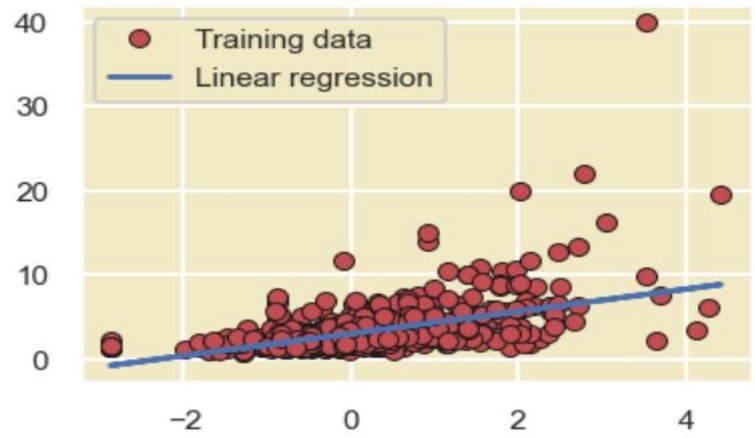
IV.B. Xây dựng mô hình học máy

1. Hồi quy tuyến tính đơn biến

🛠 Kết quả thu được sau khi chạy thuật toán.



Vẽ biểu đồ các giá trị của Cost Function sau mỗi lần thực hiện gradient descent.



Vẽ đường thẳng tuyến tính tốt nhất với tập dữ liệu.

IV.B. Xây dựng mô hình học máy

1. Hồi quy tuyến tính đơn biến

🛠 Tính Cost Function trên tập test.

```
computeCostFunction(X_test, y_test, theta)
```

```
array([2.62837948])
```

🛠 So sánh giá trị dự đoán của các hàm tự cài đặt và của các hàm có sẵn của thư viện sklearn.

	Predictions by Implementation	Predictions by Sklearn
0	2.453133	2.453246
1	2.185000	2.185106
2	2.882146	2.882270
3	5.000400	5.000578
4	2.882146	2.882270
...	...	...
221	4.330067	4.330228
222	2.667640	2.667758
223	2.185000	2.185106
224	3.847427	3.847575
225	3.525666	3.525807

IV.B. Xây dựng mô hình học máy

1. Hồi quy tuyến tính đơn biến

👉 Như đã đề cập ở trên, nhóm sử dụng 2 cách để tìm ra theta tốt nhất cho hypothesis. Một là sử dụng Gradient Descent như đã đề cập ở trên, và cách thứ hai tiếp theo đây là sử dụng Normal Equation (Biểu thức chính quy).

🗺 Normal Equation

Vector tham số θ sẽ được tính theo công thức sau:

$$\theta = \left(X^T X\right)^{-1} X^T \vec{y}$$

với X và y lần lượt là ma trận đầu vào và vector của biến phụ thuộc như đã được đề cập ở mục **Vectorization** ở trên.

```
def normalEquation(X, y):
    theta = np.zeros(X.shape[1])

    theta=np.dot(np.dot(np.linalg.inv(np.dot(X.T,X)),X.T),y.reshape(-1,1))
    theta=theta.squeeze()

    return theta
```

```
theta = normalEquation(X_train, y_train)
theta
```

```
array([3.07189877, 1.31471994])
```

👉 Nhận thấy sai số là rất ít giữa 2 phương pháp.

IV.B. Xây dựng mô hình học máy

1. Hồi quy tuyến tính đơn biến

So sánh giữa Gradient Descent và Normal Equation

Gradient Descent	Normal Equation
Cần lựa chọn α phù hợp	Không cần lựa chọn α
Cần nhiều vòng lặp	Không cần nhiều vòng lặp
Làm việc tốt thậm chí với tập dữ liệu có kích thước rất lớn	Rất chậm với tập dữ liệu có kích thước rất lớn do cần tính $(X^T X)^{-1}$

IV.B. Xây dựng mô hình học máy

2. Hồi quy tuyến tính đa biến

IV.B. Xây dựng mô hình học máy

2. Hồi quy tuyến tính đa biến

💬 Nhận xét

- Trong một tập dữ liệu với rất nhiều thuộc tính (biến) đầu vào và một biến đầu ra

  => Không dễ lựa chọn ra một thuộc tính để xây dựng hypothesis cho đường hồi quy tuyến tính.
- Vd bài toán **dự đoán giá bán xe ô tô,** dựa vào các thuộc tính khác như dung tich xilanh, số cửa, động cơ,...

  => khó khăn hơn nhiều do mỗi một thuộc tính đều đóng góp một ít tầm ảnh hưởng vào giá bán của xe ô tô đó.

  → Giải pháp: **lựa chọn tất cả** để xây dựng hypothesis cho mô hình hồi quy.

💬 Vấn đề gặp phải

Khi dùng nhiều biến để xây dựng hypothesis thì hypothesis sẽ rất phức tạp, chẳng hạn như một đầu vào x với n thuộc tính ($x_1, x_2, \ldots, x_n$) được chọn để xây dựng hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

. Điều này sẽ có thể tạo nên một đường không phải dạng đường thẳng mà nó là một dạng đường ngoằn ngòe phức tạp nào đó, có thể khớp rất tốt với tập dữ liệu huấn luyện nhưng khi áp dụng vào thực tế thì không, đây được gọi là vấn đề over fitting.

Ngoài ra còn có trường hợp **đường tuyến tính không còn phù hợp** và phải sử dụng hồi quy đa thức.

IV.B. Xây dựng mô hình học máy

2. Hồi quy tuyến tính đa biến

💬 **Quay trở lại với vấn đề một hypothesis dạng tuyến tính quá phức tạp:**

→ Một trong những cách được đề xuất để khắc phục vấn đề này là **Regularization** ('Chính quy hóa').

🗺 Regularization

- Ý tưởng là: thêm vào cost function một đại lượng nữa là **tổng các tích của một hằng số λ với θ_j với j=[1,n] nếu x có n thuộc tính.**
- Đại lượng này tác động đến Cost Function như sau:

$$J(\theta) = \frac{1}{2m}[\sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \left(\theta_j\right)^2]$$

  - Cùng nhớ lại mục tiêu của hồi quy tuyến tính là minimize hàm Cost Function do đó khi thêm một đại lượng là $\lambda \sum_{j=1}^{n} \left(\theta_j\right)^2$ thì để cho có thể minimize được Cost Function bắt buộc các tham số θ_j phải nhỏ và ta gọi việc 'ép buộc' này là phạt (penalize) các tham số θ.
  - Tuy nhiên nhìn vào công thức của đại lượng được thêm vào bạn chắc chắn sẽ chú ý đến hằng số λ, vậy việc chọn λ ảnh hưởng như nào đến quá trình Regularization.
  - Nếu chúng ta thiết lập λ rất nhỏ thì việc thực hiện regularization sẽ không còn ý nghĩa nữa, còn nếu như thiết lập λ rất lớn thì việc penalize các tham số θ là rất nặng dẫn đến θ_j (j=[1, n]) sẽ xấp xỉ 0 và lúc này hypothesis trở thành một hàm hằng $h_\theta(x) = \theta_0$ dẫn đến hiện tượng Underfitting.
  - Để dễ dàng hơn nhóm sẽ chọn λ=1 cho quá trình này.

IV.B. Xây dựng mô hình học máy

2. Hồi quy tuyến tính đa biến

🗺 Gradient Descent cho Regularization trên hồi quy tuyến tính

$$\theta_0 = \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_0^{(i)}$$

$$\theta_j = \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$ với j=[1, n] nếu x có n thuộc tính.

Tất nhiên việc cập nhật các θ_j cũng phải diễn ra một cách đồng thời.

🗺 Normal Equation với Regularization

$$\theta = \left(X^T X + \lambda A\right)^{-1} X^T \vec{y}$$

với A là ma trận chứa các giá trị 0 và chỉ có các giá trị trên đường chéo chính có giá trị là 1 nhưng trừ phần tử ở vị trí [0, 0] cũng bằng 0.

IV.B. Xây dựng mô hình học máy

2. Hồi quy tuyến tính đa biến

🛠 Cài đặt

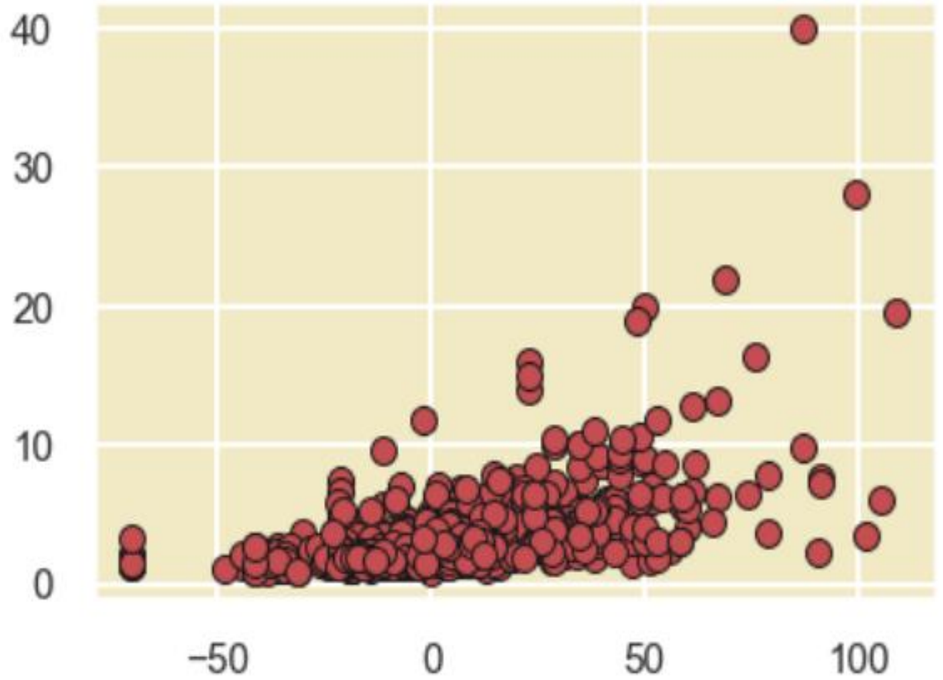
👉 Việc đầu tiên cũng sẽ là lấy ra thuộc tính và biến đầu ra từ tập dữ liệu

```
data2=new_df[:,:]
data2
```

```
array([[ 2.35, 66.  ,  2.  ,  2.  ],
       [ 3.42, 71.  ,  2.  ,  2.  ],
       [ 2.4 , 56.  ,  2.  ,  2.  ],
       ...,
       [ 2.2 , 83.5 ,  2.  ,  2.  ],
       [ 3.35, 69.  ,  2.  ,  2.  ],
       [ 1.68, 58.  ,  2.  ,  2.  ]])
```

```
X2=data2[:,1:]
y2=data[:,0]
```

🗺 PCA giảm chiều dữ liệu



IV.B. Xây dựng mô hình học máy

2. Hồi quy tuyến tính đa biến

🛠 Cài đặt

👉 Các bước tiếp theo vẫn là **add bias unit, split data, và feature scaling.**

```
#add bias unit
X2 = np.concatenate([np.ones((len(X2), 1)), X2], axis=1)
```

```
#train_test_split
np.random.seed(13)
X2_train,X2_test,y2_train,y2_test=train_test_split(X2,y2,test_size=0.2)
```

```
#feature scaling for 'DienTich' attributes.
sc2=StandardScaler()
X2_train[:,[1]]=sc2.fit_transform(X2_train[:,[1]])
```

IV.B. Xây dựng mô hình học máy

2. Hồi quy tuyến tính đa biến

🛠 Implementation Cost Function with Regularization

```
def computeCostFunctionReg(X, y, theta, lambda_):
    m = y.size  # number of training examples

    # init the cost function's value is equal to 0.
    J = 0

    # compute cost function J.
    theta_to_compute=theta[1:]
    J=(1/(2*m))*(np.sum(np.square(np.dot(X,theta.reshape(-1,1))-y.reshape(-1,1)),axis=0) + lambda_*np.sum(np.square(theta_to_compute)))

    return J
```

IV.B. Xây dựng mô hình học máy

2. Hồi quy tuyến tính đa biến

🛠 Implementation Gradient Descent with Regularization

```
def gradientDescentReg(X,y,theta,alpha,lambda_,num_iters):
    m = y.shape[0]  # number of training examples

    theta = theta.copy()

    J_records = [] # to track value of J throw each iteration.

    for i in range(num_iters):
        temp=theta.copy()
        for j in range(len(theta)):
            if j==0:
                theta[0]=temp[0]-alpha*(1/m)*np.sum((np.dot(X,temp.reshape(-1,1))-y.reshape(-1,1)).squeeze())
            else:
                theta[j]=temp[j]*(1-(alpha*(lambda_/m)))-alpha*(1/m)*np.sum((np.dot(X,temp.reshape(-1,1))-y.reshape(-1,1)).squeeze()*X[:,j])

        # save the cost J in every iteration
        J_records.append(computeCostFunctionReg(X, y, theta, lambda_))

    return theta, J_records
```

IV.B. Xây dựng mô hình học máy

2. Hồi quy tuyến tính đa biến

🛠 Sau khi cài đặt các hàm, tiến hành chạy thuật toán gradient descent.

```
# initialize fitting parameters
theta2 = np.random.rand(X2_train.shape[1])
# some gradient descent settings
iterations2 = 10000
alpha2 = 0.01 #learning rate
lambda_=1

theta2, J_history2 = gradientDescentReg(X2_train ,y2_train, theta2, alpha2, lambda_, iterations2)
```

```
theta2
```

```
array([ 2.50963541,  1.20089063,  0.42935034, -0.18216281])
```

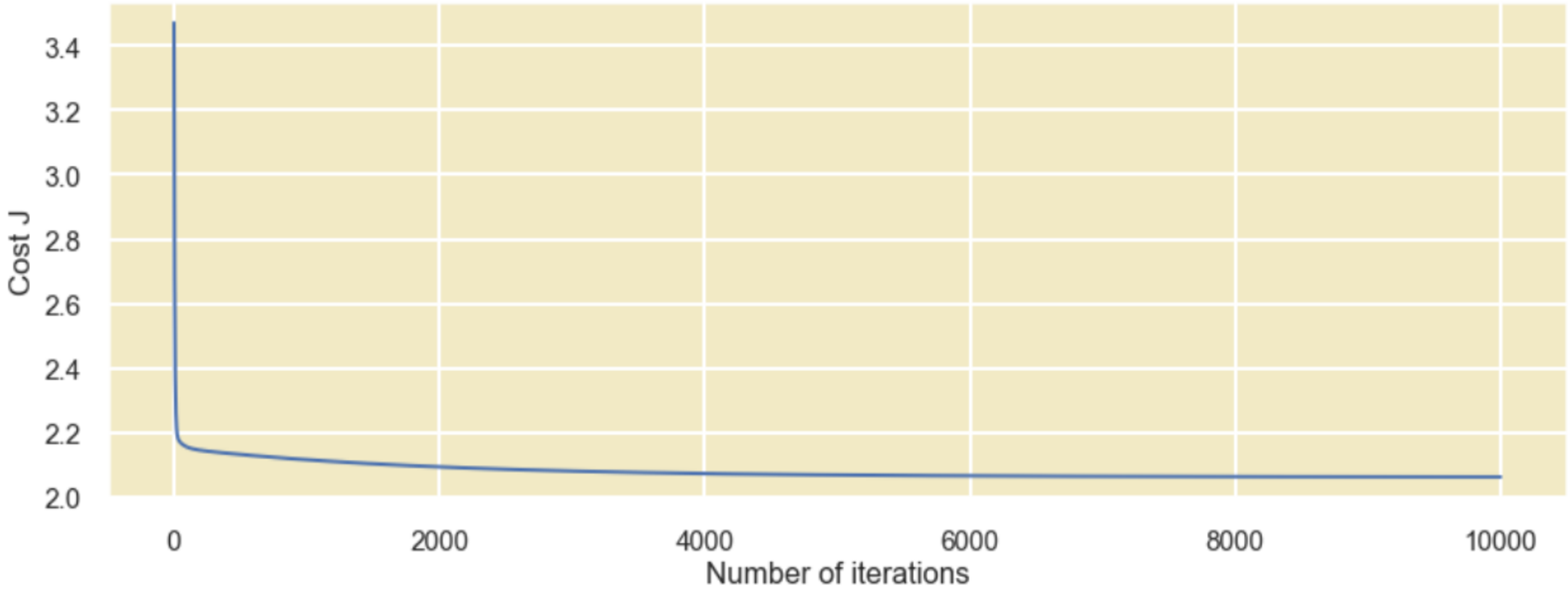
θ_0 θ_1 θ_2 θ_3

IV.B. Xây dựng mô hình học máy

2. Hồi quy tuyến tính đa biến

🛠 Kết quả thu được sau khi chạy thuật toán.

Vẽ biểu đồ các giá trị của Cost Function sau mỗi lần thực hiện gradient descent.



IV.B. Xây dựng mô hình học máy

2. Hồi quy tuyến tính đa biến

🛠 Kết quả thu được sau khi chạy thuật toán.

Tính trên tập test:

```
computeCostFunctionReg(X2_test,y2_test,theta2,lambda_)
```

```
array([2.63925256])
```

👉 Giá trị của Cost Function khi kiểm nghiệm trên tập Test là 2.639

🛠 Implementation Normal Equation with Regularization.

```
def normalEquationReg(X, y, lambda_, A):
    theta = np.zeros(X.shape[1])

    theta=np.dot(np.dot(np.linalg.inv(np.dot(X.T,X) + lambda_*A),X.T),y.reshape(-1,1))
    theta=theta.squeeze()

    return theta
```

```
A=np.diag(np.full(X2_train.shape[1],1))
A[0,0]=0
```

IV.B. Xây dựng mô hình học máy

2. Hồi quy tuyến tính đa biến

🛠 **Implementation Normal Equation with Regularization.**

```
theta2_eqn = normalEquationReg(X2_train, y2_train, lambda_, A)
theta2_eqn
```

```
array([ 2.68275386,  1.22618977,  0.38615308, -0.22465913])
```

```
computeCostFunctionReg(X2_test,y2_test,theta2_eqn,lambda_)
```

```
array([2.63380934])
```

👉 Trong trường hợp này: **Normal Equation** cho kết quả hơi tốt hơn so với **Gradient Descent**.

IV.B. Xây dựng mô hình học máy

2. Hồi quy tuyến tính đa biến

Thống kê phần trăm sai lệch của mỗi cặp giá trị dự đoán – giá trị thật.

```
statistical_df['% sai lech'].describe()

count    226.000000
mean      42.611577
std       34.197188
min        0.163634
25%       15.961642
50%       34.631247
75%       62.106314
max      149.883942
```

Ngoài ra còn một thông số khá phổ biến khác cho việc đánh giá mức độ phù hợp của các mô hình hồi quy tuyến tính là R^2 **Score**.

💬 **Nhận xét:**

- Từ độ sai lệch cho thấy mô hình đưa ra dự đoán có thể chấp nhận được cho những căn hộ có giá bán từ thấp đến trung bình.
- Những căn hộ có giá bán cao có độ sai lệch quá nhiều cho thấy những thông tin từ dữ liệu đầu vào không còn đúng khi dùng để đánh giá giá bán cho những căn hộ này.
- Có thể tính tới các yêu tố ngoại cảnh như chất lượng nội thất, chất lượng vật liệu xây dựng công trình, nhà thầu xây dựng, hay ý kiến chủ quan của người bán....

IV.B. Xây dựng mô hình học máy

3. Đánh giá mô hình (Evaluation)

IV.B. Xây dựng mô hình học máy

3. Đánh giá mô hình (Evaluation)

🗺️ Dùng $R^2 Score$ để đánh giá độ phù hợp của mô hình. Điểm số tốt nhất có thể có là 1.0 và có thể có giá trị âm (mô hình cho ra kết quả quá tệ).

🗺️ $R^2 Score$ sẽ cho biết tỷ lệ các điểm dữ liệu nằm gần đường tuyến tính như thế nào. Nếu càng nhiều điểm dữ liệu trong bộ dữ liệu nằm gần đường tuyến tình thì điểm càng cao và ngược lại.

🗺️ $R^2 Score$:

R^2 được định nghĩa bằng công thức:

$$(1 - \frac{u}{v})$$

với u = sum((y_true - y_pred)^2)

v = sum((y_true - y_true.mean( ))^2)

🛠️ **Implementation $R^2 Score$:**

```
def r2Score(y_true, y_pred):
    u = np.sum(np.square(y_true - y_pred))
    v = np.sum(np.square(y_true - np.mean(y_true)))
    return 1 - (u/v)
```

IV.B. Xây dựng mô hình học máy

3. Đánh giá mô hình (Evaluation)

💬 Đánh giá mô hình hồi quy tuyến tính đơn biến.

```
print(r2Score(y_test.round(2),predictions))
#R2 score by sklearn.
print(regr.score(X_test,y_test))
```

```
0.3175430095921882
0.31752695423895017
```

💬 Đánh giá mô hình hồi quy tuyến tính đa biến.

```
regr2 = LinearRegression()
regr2.fit(X2_train, y2_train)
```

```
LinearRegression()
```

```
print(r2Score(y2_test.round(2), np.dot(X2_test,theta2).round(2)))
#R2 score from sklearn.
print(regr2.score(X2_test,y2_test))
```

```
0.3155878692096513
0.31717337828410985
```

💬 **Nhận xét:**

- Từ $R^2 Score$ cho thấy các nhận xét như sau:
  - Bộ dữ liệu có mức phù hợp thấp với mô hình hồi quy tuyến tính.
  - Có thể do số lượng dự liệu chưa đủ lớn ảnh hưởng đến việc học tập của mô hình.
  - Giá của những ngôi nhà giá cao chưa phù hợp với thông tin đầu vào.

IV.B. Xây dựng mô hình học máy

4. Hồi quy đa thức (Polynomial Regression) đơn giản với đơn biến

IV.B. Xây dựng mô hình học máy

4. Hồi quy đa thức (Polynomial Regression) đơn giản với đơn biến

🗺 Hồi quy đa thức:

- Là việc sử dụng một hypothesis có bậc lớn hơn 1 để biểu diễn các mối quan hệ phi tuyến tính.
- Trong thuật toán hồi quy tuyến tính, nó chỉ hoạt động tốt khi mối quan hệ trong bộ dữ liệu là tuyến tính. Nhưng giả sử nếu chúng ta có dữ liệu phi tuyến tính thì hồi quy tuyến tính sẽ không thể vẽ một đường phù hợp nhất và nó sẽ thất bại trong việc khớp dữ liệu cũng như dự đoán sau này.
- Khi thực hiện hồi quy đa thức đơn biến thì hypothesis sẽ có dạng:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_n x^n$$

  với bậc là n.

💬 Nhận xét:

- Nhóm sẽ sử dụng thuộc tính 'DienTich' để thực hiện hồi quy đa thức với bậc là 2 để xem liệu có thể tăng hiệu suất dự đoán so với hồi quy tuyến tính.
- Trong phần này nhóm sẽ không tự cài đặt mà sẽ dùng thư viện sklearn để hỗ trợ.

IV.B. Xây dựng mô hình học máy

4. Hồi quy đa thức (Polynomial Regression) đơn giản với đơn biến

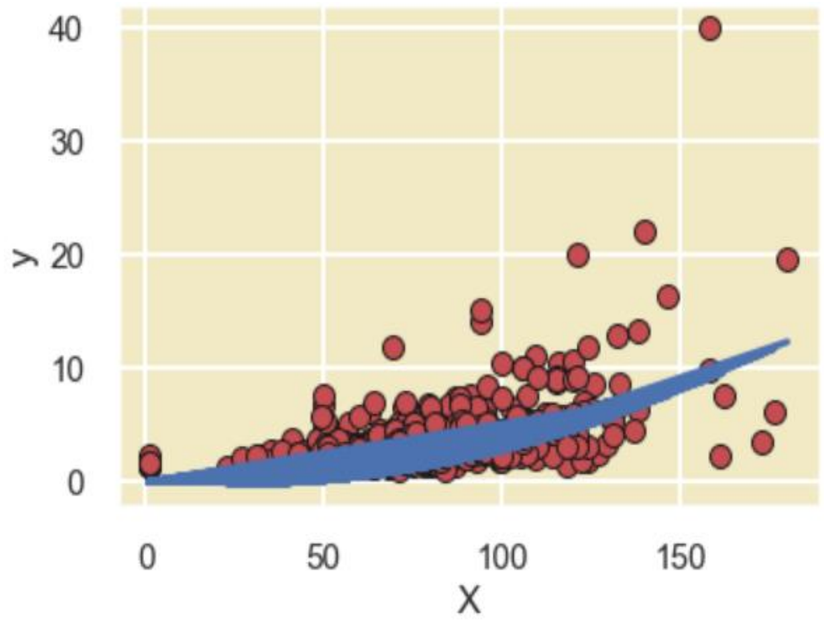
```
X3=data[:,1]
y3=data[:,0]
```

```
np.random.seed(13)
X3_train,X3_test,y3_train,y3_test=train_test_split(X3,y3,test_size=0.2)
```

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2, include_bias=True)
X3_train_trans = poly.fit_transform(X3_train.reshape(-1,1))
X3_test_trans = poly.transform(X3_test.reshape(-1,1))
```

```
lr = LinearRegression()
lr.fit(X3_train_trans, y3_train)
y_pred = lr.predict(X3_test_trans)
lr.score(X3_test_trans,y3_test)
```

```
0.4323398846052242
```



Nhận xét:

- $R^2 score$ tăng lên nhưng mô hình vẫn không khả quan.
- Điểm số tăng lên là do sai lệch ở mỗi điểm dữ liệu dự đoán và điểm dữ liệu thật giảm đi so với hồi quy tuyến tính nhưng sẽ ít đi các điểm gần đúng hơn so với hồi quy tuyến tính.

IV.B. Xây dựng mô hình học máy

5. Tổng kết

- Đối với hồi quy tuyến tính: Các thuộc tính đầu vào có thể có ảnh hưởng đến 31% giá bán của căn hộ.
- Mô hình có thể dự đoán giá của các căn hộ có giá bán từ thấp đến trung bình, nhưng không dự đoán tốt với các căn hộ có giá bán cao.
- Những yếu tố có thể ảnh hưởng đến mô hình hồi quy:
  - Còn nhiều yếu tố ngoại cảnh ảnh hưởng đến giá bán như chất lượng xây dựng, chất lượng nội thất, giá thầu,...
  - Có thể nhiều chủ căn hộ đăng giá theo cảm tính, khi nhìn vao bộ dữ liệu có rất nhiều điểm dữ liệu xếp chồng lên nhau, điều này chứng tỏ với những kích thước gần nhau nhưng giá rất khác biệt nhau.

THANKS
FOR WATCHING