BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KINH TẾ - ĐẠI HỌC ĐÀ NẴNG
KHOA THƯƠNG MẠI ĐIỆN TỬ

— o0o —

BÀI TẬP NHÓM

HỌC PHẦN: PHÂN TÍCH DỮ LIỆU BẰNG PYTHON

ĐỀ TÀI: TIỀN XỬ LÝ DỮ LIỆU - DATA PREPROCESSING

Giáo viên hướng dẫn	TS. Lê Diên Tuấn
Lớp học phần	MIS3041_48K29.1
Nhóm thực hiện	Nhóm 6
Thành viên nhóm	Hoàng Dương Thảo Hà
	(Nhóm trưởng - 0825915736)
	Hoàng Thị Phương Đông
	Nguyễn Lê Nhật Hồng
	Trần Hoài Huệ

Đà Nẵng, ngày 01 tháng 05 năm 2025

Mục lục

Danh mục hình ảnh

Phần trăm đóng góp

STT	Họ và tên	Mã Sinh viên	Phần trăm đóng góp
1	Hoàng Dương Thảo Hà	221124029112	100%
2	Hoàng Thị Phương Đông	221124029110	100%
3	Nguyễn Lê Nhật Hồng	221124029116	100%
4	Trần Hoài Huệ	221124029117	100%

1 Giới thiệu tổng quan

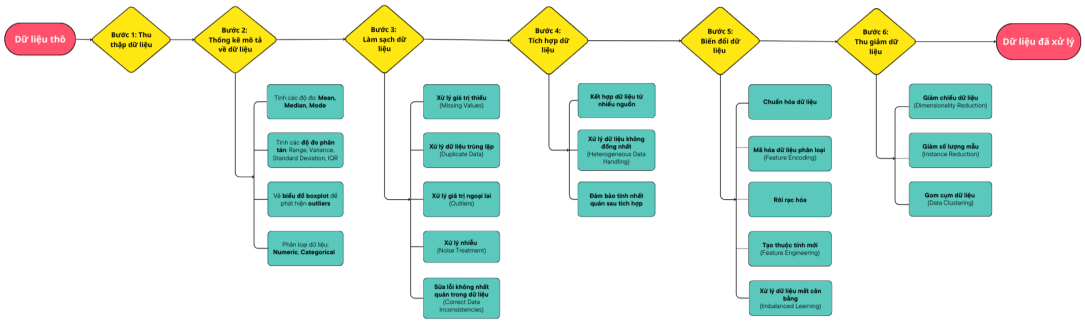
Trong phân tích dữ liệu và học máy, **tiền xử lý dữ liệu (Data Preprocessing)** là bước quan trọng ảnh hưởng trực tiếp đến chất lượng của các mô hình dự đoán. Dữ liệu thô thường chứa nhiều vấn đề như giá trị thiếu (missing values), nhiễu (noise), dữ liệu không cân bằng (imbalanced data), giá trị ngoại lai (outliers). Theo khảo sát của Anaconda[1], các nhà khoa học dữ liệu dành **khoảng 45% thời gian** cho công đoạn làm sạch và chuẩn bị dữ liệu. Do đó, việc áp dụng các kỹ thuật tiền xử lý hiệu quả là yếu tố then chốt để đảm bảo kết quả phân tích đáng tin cậy.

Đề tài tập trung nghiên cứu các kỹ thuật tiền xử lý dữ liệu áp dụng trên **bộ dữ liệu về thị trường bất động sản**, chứa thông tin về giá nhà, diện tích, số phòng ngủ, phòng tắm, cùng các yếu tố khác như vị trí, tiện ích đi kèm. Kết quả sẽ được đánh giá thông qua các biểu đồ trực quan và chỉ số thống kê, giúp làm rõ tầm quan trọng của việc chuẩn bị dữ liệu trước khi đưa vào phân tích.

Tiền xử lý dữ liệu không chỉ dừng lại ở việc **"làm sạch"** dữ liệu mà còn là bước quyết định chất lượng các mô hình phân tích. Bộ dữ liệu bất động sản trong nghiên cứu này là một ví dụ điển hình cho thấy tầm quan trọng của việc áp dụng đúng các kỹ thuật tiền xử lý trong thực tế.

2 Cơ sở lý thuyết

2.1 Quy trình nghiên cứu



Hình 1: Quy trình tiền xử lý dữ liệu

2.2 Quá trình triển khai

2.2.1 Làm sạch dữ liệu

Làm sạch dữ liệu là quá trình xử lý giá trị thiếu, loại bỏ nhiễu và sửa không nhất quán để đảm bảo dữ liệu chính xác và đáng tin cậy cho khai thác.

Giá trị thiếu: Giá trị thiếu xảy ra khi một số thuộc tính không có giá trị. Các phương pháp xử lý:

- **Bỏ qua bản ghi**: Loại bản ghi thiếu nhãn lớp, nhưng lãng phí nếu bản ghi còn thông tin hữu ích.
- **Điền thủ công**: Nhập giá trị thiếu, nhưng tốn thời gian với tập dữ liệu lớn.
- **Điền giá trị cố định**: Dùng giá trị như "Unknown", dễ gây sai lệch nếu lặp lại nhiều.
- **Dùng trung bình/trung vị**: Điền bằng trung bình (dữ liệu đối xứng) hoặc trung vị (dữ liệu lệch) của thuộc tính.
- **Dùng trung bình/trung vị theo lớp**: Điền bằng giá trị của nhóm lớp.
- **Dùng giá trị có khả năng nhất**: Dự đoán bằng hồi quy, cây quyết định, tận dụng các thuộc tính khác, chính xác nhưng phức tạp.

***Lưu ý**: Giá trị thiếu không luôn là lỗi.*

Dữ liệu nhiễu: Nhiễu là lỗi ngẫu nhiên hoặc sự biến động trong một biến được đo lường. Làm thế nào để làm mịn dữ liệu để loại bỏ nhiễu? Dưới đây là các kỹ thuật làm mịn dữ liệu:

- **Phân ô (Binning)**: Chia dữ liệu thành ô, thay giá trị bằng trung bình, trung vị hoặc ranh giới ô. Phân ô cũng dùng để rời rạc hóa hoặc giảm dữ liệu.
- **Hồi quy**: Điều chỉnh dữ liệu theo đường thẳng (hồi quy tuyến tính) hoặc bề mặt (hồi quy đa biến).
- **Phân tích ngoại lai**: Dùng phân cụm để phát hiện giá trị ngoài cụm, có thể là nhiễu.

Làm sạch dữ liệu là quy trình lặp phát hiện và sửa sai lệch do lỗi nhập liệu, biểu mẫu kém, hoặc mã không thống nhất, gồm các bước:

- **Phát hiện sai lệch**: Dùng siêu dữ liệu (loại, phạm vi) và quy tắc (duy nhất, liên tục, rỗng).
- **Biến đổi dữ liệu**: Sửa lỗi, thống nhất định dạng. Công cụ ETL hỗ trợ biến đổi, nhưng cần kịch bản tùy chỉnh.

2.2.2 Tích hợp dữ liệu

Tích hợp dữ liệu kết hợp dữ liệu từ nhiều nguồn (cơ sở dữ liệu, tệp phẳng) thành kho thống nhất, giảm dư thừa và không nhất quán, nâng cao độ chính xác khai thác.

Nhận dạng thực thể: Tích hợp dữ liệu gặp thách thức khi khớp lược đồ và đối tượng. **Vấn đề nhận dạng thực thể**: Xác định các thực thể tương đương, ví dụ *customer_id* và *cust_number* đều là mã khách hàng.

- **Siêu dữ liệu**: Dùng tên, ý nghĩa, kiểu dữ liệu để thống nhất.
- **Cấu trúc dữ liệu**: Đảm bảo ràng buộc (phụ thuộc hàm, tham chiếu).

Phân tích dư thừa và tương quan: Dư thừa xảy ra khi thuộc tính suy ra từ thuộc tính khác, ví dụ *doanh thu hàng năm* từ *doanh thu hàng tháng*. **Phân tích tương quan** phát hiện:

- **Kiểm định** χ^2 (dữ liệu danh nghĩa): Đo mối quan hệ, ví dụ khảo sát 1500 người cho $\chi^2 = 507.93$, chỉ ra giới tính và sở thích đọc tương quan mạnh.
- **Hệ số tương quan** (dữ liệu số): Đo mức độ thay đổi cùng nhau, ví dụ giá cổ phiếu AllElectronics và HighTech có hiệp phương sai 7, tăng cùng chiều.

Bản ghi trùng lặp: Ngoài dư thừa giữa các thuộc tính, cần phát hiện bản ghi trùng lặp (nhiều bản ghi giống hệt nhau cho một trường hợp dữ liệu duy nhất).

Xung đột giá trị dữ liệu: Giá trị của cùng một thực thể có thể khác nhau giữa các nguồn do cách biểu diễn, tỷ lệ/ mã hóa hoặc mức trừu tượng.

2.2.3 Giảm dữ liệu

Giảm dữ liệu tạo biểu diễn nhỏ hơn, giữ tính toàn vẹn dữ liệu, tăng hiệu quả khai thác mà kết quả gần giống dữ liệu gốc.

- **Giảm chiều**: Giảm số thuộc tính bằng biến đổi wavelet, PCA, hoặc lựa chọn tập con thuộc tính.
- **Giảm số lượng**: Thay thế dữ liệu gốc bằng các biểu diễn nhỏ hơn. Các phương pháp là tham số (dùng mô hình để ước lượng dữ liệu, chỉ lưu tham số như hồi quy, mô hình log-linear) hoặc phi tham số (lưu biểu diễn giảm như biểu đồ tần suất, phân cụm, lấy mẫu, hoặc tổng hợp khối dữ liệu).
- **Nén dữ liệu**: Áp dụng biến đổi để tạo biểu diễn "nén" của dữ liệu gốc. Nếu dữ liệu gốc có thể được khôi phục hoàn toàn từ dữ liệu nén, gọi là nén không mất dữ liệu. Nếu chỉ khôi phục được gần đúng, gọi là nén mất dữ liệu. Các kỹ thuật giảm chiều và giảm số lượng cũng được xem là dạng nén dữ liệu.

Biến đổi Wavelet: Chuyển vector dữ liệu thành hệ số wavelet, cắt bớt hệ số yếu để tạo biểu diễn thưa thớt, nhanh hơn. Hiệu quả với dữ liệu thưa, lệch.

Phân tích thành phần chính (PCA): Tìm $k \leq n$ vector trực giao để chiếu dữ liệu vào không gian nhỏ hơn. Quy trình: chuẩn hóa dữ liệu, tính vector trực giao, sắp xếp theo phương sai, loại thành phần yếu. Hiệu quả với dữ liệu thưa, dùng trong hồi quy, phân cụm.

Lựa chọn tập con thuộc tính: Loại thuộc tính không liên quan/dư thừa. Phương pháp: chọn tiến, loại lùi, kết hợp, hoặc cây quyết định. Mục tiêu: tập tối thiểu giữ phân phối lớp.

Hồi quy và mô hình Log-Linear: Hồi quy tuyến tính mô hình dữ liệu theo đường thẳng, log-linear xấp xỉ phân phối đa chiều. Chỉ lưu tham số, hiệu quả với dữ liệu thưa, lệch, nhưng hồi quy tốn tính toán với chiều cao.

Biểu đồ tần suất: Chia dữ liệu thành ô (độ rộng/tần suất bằng nhau), xấp xỉ phân phối. Hiệu quả với dữ liệu thưa, dày, lệch.

Phân cụm: Chia dữ liệu thành cụm giống nhau, thay dữ liệu gốc bằng biểu diễn cụm. Hiệu quả với dữ liệu có cụm rõ ràng.

Lấy mẫu: Biểu diễn dữ liệu lớn bằng mẫu nhỏ, ví dụ mẫu ngẫu nhiên, mẫu cụm, hoặc phân tầng. Chi phí tỷ lệ với kích thước mẫu, hiệu quả với dữ liệu lệch.

Tổng hợp khối dữ liệu: Tổng hợp theo chiều, ví dụ doanh thu quý thành năm. Khối dữ liệu lưu thông tin đa chiều, hỗ trợ khai thác nhanh ở nhiều mức trừu tượng.

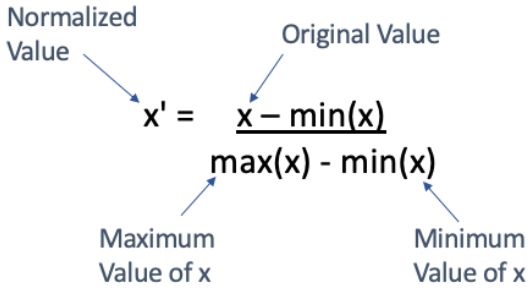
2.2.4 Biến đổi và rời rạc hóa dữ liệu:

Biến đổi dữ liệu điều chỉnh dữ liệu cho khai thác hiệu quả và dễ hiểu, bao gồm rời rạc hóa (thay giá trị số bằng nhãn).

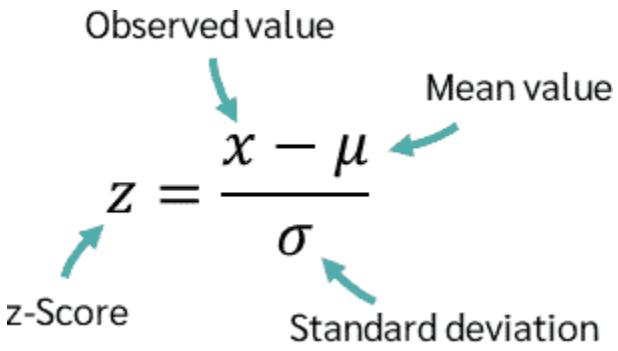
- **Làm mịn**: Loại nhiễu bằng phân ô, hồi quy, phân cụm.
- **Xây dựng thuộc tính**: Tạo thuộc tính mới, ví dụ diện tích từ chiều cao, chiều rộng.
- **Tổng hợp**: Tính tổng hoặc giá trị tổng hợp (ví dụ, doanh thu hàng tháng từ doanh thu hàng ngày).
- **Chuẩn hóa**: Đưa dữ liệu về phạm vi $[-1, 1]$ hoặc $[0, 1]$.
- **Rời rạc hóa**: Thay giá trị số bằng nhãn khoảng (tuổi: 0–10) hoặc khái niệm (trẻ, trưởng thành).
- **Tạo hệ thống khái niệm**: Tổng quát hóa danh nghĩa, ví dụ phố $\rightarrow$ thành phố.

Biến đổi dữ liệu bằng chuẩn hóa: Đưa dữ liệu về phạm vi chung:

- **Min-max**:



- **Z-score**:



- **Thập phân**: Chia giá trị lớn nhất.

Rời rạc hóa

- **Phân ô**: Chia ô độ rộng/tần suất bằng nhau, thay bằng trung bình/trung vị.
- **Biểu đồ tần suất**: Tương tự phân ô, tạo hệ thống khái niệm đa cấp.
- **Phân cụm, cây quyết định, ChiMerge**: Phân cụm (không giám sát), cây quyết định (giám sát, entropy thấp), ChiMerge (hợp nhất khoảng dựa trên χ^2).

Hệ thống khái niệm danh nghĩa: Tạo hệ thống tự động từ lược đồ, ví dụ quốc gia $\rightarrow$ tỉnh $\rightarrow$ thành phố, dựa trên số giá trị riêng biệt hoặc ngữ nghĩa lược đồ.

3 Triển khai thực nghiệm

3.1 Mô tả dataset

- Tên bộ dữ liệu: **Housing Prices Dataset**

- Bộ dữ liệu này gồm 13 cột và 545 dòng, chứa thông tin về các ngôi nhà, bao gồm nhiều đặc điểm khác nhau như diện tích, số phòng, và giá bán. Đây là một tập dữ liệu hữu ích để nghiên cứu các yếu tố ảnh hưởng đến giá nhà và áp dụng các mô hình dự báo giá bất động sản. Dữ liệu này sẽ được sử dụng để phân tích các yếu tố tác động đến giá trị bất động sản, từ đó xây dựng các mô hình học máy để dự đoán giá nhà.

3.2 Thống kê mô tả về dữ liệu

3.2.1 Thông tin về bộ dữ liệu

*** Mã code:**

```
# Thong tin ve bo du lieu
df.info()
```

*** Kết quả:**

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             530 non-null    float64
 1   area              545 non-null    int64
 2   bedrooms          544 non-null    float64
 3   bathrooms         542 non-null    float64
 4   stories           540 non-null    float64
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          536 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   535 non-null    object
 10  parking           536 non-null    float64
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: float64(5), int64(1), object(7)
memory usage: 55.5+ KB
```

Hình 2: Thông tin về bộ dữ liệu

*** Nhận xét:** Dữ liệu gồm 545 dòng và 13 cột, trong đó có 7 cột chứa giá trị null, kiểu dữ liệu bao gồm: float64 (5 cột), int64 (1 cột), object (7 cột).

3.2.2 Thống kê mô tả cho các biến số

*** Mã code:**

```
# Thong ke mo ta cho cac bien so
df.describe()
```

*** Kết quả:**

	price	area	bedrooms	bathrooms	stories	parking
count	5.300000e+02	545.000000	544.000000	542.000000	540.000000	536.000000
mean	4.730506e+06	5150.541284	2.963235	1.284133	1.803704	0.697761
std	1.860027e+06	2170.141023	0.737405	0.501833	0.871304	0.861467
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	4.273500e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

Hình 3: Thống kê mô tả cho các biến số

*** Nhận xét:**

- Phân phối không đồng đều: Các biến như price, area có phân phối lệch phải. price có trung bình là 4.73 triệu nhưng giá trị lớn nhất lên đến 13.3 triệu; tương tự, area có trung bình là 5150 nhưng giá trị lớn nhất là 16200. Điều này cho thấy tồn tại các giá trị lớn bất thường (outliers) và dữ liệu bị lệch phải rõ rệt.
- Biến price và area có độ biến thiên lớn: Độ lệch chuẩn của price là 1.86 triệu và của area là 2170, đều khá cao so với trung bình. Điều này cho thấy dữ liệu phân tán rộng, cần xử lý chuẩn hóa hoặc biến đổi log để giảm ảnh hưởng của các giá trị lớn đến mô hình.
- Hầu hết các biến còn lại có phân phối khá hợp lý: Các biến như bedrooms, bathrooms, và stories có giá trị trung vị trùng hoặc gần với trung bình (bedrooms trung vị là 3, trung bình là 2.96), thể hiện sự tập trung quanh các giá trị phổ biến như 2–3 phòng ngủ, 1–2 phòng tắm.

3.2.3 Thống kê mô tả các biến phân loại

*** Mã code:**

```
# Thong ke mo ta cho cac bien phan loai
df.select_dtypes(include=['object']).describe()
```

*** Kết quả:**

	mainroad	guestroom	basement	hotwaterheating	airconditioning	prefarea	furnishingstatus
count	545	545	536	545	535	545	545
unique	2	2	2	2	2	2	3
top	yes	no	no	no	no	no	semi-furnished
freq	468	448	349	520	364	417	227

Hình 4: Thống kê mô tả các biến phân loại

*** Nhận xét:**

- Phần lớn các biến là dạng nhị phân (yes/no) với 2 giá trị duy nhất, bao gồm: `mainroad`, `guestroom`, `basement`, `hotwaterheating`, `airconditioning`, và `prefarea`.
- Các biến có xu hướng mất cân đối (thiên lệch phân bố):
  - `mainroad`: 468/545 quan sát có nhà mặt đường (chiếm 85.9%) → phần lớn nhà đều nằm trên mặt đường.
  - `guestroom`: chỉ có 97/545 có phòng khách cho khách (17.8%), còn lại không có.
  - `basement`, `hotwaterheating`, `airconditioning`, `prefarea` cũng đều có tỷ lệ "no"chiếm ưu thế, `basement` có 349/536 là "no"(65.1%).
- Biến `furnishingstatus` có 3 giá trị phân loại: `semi-furnished` là phổ biến nhất với 227/545 dữ liệu (41.6%), tiếp theo là `unfurnished` và `furnished` → Cần sử dụng one-hot encoding hoặc dummy encoding để đưa vào mô hình.

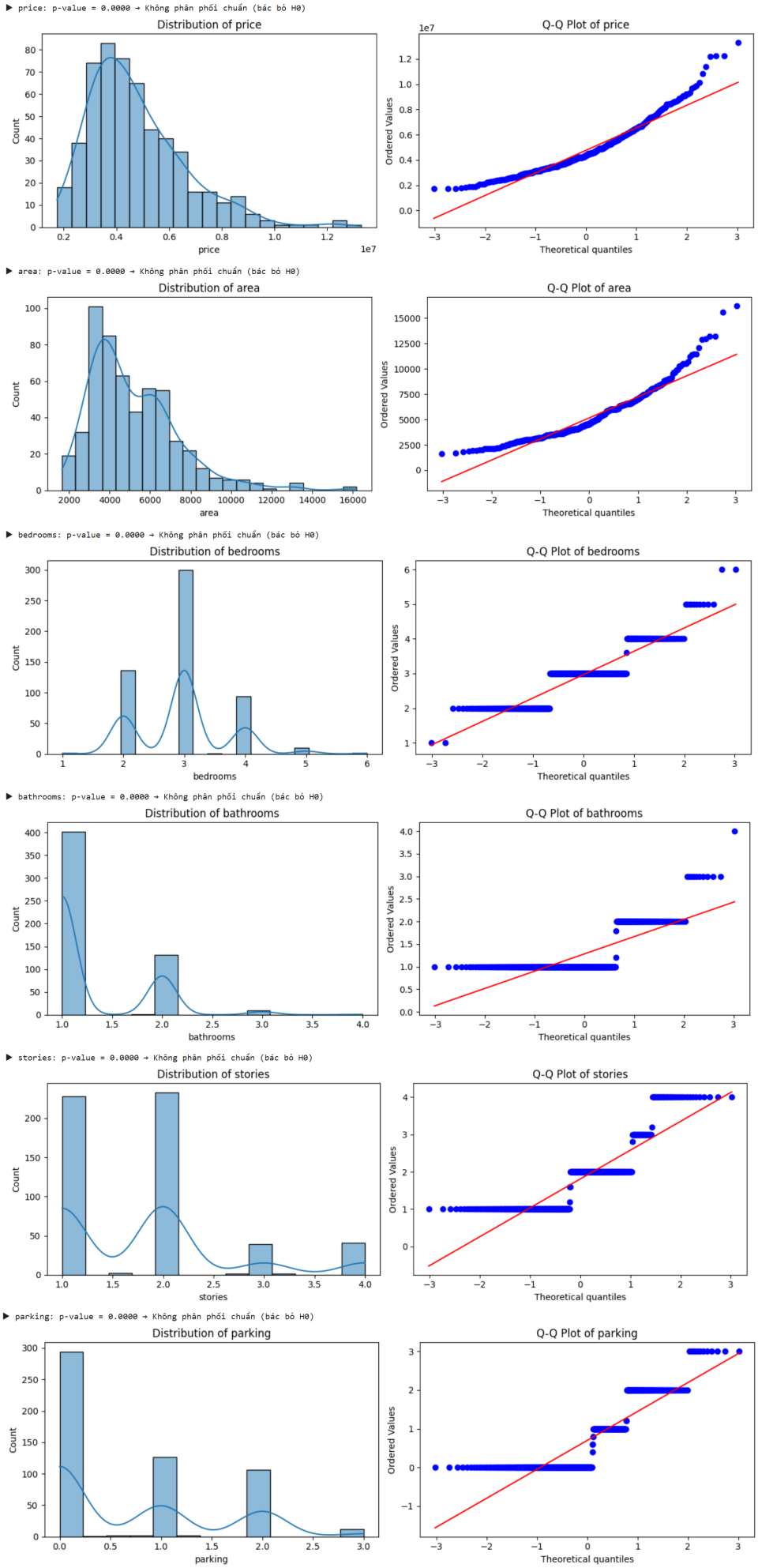
3.2.4 Kiểm tra phân phối của dữ liệu số

*** Mã code:**

```
# Vong lap kiem dinh va truc quan
for col in numerical_cols:
    data = df_missing[col].dropna()

    # 1. Kiem dinh Shapiro-Wilk
    stat, p = shapiro(data)
    print(f'>> {col}: p-value = {p:.4f} ->', end=' ')
    if p > 0.05:
        print("Phan phoi co the la chuan (khong du bang chung bac bo H0)")
```

```
else:
    print("Khong phan phoi chuan (bac bo H0)")

# 2. Bieu do phan phoi
plt.figure(figsize=(12,4))

# Bieu do histogram + KDE
plt.subplot(1,2,1)
sns.histplot(data, kde=True)
plt.title(f'Phan phoi cua {col}')

# Bieu do Q-Q
plt.subplot(1,2,2)
probplot(data, dist="norm", plot=plt)
plt.title(f'Q-Q Plot cua {col}')

plt.tight_layout()
plt.show()
```

*** Kết quả:**



Hình 5: Phân phối của dữ liệu số

*** Nhận xét:** Tất cả các cột số đều không có phân phối chuẩn. Biểu đồ histogram cho thấy phân phối lệch phải rõ rệt. Biểu đồ Q-Q plot cho thấy các điểm dữ liệu chênh lệch đáng kể khỏi đường thẳng tham chiếu.

3.3 Làm sach dữ liệu

3.3.1 Kiểm tra dữ liệu null

*** Mã code:**

```
# Tinh so luong null va phan tram null
null_count = df.isnull().sum() # So luong gia tri null
null_percentage = (null_count / len(df)) * 100 # Ty le %

# Tao DataFrame hien thi ket qua
null_summary = pd.DataFrame({
    'Column': null_count.index, # Ten cot
    'Null Count': null_count.values, # So luong gia tri null
    'Null Percentage': null_percentage.values # Phan tram gia tri null
})

# Sap xep theo so luong null giam dan
null_summary = null_summary.sort_values(by="Null Count", ascending=False)

# In ket qua
print(null_summary)
```

*** Kết quả:**

	Column	Null Count	Null Percentage
0	price	15	2.752294
9	airconditioning	10	1.834862
7	basement	9	1.651376
10	parking	9	1.651376
4	stories	5	0.917431
3	bathrooms	3	0.550459
2	bedrooms	1	0.183486
1	area	0	0.000000
5	mainroad	0	0.000000
8	hotwaterheating	0	0.000000
6	guestroom	0	0.000000
11	prefarea	0	0.000000
12	furnishingstatus	0	0.000000

Hình 6: Kiểm tra dữ liệu thiếu

*** Nhận xét:**

- Biến mục tiêu price có tỷ lệ thiếu cao nhất (15 dòng), cần xử lý cẩn thận vì ảnh

hưởng trực tiếp đến mô hình.

- Một số biến đặc trưng có tỷ lệ thiếu trung bình: airconditioning (10 dòng thiếu – 1.83%), basement và parking (mỗi biến thiếu 9 dòng – 1.65%).
- Các biến thiếu có tỷ lệ thiếu ít: stories (5 dòng), bathrooms (3 dòng), bedrooms (1 dòng).

* **Giải pháp:** Điền các giá trị null bằng các mô hình học máy thay vì xóa các bản ghi có giá trị thiếu để có độ chính xác cao hơn, không làm sai lệch phân phối mà vẫn giữ lại đầy đủ thông tin với bộ dữ liệu không quá lớn (chỉ 545 dòng).

- Đối với biến mục tiêu price: Sử dụng phương pháp Regression Imputation vì nó phụ thuộc vào các biến đầu vào như area, bedrooms, parking, v.v. Nếu dùng mean hoặc median sẽ làm giảm phương sai, làm méo phân phối và phá vỡ mối quan hệ đầu vào $\rightarrow$ đầu ra.
- Đối với các biến phân loại airconditioning, basement, prefarea: Sử dụng phương pháp Classification Imputation (mô hình Random Forest Classifier) vì mô hình sẽ tận dụng được quan hệ phức tạp giữa các đặc trưng đầu vào (area, bedrooms, stories) để ước lượng xác suất hợp lý hơn là điền bằng mode.
- Đối với các biến số parking, stories, bathrooms, bedrooms: Sử dụng phương pháp KNN imputer. Đây là những biến có quan hệ gần gũi, mang tính kết cấu (cấu hình căn nhà), nên việc dự đoán dựa trên các quan sát "gần giống nhau"trong không gian nhiều chiều là phù hợp.

*** Mã code xử lý dữ liệu null**

```
# === 1. Nhom bien theo phuong phap xu ly ===
target_col = 'price'
num_knn_features = ['bedrooms', 'bathrooms', 'stories', 'parking']
cat_classify_features = ['airconditioning', 'basement', 'prefarea']
cat_no_missing = ['mainroad', 'guestroom', 'hotwaterheating', 'furnishingstatus']
num_no_missing = ['area']

# === 2. Xu ly cac bien SO bang KNN (tru 'price') ===
knn_imputer = KNNImputer(n_neighbors=5)
df_knn_imputed = pd.DataFrame(
    knn_imputer.fit_transform(df[num_knn_features]),
```

```
    columns=num_knn_features,
    index=df.index
)

# === 3. Ma hoa bien phan loai (gom ca missing va khong missing) ===
all_cat_features = cat_classify_features + cat_no_missing
encoder = OrdinalEncoder(
    handle_unknown='use_encoded_value',
    unknown_value=np.nan
)
df_cat_encoded = pd.DataFrame(
    encoder.fit_transform(df[all_cat_features]),
    columns=all_cat_features,
    index=df.index
)

# === 4. Impute cac bien phan loai thieu bang Classification Imputer ===
cat_imputer = IterativeImputer(
    estimator=RandomForestClassifier(n_estimators=100, random_state=42),
    max_iter=10,
    random_state=42
)
df_cat_imputed = pd.DataFrame(
    cat_imputer.fit_transform(df_cat_encoded),
    columns=all_cat_features,
    index=df.index
)

# Lam tron va chuyen ve int de giai ma
df_cat_imputed = df_cat_imputed.round().astype(int)

# Giai ma ve dang nhan ban dau
df_cat_final = pd.DataFrame(
    encoder.inverse_transform(df_cat_imputed),
    columns=all_cat_features,
    index=df.index
)

# === 5. Impute bien muc tieu 'price' bang hoi quy ===
# Tao tap train/test dua vao cho thieu gia
train_idx = df['price'].notnull()
test_idx = df['price'].isnull()

# Tap dac trung dung de du doan 'price'
```

```
features_for_price = num_knn_features + num_no_missing + all_cat_features
X_train = pd.concat([
    df_knn_imputed.loc[train_idx],
    df[num_no_missing].loc[train_idx],
    df_cat_imputed.loc[train_idx]
], axis=1)
y_train = df.loc[train_idx, 'price']

X_test = pd.concat([
    df_knn_imputed.loc[test_idx],
    df[num_no_missing].loc[test_idx],
    df_cat_imputed.loc[test_idx]
], axis=1)

# Hoi quy de du doan gia
rf_reg = RandomForestRegressor(n_estimators=100, random_state=42)
rf_reg.fit(X_train, y_train)
predicted_prices = rf_reg.predict(X_test)

# Gop lai voi gia goc
df_price_filled = df['price'].copy()
df_price_filled.loc[test_idx] = predicted_prices

# === 6. Tap hop du lieu day du sau khi xu ly ===
df_missing = pd.concat([
    df_price_filled.rename('price'),
    df[num_no_missing], # 'area'
    df_knn_imputed[num_knn_features],
    df_cat_final[all_cat_features]
], axis=1)

# === 7. Kiem tra lan cuoi ===
# Tinh so luong null va phan tram null
null_count = df_missing.isnull().sum()
null_percentage = (null_count / len(df_missing)) * 100

# Tao DataFrame hien thi ket qua
null_summary = pd.DataFrame({
    'Column': null_count.index,
    'Null Count': null_count.values,
    'Null Percentage': null_percentage.values
})

# Sap xep theo so luong null giam dan
```

```
null_summary = null_summary.sort_values(by="Null Count", ascending=False)

# In ket qua
print(null_summary)
```

*** Kết quả sau khi đã xử lý null:**

	Column	Null Count	Null Percentage
0	price	0	0.0
1	area	0	0.0
2	bedrooms	0	0.0
3	bathrooms	0	0.0
4	stories	0	0.0
5	parking	0	0.0
6	airconditioning	0	0.0
7	basement	0	0.0
8	prefarea	0	0.0
9	mainroad	0	0.0
10	guestroom	0	0.0
11	hotwaterheating	0	0.0
12	furnishingstatus	0	0.0

Hình 7: Dữ liệu sau khi xử lý null

3.3.2 Kiểm tra giá trị trùng lặp

*** Mã code:**

```
num_duplicates = df_missing.duplicated().sum()
print(f"Co {num_duplicates} dong trung lap")
```

***Kết quả:**

```
Có 0 dòng trùng lặp
```

Hình 8: Kết quả kiểm tra trùng lặp dữ liệu

*** Nhận xét:** Bộ dữ liệu không có giá trị trùng lặp nào.

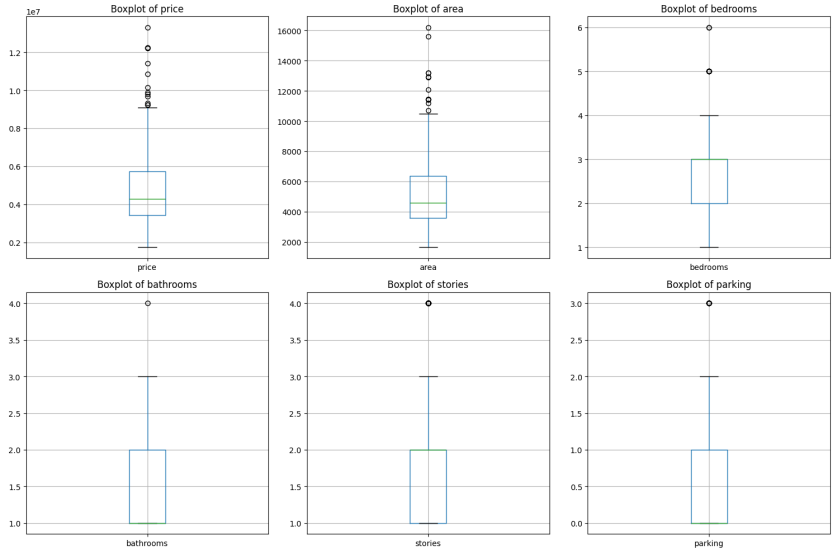
3.3.3 Kiểm tra giá trị ngoại lai

*** Mã code:**

```
# Tao figure va cac truc con (subplots)
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))

# Lam phang mang axes de de lap
axes = axes.flatten()

# Lap qua cac bien so de ve bieu do hop (boxplot)
```

```
for i, feature in enumerate(numerical_cols):
    if feature in df.columns:
        df.boxplot(column=feature, ax=axes[i]) # Ve boxplot cho tung bien
        axes[i].set_title(f'Boxplot of {feature}')

# Can chinh lai bo cuc de tranh bi chong lan
plt.tight_layout()
plt.show()
```

***Kết quả:**



Hình 9: Kết quả kiểm tra giá trị ngoại lai

*** Nhận xét:** Hầu hết các biến số đều có sự xuất hiện của các giá trị ngoại lai, đặc biệt là biến price và area có số lượng ngoại lai lớn nhất. Các giá trị ngoại lai này có thể là những điểm dữ liệu thực tế nhưng khác biệt, hoặc có thể là lỗi nhập liệu, cho thấy sự biến thiên lớn trong dữ liệu.

*** Giải pháp:** Sử dụng phương pháp biến đổi IQR cho các giá trị ngoại lai để giảm ảnh hưởng của các giá trị cực đoan mà vẫn có thể bảo toàn dữ liệu, tránh được việc loại bỏ quá nhiều điểm dữ liệu làm mất đi thông tin quan trọng, ngoài ra phương pháp IQR không dựa trên giả định rằng dữ liệu tuân theo phân phối chuẩn phù hợp với bộ dữ liệu có phân phối lệch.

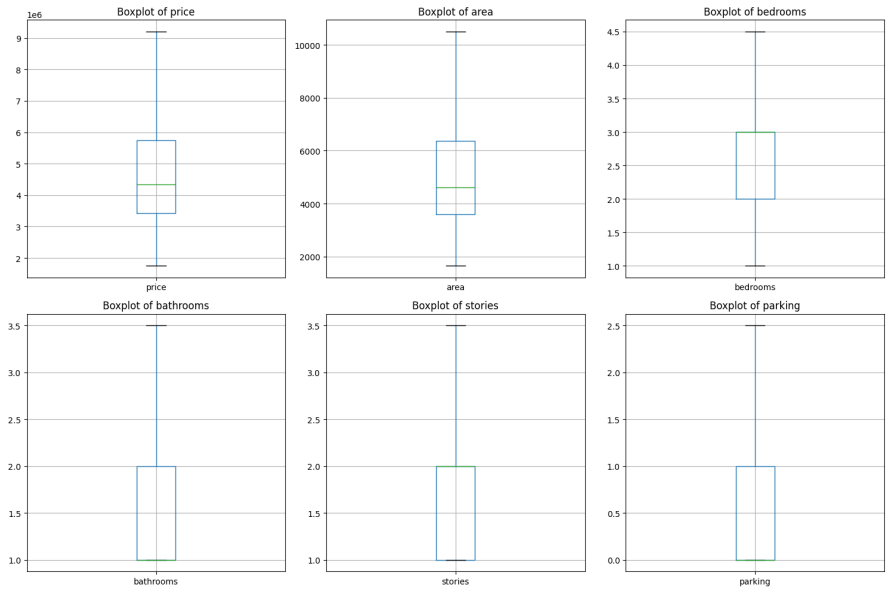
*** Mã code xử lý giá trị ngoại lai:**

```
# Ham xu ly outliers bang phuong phap IQR clipping
def clip_outliers_iqr(df, columns):
```

```
    df_clipped = df_missing.copy()
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df_clipped[col] = df[col].clip(lower=lower_bound, upper=upper_bound)
        print(f"[{col}] Clipped to range: [{lower_bound:.2f}, {upper_bound:.2f}]")
    return df_clipped

# Ap dung ham len df_missing (hoac df_cleaned cua ban)
df_clipped = clip_outliers_iqr(df_missing, numerical_cols)

# Tao figure va cac truc con (subplots)
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))

# Lam phang mang axes de de lap
axes = axes.flatten()

# Lap qua cac bien so va ve bieu do hop (boxplot)
for i, feature in enumerate(numerical_cols):
    if feature in df_clipped.columns:
        df_clipped.boxplot(column=feature, ax=axes[i]) # Ve boxplot cho du lieu da xu
            ly
        axes[i].set_title(f'Boxplot of {feature}')

# Dieu chinh bo cuc de tranh bi chong lan
plt.tight_layout()
plt.show()
```

---

*Kết quả xử lý giá trị ngoại lai:




```
[price] Clipped to range: [-35000.00, 9205000.00]
[area] Clipped to range: [-540.00, 10500.00]
[bedrooms] Clipped to range: [0.50, 4.50]
[bathrooms] Clipped to range: [-0.50, 3.50]
[stories] Clipped to range: [-0.50, 3.50]
[parking] Clipped to range: [-1.50, 2.50]
```

Hình 10: Kết quả sau khi xử lý giá trị ngoại lai

3.4 Trực quan hoá dữ liệu

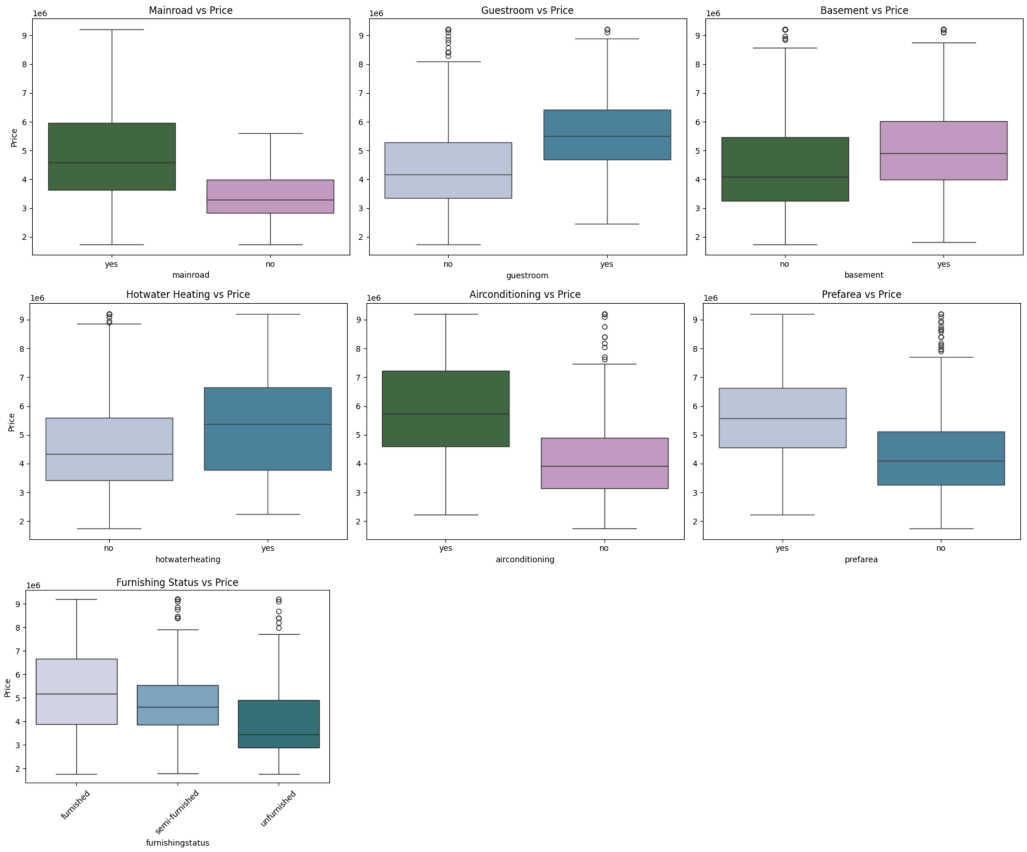
3.4.1 Biểu đồ Boxphot

* Mã code:

```
# === Boxplot: Cac bien phan loai so voi Price ===

plt.figure(figsize=(18, 10))

# Bieu do 1: Mainroad
plt.subplot(2, 3, 1)
plt.title('Mainroad vs Price')
sns.boxplot(x=df_clipped.mainroad, y=df_clipped.price, palette="cubehelix")
plt.ylabel('Price')

# Bieu do 2: Guestroom
plt.subplot(2, 3, 2)
plt.title('Guestroom vs Price')
sns.boxplot(x=df_clipped.guestroom, y=df_clipped.price, palette="PuBuGn")
plt.ylabel('')
```

```
# Bieu do 3: Basement
plt.subplot(2, 3, 3)
plt.title('Basement vs Price')
sns.boxplot(x=df_clipped.basement, y=df_clipped.price, palette="cubehelix")
plt.ylabel('')

# Bieu do 4: Hotwaterheating
plt.subplot(2, 3, 4)
plt.title('Hotwater Heating vs Price')
sns.boxplot(x=df_clipped.hotwaterheating, y=df_clipped.price, palette="PuBuGn")
plt.ylabel('Price')

# Bieu do 5: Airconditioning
plt.subplot(2, 3, 5)
plt.title('Airconditioning vs Price')
sns.boxplot(x=df_clipped.airconditioning, y=df_clipped.price, palette="cubehelix")
plt.ylabel('')

# Bieu do 6: Prefarea
plt.subplot(2, 3, 6)
plt.title('Prefarea vs Price')
sns.boxplot(x=df_clipped.prefarea, y=df_clipped.price, palette="PuBuGn")
plt.ylabel('')

plt.tight_layout()
plt.show()

# === Bieu do Furnishingstatus rieng vi co nhieu nhan hon ===
plt.figure(figsize=(6, 5))
plt.title('Furnishing Status vs Price')
sns.boxplot(x=df_clipped.furnishingstatus, y=df_clipped.price, palette="PuBuGn")
plt.xticks(rotation=45)
plt.ylabel('Price')
plt.tight_layout()
plt.show()
```

*** Kết quả:**



Hình 11: Biểu đồ thể hiện mối quan hệ giữa giá nhà và các biến độc lập.

*** Nhận xét:** Các biểu đồ hộp (boxplot) trên thể hiện mối quan hệ giữa giá nhà và các biến độc lập.

- Việc nhà ở gần đường chính (mainroad = yes) và có phòng khách (guestroom = yes) cho thấy có mối tương quan rõ rệt với giá nhà. Cụ thể, giá trung bình của các ngôi nhà có những đặc điểm này cao hơn đáng kể so với các nhà không có, phản ánh vai trò quan trọng của vị trí giao thông thuận lợi và tiện nghi bổ sung trong việc định giá.

- Đối với những căn nhà có tầng hầm (basement = yes) và hệ thống nước nóng (hotwaterheating = yes) có tương quan nhẹ với giá nhà. Giá trung bình của các nhà có những đặc điểm này chỉ cao hơn một chút so với các nhà không có, cho thấy tác động của chúng không quá nổi bật trong tập dữ liệu.

- Nhà có điều hòa (airconditioning = yes) và nằm trong khu vực ưu tiên (prefarea = yes) thể hiện mối tương quan tích cực với giá nhà, với mức giá trung bình cao hơn so với các nhà không có những đặc điểm này. Ngoài ra, nhà có nội thất đầy đủ

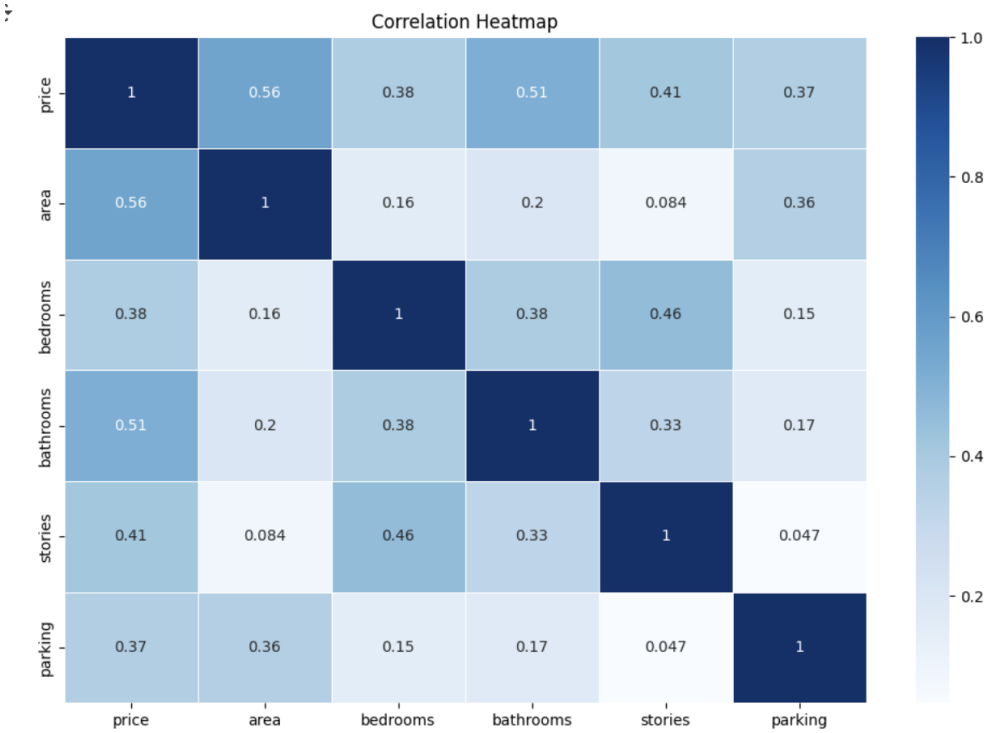
(furnished) cũng có xu hướng có giá cao hơn so với các trạng thái khác, nhấn mạnh vai trò của tiện nghi và vị trí đắc địa trong việc tăng giá trị bất động sản.

3.4.2 Biểu đồ tương quan

*** Mã code:**

```
# Tinh toan ma tran tuong quan giua cac bien so
cor_matrix = df_clipped[numerical_cols].corr()
plt.figure(figsize=(12, 8))

# Ve heatmap hien thi he so tuong quan
sns.heatmap(
    cor_matrix,
    annot=True,
    cmap='Blues',
    linewidths=0.5
)
plt.title('Correlation Heatmap')

# Hien thi bieu do
plt.show()
```

*** Kết quả**



Hình 12: Biểu đồ tương quan giữa các biến.

*** Nhận xét:**

Biểu đồ heatmap hệ số tương quan (Correlation Heatmap) trên thể hiện mối quan hệ tuyến tính giữa các đặc trưng (price, area, bedrooms, bathrooms, stories, parking) trong tập dữ liệu giá nhà. Mối mối quan hệ giữa price và area có hệ số tương quan 0.56 cho thấy mối quan hệ dương vừa phải giữa diện tích nhà và giá. Điều này hợp lý vì nhà có diện tích lớn thường có giá cao hơn, nhưng mối quan hệ không hoàn toàn tuyến tính, có thể do các yếu tố khác (như vị trí) cũng ảnh hưởng.

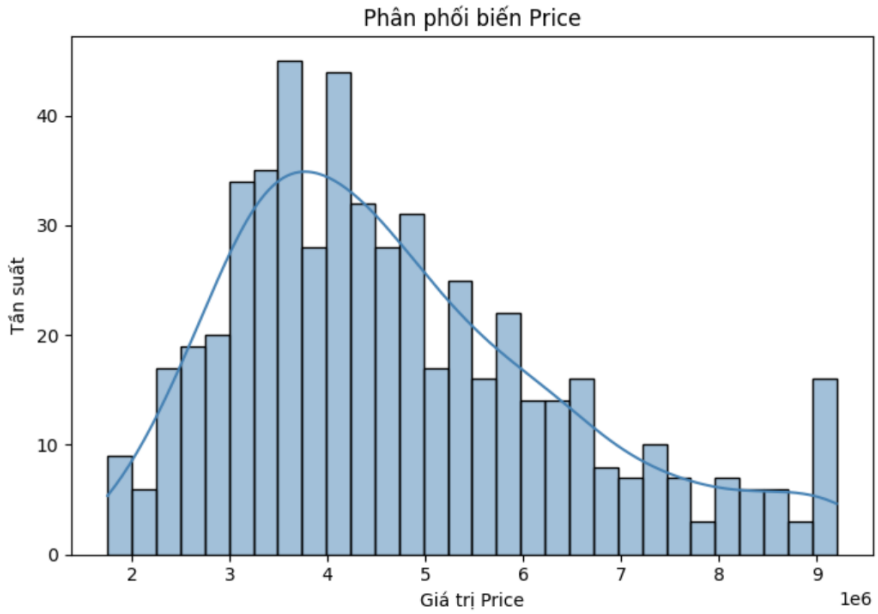
3.5 Biến đổi dữ liệu

3.5.1 Kiểm tra phân phối biến mục tiêu

*** Mã code:**

```
# Tao figure
plt.figure(figsize=(7, 5))

# Ve histogram the hien phan phoi cua bien 'price' voi duong KDE (ham mat do nhan)
sns.histplot(
    df_clipped['price'],
    kde=True,
    bins=30,
    color='steelblue'
)

# Dat tieu de
plt.title('Phan phoi bien Price')

# Gan nhan truc
plt.xlabel('Gia tri Price')
plt.ylabel('Tan suat')

# Can chinh bo cuc de khong bi chong cheo
plt.tight_layout()
# Hien thi bieu do
plt.show()
```

***Kết quả:**



Hình 13: Kết quả kiểm tra phân phối biến mục tiêu

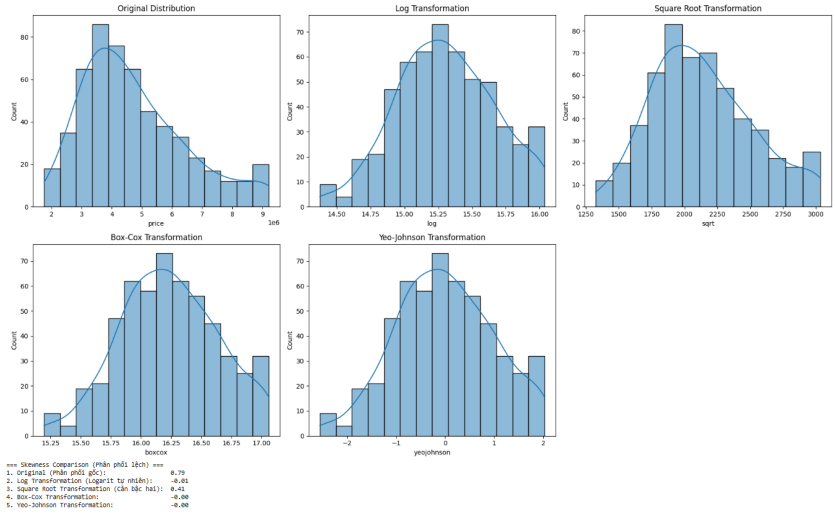
Biểu đồ histogram của biến mục tiêu "Price"(giá nhà) cho thấy phân phối dữ liệu bị lệch phải, với đỉnh nằm trong khoảng từ 4 đến 6 và có một đuôi dài kéo về phía các giá trị cao hơn. Điều này có nghĩa là phần lớn giá nhà tập trung ở mức thấp đến trung bình, nhưng có một số ít ngôi nhà có giá rất cao. Việc này có thể ảnh hưởng đến hiệu quả của các mô hình tuyến tính như Linear Regression vốn giả định phân phối chuẩn của biến mục tiêu để ước lượng chính xác hơn. Vì vậy, để cải thiện hiệu quả dự báo của mô hình, nên biến đổi biến mục tiêu bằng bốn phương pháp để giảm skewness, giúp mô hình học tốt hơn và kết quả hồi quy chính xác hơn.

*** Mã code xử lý:**

```
def analyze_skewness(df_clipped, column):
    data_clean = df_clipped[[column]].dropna().copy()

    # Tao cac bien transform
    data_clean['log'] = np.log(data_clean[column] + 1)
    data_clean['sqrt'] = np.sqrt(data_clean[column])
    data_clean['boxcox'], _ = stats.boxcox(data_clean[column][data_clean[column] >
        0])

    # Yeo-Johnson (cho ca so 0 va am neu co)
    pt = PowerTransformer(method='yeo-johnson')
    data_clean['yeojohnson'] = pt.fit_transform(data_clean[[column]])
```

```
    # === Ve bieu do ===
    fig, ax = plt.subplots(2, 3, figsize=(18, 10))
    ax = ax.flatten()

    sns.histplot(data_clean[column], kde=True, ax=ax[0])
    ax[0].set_title('Phan phoi Goc')
    sns.histplot(data_clean['log'], kde=True, ax=ax[1])
    ax[1].set_title('Bien Doi Logarit')
    sns.histplot(data_clean['sqrt'], kde=True, ax=ax[2])
    ax[2].set_title('Bien Doi Can Bac Hai')
    sns.histplot(data_clean['boxcox'], kde=True, ax=ax[3])
    ax[3].set_title('Bien Doi Box-Cox')
    sns.histplot(data_clean['yeojohnson'], kde=True, ax=ax[4])
    ax[4].set_title('Bien Doi Yeo-Johnson')
    ax[5].axis('off')

    plt.tight_layout()
    plt.show()

    # === In do lech Skewness ===
    print("=== So sanh Do Lech (Phan phoi lech) ===")
    print(f"1. Goc (Phan phoi goc): {data_clean[column].skew():.2f}")
    print(f"2. Bien Doi Log (Logarit tu nhien): {data_clean['log'].skew():.2f}")
    print(f"3. Bien Doi Can Bac Hai: {data_clean['sqrt'].skew():.2f}")
    print(f"4. Bien Doi Box-Cox: {pd.Series(data_clean['boxcox']).skew():.2f}")
    print(f"5. Bien Doi Yeo-Johnson: {data_clean['yeojohnson'].skew():.2f}")
analyze_skewness(df_clipped, 'price')
```

*Kết quả:



Hình 14: Kết quả thử nghiệm 4 phương pháp

***Nhận xét:**

Để giải quyết vấn đề trên, nhóm 6 đã thử nghiệm bốn phương pháp biến đổi phân phối:

- Log Transformation (Logarit tự nhiên): Độ lệch giảm xuống 0.01. Phân phối trở nên gần chuẩn, với biểu đồ histogram cân đối hơn, không còn đuôi dài đáng kể.
- Square Root Transformation (Căn bậc hai): Độ lệch giảm xuống 0.41. Phân phối vẫn còn lệch phải, cho thấy phương pháp này không đủ mạnh để chuẩn hóa hoàn toàn dữ liệu.
- Box-Cox Transformation: Độ lệch đạt 0.00, phân phối gần chuẩn hoàn toàn. Phương pháp này hiệu quả nhưng yêu cầu dữ liệu dương và phức tạp hơn trong triển khai.
- Yeo-Johnson Transformation: Độ lệch cũng đạt 0.00, tương tự Box-Cox. Phương pháp này linh hoạt hơn vì có thể xử lý cả dữ liệu âm, nhưng không cần thiết trong trường hợp giá nhà (luôn dương).

Trong các phương pháp thử nghiệm, Log Transformation là lựa chọn tối ưu vì nó vừa hiệu quả trong việc chuẩn hóa phân phối, vừa đơn giản và có ý nghĩa thực tiễn cao. Box-Cox và Yeo-Johnson cũng đạt kết quả tốt, nhưng phức tạp hơn và không mang lại lợi ích vượt trội trong trường hợp này. Square Root Transformation ít hiệu quả hơn, nên không được ưu tiên.

3.5.2 Mã hóa dữ liệu phân loại

Các biến như mainroad, guestroom, basement, v.v., là các biến nhị phân, chỉ có hai giá trị: "yes"hoặc "no"(có hoặc không). Để mô hình học máy có thể xử lý, chúng ta chuyển các giá trị dạng chuỗi ("yes"/"no") thành dạng số (1 hoặc 0), trong đó:

- yes hoặc Yes được ánh xạ thành 1.
- no hoặc No được ánh xạ thành 0.

=> Việc chuyển thành 1 hoặc 0 giúp mô hình hiểu được ý nghĩa nhị phân của các biến này mà không cần thêm chiều dữ liệu mới, đồng thời đảm bảo tính tuyến tính và khả năng diễn giải trong các mô hình như hồi quy tuyến tính.

Biến furnishingstatus là một biến phân loại có 3 giá trị: "unfurnished", "semi-furnished", và "furnished". Đây là biến không có thứ tự (nominal), nghĩa là không có mối quan hệ

thứ tự tự nhiên giữa các giá trị. Nếu mã hóa bằng số (ví dụ: 0, 1, 2), mô hình có thể hiểu sai rằng có mối quan hệ thứ tự, dẫn đến kết quả sai lệch. Vì vậy, biến furnishingstatus sẽ được mã hóa dummies để tạo các cột nhị phân độc lập (0 hoặc 1), không áp đặt mối quan hệ số học giữa các danh mục, đảm bảo mỗi danh mục được đối xử công bằng.

*** Mã code:**

```
# 1. Chuyen cac bien phan loai nhi phan thanh 1/0
binary_cols = ['mainroad', 'guestroom', 'basement',
'hotwaterheating', 'airconditioning', 'prefarea']
for col in binary_cols:
   df_log[col] = df_log[col].map({'yes': 1, 'no': 0, 'Yes': 1, 'No': 0})

# 2. Xu ly bien 'furnishingstatus' bang ma hoa dummies
# Ap dung dummies encoding
df_log = pd.get_dummies(df_log, columns=['furnishingstatus'], drop_first=True)

# Chuyen cac cot boolean thanh so nguyen (0/1)
boolean_cols = ['furnishingstatus_semi-furnished', 'furnishingstatus_unfurnished']
for col in boolean_cols:
   df_log[col] = df_log[col].astype(int)

# Hien thi 5 dong dau tien cua DataFrame
df_log.head()
```

*** Kết quả:**

	price	area	bedrooms	bathrooms	stories	parking	airconditioning	basement	prefarea	mainroad	guestroom	hotwaterheating	price_log	furnishingstatus_semi-furnished	furnishingstatus_unfurnished
0	9205000.0	7420	4.0	2.0	3.0	2.0	1	0	1	1	0	0	16.035257	0	0
1	9205000.0	8960	4.0	3.5	3.5	2.5	1	0	0	1	0	0	16.035257	0	0
2	9205000.0	9960	3.0	2.0	2.0	2.0	0	1	1	1	0	0	16.035257	1	0
3	9205000.0	7500	4.0	2.0	2.0	2.5	1	1	1	1	0	0	16.035257	0	0
4	9205000.0	7420	4.0	1.0	2.0	2.0	1	1	0	1	1	0	16.035257	0	0

Hình 15: Kết quả sau khi mã hóa dữ liệu phân loại

3.5.3 Chuẩn hóa dữ liệu

Việc chia tỷ lệ dữ liệu là cần thiết để đảm bảo các đặc trưng có cùng thang đo, từ đó cải thiện hiệu suất của mô hình học máy, đặc biệt với các thuật toán nhạy cảm với độ lớn của dữ liệu như hồi quy tuyến tính. Ba phương pháp chia tỷ lệ được thử nghiệm bao gồm: StandardScaler, MinMaxScaler, và RobustScaler để phân tích kết quả để lựa chọn phương pháp tối ưu.

*** Mã code:**

```
# Chuan hoa dac trung cho tat ca cac bien
# Xac dinh tat ca cac dac trung can chuan hoa
features = ['area', 'bedrooms', 'bathrooms', 'stories', 'parking',
         'mainroad', 'guestroom', 'basement', 'hotwaterheating',
         'airconditioning', 'prefarea', 'furnishingstatus_semi-furnished',
         'furnishingstatus_unfurnished']

# 1. StandardScaler (Chuan hoa Z-score)
scaler_std = StandardScaler()
df_log_std = df_log.copy()
df_log_std[features] = scaler_std.fit_transform(df_log[features])

# 2. MinMaxScaler (Chuan hoa ve khoang 0-1)
scaler_minmax = MinMaxScaler()
df_log_minmax = df_log.copy()
df_log_minmax[features] = scaler_minmax.fit_transform(df_log[features])

# 3. RobustScaler (Su dung trung vi va IQR, phu hop voi du lieu co ngoai le)
scaler_robust = RobustScaler()
df_log_robust = df_log.copy()
df_log_robust[features] = scaler_robust.fit_transform(df_log[features])

# So sanh cac thuoc tinh thong ke cua du lieu da chuan hoa
print("\nThong ke StandardScaler:")
df_log_std[features].describe().loc[['mean', 'std']].round(3)

print("\nThong ke MinMaxScaler:")
df_log_minmax[features].describe().loc[['min', 'max']].round(3)

print("\nThong ke RobustScaler:")
df_log_robust[features].describe().round(3)
```

*** Kết quả:**

Thống kê StandardScaler:

	area	bedrooms	bathrooms	stories	parking	mainroad	guestroom	basement	hotwaterheating	airconditioning	prefarea	furnishingstatus_semi-furnished	furnishingstatus_unfurnished
mean	0.000	0.000	-0.000	-0.000	0.000	-0.000	0.000	-0.000	-0.000	-0.000	0.000	-0.000	-0.000
std	1.001	1.001	1.001	1.001	1.001	1.001	1.001	1.001	1.001	1.001	1.001	1.001	1.001

Thống kê MinMaxScaler:

	area	bedrooms	bathrooms	stories	parking	mainroad	guestroom	basement	hotwaterheating	airconditioning	prefarea	furnishingstatus_semi-furnished	furnishingstatus_unfurnished
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
max	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Thống kê RobustScaler:

	area	bedrooms	bathrooms	stories	parking	mainroad	guestroom	basement	hotwaterheating	airconditioning	prefarea	furnishingstatus_semi-furnished	furnishingstatus_unfurnished
count	545.000	545.000	545.000	545.000	545.000	545.000	545.000	545.000	545.000	545.000	545.000	545.000	545.000
mean	0.182	-0.050	0.283	-0.231	0.686	-0.141	0.178	0.347	0.046	0.316	0.235	0.417	0.327
std	0.727	0.697	0.497	0.781	0.829	0.349	0.383	0.476	0.209	0.465	0.424	0.493	0.469
min	-1.069	-2.000	0.000	-1.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
25%	-0.362	-1.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
50%	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
75%	0.638	0.000	1.000	0.000	1.000	0.000	0.000	1.000	0.000	1.000	0.000	1.000	1.000
max	2.138	1.500	2.500	1.500	2.500	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Hình 16: Kết quả sau khi chuẩn hóa dữ liệu

*** Nhận xét:**

- Mean $\approx 0,$ Std ≈ 1 => phân phối đúng chuẩn.
- MinMaxScaler: Dữ liệu bị nén mạnh về khoảng [0, 1] => Rất dễ méo mó nếu có chỉ vài giá trị lớn, như area hoặc bathrooms có giá trị cực đại bất thường $\rightarrow$ ảnh hưởng toàn bộ scale.
- RobustScaler: Median = 0, IQR = 1 (tức là dữ liệu phân bố quanh trung vị, ổn định). Biến bathrooms hoặc parking có vẻ có một số giá trị lớn (như bathrooms lên đến 2.5) nhưng vẫn không làm lệch toàn bộ phân phối.

=> StandardScaler là phương pháp chia tỷ lệ phù hợp nhất cho tập dữ liệu giá nhà, nhờ khả năng chuẩn hóa dữ liệu về phân phối có trung bình 0 và độ lệch chuẩn 1, có thể hoạt động tốt vì hồi quy tuyến tính thường giả định các tính năng phân phối chuẩn. Mặc dù RobustScaler có lợi thế khi xử lý giá trị ngoại lai và MinMaxScaler đảm bảo thang đo đồng nhất, nhưng các đặc điểm của dữ liệu sau tiền xử lý và yêu cầu của mô hình khiến StandardScaler trở thành lựa chọn tối ưu. Phương pháp này sẽ được áp dụng trong các bước tiếp theo để xây dựng mô hình dự đoán giá nhà hiệu quả.

3.6 Giảm chiều dữ liệu

3.6.1 Kiểm tra hiện tượng đa cộng tuyến

Hệ số VIF (Variance Inflation Factor) được sử dụng để đánh giá mức độ đa cộng tuyến giữa các đặc trưng trong tập dữ liệu giá nhà. Đa cộng tuyến xảy ra khi các đặc trưng có mối quan hệ tuyến tính mạnh với nhau, có thể làm giảm độ chính xác và khả năng diễn giải của mô hình hồi quy tuyến tính. Một hệ số VIF cao cho thấy đặc trưng đó có mối

quan hệ tuyến tính mạnh với các đặc trưng khác.

*** Mã code:**

```
# Bo cot 'price' va 'price_log' vi chung la bien muc tieu, chi giu cac bien dau vao
X = df_std.drop(['price', 'price_log'], axis=1)

# Tao DataFrame de luu ket qua VIF
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in
    range(X.shape[1])]
# VIF > 5 hoac > 10 thuong cho thay da cong tuyen nghiem trong giua cac bien

# In ra danh sach cac bien va he so VIF tuong ung, sap xep theo gia tri VIF giam dan
print(" He so VIF (Variance Inflation Factor):")
print(vif_data.sort_values("VIF", ascending=False))
```

*** Kết quả:**

```
 Hệ số VIF (Variance Inflation Factor):
                            Feature       VIF
12     furnishingstatus_unfurnished  1.676785
11  furnishingstatus_semi-furnished  1.580599
3                           stories  1.512918
1                          bedrooms  1.458371
0                              area  1.365287
6                          basement  1.305193
2                         bathrooms  1.279021
4                           parking  1.223903
9                         guestroom  1.211555
5                   airconditioning  1.205716
8                          mainroad  1.181868
7                          prefarea  1.158430
10                 hotwaterheating  1.045219
```

Hình 17: Kiểm tra hiện tượng đa cộng tuyến

*** Nhận xét:**

- Tất cả các đặc trưng đều có giá trị VIF nhỏ hơn 5, với giá trị cao nhất là 1.676785 (furnishingstatusunfurnished) và thấp nhất là 1.045219 (hotwaterheating).
- Theo quy tắc chung, VIF < 5 cho thấy không có vấn đề đa cộng tuyến đáng kể. Các đặc trưng trong tập dữ liệu này không có mối quan hệ tuyến tính mạnh với nhau,

đảm bảo rằng mô hình hồi quy tuyến tính sẽ không bị ảnh hưởng bởi hiện tượng đa cộng tuyến.

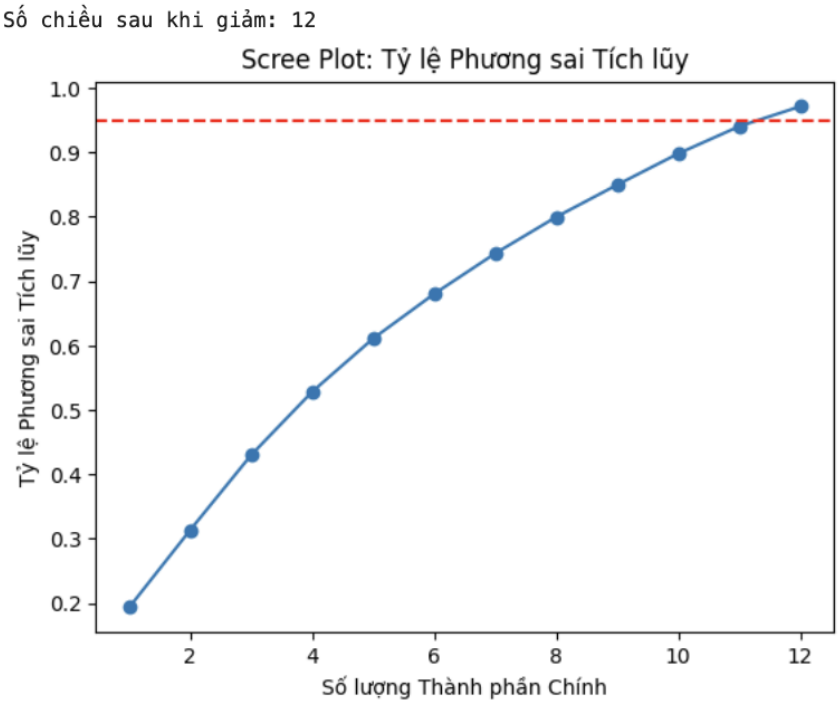
3.6.2 Giảm chiều dữ liệu bằng PCA (Principal Component Analysis)

PCA là một kỹ thuật biến đổi tuyến tính, tạo ra các thành phần chính (principal components) là các tổ hợp tuyến tính của các đặc trưng ban đầu. Phương pháp này sẽ giúp giảm chiều bằng cách loại bỏ tương quan giữa các đặc trưng, chuyển đổi chúng thành các thành phần chính độc lập. Điều này đặc biệt hữu ích trong tập dữ liệu giá nhà, vì các đặc trưng như area, bedrooms, và stories có thể tương quan với nhau.

*** Mã code:**

```
# Chay PCA
pca = PCA(n_components=0.95)
X_pca = pca.fit_transform(df_std[features])
print(f"So chieu sau khi giam: {X_pca.shape[1]}")

# Ve Scree Plot
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
plt.plot(range(1, len(cumulative_variance) + 1), cumulative_variance, marker='o')
plt.axhline(y=0.95, color='r', linestyle='--')
plt.xlabel('So luong Thanh phan chinh')
plt.ylabel('Ty le Phuong sai Tich luy')
plt.title('Scree Plot: Ty le Phuong sai Tich luy')
plt.show()
```

*** Kết quả:**



Hình 18: Biểu đồ Scree Plot

*** Nhận xét:** Biểu đồ cho thấy tỷ lệ phương sai tích lũy (cumulative variance ratio) tăng dần theo số lượng thành phần chính (principal components). Với 12 thành phần chính, tỷ lệ phương sai tích lũy đạt 0.95 (95%).

3.6.3 Tích chọn đăc trưng quan trọng bằng RandomForest

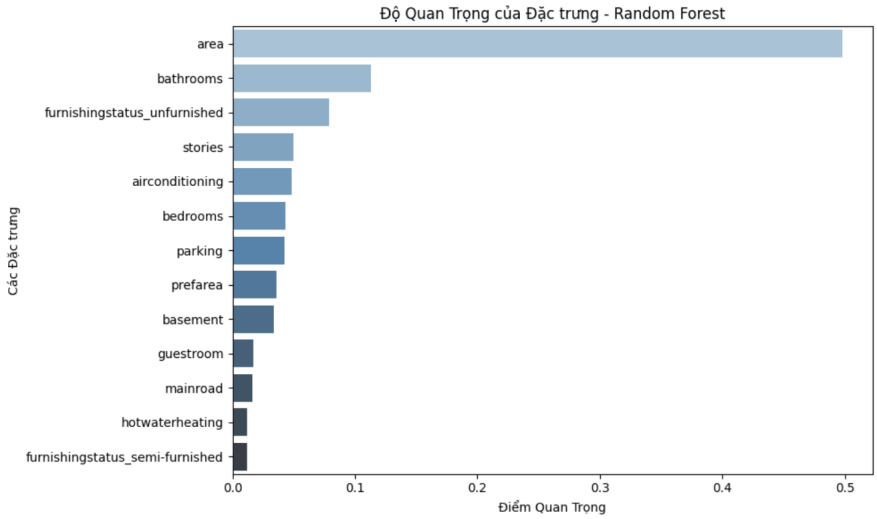
Phương pháp trích chọn đặc trưng quan trọng bằng Random Forest là một kỹ thuật thuộc nhóm lựa chọn đặc trưng (feature selection), nhằm xác định và giữ lại những đặc trưng có ảnh hưởng lớn nhất đến biến mục tiêu trong một tập dữ liệu. Phương pháp này tận dụng mô hình Random Forest để đánh giá mức độ quan trọng của từng đặc trưng dựa trên vai trò của chúng trong việc dự đoán kết quả.

*** Mã code:**

```
# Trich chon du lieu dau vao va dau ra
X_feature_selection = df_std.drop(columns=['price', 'price_log'])
y_feature_selection = df_std['price_log']

# Huan luyen mo hinh Random Forest de trich chon dac trung
model_feature_selection = RandomForestRegressor(n_estimators=100, random_state=42)
model_feature_selection.fit(X_feature_selection, y_feature_selection)

# Lay va sap xep do quan trong cua cac dac trung
```

```
feature_importance_scores = pd.Series(model_feature_selection.feature_importances_,
index=X_feature_selection.columns)
feature_importance_scores = feature_importance_scores.sort_values(ascending=False)

# Ve bieu do the hien do quan trong cua cac dac trung
plt.figure(figsize=(10, 6))
sns.barplot(x=feature_importance_scores.values, y=feature_importance_scores.index,
palette='Blues_d')
plt.title("Do Quan Trong cua Dac trung - Random Forest")
plt.xlabel("Diem Quan Trong")
plt.ylabel("Cac Dac trung")
plt.tight_layout()
plt.show()
```

*** Kết quả:**



Hình 19: Độ quan trọng của các đặc trưng

*** Nhận xét:** Biểu đồ độ quan trọng của các đặc trưng trong dự đoán giá nhà (pricelog), được tính toán dựa trên mô hình Random Forest, cung cấp cái nhìn rõ ràng về vai trò của từng đặc trưng:

- Các đặc trưng quan trọng nhất: area (diện tích) và bathrooms (số phòng tắm) nổi bật với mức độ quan trọng lần lượt khoảng 0.48 và 0.18. Đây là hai yếu tố chính quyết định giá trị nhà, phản ánh vai trò quan trọng của diện tích và tiện nghi trong thị trường bất động sản.
- Các đặc trưng có ảnh hưởng trung bình: furnishingstatus unfurnished, stories (số

tầng), airconditioning (điều hòa), bedrooms (số phòng ngủ), và parking (bãi đỗ xe) cũng góp phần vào dự đoán, nhưng với mức độ quan trọng dưới 0.1, cho thấy tác động của chúng ít hơn so với area và bathrooms.

- Các đặc trưng ít ảnh hưởng: prefarea (khu vực ưu tiên), basement (hầm), guestroom (phòng khách), mainroad (gần đường chính), hotwaterheating (nước nóng),... có độ quan trọng rất thấp (dưới 0.03), cho thấy chúng gần như không đáng kể trong việc dự đoán giá nhà.

Về cả 2 phương pháp, PCA đóng vai trò giảm chiều dữ liệu và loại bỏ tương quan giữa các đặc trưng, trong khi Random Forest hỗ trợ lựa chọn các đặc trưng quan trọng, đảm bảo mô hình vừa đạt hiệu quả dự đoán vừa giữ được tính minh bạch. Sự kết hợp này tối ưu hóa hiệu suất mô hình dự đoán giá nhà, cân bằng giữa độ chính xác và sự đơn giản trong phân tích.

Tuy nhiên, với số lượng 13 đặc trưng ban đầu, hiệu quả của việc giảm chiều bằng PCA không mang lại lợi ích đáng kể về mặt tính toán, vì số chiều giảm (từ 13 xuống 12 thành phần chính để giữ 95% phương sai) không đủ lớn để cải thiện rõ rệt tốc độ xử lý hoặc giảm nguy cơ overfitting. Do đó, việc áp dụng PCA có thể không thực sự cần thiết trong trường hợp này, trừ khi tập dữ liệu mở rộng đáng kể về số đặc trưng trong tương lai.

3.7 Xây dựng mô hình

Sử dụng mô hình hồi quy tuyến tính Linear Regression để kiểm chứng tác động của các bước tiền xử lý dữ liệu đến hiệu quả dự đoán giá nhà.

* Thiết lập mô hình:

- Thuật toán sử dụng: Linear Regression
- Biến mục tiêu: pricelog (sau khi log-transform)
- Biến đầu vào: Các đặc trưng đã được xử lý (làm sạch, chuẩn hóa, mã hóa, giảm chiều)
- Thang đo đánh giá:
  - Hệ số xác định R bình phương
  - MAE (Sai số tuyệt đối trung bình)

- MSE (Sai số bình phương trung bình)
- RMSE (Sai số căn bậc hai)

* Thiết lập tập huấn luyện và kiểm tra:

- Chia tập dữ liệu thành:
  - Tập huấn luyện (80%)
  - Tập kiểm tra (20%)
- Áp dụng phương pháp train-test-split với random-state = 42 để đảm bảo tính tái lập.

***Mã code:**

```
# === 1. Tach bien dau vao (X) va bien muc tieu (y) ===
X = df_std.drop(columns=['price_log', 'price'])
y = df_std['price_log']

# === 2. Chia tap du lieu thanh tap huan luyen va tap kiem tra ===
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.2, random_state=42
)

# === 3. Khoi tao va huan luyen mo hinh hoi quy tuyen tinh ===
from sklearn.linear_model import LinearRegression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# === 4. Du doan gia tri dau ra tren tap kiem tra ===
y_pred = lr_model.predict(X_test)

# === 5. Danh gia hieu suat mo hinh ===
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
import numpy as np

r2 = r2_score(y_test, y_pred)     # He so xac dinh R
mae = mean_absolute_error(y_test, y_pred) # Sai so tuyet doi trung binh
mse = mean_squared_error(y_test, y_pred) # Sai so binh phuong trung binh
rmse = np.sqrt(mse)               # Can bac hai cua MSE

# === 6. In ket qua ra man hinh ===
print(f"R Score (He so xac dinh): {r2:.4f}")
```

```
print(f"MAE (Sai so tuyet doi trung binh): {mae:.4f}")
print(f"MSE (Sai so binh phuong trung binh): {mse:.4f}")
print(f"RMSE (Sai so can bac hai): {rmse:.4f}")
```

*** Kết quả:**

```
R² Score (Hệ số xác định): 0.6568
MAE (Sai số tuyệt đối trung bình): 0.1959
MSE (Sai số bình phương trung bình): 0.0608
RMSE (Sai số căn bậc hai): 0.2466
```

Hình 20: Các chỉ số đánh giá mô hình

*** Nhận xét:** Kết quả huấn luyện mô hình hồi quy tuyến tính (Linear Regression) để dự đoán giá nhà (price log) cho thấy các chỉ số hiệu suất như sau:

- R² Score (Hệ số xác định): Đạt 0.6568, nghĩa là mô hình giải thích được khoảng 65.68% phương sai của biến mục tiêu (price log). Đây là một kết quả khá tốt, cho thấy mô hình có khả năng dự đoán giá nhà ở mức chấp nhận được, nhưng vẫn còn khoảng 34.32% phương sai chưa được giải thích, có thể do các yếu tố chưa được xem xét trong dữ liệu hoặc hạn chế của mô hình tuyến tính khi xử lý mối quan hệ phi tuyến.
- MAE (Sai số tuyệt đối trung bình): 0.1959, nghĩa là sai số trung bình giữa giá trị dự đoán và thực tế (trên thang logarit) là khoảng 0.1959. Giá trị này cho thấy mô hình có độ chính xác tương đối cao, nhưng sai số vẫn đáng kể, đặc biệt khi chuyển đổi ngược từ price log về giá trị gốc, sai số này có thể được khuếch đại do tính chất của phép biến đổi logarit.
- MSE (Sai số bình phương trung bình): 0.0608, cho thấy sai số bình phương trung bình giữa giá trị dự đoán và thực tế ở mức thấp. Tuy nhiên, chỉ số này nhạy cảm với các sai số lớn, nên cần xem xét thêm các chỉ số khác để đánh giá toàn diện.
- RMSE (Sai số căn bậc hai trung bình): 0.2466, tương đối gần với MAE (0.1959), cho thấy không có nhiều sai số lớn bất thường trong dự đoán. RMSE cũng ở mức chấp nhận được, nhưng tương tự MAE, khi chuyển đổi ngược về giá trị gốc, sai số này có thể lớn hơn đáng kể.

  *** So sánh mô hình:**

Chỉ số đánh giá	Mô hình tiền xử lý đầy đủ (1)	Mô hình tiền xử lý đơn giản (2)	So sánh và nhận xét
R^2 (Hệ số xác định)	0.6568	0.5939	Mô hình 1 tốt hơn - giải thích được nhiều phương sai hơn
MAE	0.1959	0.2035	Mô hình 1 tốt hơn - sai số tuyệt đối trung bình thấp hơn
MSE	0.0608	0.0678	Mô hình 1 tốt hơn - sai số bình phương thấp hơn
RMSE	0.2466	0.2603	Mô hình 1 tốt hơn - sai số trung bình sau căn nhỏ hơn
Ước lệch trung bình	$\approx 21\%(exp(0.1950)-1)$	$\approx 22.6\%(exp(0.2035)-1)$	Mô hình 1 dự đoán gần đúng hơn về giá trị gốc

4 Kết luận

4.1 Tổng kết kết quả

Báo cáo đã trình bày quy trình tiền xử lý dữ liệu (Data Preprocessing) toàn diện cho bộ dữ liệu bất động sản (Housing Prices Dataset), từ khâu làm sạch, tích hợp, giảm chiều, đến biến đổi dữ liệu, nhằm tối ưu hóa hiệu suất của mô hình dự đoán giá nhà. Các kết quả chính thu được như sau:

- **Làm sạch dữ liệu**:
  - Giá trị thiếu được xử lý bằng các phương pháp phù hợp với từng loại biến: KNN Imputer cho biến số, Classification Imputer (MICE + Random Forest) cho biến phân loại, và Regression Imputer cho biến mục tiêu price.
  - Giá trị ngoại lai được điều chỉnh bằng phương pháp IQR, giảm skewness mà không làm mất thông tin quan trọng.
  - Kiểm tra phân phối bằng Shapiro-Wilk và Q-Q plot cho thấy dữ liệu không chuẩn, cần biến đổi log để phù hợp với mô hình tuyến tính.
- **Biến đổi dữ liệu tối ưu**:
  - Log Transformation giúp chuẩn hóa phân phối của biến mục tiêu price, giảm độ lệch từ 1.38 xuống 0.01.
  - Mã hóa biến phân loại bằng Dummy Encoding cho furnishingstatus và Nhị phân (0/1) cho các biến yes/no, đảm bảo tính khách quan.

- StandardScaler được lựa chọn để chuẩn hóa dữ liệu số, cân bằng ảnh hưởng giữa các đặc trưng.

– **Giảm chiều dữ liệu**:

- Phân tích VIF (<5) và PCA (95% phương sai với 12 thành phần) xác nhận không có đa cộng tuyến nghiêm trọng.
- Random Forest chỉ ra area và bathrooms là hai đặc trưng quan trọng nhất, giải thích 66% phương sai của giá nhà.

– **Hiệu quả mô hình**:

- Mô hình Linear Regression sau tiền xử lý đạt $R^2 = 0.659$, MAE = 0.195 (sai số $\sim$21.5%), thể hiện khả năng dự đoán ổn định.

4.2 Hạn chế và định hướng phát triển

– **Hạn chế về dữ liệu**:

- Kích thước mẫu khiêm tốn (n=545) có thể ảnh hưởng đến độ tin cậy của kết quả
- Phạm vi địa lý giới hạn chưa phản ánh đa dạng các phân khúc thị trường

– **Hạn chế phương pháp**:

- Chưa tối ưu hóa siêu tham số cho các bộ imputer và mô hình hồi quy
- Giới hạn ở mô hình Linear Regression chưa khai thác các thuật toán hiện đại

– **Định hướng phát triển**:

- Mở rộng nghiên cứu với các bộ dữ liệu đa khu vực và đa thời điểm
- Ứng dụng các phương pháp deep learning cho bài toán imputation
- Phát triển hệ thống tự động hóa quy trình tiền xử lý
- Tích hợp thêm các nguồn dữ liệu bổ sung như hình ảnh và GIS

4.3 Tầm quan trọng thực tiễn

Tiền xử lý dữ liệu là bước nền tảng và có vai trò quyết định trong toàn bộ quy trình phân tích dữ liệu và xây dựng mô hình học máy. Chất lượng của dữ liệu đầu vào trực

tiếp ảnh hưởng đến độ tin cậy và hiệu quả của mô hình đầu ra. Nếu dữ liệu chưa được xử lý kỹ lưỡng, mô hình sẽ dễ mắc sai lệch, dẫn đến những kết luận không chính xác.

Giai đoạn tiền xử lý không chỉ giúp làm sạch, chuẩn hóa và biến đổi dữ liệu về định dạng phù hợp (thường là dạng số, không chứa ngoại lệ) mà còn là cầu nối quan trọng giữa dữ liệu thực tiễn và yêu cầu kỹ thuật của thuật toán. Việc phát hiện và xử lý kịp thời các vấn đề như nhiễu, giá trị thiếu hay phân bố không đồng đều sẽ giúp tiết kiệm thời gian, công sức và tránh phải lặp lại toàn bộ quy trình phân tích.

Trong lĩnh vực phân tích bất động sản, nghiên cứu cho thấy tiền xử lý dữ liệu đóng vai trò thiết yếu, cụ thể là:

- Cải thiện độ chính xác trong việc dự đoán giá bất động sản.
- Giảm thiểu sai số do dữ liệu bị nhiễu, thiếu hoặc sai định dạng.
- Cung cấp góc nhìn đầy đủ và có hệ thống về các yếu tố ảnh hưởng đến giá nhà như diện tích, số phòng tắm, vị trí địa lý,...

Tài liệu

[1] Anaconda. (2020). *State of Data Science 2020*. [Online] Available: https://www.anaconda.com/state-of-data-science-2020

[2] Lê Diên Tuấn (2023). *Bài giảng học phần Phân tích dữ liệu bằng Python*. Khoa Thương mại Điện tử, Đại học Kinh tế - Đại học Đà Nẵng.

LIÊN KẾT VIDEO BÁO CÁO

Tên video: Group-6-48K29.1 - Báo cáo đề tài Tiền xử lý dữ liệu - Data Preprocessing

https://www.youtube.com/watch?v=HV88o5z6zqU