UỶ BAN NHÂN DÂN

THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC SÀI GÒN

BÁO CÁO MÔN HỌC PHÂN TÍCH DỮ LIỆU
Đề tài: ÁP DỤNG MÔ HÌNH LINEAR REGRESSION DỰ ĐOÁN GIÁ NHÀ

Thành viên tham gia:

Dương Ngọc Nguyên - 3121411148
Hồ Đăng Hoàng - 3121411076
Đặng Anh Tú - 3121411219

Giáo viên hướng dẫn:

Thầy Trịnh Tấn Đạt
Cô Nguyễn Thị Tuyết Nam

TP.Hồ Chí Minh, Tháng 5 Năm 2024

Lời cảm ơn

Trong quá trình học tập và làm việc nhóm chúng em đã gặp một số khó khăn và nhiều thắc mắc nhưng vẫn được sự tận tâm dạy dỗ và giúp đỡ từ thầy Đạt và cô Nam rất nhiều, chúng em vô cùng biết ơn và trân trọng sự giúp đỡ của thầy và cô đã không ngại khó khăn vẫn luôn giúp đỡ hỗ trợ chúng em trong quá trình học tập.

Song tuy nhiên, nhóm chúng em nhận thức rằng trong quá trình học tập và làm việc, chúng em vẫn còn mắc phải một số lỗi và thiếu sót. Nhưng nhờ vào sự kiên nhẫn và sự chỉ dạy tận tình của thầy Đạt và cô Nam, chúng em đã có cơ hội học hỏi và khắc phục những khuyết điểm đó.

Sự quan tâm và sự hỗ trợ không ngừng nghỉ từ phía thầy và cô đã là nguồn động viên lớn lao giúp chúng em vượt qua những thách thức trong quá trình nắm bắt kiến thức và kỹ năng mới. Bằng cách này, chúng em đã có thể tiến bộ và phát triển không chỉ trong lĩnh vực học thuật mà còn trong việc phát triển bản thân.

Lời mở đầu

Trong lĩnh vực Phân tích khai phá dữ liệu(Data Exploratory Analysis) và Máy học(Machine Learning), mô hình Linear Regression(Hồi qui tuyến tính) là một công cụ cực kỳ phổ biến và mạnh mẽ được sử dụng để dự đoán giá trị của một biến phụ thuộc dựa trên các biến độc lập khác. Trong môi trường bất động sản, linear regression có thể được áp dụng để dự đoán giá nhà dựa trên các yếu tố như diện tích, vị trí, số phòng, và các yếu tố khác.

Mô hình Linear Regression hoạt động dựa trên giả định rằng có một mối quan hệ tuyến tính giữa biến phụ thuộc và các biến độc lập. Điều này có nghĩa là giá trị của biến phụ thuộc có thể được biểu diễn dưới dạng một tổ hợp tuyến tính của các biến độc lập, cộng với một hệ số chặn (Intercept).

Trong trường hợp dự đoán giá nhà, các biến độc lập có thể bao gồm diện tích, số phòng ngủ, số phòng tắm, vị trí, tiện ích xung quanh, và nhiều yếu tố khác. Mô hình linear regression có thể học từ dữ liệu huấn luyện để tìm ra mối quan hệ tuyến tính giữa các biến độc lập này và giá nhà, từ đó có thể dự đoán giá của các căn nhà mới dựa trên các yếu tố tương tự.

Trong báo cáo này, chúng ta sẽ khám phá sâu hơn về cách mô hình Linear Regression được áp dụng vào dự đoán giá nhà, bao gồm cách xây dựng mô hình, đánh giá hiệu suất, và ứng dụng thực tế trong lĩnh vực bất động sản.

Mục lục

Danh sách các hình

Chương 1

Tổng quan

1.1 Giới thiệu

1.1.1 Nhu cầu thực tế

Trong thời đại ngày nay, nhu cầu về dự đoán giá nhà đã trở thành một yếu tố quan trọng trong thị trường bất động sản, khi mà xã hội đang chứng kiến sự phát triển vượt bậc và mở rộng. Như một phản ứng tự nhiên của sự phát triển này, nhu cầu về các công cụ dự đoán giá nhà cũng đã tăng lên.

Trong cuộc sống hiện đại, việc đưa ra dự đoán chính xác về giá trị của một căn nhà là rất quan trọng đối với cả người mua và người bán. Điều này giúp người mua đưa ra quyết định thông minh về việc đầu tư vào bất động sản, đồng thời hỗ trợ người bán trong việc định giá phù hợp với thị trường.

Nhu cầu thực tế của việc dự đoán giá nhà không chỉ giúp cho các nhà đầu tư và nhà môi giới bất động sản đưa ra quyết định thông minh về việc mua và bán, mà còn hỗ trợ người mua nhà trong việc đánh giá tính hợp lý của giá bán. Điều này giúp tạo ra một thị trường bất động sản đáng tin cậy và minh bạch hơn.

Trong bối cảnh nhu cầu ngày càng tăng của các cá nhân và tổ chức đối với dự đoán chính xác về giá nhà, các công cụ và phương pháp dự đoán giá nhà đóng vai trò quan trọng trong việc cung cấp một cơ sở thông tin đáng tin cậy và hỗ trợ quyết định.

1.1.2 Bài toán dự đoán giá nhà

Dự đoán giá nhà là một bài toán trong lĩnh vực dự đoán giá cả bất động sản. Hệ thống có thể được huấn luyện để phân tích các đặc điểm của một căn nhà và dự đoán giá trị của nó dựa trên các yếu tố như diện tích, vị trí địa lý, tiện ích xung quanh, năm xây dựng, và các yếu tố khác.

1.1.3 Các ứng dụng hiện nay

Bài toán dự đoán giá nhà có thể được áp dụng rộng rãi trong trong cuộc sống đắt đỏ ngày nay để người dân có thể đưa ra một quyết định chuẩn xác để có thể mua bán với giá trị minh bạch và chính xác nhất, nó thu hút nhiều nhà đầu tư trong thời đại hiện nay và một số ứng dụng thực tiễn trong cuộc sống:

- Hệ thống định giá bất động sản: Linear regression sẽ cố gắng xây dựng một mô hình tuyến tính để dự đoán giá trị của các căn nhà dựa trên các biến đầu vào như diện tích, vị trí, tiện ích xung quanh, và các yếu tố khác, cung cấp một cái nhìn tổng quan hơn về mối liên hệ tuyến tính giữa chúng.

- Hệ thống đầu tư bất động sản: Sử dụng để ước tính lợi nhuận từ các dự án đầu tư bằng cách dự đoán giá trị tương lai của các căn nhà dựa trên các yếu tố kinh tế và thị trường.

- Hệ thống tư vấn mua bán nhà: Sử dụng để tư vấn cho khách hàng về giá trị của các căn nhà và đưa ra các gợi ý mua bán dựa trên dự đoán về giá nhà.

- Hệ thống định giá cho vay nhà: Linear regression có thể được sử dụng để đánh giá khả năng trả nợ của người vay và xác định giá trị tài sản cần thiết để đảm bảo cho vay. Tuy nhiên, điều này có thể bị hạn chế nếu mô hình không mô tả được các mối quan hệ phi tuyến tính hoặc phức tạp giữa các biến đầu vào.

- Hệ thống dự đoán xu hướng thị trường: Được sử dụng để dự đoán xu hướng thị trường bất động sản và đưa ra các chính sách phù hợp. Tuy nhiên, điều này có thể đòi hỏi mô hình phải được cập nhật thường xuyên để phản ánh các biến đổi phức tạp trong thị trường.

1.1.4 Những khó khăn của bài toán trong dự đoán giá nhà

- Khả năng biểu diễn mối quan hệ phức tạp: Linear regression có thể không thể biểu diễn được các mối quan hệ phức tạp hoặc phi tuyến tính giữa các biến đầu vào và giá nhà.

- Điều kiện độc lập và phân phối chuẩn: Mô hình linear regression yêu cầu rằng các biến đầu vào độc lập với nhau và phân phối chuẩn. Trong thực tế, có thể có sự tương quan giữa các biến đầu vào hoặc các biến có phân phối không chuẩn, điều này có thể ảnh hưởng đến hiệu suất của mô hình.

- Outliers(dữ liệu ngoại lai) và dữ liệu nhiễu: Các outliers và nhiễu trong dữ liệu có thể ảnh hưởng đến khả năng dự đoán của mô hình linear regression. Linear Regression nhạy cảm với nhiễu, và các outliers có thể làm biến đổi mô hình dự đoán một cách không mong muốn.

- Overfitting hoặc underfitting: Linear regression có thể dễ bị overfitting (quá mức) hoặc underfitting (thiếu mức) nếu không được điều chỉnh chính xác. Overfitting xảy ra khi mô hình quá phức tạp và chỉ tốt trên dữ liệu huấn luyện mà không tốt trên dữ liệu mới, trong khi underfitting xảy ra khi mô hình quá đơn giản và không thể mô hình hóa được các mẫu dữ liệu.

Tuy bài toán có nhiều khó khăn khách quan lẫn chủ quan gặp phải trong quá trình xây dựng nhưng đây là một bài toán đủ sức đem lại sự hiệu quả và đưa ra quyết định tốt hơn cho con người chọn lựa. Tuy còn nhiều hạn chế nhưng ít nhiều nó đã đóng góp một phần nào đó cho con người và kinh tế chúng ta. Hầu hết các nghiên cứu đều cố gắng giải quyết bài toán từ trường hợp đơn giản dễ phân tích để dự đoán nhất rồi dần dần phát triển thuật toán để khắc phục những khó khăn mà đã được nêu lên ở trên đưa ra một kết quả với mức độ chính xác tốt nhất có thể…

Chương 2

Cơ sở lý thuyết

2.1 Giới thiệu

Linear Regression là một thuật toán học có giám sát (supervised learning) trong Machine Learning, nó là một phương pháp nó là một phương pháp thống kê dùng để ước lượng mối quan hệ giữa các biến độc lập (input features) và biến phụ thuộc (output target). Linear Regression giả định rằng sự tương quan giữa các biến là tuyến tính, từ đó tìm ra hàm tuyến tính tốt nhất để biểu diễn mối quan hệ này. Thuật toán này dự báo giá trị của biến output từ các giá trị của các biến đầu vào.

2.1.1 Liner Regression

Ý tưởng

Nảy sinh ý tưởng: Ý tưởng về Linear Regression có thể nảy sinh từ nhu cầu của con người trong việc hiểu và dự đoán các mối quan hệ giữa các biến. Trong thời gian lâu dài, con người luôn quan tâm đến việc dự đoán và ước tính giá trị dựa trên dữ liệu có sẵn.

Phát triển ý tưởng ban đầu: Những nhà toán học và nhà thống kê đã bắt đầu phát triển ý tưởng về mối quan hệ tuyến tính giữa các biến và cách biểu diễn nó bằng các mô hình toán học.

Nghiên cứu lý thuyết: Nghiên cứu lý thuyết về các phương pháp thống kê và mô hình hóa dữ liệu đã dẫn đến sự phát triển của Linear Regression và các phương pháp tương tự.

Một số áp dụng trong thực tế:

- Dự báo giá cả: dự đoán giá nhà, giá cổ phiếu, giá nhiên liệu dựa trên các yếu tố như vị trí, kích thước, chất lượng, lượng cung cầu, …

- Dự báo điểm số: Dự đoán điểm số của học sinh dựa trên thời gian học, sự nổ lực, kỹ năng, và trình độ của giáo viên, …

- Dự báo sản phẩm: Dự đoán đầu ra sản xuất dựa trên thời gian, công suất, nguyên vật liệu, lao động, ….

- Phân tích chuỗi thời gian: dự đoán xu hướng và chu kỳ của các chuỗi dữ liệu như bất động sản, thời tiết, xu hướng sản xuất, …

Phát triển và cải tiến: Linear Regression không ngừng được phát triển và cải tiến qua các nghiên cứu và ứng dụng thực tế. Các phương pháp mới và các biến thể của linear regression được phát triển để xử lý các vấn đề phức tạp hơn trong dữ liệu thực tế.

2.1.2 Mô hình toán học

1 Multivariable linear regression

$$y \approx f(x) = y\hat{}$$
$$f(x) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_0 \quad (1)$$

Trong đó, w_1, w_2, w_3, w_0 là các hệ số tương ứng với từng biến độc lập $x1, x2, x3$ và w_0 còn được gọi là bias hay là hệ số chặn(intercept). Mối quan hệ $y \approx f(x)$ bên trên là một mối quan hệ tuyến tính (Linear). Bài toán chúng ta đang làm là một bài toán thuộc loại Regression(Hồi qui). Bài toán đi tìm các hệ số tối ưu $\{w_1, w_2, w_3, w_0\}$ chính vì vậy được gọi là bài toán Linear Regression.

Trong phương trình (1) phía trên nếu chúng ta đặt $w=[w_1,w_2,w_3,w_0]^T$ là vector (cột) hệ số cần phải tối ưu

và $\bar{\bar{\mathbf{x}}}=[\mathbf{1}, \mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}]$ (đọc là *x bar* trong tiếng Anh) là vector (hàng) dữ liệu đầu vào *mở rộng*. Số 11 ở đầu được thêm vào để phép tính đơn giản hơn và thuận tiện cho việc tính toán. Khi đó, phương trình (1) có thể được viết lại dưới dạng:

$$\boldsymbol{y} \approx \bar{\bar{\mathbf{x}}}\mathbf{w} = \mathbf{y}\hat{}$$

2 Sai số dự đoán

Giá trị sai số càng nhỏ càng tốt

$$\frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{e}^{\mathbf{2}} = \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{y} - \boldsymbol{y}\hat{})^{\mathbf{2}} = \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{y} - \bar{\boldsymbol{x}}\boldsymbol{w})^{\mathbf{2}}$$

Trong đó hệ số 1/2 là để thuận tiện cho việc tính toán (khi tính đạo hàm thì số 1/2 sẽ bị triệt tiêu). Chúng ta cần e^2 vì $e=y-y\hat{}$ có thể là một số âm, việc nói e nhỏ nhất sẽ không đúng vì khi $e=-\infty$ là rất nhỏ nhưng sự sai lệch là rất lớn.

3 Hàm mất mát

Điều tương tự xẩy ra với tất cả các cặp (input, out come) (x_i, y_i), $I = 1, 2, \ldots, N$, với N là số lượng dữ liệu quan sát được. Điều chúng ta muốn tìm w để hàm số nhỏ nhất:

$$\mathcal{L}(\mathbf{w}) = \frac{\mathbf{1}}{\mathbf{2}}\sum_{\mathbf{i=1}}^{\mathbf{N}}(\mathbf{y_i} - \bar{\mathbf{x}}_\mathbf{i}\mathbf{w})^{\mathbf{2}} \quad (\mathbf{2})$$

Hàm $\mathcal{L}(w)$ được gọi là hàm mất mát (loss function) của bài toán Linear Regression. Chúng ta luôn mong muốn rằng sự mất (sai số) là nhỏ nhất, điều đó đồng nghĩa với việc tìm vector hệ số **w** sao cho giá trị của hàm mất mát này càng nhỏ càng tốt. Giá trị của **w** làm cho hàm mất mát đạt giá trị nhỏ nhất được gọi là *điểm tối ưu* (optimal point), ký hiệu:

$$\boldsymbol{w}^* = \mathbf{arg\,min}\mathcal{L}(\mathbf{w})$$

Trước khi đi tìm lời giải, chúng ta đơn giản hóa phép toán trong phương trình hàm mất mát (2). Đặt $\mathbf{y} = [\mathbf{y_1}; \mathbf{y_2}; \ldots; \mathbf{y_N}]$ là một

vector cột chứa tất cả các *output* của *training data*; $\bar{\bar{\mathbf{x}}}=[\bar{\bar{\mathbf{x}}}_1;\bar{\bar{\mathbf{x}}}_2;\ldots;\bar{\bar{\mathbf{x}}}]$ là ma trận dữ liệu đầu vào (mở rộng) mà mỗi hàng của nó là một điểm dữ liệu. Khi đó hàm số mất mát $L(\mathrm{w})$ được viết dưới dạng ma trận đơn giản hơn:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(\mathbf{y_i} - \bar{\mathbf{x}}_\mathbf{i}\mathbf{w})^2 = \frac{1}{2}\| \mathbf{y} - \bar{\mathbf{X}}\mathbf{w} \|_2^2 \quad (3)$$

Với $\|z\|_2$ là Euclidean norm (chuẩn Euclid, hay khoảng cách Euclid), nói cách khác $\| z \|_2^2$ là tổng của bình phương mỗi phần tử của vector $\mathbf{z}$. Tới đây, ta đã có một dạng đơn giản của hàm mất mát được viết như phương trình (3).

4 Nghiệm cho bài toán Linear Regression

Đạo hàm theo $\mathbf{w}$ của hàm mất mát là:

$$\frac{\partial \mathcal{L}(\mathrm{w})}{\partial \mathbf{w}} = \bar{\mathbf{X}}^\mathbf{T}(\bar{\mathbf{X}}\mathbf{w} - \mathbf{y})$$

Phương trình đạo hàm bằng 0 tương đương với

$$\bar{\mathbf{X}}^\mathbf{T}\bar{\mathbf{X}}\mathbf{w} = \bar{\mathbf{X}}^\mathbf{T}\mathbf{y} \triangleq \mathbf{b} \quad (4)$$

(Ký hiệu $\bar{\mathbf{X}}^\mathbf{T}\mathbf{y} \triangleq \mathbf{b}$ nghĩa là đặt $\bar{\mathbf{X}}^\mathbf{T}\mathbf{y}$ bằng $\mathbf{b}$)

Nếu ma trận vuông $\mathbf{A} \triangleq \bar{\mathbf{X}}^\mathbf{T}\bar{\mathbf{X}}$ khả nghịch (non-singular hay invertible) thì phương trình **(4)** có nghiệm duy nhất: $\mathbf{w = A^{-1}b}$

Với khái niệm nghịch đảo, điểm tối ưu của bài toán Linear Regression có dạng:

$$\mathbf{w} = \mathbf{A}\dagger\ \mathbf{b} = (\bar{\mathbf{X}}^\mathbf{T}\bar{\mathbf{X}})\dagger\ \bar{\mathbf{X}}^\mathbf{T}\mathbf{y} \quad (5)$$

2.1.3 Ví dụ về bài toán Linear Regression bằng Python

Ta sẽ so sánh nghiệm của bài toán khi giải theo phương trình (5) và tìm được nghiệm khi dùng thư viện scikit-learn của Python. Ta sẽ dùng bảng dữ liệu cân nặng của hình 2.1 dưới đây.

Chiều cao (cm)	Cân nặng (kg)	Chiều cao (cm)	Cân nặng (kg)
147	49	168	60
150	50	170	72
153	51	173	63
155	52	175	64
158	54	178	66
160	56	180	67
163	58	183	68
165	59		

Hình 2.1 Bảng dữ liệu về cân nặng và chiều cao

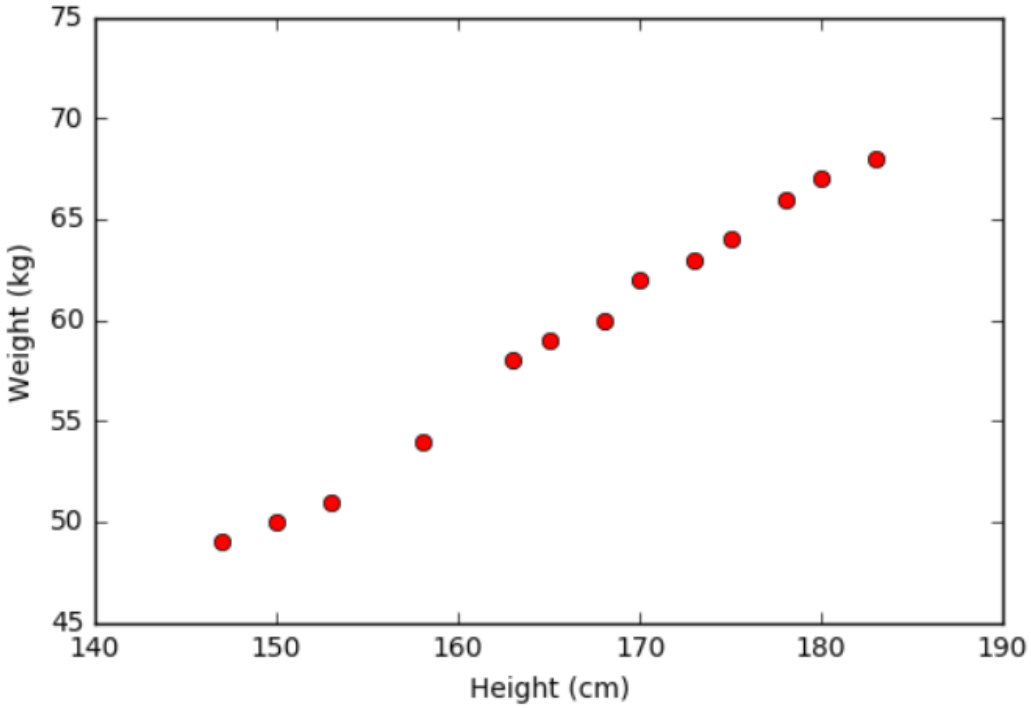
Cân nặng tỉ lệ thuận với chiều cao nên ta có thể sử dụng mô hình Linear Regression để kiểm tra độ chính xác của mô hình này tìm được, chúng ta sẽ dùng cột 155 và 160 cm để kiểm thử. Các cột còn lại ta sẽ dùng để huấn luyện cho mô hình.

Trước tiên ta sẽ dùng hai thư viện numpy và matplotlib cho việc vẽ hình.

```
# To support both python 2 and python 3
from __future__ import division, print_function, unicode_literals
import numpy as np
import matplotlib.pyplot as plt
```

Ta cho đầu vào dữ liệu và vẽ đồ thị biểu diễn

```
# height (cm)
X = np.array([[147, 150, 153, 158, 163, 165, 168, 170, 173, 175, 178, 180, 183]]).T
# weight (kg)
y = np.array([[ 49, 50, 51,  54, 58, 59, 60, 62, 63, 64, 66, 67, 68]]).T
# Visualize data
plt.plot(X, y, 'ro')
plt.axis([140, 190, 45, 75])
plt.xlabel('Height (cm)')
plt.ylabel('Weight (kg)')
plt.show()
```



Hình 2.2 Đồ thị biểu diễn phân bố dữ liệu

Dữ liệu trên được sắp xếp theo gần như một đường thẳng nên mô hình Linear Regression khả năng cao sẽ cho một kết quả tốt.

Ta có công thức: Cân nặng = w_1* (chiều cao) + w_0

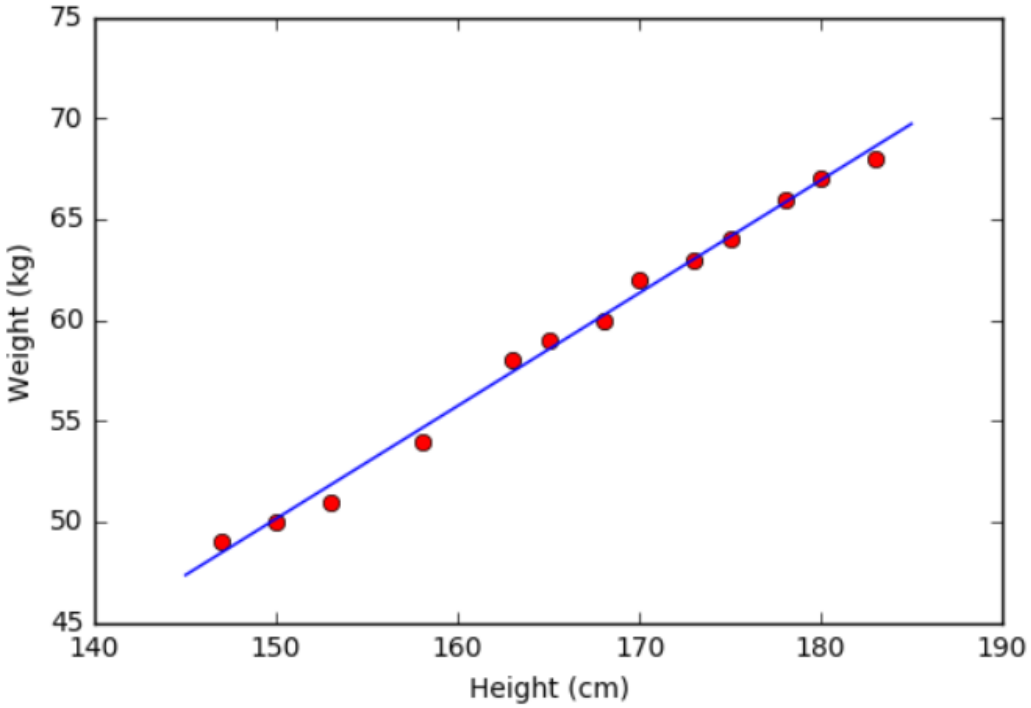
Dựa vào công thức (5) để tính toán các hệ số w_1 và w_0 để tính bằng numpy.linalg.pinv(A)

```
# Building Xbar
one = np.ones((X.shape[0], 1))
Xbar = np.concatenate((one, X), axis = 1)

# Calculating weights of the fitting line
A = np.dot(Xbar.T, Xbar)
b = np.dot(Xbar.T, y)
w = np.dot(np.linalg.pinv(A), b)
print('w = ', w)
# Preparing the fitting line
w_0 = w[0][0]
w_1 = w[1][0]
x0 = np.linspace(145, 185, 2)
y0 = w_0 + w_1*x0

# Drawing the fitting line
plt.plot(X.T, y.T, 'ro')     # data
plt.plot(x0, y0)               # the fitting line
plt.axis([140, 190, 45, 75])
plt.xlabel('Height (cm)')
plt.ylabel('Weight (kg)')
plt.show()
```

```
w =  [[-33.73541021]
 [  0.55920496]]
```



Hình 2.3 Phương trình tuyến tính sau khi huấn luyện

Từ hình 2.3 ta thấy các điểm dữ liệu màu đỏ nằm khá gần đường thẳng dự đoán màu xanh. Mô hình Linear Regression hoạt động tốt với tập dữ liệu huấn luyện. Bây giờ ta sẽ sử dụng mô hình này để dự đoán cân nặng của hai người có chiều cao 155 và 160cm mà khi nãy ta đã không dùng để huấn huyện.

Ta sẽ sử dụng thư viện scikit-learn của Python để tìm nghiệm.

```
from sklearn import datasets, linear_model

# fit the model by Linear Regression
regr = linear_model.LinearRegression(fit_intercept=False) # fit_intercept = False for calculating the b
regr.fit(Xbar, y)

# Compare two results
print( 'Solution found by scikit-learn  : ', regr.coef_ )
print( 'Solution found by (5): ', w.T)
```

Hình 2.4 Thư viện scikit-learn

Chúng ta nhận được hai kết quả thu được như nhau

```
Solution found by scikit-learn  :  [[  -33.73541021 0.55920496]]
Solution found by (5):  [[  -33.73541021 0.55920496 ]]
```

Hình 2.5 Kết quả thu được sau khi đưa vào dự đoán

Chương 3

Phương pháp đề xuất

3.1 Các bước tiền xử lý dữ liệu

3.1.1 Khám phá cấu trúc dữ liệu

Bước đầu tiên là phân tích khám phá, giúp hiểu về cấu trúc dữ liệu. Thường thì bước này bị bỏ qua, nhưng đó là lỗi tồi tệ nhất vì sau này sẽ làm cho mô hình cho ra lỗi hoặc không hoạt động như mong đợi.

Chia việc phân tích khám phá thành ba phần:

- Kiểm tra cấu trúc của bộ dữ liệu, các thống kê, các giá trị thiếu, các bản sao, các giá trị duy nhất của các biến phân loại.
- Hiểu ý nghĩa và phân phối của các biến.
- Nghiên cứu mối quan hệ giữa các biến.

Để phân tích cách bộ dữ liệu được tổ chức, có các phương pháp Pandas sau:

- print(df.head())
  - Thể hiện 5 dòng đầu tiên của tập dữ liệu
- print(df.info())
  - Thể hiện cấu trúc dữ liệu của tập dữ liệu
- print(df.isnull().sum())
  - Thể hiện tổng hợp số lượng cột dòng dữ liệu bị mất mát
- print(df.duplicated().sum())
  - Thể hiện giá các dữ liệu bị trùng lắp
- print(df.describe())
  - Thể hiện chi tiết kiểu dữ liệu của từng cột đặc trưng

3.1.2 Xử lí dữ liệu thiếu, mất mát

Trong bước đầu tiên, chúng ta đã sơ lượt xem có giá trị thiếu trong mỗi cột không. Trong trường hợp có giá trị thiếu, chúng ta cần phải xử lí vấn đề này thật kỹ. Cách đơn giản nhất có thể là loại bỏ các biến hoặc các hàng chứa giá trị không hợp lệ hay còn gọi là NaN, nhưng đôi khi dữ liệu bị thiếu chiếm không ít phần thi việc loại bỏ sẽ gây mất mát nhiều dữ liệu cho việc huấn luyện và chúng ta sẽ có các cách khác bên dưới:

- Loại bỏ dữ liệu null: Phương pháp đơn giản nhất là loại bỏ các quan sát hoặc biến chứa giá trị null khỏi tập dữ liệu của bạn. Tuy nhiên, điều này có thể dẫn đến mất mát thông tin đáng kể.

- Lấp đầy dữ liệu null: Bạn có thể điền vào các giá trị null bằng các giá trị thay thế như mean, median, mode hoặc giá trị cố định. Việc này giúp giữ lại dữ liệu và không gây mất mát lớn. Phương pháp này thường được sử dụng cho các biến số.

Nếu chúng ta đang xử lí một biến số, có một số phương pháp để điền vào giá trị thiếu. Phương pháp phổ biến nhất là điền vào các giá trị thiếu bằng giá trị trung bình/median của cột đó:

- df['column_name'].fillna(df['column_name'].mean())
- df['column_name'].fillna(df['column_name'].median())

3.1.3 Xử lí dữ liệu trùng lắp và ngoại lệ

Xử lí dữ liệu trùng lặp và ngoại lệ là các bước quan trọng trong tiền xử lí dữ liệu để đảm bảo dữ liệu đạt được chất lượng cao và phù hợp cho mô hình hóa. Dưới đây là một số phương pháp phổ biến để xử lí dữ liệu trùng lặp và ngoại lệ:

Xử lí dữ liệu Trùng Lặp:

- Xác định dữ liệu trùng lặp: Sử dụng phương pháp như df.duplicated() để xác định các quan sát trùng lặp trong tập dữ liệu.

- Loại bỏ dữ liệu trùng lặp: Sử dụng phương thức drop_duplicates() để loại bỏ các hàng dữ liệu bị trùng lặp. Khi gọi df.drop_duplicates(), nó sẽ trả về một bản sao của DataFrame đã loại bỏ các hàng bị trùng lặp.

Xử lí dữ liệu Ngoại Lệ:

- Xác định và đánh dấu ngoại lệ: Sử dụng các phép đo như z-score hoặc IQR để xác định các quan sát ngoại lệ trong dữ liệu.

- Loại bỏ ngoại lệ: Có thể loại bỏ ngoại lệ khỏi dữ liệu nếu chúng là các giá trị nhiễu không mong muốn và không đại diện cho xu hướng chung của dữ liệu.

- Thay thế hoặc cắt ngoại lệ: Thay vì loại bỏ ngoại lệ hoàn toàn, cũng có thể thay thế chúng bằng các giá trị cận trên hoặc cận dưới, hoặc cắt chúng ở mức cận trên và dưới nhất định.

3.1.4 Phân chia dữ liệu để huấn luyện

Phân chia dữ liệu thành hai phần tập huấn luyện và kiểm định là một bước quan trọng trong việc phát triển mô hình máy học. Cách phổ biến nhất là sử dụng 70% cho huấn luyện, 30% cho kiểm định. Khi lượng dữ liệu tăng lên, phần trăm cho tập huấn luyện nên tăng và phần trăm cho tập kiểm định sẽ giảm đi.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Hình 3.1 Chia dữ liệu với scikit-learn

Chia dữ liệu: Hàm train_test_split được sử dụng để chia dữ liệu thành tập huấn luyện và tập kiểm tra. Trong đoạn code, x là các đặc trưng (features), y là biến mục tiêu (target), và test_size=0.30 chỉ định rằng 30% của dữ liệu sẽ được sử dụng cho tập kiểm tra, trong khi 70% còn lại sẽ được sử dụng cho tập huấn luyện. Điều này được thể hiện qua đối số test_size. Đối số random_state=42 đảm bảo rằng việc chia dữ liệu sẽ được thực

hiện một cách ngẫu nhiên nhưng sẽ tạo ra các kết quả nhất quán cho mỗi lần chạy.

Gán kết quả đã chia: Kết quả của việc chia dữ liệu sẽ được gán vào bốn biến: x_train, x_test, y_train, và y_test. x_train và y_train là tập dữ liệu được sử dụng để huấn luyện mô hình, trong khi x_test và y_test là tập dữ liệu được sử dụng để đánh giá hiệu suất của mô hình đã huấn luyện.

Cuối cùng, in ra kích thước của tập huấn luyện (x_train.shape), tập kiểm tra (x_test.shape), nhãn tập huấn luyện (y_train.shape) và nhãn tập kiểm tra (y_test.shape) để kiểm tra xem phân chia dữ liệu đã được thực hiện đúng cách hay không.

3.1.5 Feature Scaling cho dữ liệu

Phương pháp Scaling features là một bước quan trọng trong tiền xử lý dữ liệu cho các mô hình học máy, như Linear Regression, Logistic Regression, KNN, Support Vector Machine và Neural Networks. Scaling giúp các biến đặc trưng có cùng phạm vi giá trị, mà không thay đổi phân phối của chúng. Dưới đây là mô tả về kỹ thuật Scaling features phổ biến với Normalization (Min-Max Scaling):

- Normalization, còn được gọi là Min-Max Scaling, chuyển đổi giá trị của một biến thành một phạm vi giữa 0 và 1.
- Phương pháp này đơn giản là trừ giá trị nhỏ nhất của biến khỏi giá trị của biến, và sau đó chia cho sự khác biệt giữa giá trị lớn nhất và giá trị nhỏ nhất của biến đó.
- Điều này có thể được thực hiện bằng cách sử dụng MinMaxScaler từ scikit-learn.

```
from sklearn.preprocessing import MinMaxScaler

mmscaler = MinMaxScaler(feature_range=(0,1))
x_train_scaled = mmscaler.fit_transform(x_train)
x_test_scaled = mmscaler.transform(x_test)

x_train = pd.DataFrame(x_train_scaled)
x_test = pd.DataFrame(x_test_scaled)
```

3.2 Input và Output của mô hình

3.2.1 Input dữ liệu

1. PRT_ID: Mã ngôi nhà
2. AREA: Khu vực nhà ở Chennai
3. INT_SQFT: Diện tích mặt sàn
4. DATE_SALE: Ngày bán
5. DIST_MAINROAD: Khoảng cách từ nhà đến đường chính
6. N_BEDROOM: Số lượng phòng ngủ
7. N_BATHROOM: Số lượng phòng tắm
8. N_ROOM: Tổng số lượng phòng
9. SALE_COND: Điều kiện bán hàng
10. PARK_FACIL: Có chỗ đậu xe
11. DATE_BUILD: Ngày xây dựng ngôi nhà
12. BUILDTYPE: Mục đích xây dựng ngôi nhà
13. UTILITY_AVAIL: Cơ sở vật chất sẵn có
14. STREET: Loại đường gần nhà như thế nào
15. MZZONE: Nhà ở thuộc vùng nào(RL, RH, RM, C, A)
16. QS_ROOMS: Điểm số chất lượng phòng
17. QS_BATHROOM: Điểm số chất lượng phòng tắm
18. QS_BEDROOM: Điểm số chất lượng phòng ngủ

19. QS_OVERALL: Điểm số chất lượng tổng quát

20. REG_FEE: Phí đăng ký sau bán nhà

21. COMMIS: Phí hoa hồng sau khi bán nhà

22. SALES_PRICE: Giá tiền khi bán nhà

3.2.2 Output dữ liệu

Giá nhà dự đoán qua tập dữ liệu đầu vào được huấn luyện bằng mô hình Linear Regression.

3.3 Đề xuất mô hình sử dụng

Trong lĩnh vực trí tuệ nhân tạo (AI), việc chọn mô hình phù hợp là một phần quan trọng để giải quyết các vấn đề dự đoán và phân tích dữ liệu. Một trong những mô hình đơn giản và phổ biến nhất trong AI là Linear Regression.

Linear Regression là một phương pháp dự đoán giá trị đầu ra dựa trên các biến đầu vào bằng cách xây dựng một mô hình tuyến tính giữa các biến đó. Trong mô hình này, chúng ta giả định rằng có một mối quan hệ tuyến tính giữa các biến đầu vào và biến đầu ra. Mô hình này có thể được sử dụng để dự đoán giá trị liên tục, như giá cổ phiếu, doanh số bán hàng, hoặc bất kỳ biến số liên tục nào khác.

- Đơn giản và Dễ hiểu: Linear Regression là một trong những mô hình đơn giản nhất trong học máy và dễ hiểu. Điều này làm cho nó trở thành lựa chọn phổ biến đối với các vấn đề đơn giản hoặc khi cần một giải pháp nhanh chóng và dễ giải thích.

- Khả năng giải thích: Mô hình Linear Regression cung cấp thông tin về mức độ ảnh hưởng của mỗi biến đầu vào đến biến đầu ra, giúp ta hiểu rõ hơn về các mối quan hệ tuyến tính trong dữ liệu.

- Thời gian huấn luyện nhanh: Vì Linear Regression có cấu trúc đơn giản, thời gian huấn luyện của nó thường rất nhanh, đặc biệt là đối với các tập dữ liệu lớn.

Hơn nữa Linear Regression là một công cụ mạnh mẽ và linh hoạt trong học máy, đặc biệt là khi áp dụng cho các bài toán dự đoán giá trị liên tục. Mặc dù đơn giản, nhưng nó có thể cung cấp các dự đoán chính xác và cung cấp thông tin giá trị cho việc phân tích dữ liệu. Hãy xem xét sử dụng Linear Regression cho bài toán của bạn nếu bạn đang tìm kiếm một giải pháp đơn giản và hiệu quả.

Một số mô hình đề xuất khác để thực hiện mô hình AI dự đoán :

- Decision Tree Regression: Mô hình này phù hợp khi có sự không tuyến tính trong dữ liệu hoặc khi có sự tương tác phức tạp giữa các biến đầu vào. Decision tree có khả năng xử lý các biến phân loại và liên tục, và thường không yêu cầu chuẩn hoá dữ liệu.

- Random Forest Regression: Random forest là một phương pháp ensemble của decision tree, tức là nó kết hợp nhiều cây quyết định để tạo ra dự đoán. Random forest thường cho kết quả tốt hơn đối với dữ liệu có tính tương tác cao và giảm được vấn đề overfitting.

- Gradient Boosting Regression: Gradient boosting là một phương pháp ensemble khác, nhưng nó xây dựng các cây quyết định một cách tuần tự theo chiều hướng làm giảm sai số của mô hình trước đó. Gradient boosting thường cho kết quả rất tốt và được sử dụng phổ biến.

- Support Vector Regression (SVR): SVR là một phương pháp phù hợp cho các bài toán có dữ liệu có tính chất phi tuyến tính và có số lượng biến lớn. SVR có khả năng tìm ra ranh giới tốt giữa các điểm dữ liệu và thích hợp cho việc xử lý dữ liệu nhiễu.

- Neural Network Regression: Mạng nơ-ron có thể được sử dụng cho các bài toán dự đoán giá nhà với dữ liệu phức tạp và

có tính tương tác cao. Mạng nơ-ron có khả năng học được các mối quan hệ phi tuyến tính trong dữ liệu và có thể cho kết quả tốt nếu được huấn luyện một cách thích hợp.

Chương 4

Thực nghiệm và đánh giá kết quả

4.1 Mô tả, thống kê và kết quả các bước tiền xử lý dữ liệu

4.1.1 Mô tả dữ liệu

Đọc dữ liệu từ một tệp CSV có tên là "Chennai houseing sale.csv" và lưu dữ liệu vào một DataFrame có tên là df.

	PRT_ID	AREA	INT_SQFT	DATE_SALE	DIST_MAINROAD	N_BEDROOM	N_BATHROOM	N_ROOM	SALE_COND
0	P03210	Karapakkam	1004	04-05-2011	131	1.0	1.0	3	AbNormal
1	P09411	Anna Nagar	1986	19-12-2006	26	2.0	1.0	5	AbNormal
2	P01812	Adyar	909	04-02-2012	70	1.0	1.0	3	AbNormal
3	P05346	Velachery	1855	13-03-2010	14	3.0	2.0	5	Family
4	P06210	Karapakkam	1226	05-10-2009	84	1.0	1.0	3	AbNormal
...	...	...	...	...	...	...	...	...	...
7104	P03834	Karapakkam	598	03-01-2011	51	1.0	1.0	2	AdjLand
7105	P10000	Velachery	1897	08-04-2004	52	3.0	2.0	5	Family
7106	P09594	Velachery	1614	25-08-2006	152	2.0	1.0	4	Normal Sale
7107	P06508	Karapakkam	787	03-08-2009	40	1.0	1.0	2	Partial
7108	P09794	Velachery	1896	13-07-2005	156	3.0	2.0	5	Partial

7109 rows × 22 columns

PARK_FACIL	DATE_BUILD	BUILDTYPE	UTILITY_AVAIL	STREET	MZZONE	QS_ROOMS	QS_BATHROOM	QS_BEDROOM	QS_OVERAL
Yes	15-05-1967	Commercial	AllPub	Paved	A	4.0	3.9	4.9	4.33
No	22-12-1995	Commercial	AllPub	Gravel	RH	4.9	4.2	2.5	3.76
Yes	09-02-1992	Commercial	ELO	Gravel	RL	4.1	3.8	2.2	3.09
No	18-03-1988	Others	NoSewr	Paved	I	4.7	3.9	3.6	4.01
Yes	13-10-1979	Others	AllPub	Gravel	C	3.0	2.5	4.1	3.29
...	...	...	...	...	...	...	...	...	.
No	15-01-1962	Others	ELO	No Access	RM	3.0	2.2	2.4	2.52
Yes	11-04-1995	Others	NoSeWa	No Access	RH	3.6	4.5	3.3	3.92
No	01-09-1978	House	NoSeWa	Gravel	I	4.3	4.2	2.9	3.84
Yes	11-08-1977	Commercial	ELO	Paved	RL	4.6	3.8	4.1	4.16
Yes	24-07-1961	Others	ELO	Paved	I	3.1	3.5	4.3	3.64

PE	UTILITY_AVAIL	STREET	MZZONE	QS_ROOMS	QS_BATHROOM	QS_BEDROOM	QS_OVERALL	REG_FEE	COMMIS	SALES_PRICE
ial	AllPub	Paved	A	4.0	3.9	4.9	4.330	380000	144400	7600000
ial	AllPub	Gravel	RH	4.9	4.2	2.5	3.765	760122	304049	21717770
ial	ELO	Gravel	RL	4.1	3.8	2.2	3.090	421094	92114	13159200
ers	NoSewr	Paved	I	4.7	3.9	3.6	4.010	356321	77042	9630290
ers	AllPub	Gravel	C	3.0	2.5	4.1	3.290	237000	74063	7406250
...	...	...	...	...	...	...	...	...	...	...
ers	ELO	No Access	RM	3.0	2.2	2.4	2.520	208767	107060	5353000
ers	NoSeWa	No Access	RH	3.6	4.5	3.3	3.920	346191	205551	10818480
ise	NoSeWa	Gravel	I	4.3	4.2	2.9	3.840	317354	167028	8351410
ial	ELO	Paved	RL	4.6	3.8	4.1	4.160	425350	119098	8507000
ers	ELO	Paved	I	3.1	3.5	4.3	3.640	349177	79812	9976480

Hình 4.1 Xuất dữ liệu khi đọc từ file csv

- PRT_ID: Định danh duy nhất cho mỗi bất động sản.
- AREA: Khu vực hoặc vị trí nơi bất động sản đặt.
- INT_SQFT: Diện tích nội thất của bất động sản tính bằng feet vuông.
- DATE_SALE: Ngày bán bất động sản.
- DIST_MAINROAD: Khoảng cách từ bất động sản đến con đường chính.
- N_BEDROOM: Số phòng ngủ trong bất động sản.
- N_BATHROOM: Số phòng tắm trong bất động sản.
- N_ROOM: Tổng số phòng trong bất động sản.
- SALE_COND: Tình trạng bán hàng (ví dụ: mới, tái bán, vv.).
- PARK_FACIL: Sự có sẵn các cơ sở đỗ xe (ví dụ: Có, Không).
- DATE_BUILD: Ngày xây dựng của bất động sản.
- BUILDTYPE: Loại công trình (ví dụ: Nhà, Căn hộ, vv.).
- UTILITY_AVAIL: Sự có sẵn các tiện ích (ví dụ: AllPub, NoSeWa, vv.).
- STREET: Loại đường tiếp cận bất động sản (ví dụ: Lát gạch, Sỏi, vv.).
- MZZONE: Phân loại khu vực (zoning) cho bất động sản.
- QS_ROOMS: Điểm chất lượng của các phòng trong bất động sản.
- QS_BATHROOM: Điểm chất lượng của các phòng tắm trong bất động sản.
- QS_BEDROOM: Điểm chất lượng của các phòng ngủ trong bất động sản.
- QS_OVERALL: Điểm chất lượng tổng thể của bất động sản.

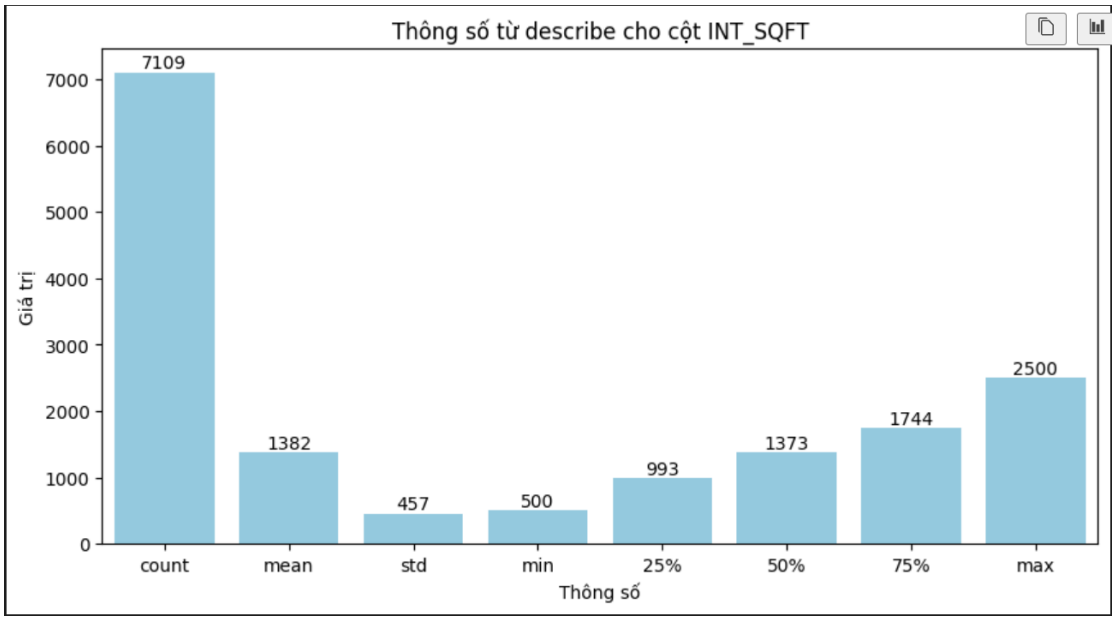
- REG_FEE: Phí đăng ký cho bất động sản.
- COMMIS: Tiền hoa hồng trả cho môi giới cho việc bán.
- SALES_PRICE: Giá bán của bất động sản.(Target Column)

4.1.2 Thống kê data

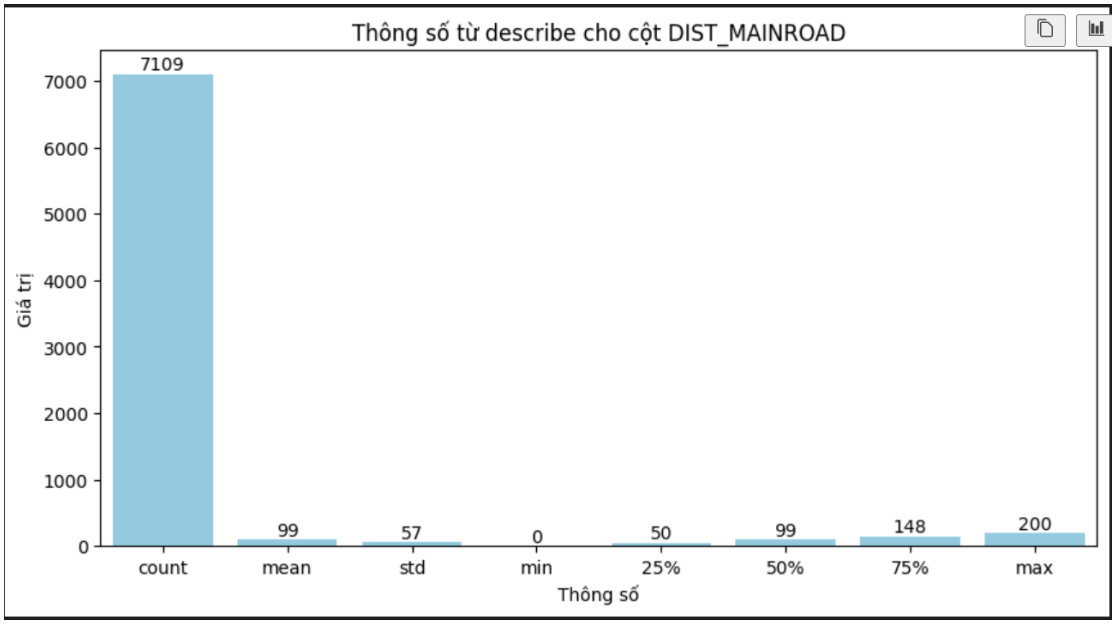
Sử dụng .describe() trong pandas để tạo ra một tóm tắt thống kê của dữ liệu trong DataFrame. Kết quả được trả về bao gồm các thông số như số lượng quan sát, giá trị trung bình, độ lệch chuẩn, giá trị tối thiểu, các phần vị (percentiles), và giá trị tối đa của các cột số trong DataFrame.

Dưới đây là các thông số mà .describe() cung cấp:

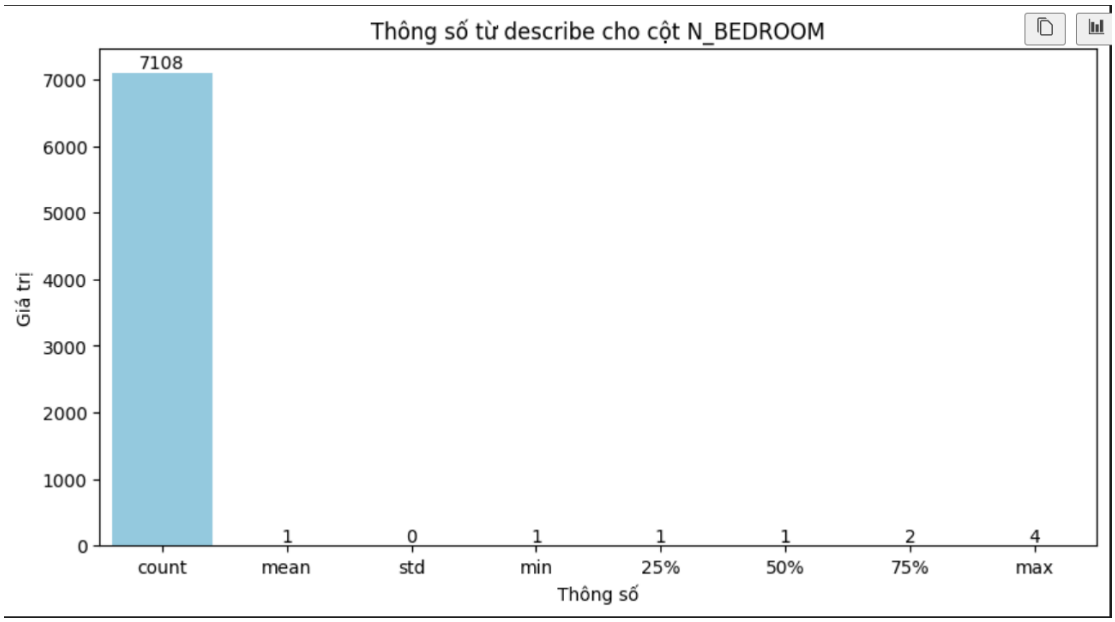
- count: Số lượng giá trị không bị thiếu (NaN).
- mean: Giá trị trung bình của dữ liệu.
- std: Độ lệch chuẩn của dữ liệu, đo lường mức độ biến động của dữ liệu.
- min: Giá trị nhỏ nhất trong dữ liệu.
- 25%, 50%, 75%: Các phần vị (percentiles) thứ 25, 50 (median), và 75 của dữ liệu.
- max: Giá trị lớn nhất trong dữ liệu.



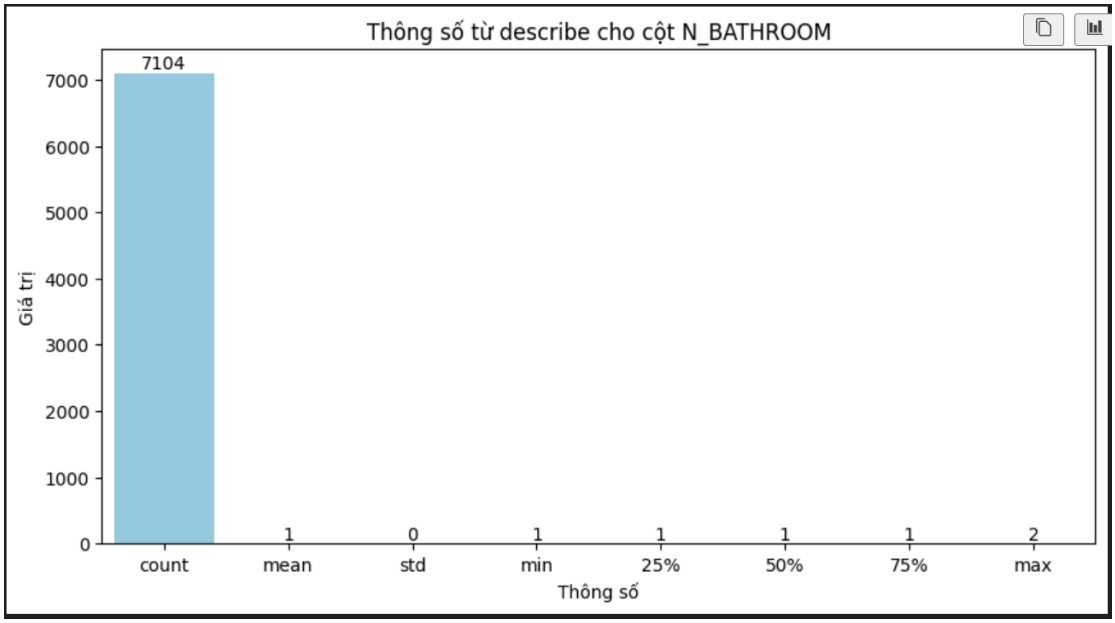
Hình 4.2 Thống kê của cột INT_SQFT



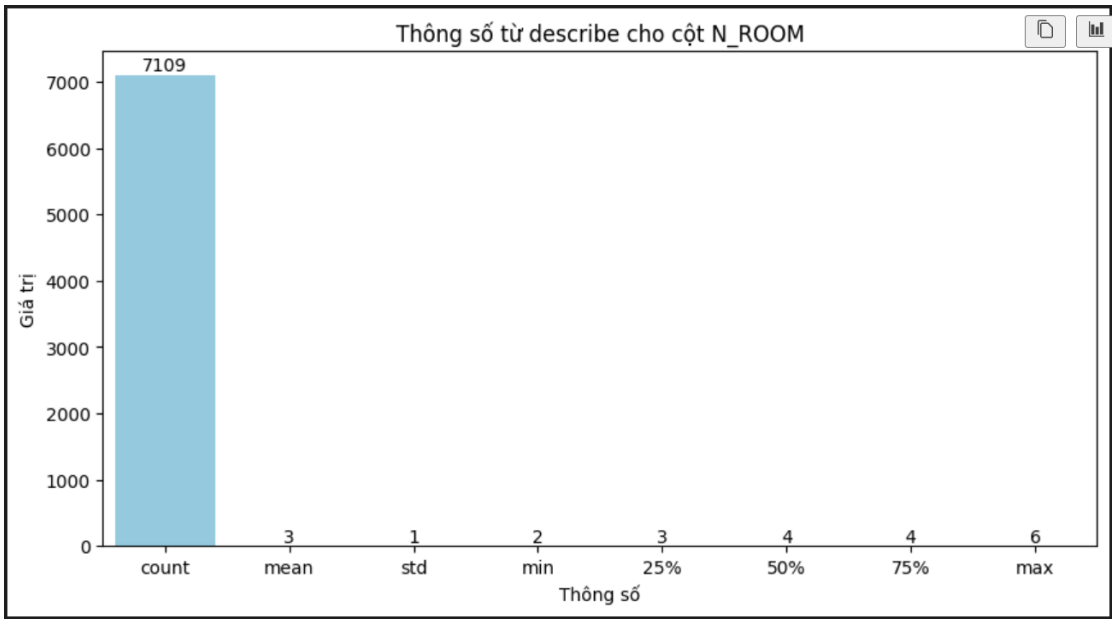
Hình 4.3 Thống kê của cột DIST_MAINROAD



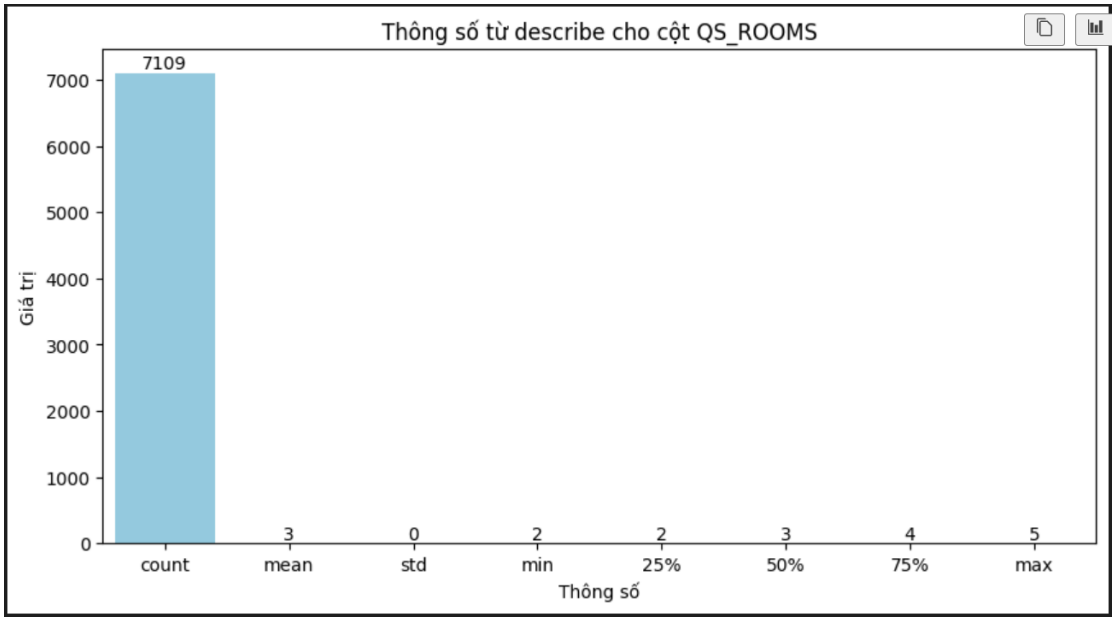
Hình 4.4 Thống kê của cột N_BEDROOM



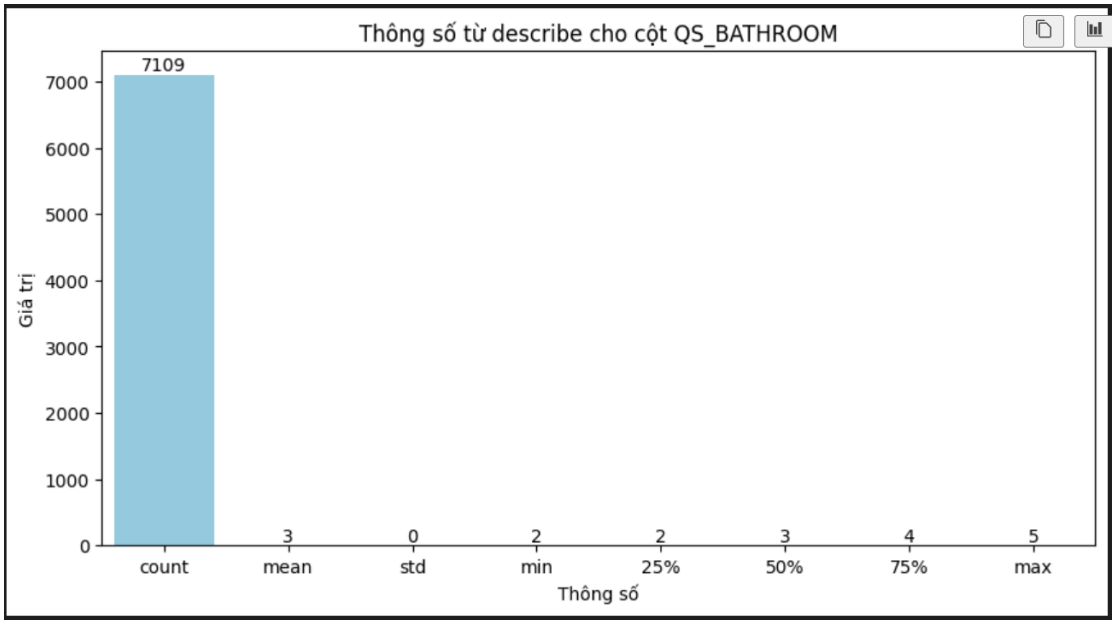
Hình 4.5 Thống kê của cột N_BATHROOM



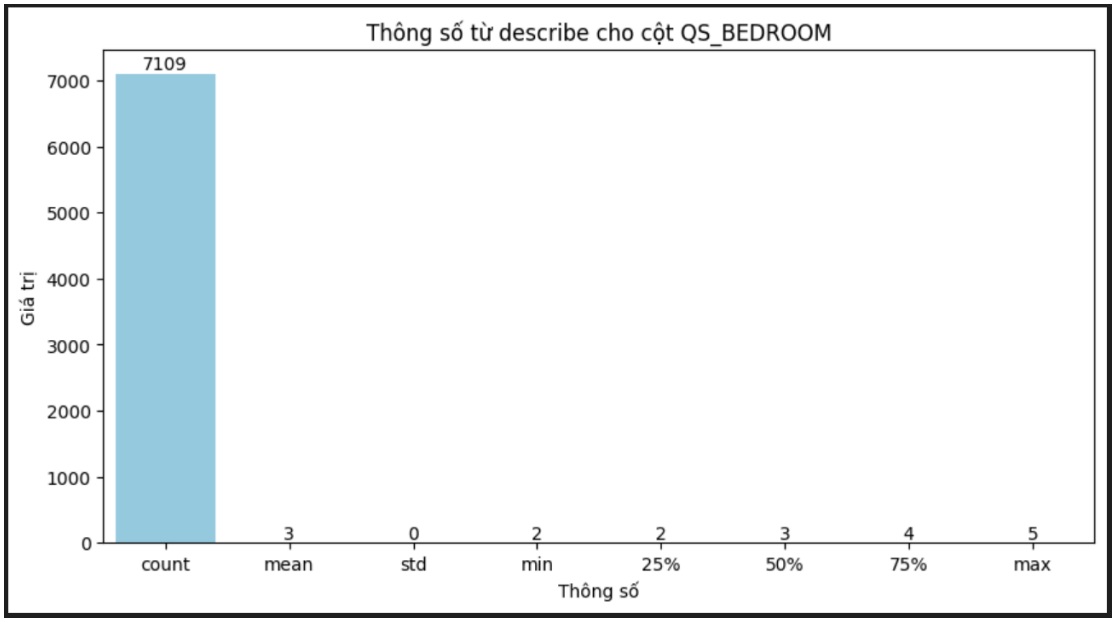
Hình 4.6 Thống kê của cột N_ROOM



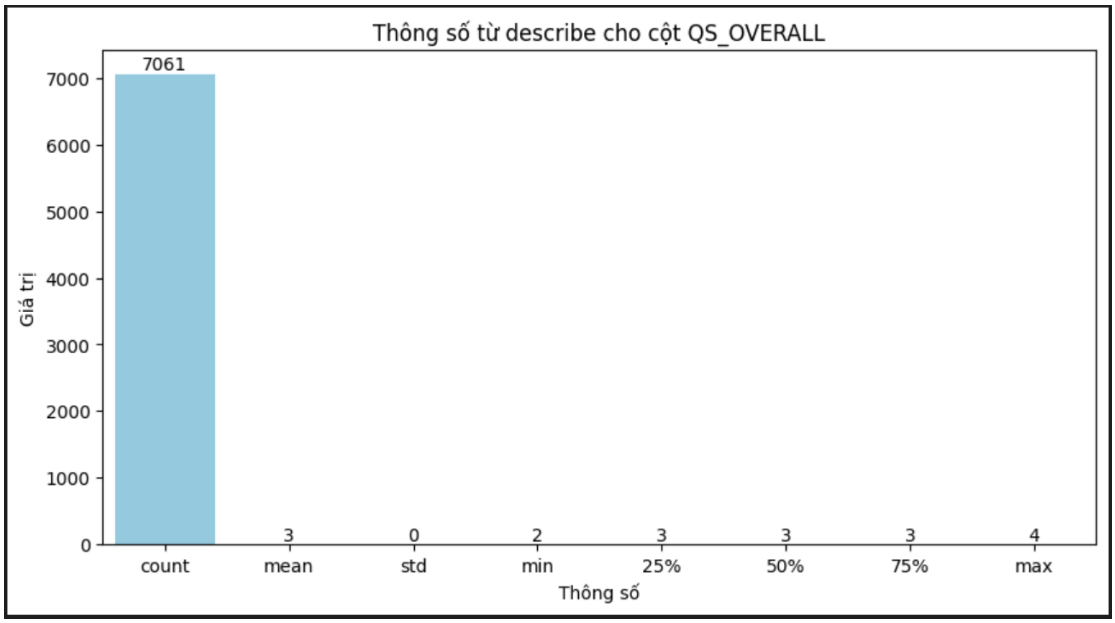
Hình 4.7 Thống kê của cột QS_ROOMS



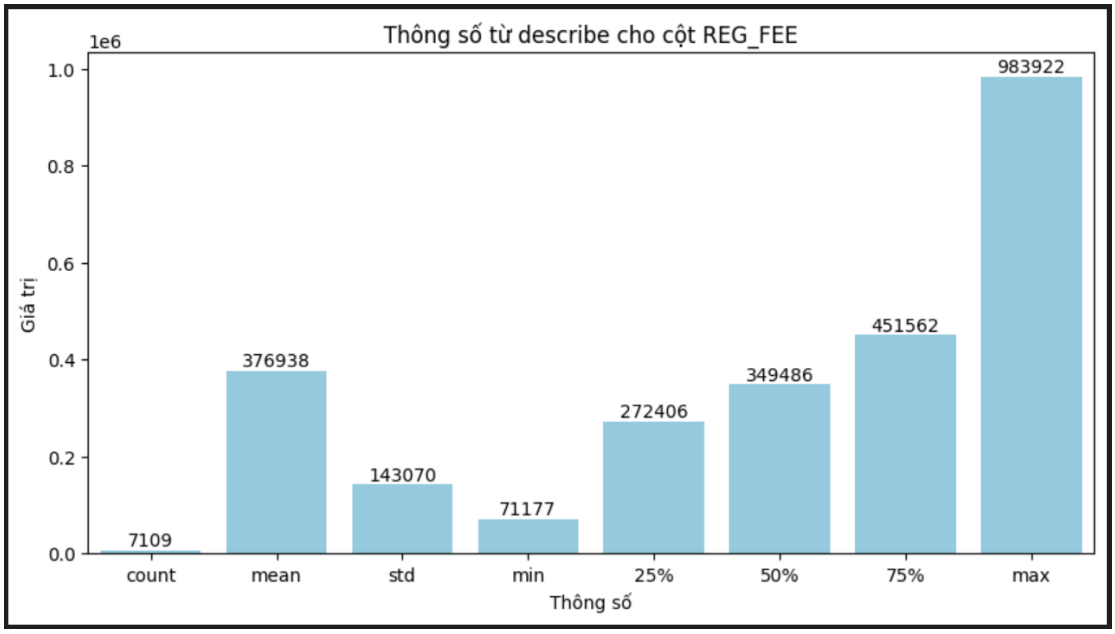
Hình 4.8 Thống kê của cột QS_BATHROOM



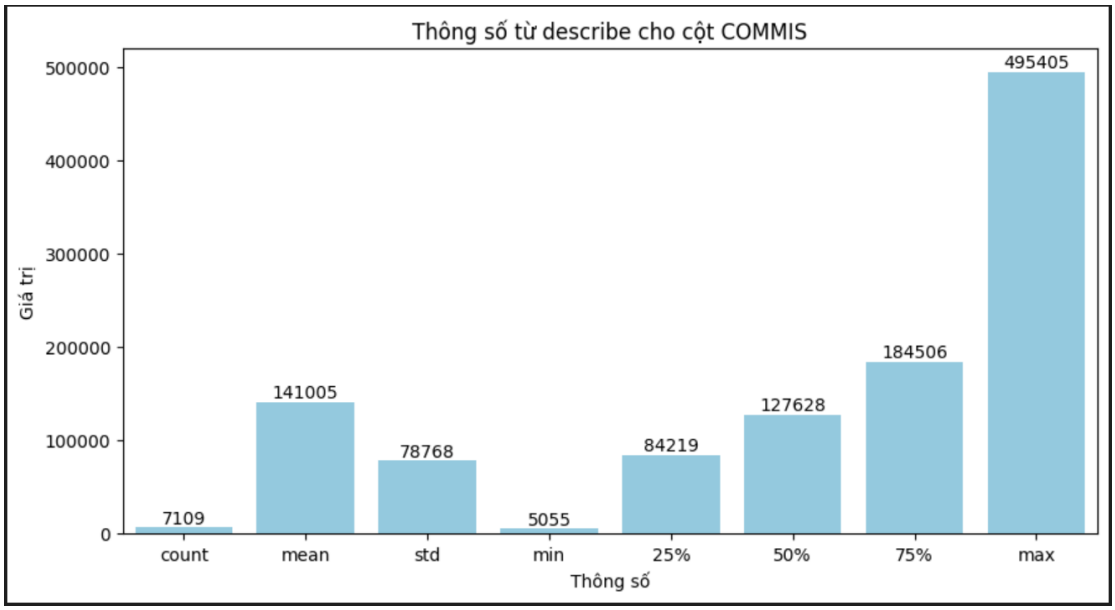
Hình 4.9 Thống kê của cột QS_BEDROOM



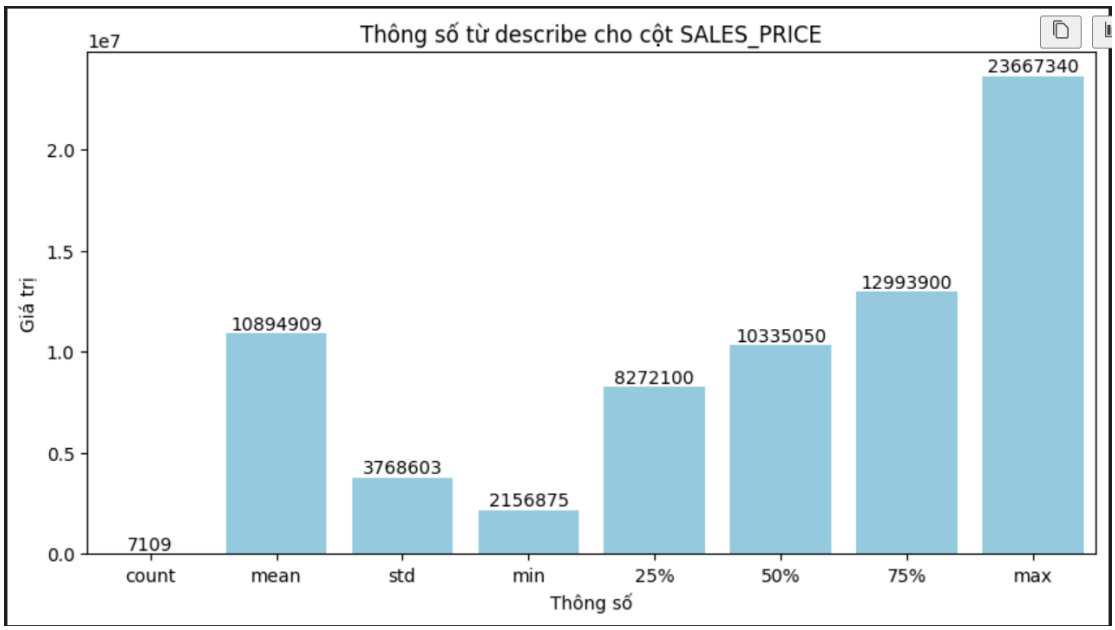
Hình 4.10 Thống kê của cột QS_OVERALL



Hình 4.11 Thống kê của cột REG_FEE



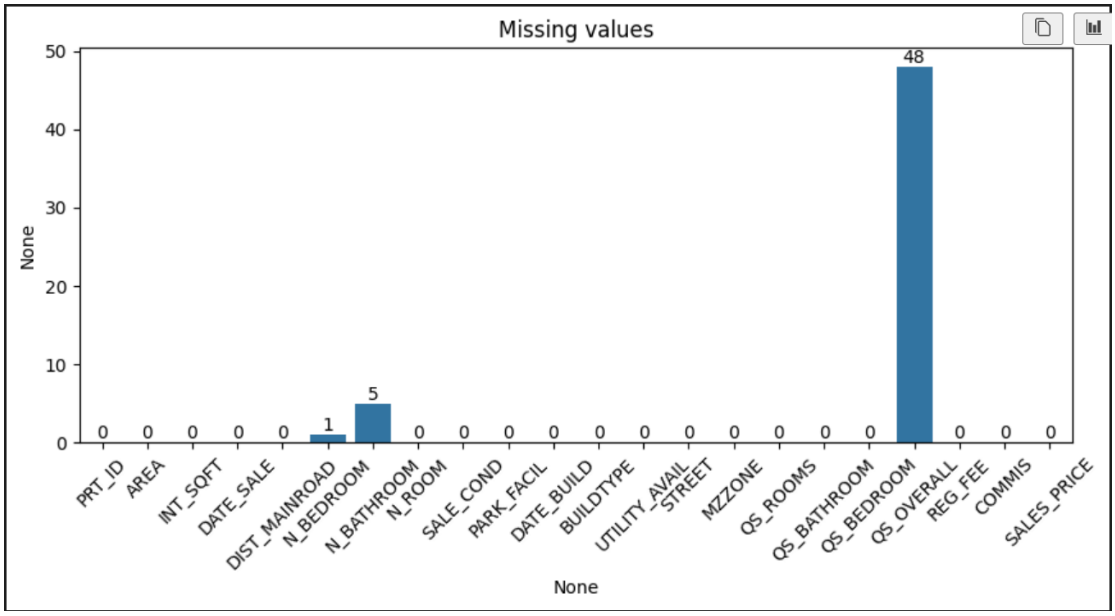
Hình 4.12 Thống kê của cột COMMIS



Hình 4.13 Thống kê của cột SALES_PRICE

4.1.3 Kết quả các bước tiền xử lý dữ liệu

Xử lí dữ liệu thiếu, mất mát



Hình 4.14 Thống kê số lượng giá trị NaN

```
# Replacing the Null Values with mean values of the data using Simple Imputer
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(missing_values=np.nan,strategy='mean',fill_value=None)
df['N_BEDROOM']=imputer.fit_transform(df[['N_BEDROOM']])
df['N_BATHROOM']=imputer.fit_transform(df[['N_BATHROOM']])
df['QS_OVERALL']=imputer.fit_transform(df[['QS_OVERALL']])
```

Hình 4.15 Thực hiện lấp đầy các giá trị NaN

Sử dụng SimpleImputer từ sklearn để thay thế các giá trị null (NaN) trong các cột 'N_BEDROOM', 'N_BATHROOM', và 'QS_OVERALL' của DataFrame 'df' bằng giá trị trung bình của từng cột.

Tiếp theo, tạo một instance của SimpleImputer với các tham số sau:

- missing_values=np.nan: Đây là giá trị cần thay thế, trong trường hợp này là NaN.
- strategy='mean': Chiến lược thay thế giá trị NaN bằng giá trị trung bình của cột.
- fill_value=None: Điều này chỉ ra rằng giá trị fill (điền vào) không được sử dụng. Mặc định là None.
-

Sau đó áp dụng SimpleImputer vào các cột 'N_BEDROOM', 'N_BATHROOM', và 'QS_OVERALL' bằng cách sử dụng phương thức fit_transform() của SimpleImputer trên mỗi cột.

Kết quả là các giá trị NaN trong các cột được thay thế bằng giá trị trung bình của cột tương ứng.

Xử lí dữ liệu trùng lắp, ngoại lệ

Kiểm tra dữ liệu trùng lặp là quá trình kiểm tra xem có bất kỳ hàng nào trong DataFrame có dữ liệu giống nhau hoàn toàn hoặc bị sai cấu trúc với hàng nào đó không. Điều này thường được thực hiện để đảm bảo tính nhất quán của dữ liệu và tránh các vấn đề không mong muốn trong phân tích.

```
AREA
['Karapakkam' 'Anna Nagar' 'Adyar' 'Velachery' 'Chrompet' 'KK Nagar'
 'TNagar' 'T Nagar' 'Chrompt' 'Chrmpet' 'Karapakam' 'Ana Nagar' 'Chormpet'
 'Adyr' 'Velchery' 'Ann Nagar' 'KKNagar']

DATE_SALE
['04-05-2011' '19-12-2006' '04-02-2012' ... '28-03-2014' '25-08-2006'
 '13-07-2005']

SALE_COND
['AbNormal' 'Family' 'Partial' 'AdjLand' 'Normal Sale' 'Ab Normal'
 'Partiall' 'Adj Land' 'PartiaLl']
PARK_FACIL
['Yes' 'No' 'Noo']

DATE_BUILD
['15-05-1967' '22-12-1995' '09-02-1992' ... '01-09-1978' '11-08-1977'
 '24-07-1961']

BUILDTYPE
['Commercial' 'Others' 'Other' 'House' 'Comercial']

UTILITY_AVAIL
['AllPub' 'ELO' 'NoSewr ' 'NoSeWa' 'All Pub']

STREET
['Paved' 'Gravel' 'No Access' 'Pavd' 'NoAccess']
```

Hình 4.16 Một số cột có các giá trị trùng lắp, sai cấu trúc

```
# Use the replace() method of the Pandas library to replace heterogeneous values in columns of a DataFrame with
# value normalization.
df.AREA.replace(['Ana Nagar','Ann Nagar'],'Anna Nagar',inplace=True)
df.AREA.replace('Karapakkam','Karapakam',inplace=True)
df.AREA.replace(['Chrompt','Chrmpet','Chormpet'],'Chrompet',inplace=True)
df.AREA.replace('KKNagar','KK Nagar',inplace=True)
df.AREA.replace('TNagar','T Nagar',inplace=True)
df.AREA.replace('Adyr','Adyar',inplace=True)
df.AREA.replace('Velchery','Velachery',inplace=True)
df.BUILDTYPE.replace('Comercial','Commercial',inplace=True)
df.BUILDTYPE.replace('Other','Others',inplace=True)
df.UTILITY_AVAIL.replace('AllPub','All Pub',inplace=True)
df.UTILITY_AVAIL.replace('NoSewr','NoSeWa',inplace=True)
df.UTILITY_AVAIL.replace('NoSewr ','NoSeWa',inplace=True)
df.SALE_COND.replace('Ab Normal','AbNormal',inplace=True)
df.SALE_COND.replace(['PartiaLl','Partiall'],'Partial',inplace=True)
df.SALE_COND.replace('Adj Land','AdjLand',inplace=True)
df.PARK_FACIL.replace('Noo','No',inplace=True)
df.STREET.replace('Pavd','Paved',inplace=True)
df.STREET.replace('NoAccess','No Access',inplace=True)
```

Hình 4.17 Xử lí đồng nhất dữ liệu

Sử dụng phương thức replace() từ thư viện Pandas để thay thế các giá trị không đồng nhất trong các cột của DataFrame bằng các giá trị được chuẩn hóa.

Các dòng như df.AREA.replace(['Ana Nagar','Ann Nagar'],'Anna Nagar',inplace=True) sẽ thay thế các giá trị 'Ana Nagar' và 'Ann Nagar' trong cột 'AREA' bằng 'Anna Nagar'. Các dòng tương tự cho các giá trị khác như 'Karapakkam' được thay thế bằng 'Karapakam', 'Chrompt', 'Chrmpet', 'Chormpet' được thay thế bằng 'Chrompet', và các giá trị khác tương tự.

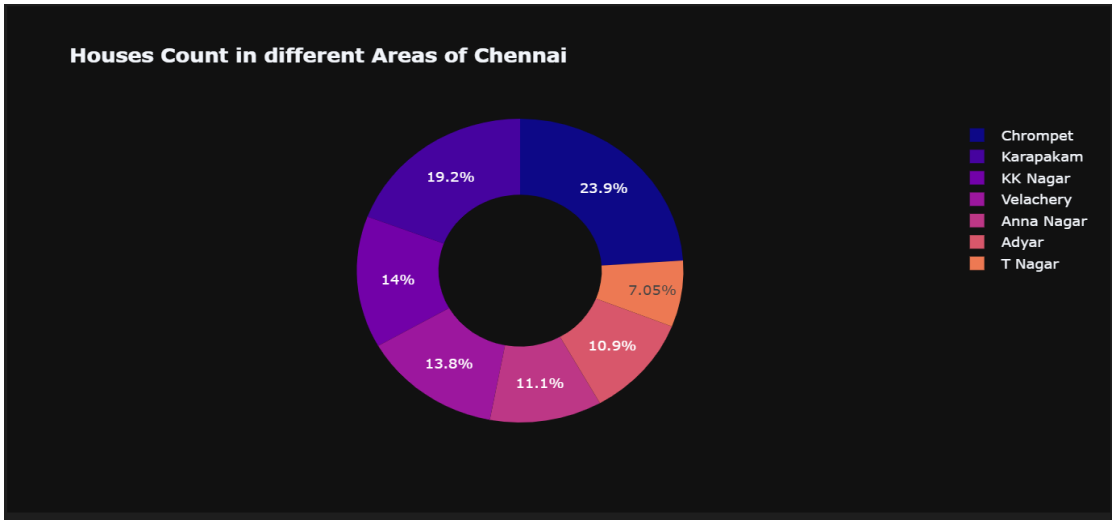
Các dòng như

df.BUILDTYPE.replace('Comercial','Commercial',inplace=True) và df.BUILDTYPE.replace('Other','Others',inplace=True) thực hiện việc chuẩn hóa các giá trị trong cột 'BUILDTYPE' bằng cách thay thế 'Comercial' thành 'Commercial' và 'Other' thành 'Others'.

Tương tự, các dòng tiếp theo thực hiện việc thay thế các giá trị trong các cột khác như 'UTILITY_AVAIL', 'SALE_COND', 'PARK_FACIL', và 'STREET' bằng các giá trị chuẩn hóa tương ứng.

Việc chuẩn hóa giá trị như vậy giúp làm cho dữ liệu trở nên đồng nhất và dễ dàng xử lý hơn trong quá trình phân tích và xử lý.

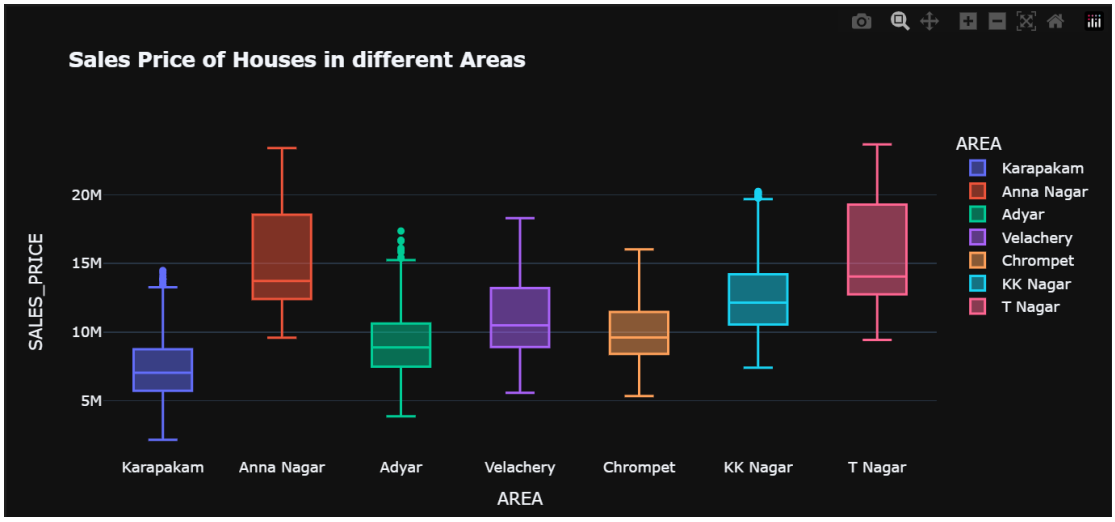
Trực quan hóa dữ liệu để huấn luyện mô hình



Hình 4.18 Thống kê phân bố nhà

Số lượng nhà ở các khu vực khác nhau tại Chennai

- Khu vực Chrompet có số lượng nhà lớn nhất so với các khu vực khác.
- Khu vực T Nagar có ít nhà nhất tại Chennai.



Hình 4.19 Thống kê sự dao động giá bán theo khu vực

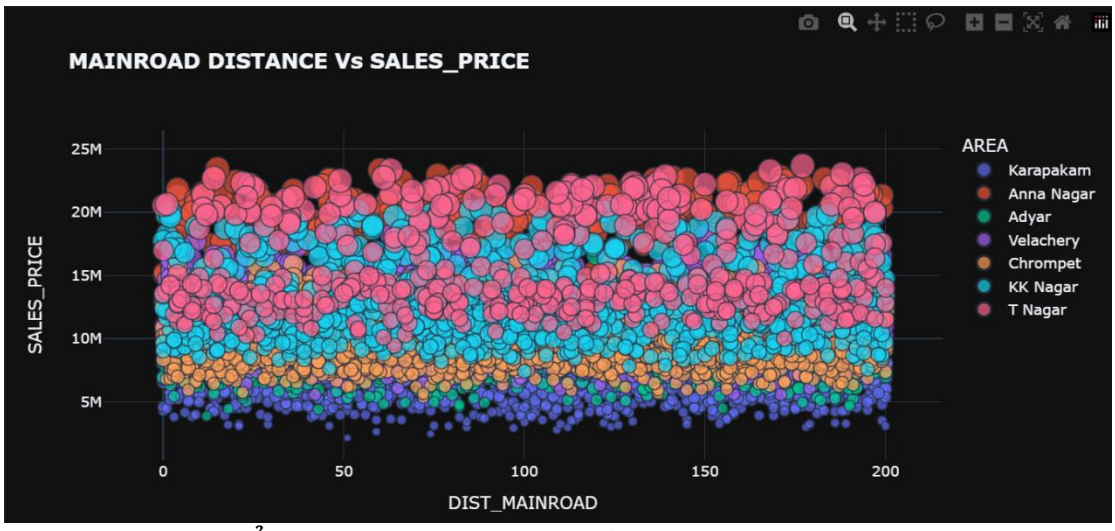
Biểu đồ sự biến đổi Giá Bán của các căn nhà theo các khu vực khác nhau tại Chennai

- Các căn nhà ở khu vực T Nagar và Anna Nagar có giá bán cao nhất và các căn nhà ở khu vực KK Nagar đứng thứ hai về giá bán cao nhất.
- Các căn nhà ở khu vực Karapakam có giá bán thấp hơn so với các căn nhà ở các khu vực khác.

Hình 4.20 Thống kê ảnh hưởng của diện tích với giá bán

Ảnh hưởng của Diện Tích căn nhà đến Giá bán

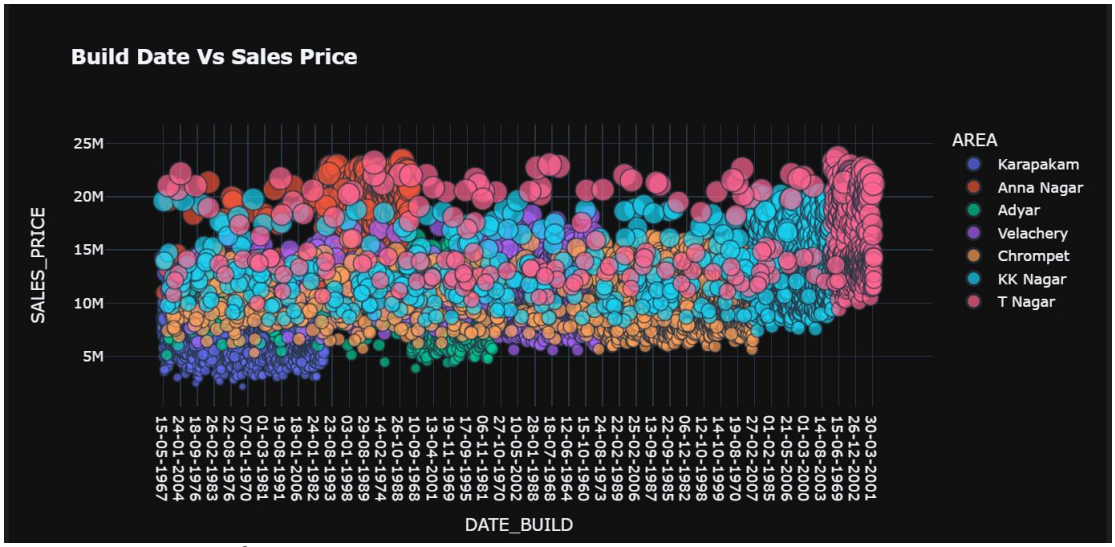
- Tăng diện tích của căn nhà sẽ tăng giá bán của căn nhà.
- Nhìn vào, các căn nhà liên quan đến cùng một khu vực có độ dài diện tích gần như giống nhau.
- Các căn nhà ở khu vực T Nagar và Anna Nagar có giá cao nhất với diện tích trong khoảng từ 1500 đến 2000 foot vuông.
- Các căn nhà ở khu vực KK Nagar có diện tích lớn từ 1400 đến 2500 foot vuông.



Hình 4.22 Ảnh hưởng của việc gần mặt đường với giá bán

Sự phụ thuộc của Giá Bán vào khoảng cách đến Đường Chính từ căn nhà

- Dường như khoảng cách đến đường chính không ảnh hưởng quá nhiềuđến giá bán của các căn nhà.
- Giá bán không khác nhau cho các căn nhà có khoảng cách đến Đường chính ngắn hơn và lớn hơn.
- Vì vậy, khoảng cách đến đường chính không ảnh hưởng nhiều đến giá bán.



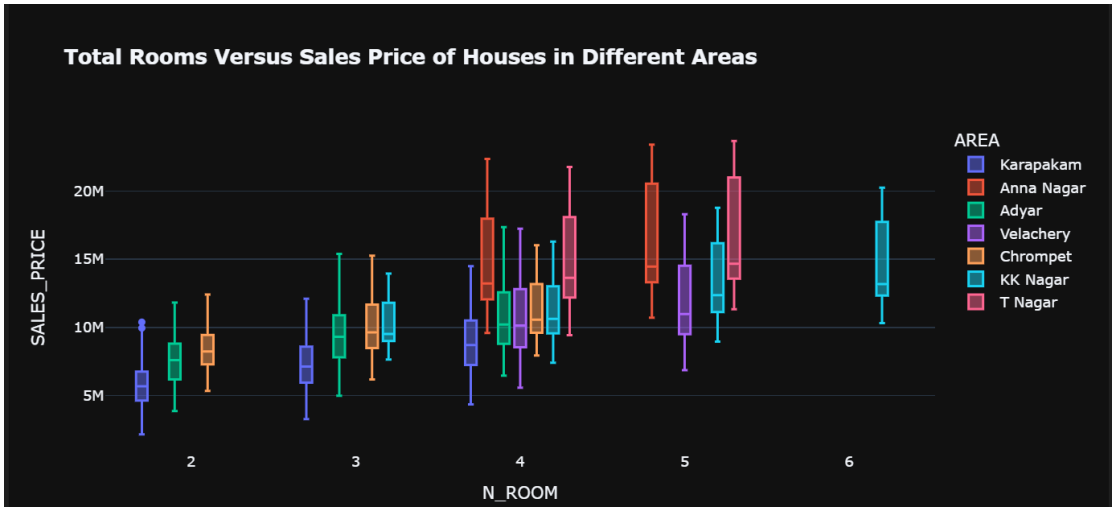
Hình 4.23 Ảnh hưởng của thời gian xây dựng so với giá bán

Sự phụ thuộc của Giá Bán vào Thời Gian Xây Dựng

- Dường như khoảng thời gian xây dựng không ảnh hưởng nhiều đến giá bán của các căn nhà.
- Giá bán không khác nhau cho các căn nhà có khoảng thời gian xây dựng cách xa nhau .
- Vì vậy, khoảng thời gian xây dựng không ảnh hưởng nhiều đến giá bán.

Hình 4.24 Ảnh hưởng của thời gian bán so với giá bán

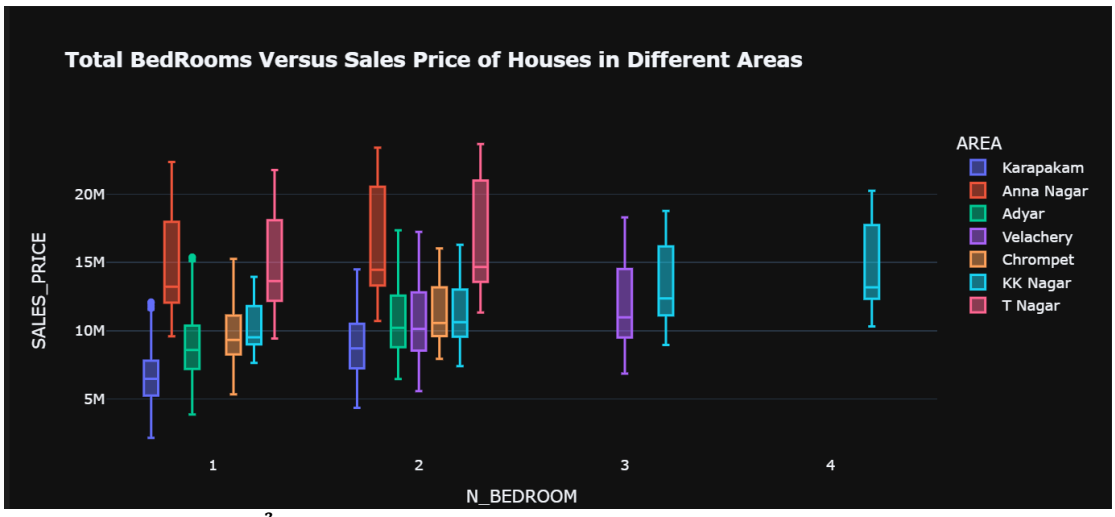
Dường như khoảng thời gian xây dựng không ảnh hưởng nhiều đến giá bán của các căn nhà, có thể xem là không ảnh hưởng.



Hình 4.25 Ảnh hưởng số lượng phòng so với giá nhà

Ảnh hưởng của Số Phòng Đến Giá Bán

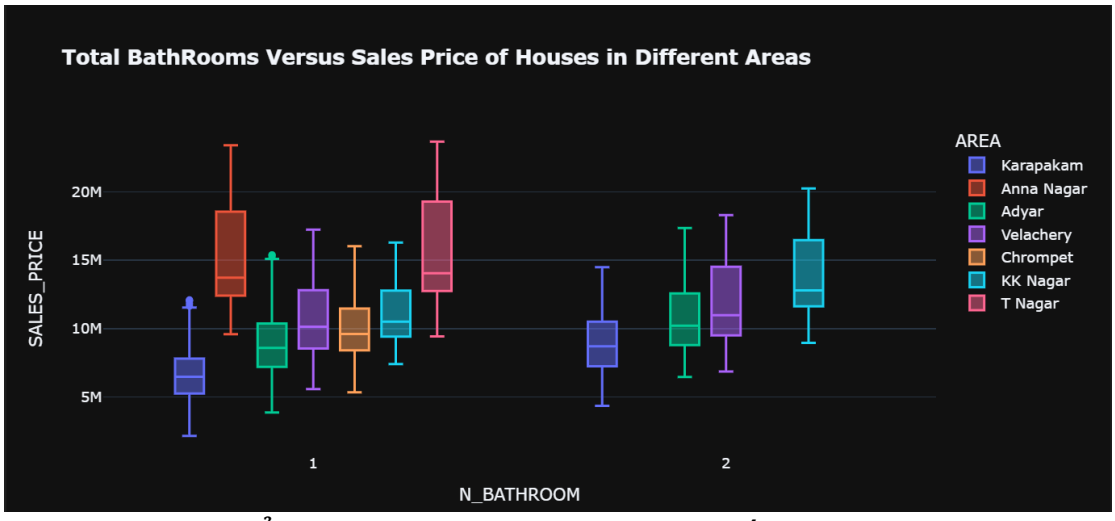
- Có thể thấy rằng việc tăng số phòng sẽ tăng giá bán của các căn nhà.
- Các căn nhà có tổng cộng 4 và 5 phòng ở khu vực Anna Nagar, Velacherry, KK Nagar và T Nagar có giá bán cao nhất.
- Các căn nhà ở khu vực Karapakam, Adyar và Chrompet có từ 2 đến bốn phòng, nhưng những căn nhà có 4 phòng có giá bán cao nhất so với các căn nhà khác.
- Khu vực Anna Nagar, Velacherry và T Nagar bao gồm các căn nhà có từ 4 đến 5 phòng.
- Khu vực KK Nagar có các căn nhà có từ hai đến sáu phòng và giá bán tăng theo sự tăng số lượng phòng.



Hình 4.26 Ảnh hưởng số lượng phòng ngủ so với giá nhà

Ảnh hưởng của Số Phòng Ngủ Đến Giá Bán?

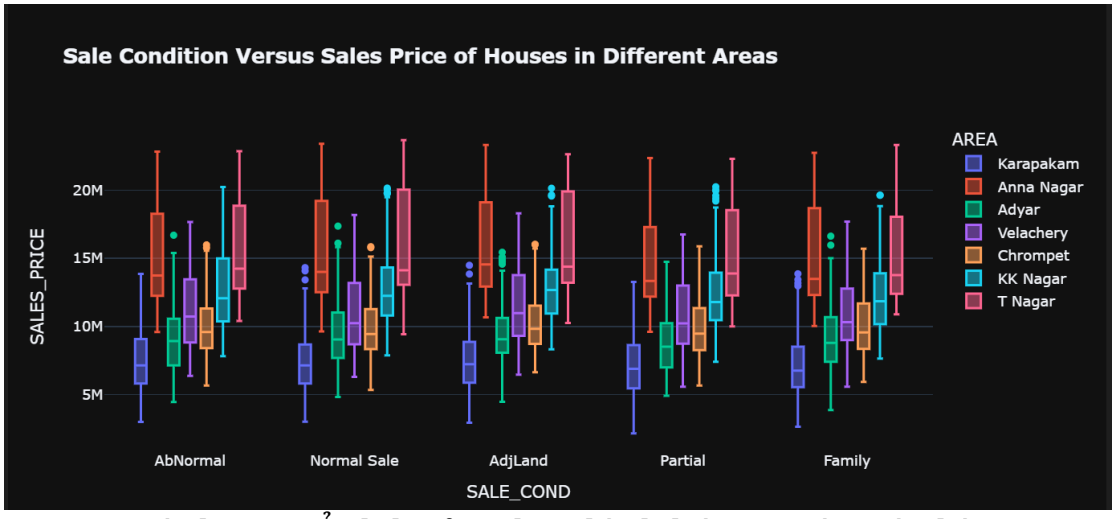
- Tăng số phòng ngủ tăng giá bán của căn nhà.
- Khu vực Anna Nagar, T Nagar, Adyar, Karapakam và Chrompet có từ 1 đến 2 phòng ngủ, nhưng các căn nhà 2 phòng ngủ có giá bán cao nhất.
- Khu vực KK Nagar có từ 1 đến 4 phòng ngủ và giá bán tăng theo số phòng ngủ.
- Khu vực Velacherry có từ 2 đến 3 phòng ngủ.
- Trong tất cả các khu vực, các căn nhà ở Anna Nagar và T Nagar có 2 phòng ngủ có giá bán cao nhất.



Hình 4.27 Ảnh hưởng số lượng phòng tắm so với giá nhà

Ảnh hưởng của Số Phòng Tắm Đến Giá Bán

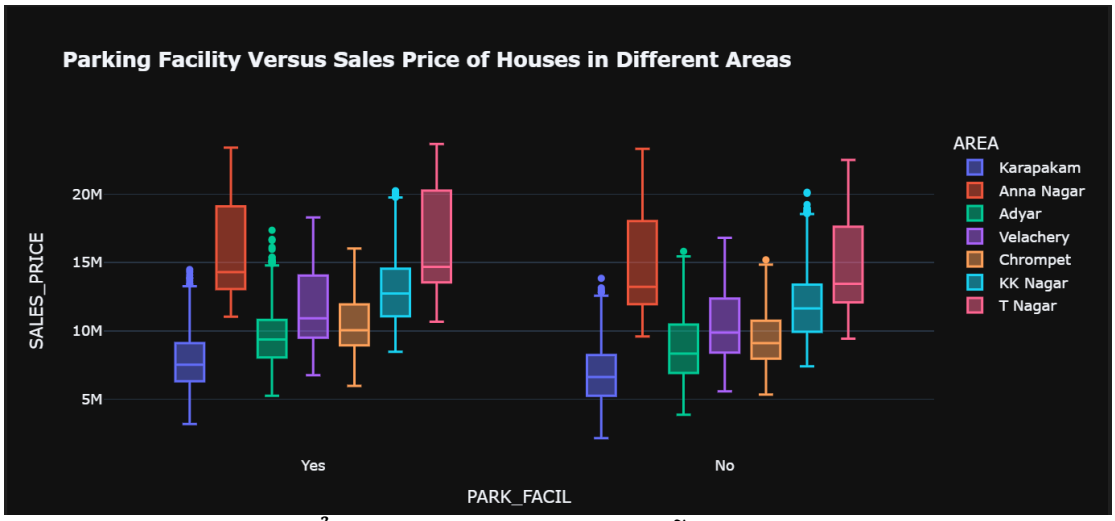
- Sự tăng số phòng tắm thường đi đôi với việc tăng giá bán ở tất cả các khu vực.
- Tuy nhiên, có một điểm đặc biệt là ở khu vực Anna Nagar và T Nagar, các căn nhà chỉ có một phòng tắm lại có giá bán cao nhất so với các khu vực khác. Điều này có thể liên quan đến yếu tố địa lý, tiện ích hoặc sự ưa chuộng từ phía người mua trong khu vực đó.



Hình 4.28 Ảnh hưởng loại hình bán so với giá nhà

Ảnh hưởng của Tình Trạng Bán Đối Với Giá Bán Của Nhà

- Các căn nhà được bán trong tình trạng Normal Sale, Family và Abnormalthường có giá bán cao nhất.
- Điều này có thể do các nhà được bảo dưỡng và bảo trì tốt hơn, phù hợp với nhu cầu của gia đình hoặc có các đặc điểm độc đáo hoặc cần sự chú ý đặc biệt, từ đó tạo ra giá trị cao hơn trong mắt người mua.

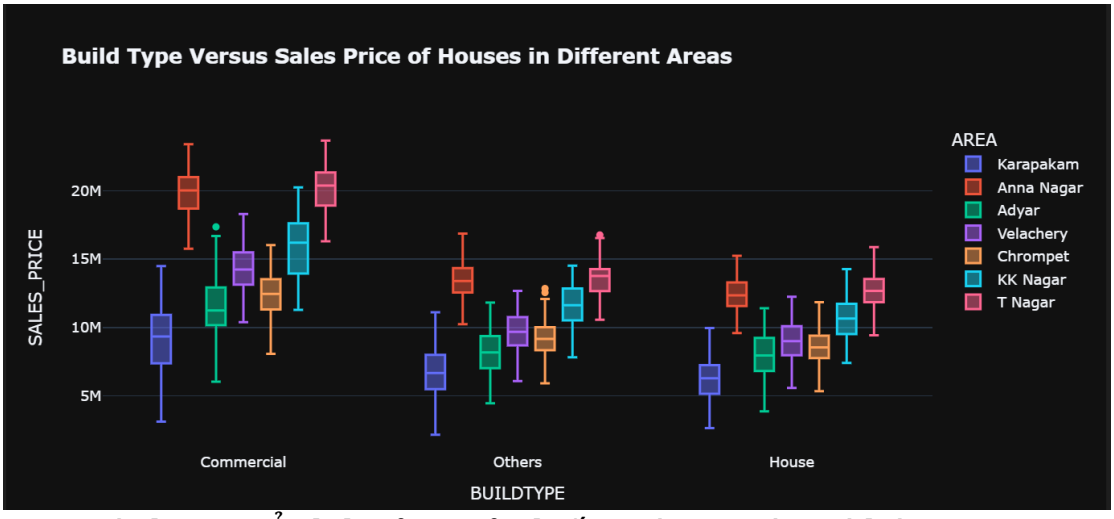


Hình 4.29 Ảnh hưởng tiện ích đỗ xe so với giá nhà

Ảnh hưởng của Tiện ích đỗ xe so với Giá Bán

- Giá bán thường cao nhất cho các căn nhà có tiện ích đỗ xe ở các khu vực khác nhau.

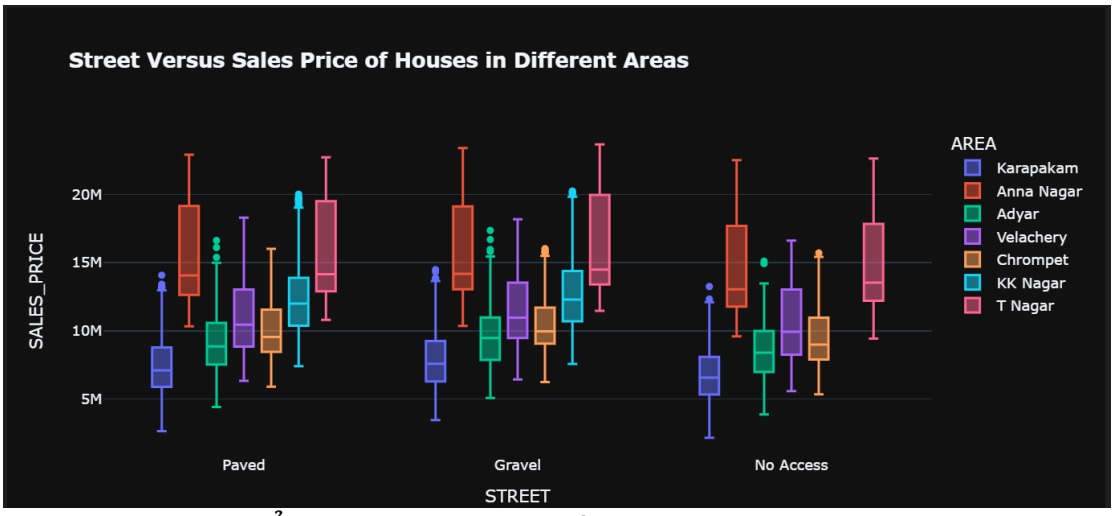
- Việc có chỗ đậu xe đóng vai trò quan trọng trong việc quyết định giá bán của căn nhà, vì tiện ích này đáp ứng nhu cầu của người mua và tạo ra một sự tiện lợi cho họ.



Hình 4.30 Ảnh hưởng của kiến trúc so với giá bán

Ảnh hưởng của Kiến Trúc so với Giá Bán

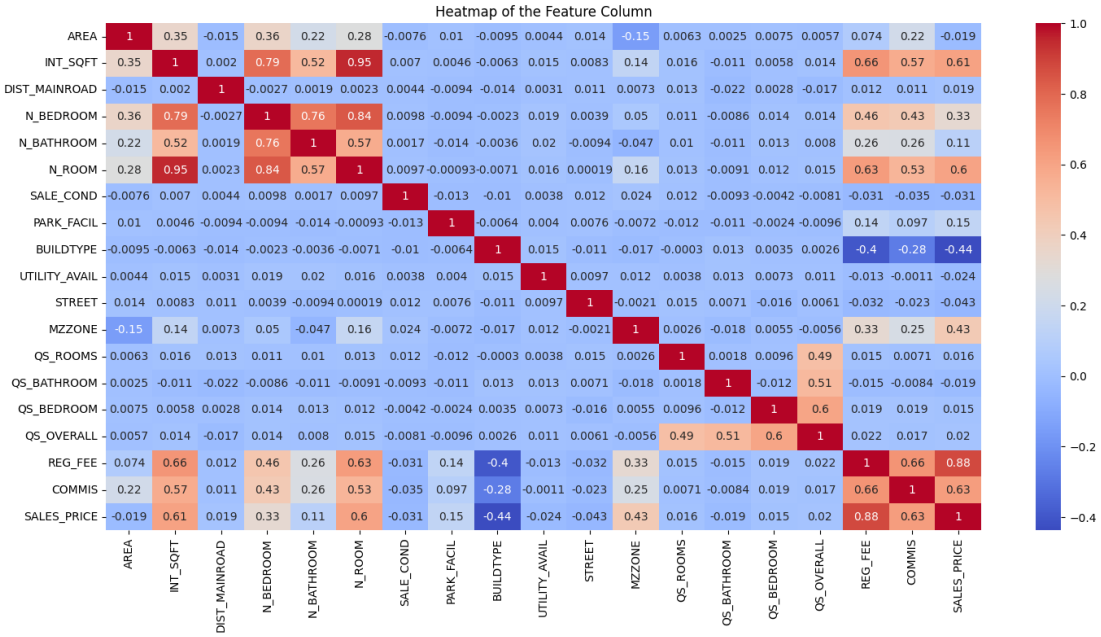
- Giá bán thường cao nhất cho các căn nhà có kiểu kiến trúc xây dựng thương mại(Commercial) ở các khu vực khác nhau.
- Kiểu kiến trúc xây dựng thương mại đóng vai trò quan trọng trong việc quyết định giá bán của căn nhà, có thể là do tính đa dạng và tiện ích mà kiểu kiến trúc này mang lại.



Hình 4.31 Ảnh hưởng của kiến trúc so với giá bán

Ảnh hưởng của Loại Đường Phố Đối Với Giá Bán Của Nhà

- Giá bán có chênh lệch một ít cho nhà ở các con phố trên các khu vực khác nhau.



Hình 4.32 Biểu đồ tương quan của các cột dữ liệu

Xác định Tương Quan: Heatmap correlation cho phép nhìn vào mức độ tương quan giữa các cặp đặc trưng trong dữ liệu. Nếu hai đặc trưng có mối quan hệ tương quan cao (gần 1 hoặc -1), thì chúng có thể cung cấp thông tin trùng lặp và có thể không cần thiết cho mô hình của.

Chọn Đặc Trưng: Bằng cách sử dụng heatmap correlation, có thể lựa chọn các đặc trưng có mức độ tương quan cao với biến mục tiêu (target variable) để làm input cho mô hình của. Điều này giúp giảm số lượng đặc trưng, giữ lại những đặc trưng quan trọng nhất và cải thiện hiệu suất của mô hình.

Loại bỏ Đặc Trưng Trùng Lặp: Nếu hai đặc trưng có mức độ tương quan cao với nhau, bạn có thể chọn loại bỏ một trong hai để giảm độ phức tạp của mô hình và giảm thiểu hiện tượng đa cộng tuyến.

Phát hiện Đặc Trưng Quan Trọng: Heatmap correlation cũng có thể giúp bạn phát hiện ra những đặc trưng quan trọng có mối quan hệ tương quan cao với biến mục tiêu.

4.2 Thông số cho mô hình, độ đo đánh giá

Thông số huấn luyện mô hình:

1. AREA: Diện tích của căn nhà.
2. INT_SQFT: Diện tích nội thất của căn nhà.
3. DIST_MAINROAD: Khoảng cách đến đường chính gần nhất.
4. N_BEDROOM: Số lượng phòng ngủ.
5. N_BATHROOM: Số lượng phòng tắm.
6. N_ROOM: Tổng số phòng trong căn nhà.
7. SALE_COND: Tình trạng bán của căn nhà.
8. PARK_FACIL: Tiện ích gửi xe có sẵn hay không.
9. BUILDTYPE: Loại cấu trúc xây dựng của căn nhà.
10. UTILITY_AVAIL: Tiện ích công cộng có sẵn hay không.
11. STREET: Kiểu đường đi trước nhà.
12. MZZONE: Loại khu vực.
13. QS_ROOMS: Đánh giá chất lượng phòng.
14. QS_BATHROOM: Đánh giá chất lượng
15. QS_BEDROOM: Đánh giá chất lượng phòng ngủ.
16. QS_OVERALL: Đánh giá chất lượng tổng thể của nhà.
17. REG_FEE: Phí đăng ký.
18. COMMIS: Tiền hoa hồng cho người môi giới.

Trong việc xây dựng mô hình dự đoán giá bán nhà, việc chọn các đặc trưng phù hợp là một bước quan trọng để đảm bảo hiệu suất của mô hình. Các đặc trưng được chọn phải phản ánh mối quan hệ có thể có giữa các biến độc lập và biến phụ thuộc.

Thông số đánh giá cho mô hình:

1.Mean Absolute Error (MAE):
MAE được tính bằng trung bình của giá trị tuyệt đối của sự chênh lệch giữa dự đoán và giá trị thực tế.

Công thức tính: MAE = (1/n) * Σ|i=1->n| (y_i - ŷ_i)

Trong đó:
- n là số lượng mẫu,
- y_i là giá trị thực tế,
- ŷ_i là giá trị dự đoán.

2.Mean Squared Error (MSE):
MSE là trung bình của bình phương của sự chênh lệch giữa dự đoán và giá trị thực tế.

Công thức tính: MSE = (1/n) * Σ(i=1->n) (y_i - ŷ_i)^2

3.Root Mean Squared Error (RMSE):
RMSE là căn bậc hai của MSE, đo lường sự chênh lệch trung bình giữa dự đoán và giá trị thực tế dưới dạng đơn vị của dữ liệu.

Công thức tính: RMSE = sqrt(MSE)

4.R-squared (R2_score):
R2_score đo lường phần trăm của phương sai trong biến phụ thuộc mà mô hình có thể giải thích. Giá trị này dao động từ 0 đến 1, với 1 cho thấy mô hình hoàn hảo.

Công thức tính:
- Tổng biến thiên do mô hình giải thích (SST): SST = Σ(y_i - ȳ)^2
- Tổng biến thiên không giải thích (SSE): SSE = Σ(y_i - ŷ_i)^2

- R2_score = 1 - (SSE / SST)

Trong đó:
- ȳ là giá trị trung bình của giá trị thực tế.

5.Root Mean Squared Log Error (RMSLE):

RMSLE đo lường sự chênh lệch giữa logarithm của dự đoán và giá trị thực tế. Nó thường được sử dụng khi biến đổi logarithm có ý nghĩa trong dữ liệu.

Công thức tính:
RMSLE = sqrt((1/n) * Σ(i=1->n) (log(1 + y_i) - log(1 + ŷ_i))^2)

6.Mean Absolute Percentage Error (MAPE):

MAPE đo lường sự chênh lệch trung bình giữa dự đoán và giá trị thực tế dưới dạng phần trăm của giá trị thực tế.

Công thức tính:
MAPE = (1/n) * Σ(i=1->n) (|y_i - ŷ_i| / |y_i|) * 100

7.Adjusted R Square:

Đây là một biến thể của R2_score, điều chỉnh dựa trên số lượng biến độc lập trong mô hình. Nó cố gắng loại bỏ các biến không cần thiết khỏi mô hình.

Công thức tính:
Adjusted R Square = 1 - [(1 - R2_score) * (n - 1) / (n - k - 1)]

Trong đó:

- n là số lượng mẫu,
- k là số lượng biến độc lập trong mô hình.

4.3 Thực nghiệm và đánh giá kết quả

Chuẩn bị dữ liệu cho mô hình

Choose Data into two set : x(features), y(target)

+ Code + Markdown

```
# storing the Dependent Variables in X and Independent Variable in Y
x=df.drop(['PRT_ID', 'DATE_SALE','DATE_BUILD','SALES_PRICE'],axis=1)
y=df['SALES_PRICE']
```

✓ 0.0s

Hình 4.33 Chia dữ liệu thành 2 tập Features(x) và Target(y)

Splitting the Data into Training set and Testing Set

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

✓ 0.0s

```
((4976, 18), (2133, 18), (4976,), (2133,))
```

Hình 4.33 Chia dữ liệu để huấn luyện

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_state=42):

- x: Là tập hợp các biến độc lập.
- y: Là tập hợp các biến phụ thuộc cần dự đoán.
- test_size=0.30: Đây là tỷ lệ của tập kiểm tra so với tổng số mẫu. Trong trường hợp này, tập kiểm tra chiếm 30% tổng số mẫu và tập huấn luyện chiếm 70%.
- random_state=42: Tham số này đảm bảo rằng việc chia dữ liệu sẽ luôn cho kết quả giống nhau nếu chúng ta chạy code này nhiều lần. Giá trị 42 ở đây là một giá trị ngẫu nhiên, nó có thể được thay đổi hoặc không cần thiết tùy theo mong muốn.

Tiến hành huấn luyện với Linear Regression Model

```
# Build the Regression / Regressor models

from sklearn.linear_model import LinearRegression

# Create objects of Regression / Regressor models with default hyper-parameters

model_lnm = LinearRegression()
```
✓ 0.0s

Hình 4.34 Import Model Linear Regression với Scikit-learn

```
# Fit the model with train data

model.fit(x_train, y_train)

# Predict the model with test data

y_pred = model.predict(x_test)

# Print the model name

print('Model Name: ', model)

# Evaluation metrics for Regression analysis

from sklearn import metrics

print('Mean Absolute Error (MAE):', round(metrics.mean_absolute_error(y_test, y_pred),3))
print('Mean Squared Error (MSE):', round(metrics.mean_squared_error(y_test, y_pred),3))
print('Root Mean Squared Error (RMSE):', round(np.sqrt(metrics.mean_squared_error(y_test, y_pred)),3))
print('R2_score:', round(metrics.r2_score(y_test, y_pred),6))
print('Root Mean Squared Log Error (RMSLE):', round(np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))),
                                                    3))
```

Hình 4.35 Huấn luyện với dữ liệu đã chuẩn bị

```
# Define the function to calculate the MAPE - Mean Absolute Percentage Error

def MAPE (y_test, y_pred):
    y_test, y_pred = np.array(y_test), np.array(y_pred)
    return np.mean(np.abs((y_test - y_pred) / y_test)) * 100

# Evaluation of MAPE

result = MAPE(y_test, y_pred)
print('Mean Absolute Percentage Error (MAPE):', round(result, 2), '%')

# Calculate Adjusted R squared values

r_squared = round(metrics.r2_score(y_test, y_pred),6)
adjusted_r_squared = round(1 - (1-r_squared)*(len(y)-1)/(len(y)-x.shape[1]-1),6)
print('Adj R Square: ', adjusted_r_squared)
```

Hình 4.36 Định nghĩa các hàm thông số đánh giá

Nhận xét sơ lượt về mô hình

```
Model Name:  LinearRegression()
Mean Absolute Error (MAE): 1031042.027
Mean Squared Error (MSE): 1687454090535.995
Root Mean Squared Error (RMSE): 1299020.435
R2_score: 0.87557
Root Mean Squared Log Error (RMSLE): 14.077
Mean Absolute Percentage Error (MAPE): 10.16 %
Adj R Square:  0.875254
```

Hình 4.37 Các thông số đánh giá mô hình

Mean Absolute Error (MAE): MAE đo lường trung bình của sự chênh lệch giữa giá trị dự đoán và giá trị thực tế. Trong trường hợp này, giá trị MAE là khoảng 1,031,042.027. Điều này cho thấy giá trị trung bình của sai số tuyệt đối giữa dự đoán và giá trị thực tế là khá lớn.

Mean Squared Error (MSE): MSE là trung bình của bình phương của sự chênh lệch giữa giá trị dự đoán và giá trị thực tế. Với giá trị MSE là khoảng 1,687,454,090,536, MSE cũng cho thấy sự chênh lệch lớn giữa dự đoán và thực tế.

Root Mean Squared Error (RMSE): RMSE là căn bậc hai của MSE, là một phép đo phổ biến để đánh giá sự chênh lệch trung bình giữa dự đoán và giá trị thực tế. Với giá trị RMSE là khoảng 1,299,020.435, cũng cho thấy sự chênh lệch lớn giữa dự đoán và thực tế.

R-squared (R2_score): R2_score là một phép đo thống kê để đánh giá mức độ phù hợp của mô hình. Nó cho biết phần trăm phương sai của biến phụ thuộc được giải thích bởi mô hình. Với giá trị R2_score là khoảng 0.87557, mô hình giải thích khoảng 87.56% sự biến thiên của dữ liệu, điều này cho thấy một mức độ phù hợp tương đối tốt của mô hình.

Root Mean Squared Log Error (RMSLE): RMSLE đo lường sự chênh lệch giữa logarithm của giá trị dự đoán và giá trị thực tế. Với giá trị RMSLE là khoảng 14.077, cho thấy một mức độ chênh lệch lớn giữa dự đoán và thực tế.

Mean Absolute Percentage Error (MAPE): MAPE đo lường trung bình của phần trăm sai số tuyệt đối giữa giá trị dự đoán và giá trị thực tế. Với giá trị MAPE là khoảng 10.16%, cho thấy mức độ sai số tương đối của mô hình.

Adj R Square: Adj R Square là một biến thể của R2_score, nhưng được sửa đổi để xem xét số lượng biến trong mô hình. Với giá trị Adj R Square là khoảng 0.875254, cũng cho thấy một mức độ phù hợp tương đối tốt của mô hình.

```
Result_prediction[['PRT_ID','AREA','Price_actually','Price_predict']]
```

✓ 0.0s

	PRT_ID	AREA	Price_actually	Price_predict
8	P03377	Chrompet	8308970	6288189.38
14	P04085	Velachery	15499680	15339365.64
15	P06328	Velachery	15714080	14174788.66
17	P02016	Chrompet	10912550	9731951.96
19	P01372	Anna Nagar	21203240	20175357.84
...	...	...	...	...
7101	P05042	Karapakkam	6211750	5627456.19
7102	P05560	Karapakkam	5643500	6829112.84
7103	P05133	Karapakkam	9387250	9324034.59
7105	P10000	Velachery	10818480	9220275.13
7107	P06508	Karapakkam	8507000	10414014.19

2133 rows × 4 columns

Hình 4.38 Giá dự đoán và Giá thật tế

Dựa trên so sánh giữa giá trị thực tế và giá trị dự đoán, có thể nhận xét rằng mô hình dự đoán tỏ ra khá chính xác. Mặc dù có một số sai số nhỏ xuất hiện, nhưng đối chúng, sự khác biệt giữa giá trị thực tế và giá trị dự đoán được duy trì ở mức tương đối thấp.

Hình 4.39 Biểu đồ phân tán dữ liệu về giá

Biểu đồ phân tán với trendline đã tạo ra một hình ảnh tương đối tốt về mối quan hệ giữa giá trị thực tế và giá trị dự đoán từ mô hình hồi quy. Các điểm dữ liệu trên biểu đồ phân bố gần dường trendline, cho thấy rằng mô hình có khả năng dự đoán giá trị dự đoán với mức độ chính xác khá cao dựa trên giá trị thực tế.

Trendline OLS cung cấp một cái nhìn tổng quan về mối quan hệ tuyến tính giữa các biến, và dường như có một mẫu tăng/giảm chung giữa giá trị thực tế và giá trị dự đoán. Điều này gợi ý rằng mô hình hồi quy có thể đã tìm được một mối quan hệ có ý nghĩa giữa các biến đầu vào và đầu ra.

Tuy nhiên, cần phải tiếp tục kiểm tra các chỉ số đánh giá mô hình để đảm bảo độ chính xác và hiệu suất của mô hình trên tập dữ liệu. Ngoài ra, việc kiểm tra trên tập dữ liệu kiểm tra độc lập cũng là cần thiết để đánh giá khả năng tổng quát hóa của mô hình trên dữ liệu mới.

Tóm lại, biểu đồ đã tạo ra một cái nhìn tổng quan đáng tin cậy về hiệu suất của mô hình hồi quy, nhưng việc tiếp tục kiểm tra và đánh giá sâu hơn là cần thiết để đảm bảo tính đáng tin cậy của mô hình.

Chương 5

Kết luận và đề xuất

5.1 Ưu điểm và nhược điểm của mô hình

Ưu điểm:

- Dễ hiểu và triển khai: Linear Regression là một mô hình đơn giản và dễ hiểu. Cách hoạt động của nó dựa trên mối quan hệ tuyến tính giữa biến độc lập và biến phụ thuộc, điều này làm cho nó dễ triển khai trong thực tế.

- Tính toán hiệu suất cao: Mô hình Linear Regression có thể được tính toán một cách hiệu quả với dữ liệu lớn, đặc biệt là khi sử dụng các phương pháp như Gradient Descent.

- Dùng được trong nhiều tình huống: Linear Regression không chỉ được sử dụng để dự đoán giá nhà mà còn có thể áp dụng trong nhiều lĩnh vực khác nhau như dự đoán doanh số bán hàng, dự đoán thu nhập cá nhân, v.v.

- Giả định đơn giản: Mô hình Linear Regression thường làm việc tốt khi dữ liệu có mối quan hệ tuyến tính, giả định đơn giản về mối quan hệ này có thể là ưu điểm khi dữ liệu thỏa mãn giả định này.

Nhược điểm:

- Độ phức tạp thấp: Mô hình Linear Regression có độ phức tạp thấp và không thể mô hình hóa các mối quan hệ phức tạp giữa biến độc lập và biến phụ thuộc.

- Nhạy cảm với nhiễu và điểm ngoại lai: Mô hình Linear Regression nhạy cảm với nhiễu và điểm ngoại lai trong dữ liệu, điều này có thể ảnh hưởng đến độ chính xác của dự đoán.

- Giả định về tuyến tính: Mô hình Linear Regression giả định mối quan hệ giữa các biến là tuyến tính, điều này có thể không phù hợp với dữ liệu thực tế nếu mối quan hệ thực sự không phải là tuyến tính.

- Khả năng giải quyết vấn đề overfitting hạn chế: Mô hình Linear Regression có khả năng giải quyết vấn đề overfitting (quá mức) hạn chế, đặc biệt là khi có nhiều biến độc lập không tuyến tính hoặc tương quan cao.

5. 2 Đề xuất cải tiến cho mô hình:

1. Thêm các biến phụ thuộc tuyến tính: Một cách đơn giản nhất để cải thiện mô hình Linear Regression là thêm các biến phụ thuộc tuyến tính. Điều này có thể giúp mô hình mô hình hóa mối quan hệ phức tạp hơn giữa biến độc lập và biến phụ thuộc.

2. Chuẩn hóa các biến độc lập: Chuẩn hóa các biến độc lập trước khi đưa vào mô hình có thể giúp mô hình hội tụ nhanh hơn và giảm thiểu ảnh hưởng của các biến có tỷ lệ khác nhau.

3. Xử lý điểm ngoại lai: Phát hiện và xử lý điểm ngoại lai có thể giúp cải thiện độ chính xác của mô hình bằng cách loại bỏ các ảnh hưởng không mong muốn từ các điểm ngoại lai.

4. Sử dụng Regularization: Sử dụng các kỹ thuật regularization như Lasso hoặc Ridge Regression có thể giúp kiểm soát overfitting và cải thiện hiệu suất của mô hình, đặc biệt là khi có nhiều biến độc lập có tương quan cao.

5. Sử dụng Polynomial Regression: Đôi khi mối quan hệ giữa biến độc lập và biến phụ thuộc không phải là tuyến tính. Trong trường hợp này, sử dụng Polynomial Regression có thể giúp mô hình mô hình hóa mối quan hệ phức tạp hơn.

6. Thử nghiệm các mô hình phức tạp hơn: Ngoài Linear Regression, bạn có thể thử nghiệm các mô hình phức tạp hơn như Decision Trees, Random Forests, hoặc Neural Networks để xem xét mô hình nào hoạt động tốt nhất cho dữ liệu của bạn.

7. Tinh chỉnh siêu tham số: Tinh chỉnh các siêu tham số của mô hình như learning rate, số lượng epochs, hoặc hệ số regularization có thể cải thiện hiệu suất của mô hình.

8. Sử dụng Ensemble Learning: Kết hợp nhiều mô hình Linear Regression thông qua các kỹ thuật như Bagging hoặc Boosting có thể tạo ra một mô hình mạnh mẽ hơn.

9. Sử dụng K-fold cross-validation là một phương pháp hiệu quả để cải tiến mô hình. K-fold cross-validation giúp đánh giá hiệu suất của mô hình một cách chính xác hơn bằng cách chia tập dữ liệu thành các tập con (folds), huấn luyện mô hình trên K-1 folds và kiểm tra trên fold còn lại. Quá trình này được lặp lại K lần với các fold khác nhau, sau đó kết quả được kết hợp để đánh giá hiệu suất trung bình của mô hình. Các ưu điểm của K-fold cross-validation bao gồm:

- Đánh giá chính xác hiệu suất của mô hình: K-fold cross-validation cung cấp một ước lượng chính xác về hiệu suất của mô hình trên toàn bộ tập dữ liệu, giúp đảm bảo tính tổng quát hóa và đánh giá chính xác hiệu suất dự đoán.

- Tận dụng tối đa dữ liệu: Mỗi mẫu dữ liệu được sử dụng để huấn luyện và kiểm tra mô hình, giúp tận dụng tối đa dữ liệu có sẵn.

- Phát hiện overfitting: K-fold cross-validation giúp phát hiện overfitting bằng cách đánh giá hiệu suất của mô hình trên các tập dữ liệu kiểm tra độc lập.

- Stability: Kết quả của K-fold cross-validation thường ổn định hơn so với việc chỉ sử dụng một tập dữ liệu kiểm tra duy nhất.

Tài liệu tham khảo

[1] Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, pages 119 - 127.

[2] Müller, A. C., & Guido, S. (2016). Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media.

[3] Raschka, S., & Mirjalili, V. (2019). Python Machine Learning. Packt Publishing.

[4] Ng, A. (2018). Machine Learning Yearning: Technical Strategy for AI Engineers, In the Era of Deep Learning. Independently published.

[5] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[6] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). An Introduction to Statistical Learning: with Applications in R. Springer.

[7] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

[8] Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer.

[9] Chatterjee, S., & Hadi, A. S. (2012). Regression Analysis by Example. John Wiley & Sons.

[10] Seber, G. A. F., & Lee, A. J. (2003). Linear Regression Analysis. John Wiley & Sons.

[11] Harrell Jr., F. E. (2015). Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis. Springer.