ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

AI23 (23TNT1), FIT@HCMUS-VNUHCM

Báo cáo Lab 1

Đề tài: Dự đoán giá xe bằng mô hình Linear Regression

Môn học: Phương pháp toán cho Trí tuệ nhân tạo

Sinh viên thực hiện:
Nguyễn Đình Hà Dương (23122002)
Nguyễn Lê Hoàng Trung (23122004)
Đinh Đức Tài (23122013)
Hoàng Minh Trung (23122014)

Giáo viên hướng dẫn:
TS. Cấn Trần Thành Trung
ThS. Nguyễn Ngọc Toàn

Ngày 5 tháng 4 năm 2025

Mục lục

1 Giới thiệu

Đây là bài báo cáo cho **Lab 1 - Dự đoán giá xe**, môn Phương pháp toán cho Trí tuệ nhân tạo, lớp Trí tuệ nhân tạo Khóa 2023 (23TNT1), Khoa Công nghệ thông tin, Trường Đại học Khoa học tự nhiên - Đại học Quốc gia TP.HCM. Trong bài báo cáo này, chúng tôi sẽ trình bày phương pháp **dự đoán giá xe** bằng **mô hình hồi quy tuyến tính** dựa trên dữ liệu huấn luyện được cho trước.

Báo cáo được thực hiện bởi nhóm các thành viên:

- Nguyễn Đình Hà Dương (23122002)
- Nguyễn Lê Hoàng Trung (23122004)
- Đinh Đức Tài (23122013)
- Hoàng Minh Trung (23122014)

Đường dẫn repository Github của báo cáo: https://github.com/ductai05/Math-For-AI [1]

Bảng phân công nhiệm vụ cho từng thành viên:

Họ và tên	MSSV	Nhiệm vụ
Nguyễn Đình Hà Dương	23122002	- Linear regression với phương pháp PCA. - Cơ sở toán học của Linear regression & Ridge regression & PCA.
Nguyễn Lê Hoàng Trung	23122004	- Polynomial linear regression. - Cơ sở toán học của Polynomial linear regression. Code testing.
Đinh Đức Tài	23122013	- Data preprocessing, visualization, encoding, normalization. - Simple linear regression & Evaluation metrics. Review report.
Hoàng Minh Trung	23122014	- Multiple linear regression - Cơ sở toán học của Multiple linear regression & Lasso regression

Các thư viện và công nghệ sử dụng:

- Numpy: thư viện Python để xử lý số học.
- Pandas: thư viện Python để thao tác và xử lý dữ liệu.
- Matplotlib: thư viện Python để trực quan hóa dữ liệu.
- Jupyter Notebook (thông qua jupyter, ipykernel): Môi trường làm việc tương tác cho phép kết hợp mã thực thi, văn bản mô tả (Markdown), công thức toán học và trực quan hóa trong cùng một tài liệu.
- Visual Studio Code: Trình soạn thảo mã nguồn (IDE).
- Git, Github: Quản lý dự án, lưu và chia sẻ source code.

2 Nền tảng toán học

2.1 Xác suất thống kê

Các kiến thức trong phần này được trích dẫn từ sách Giáo trình bài tập Xác suất thống kê [2], trường Đại học Khoa học tự nhiên, ĐHQG-HCM và một số trang thông tin khác.

2.1.1 Chuẩn hóa Z-score

Chuẩn hóa Z-score là phương pháp biến đổi dữ liệu có phân phối chuẩn bất kì về phân phối chuẩn hóa. Tức là, nếu u là giá trị chuẩn hóa của dữ liệu ban đầu thì $u \sim N(0, 1)$.

Dữ liệu sau quá trình chuẩn hóa thường được gọi là dữ liệu chuẩn hóa hoặc **điểm Z (Z-scores)**. Giá trị của Z-score thường nằm trong khoảng $[-3, 3]$.

Công thức chuẩn hóa đối với tổng thể: Nếu một biến ngẫu nhiên X tuân theo phân phối chuẩn (tức là $X \sim N(\mu, \sigma^2)$), thì điểm Z được tính bằng công thức:

$$Z = \frac{X - \mu}{\sigma}$$

Trong đó:

- Z: Điểm Z chuẩn hóa.
- X: Giá trị của biến ngẫu nhiên hoặc một giá trị cụ thể từ tổng thể.
- μ: Trung bình của tổng thể.
- σ: Độ lệch chuẩn của tổng thể.

Công thức chuẩn hóa đối với dữ liệu mẫu:

Trong thực tế, chúng ta thường làm việc với dữ liệu mẫu và không biết μ và σ. Khi đó, chúng ta sẽ ước lượng chúng bằng trung bình mẫu ($\bar{x}$) và độ lệch chuẩn mẫu (s). Điểm Z cho một giá trị cụ thể x trong mẫu được tính bằng công thức:

$$z = \frac{x - \bar{x}}{s}$$

Trong đó:

- z: Giá trị chuẩn hóa (điểm Z) của x dựa trên mẫu.
- x: Giá trị dữ liệu gốc trong mẫu.
- $\bar{x}$: Trung bình mẫu của dữ liệu (sample mean).
- s: Độ lệch chuẩn mẫu của dữ liệu.

2.1.2 Đường thẳng hồi quy tuyến tính đơn

Đường thẳng hồi quy tuyến tính đơn dùng để mô hình hóa mối quan hệ tuyến tính giữa biến độc lập thường x và biến phụ thuộc Y. Mục tiêu của đường thẳng hồi quy tuyến tính đơn là tìm ra **đường thẳng tốt nhất** mô tả mối quan hệ giữa x và Y trong một tập dữ liệu.

Phương trình của đường thẳng hồi quy tuyến tính đơn:

$$Y = \beta_0 + \beta_1 x + \varepsilon$$

Trong đó:

- β_0 và β_1 là các tham số chưa biết (được gọi lần lượt là hệ số chặn và hệ số góc của đường thẳng hồi quy).
- Y là biến phụ thuộc và x là biến độc lập;
- ε là thành phần sai số, được giả sử có phân phối chuẩn $\mathcal{N}(0, \sigma^2)$.

Trong thực tế, chúng ta không biết chính xác β_0 và β_1. Chúng ta sử dụng **dữ liệu mẫu** để ước tính chúng. Các ước lượng thường được ký hiệu là b_0 và b_1.

Phương trình đường thẳng hồi quy ước lượng:

$$\hat{y} = b_0 + b_1 x$$

Trong đó:

- $\hat{y}$ **(y mũ)**: Giá trị y **dự đoán** bởi mô hình hồi quy cho một giá trị x nhất định.
- b_0: Ước lượng của hệ số chặn β_0.
- b_1: Ước lượng của hệ số góc β_1.

Phương pháp phổ biến nhất để tìm b_0 và b_1 là **phương pháp bình phương tối thiểu (Ordinary Least Squares - OLS)**. Phương pháp này tìm cách tối thiểu hóa tổng bình phương của các sai số (ε) giữa giá trị thực tế và giá trị dự đoán.

Công thức tính b_1 và b_0 như sau:

$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

$$b_0 = \bar{y} - b_1 \bar{x}$$

Trong đó:

- x_i, y_i: Các cặp giá trị dữ liệu quan sát được.
- $\bar{x}$: Giá trị trung bình của biến x.
- $\bar{y}$: Giá trị trung bình của biến y.

2.1.3 Hiệp phương sai (Covariance)

Hiệp phương sai (Covariance) [3] là thước đo mối liên hệ tuyến tính giữa hai biến ngẫu nhiên X và Y. Ký hiệu: $cov(X, Y)$. Hiệp phương sai giữa hai biến ngẫu nhiên X và Y còn được định nghĩa là kỳ vọng của tích giữa độ lệch của X và Y so với giá trị kỳ vọng của chúng.

Công thức tính hiệp phương sai:

$$cov(X, Y) = E[(X - E(X))(Y - E(Y))]$$

Công thức tính trên tổng thể:

$$cov(X, Y) = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu_X)(y_i - \mu_Y)$$

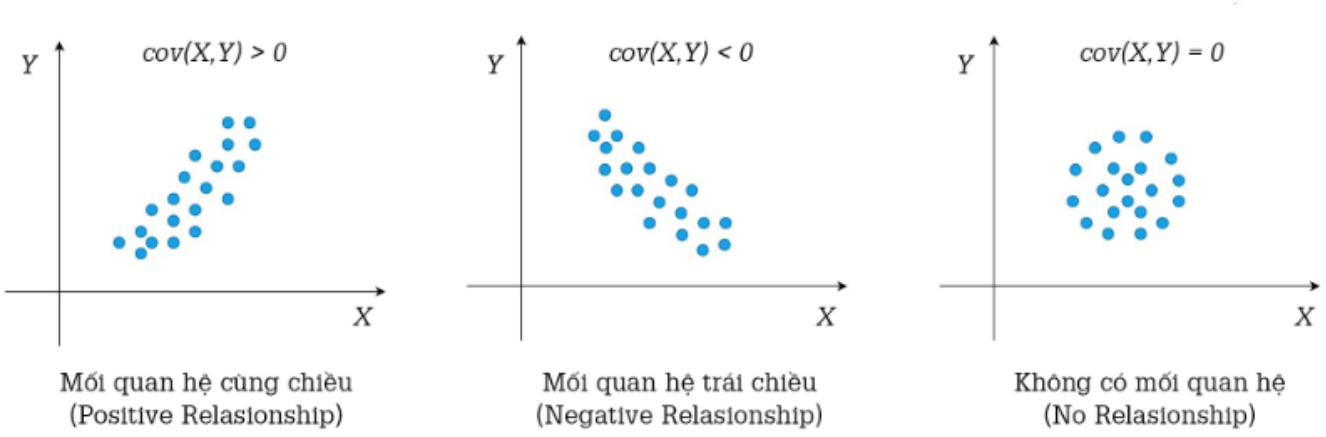
Công thức tính trên mẫu:

$$cov(X, Y) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

Trong đó:

- x_i, y_i là giá trị của quan sát thứ i.
- μ_X, μ_Y là giá trị trung bình của tổng thể.
- $\bar{x}, \bar{y}$ là giá trị trung bình của mẫu.
- N là tổng số quan sát của tổng thể.
- n là tổng số quan sát của mẫu.

Trực quan bằng đồ thị:



Hình 1: Minh họa hiệp phương sai giữa hai biến ngẫu nhiên X và Y.

Đồ thị trên minh họa ba trường hợp có thể xảy ra khi tính hiệp phương sai:

- Khi $\text{cov}(X, Y) > 0$: Hai biến X và Y có quan hệ tuyến tính thuận, khi X tăng thì Y cũng tăng.
- Khi $\text{cov}(X, Y) < 0$: Hai biến X và Y có quan hệ tuyến tính nghịch, khi X tăng thì Y giảm và ngược lại.
- Khi $\text{cov}(X, Y) = 0$: Hai biến X và Y không có mối quan hệ tuyến tính với nhau.

2.1.4 Ma trận hiệp phương sai (Covariance matrix)

Ma trận hiệp phương sai là một ma trận vuông chứa các hiệp phương sai giữa các biến trong một tập dữ liệu. Nếu một tập dữ liệu có p biến ngẫu nhiên $X_1, X_2, \ldots, X_p$, thì ma trận hiệp phương sai Σ có dạng:

$$\Sigma = \begin{bmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \ldots & \text{Cov}(X_1, X_p) \\ \text{Cov}(X_2, X_1) & \text{Var}(X_2) & \ldots & \text{Cov}(X_2, X_p) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(X_p, X_1) & \text{Cov}(X_p, X_2) & \ldots & \text{Var}(X_p) \end{bmatrix}$$

Trong đó:

- $\text{Var}(X_i) = \text{Cov}(X_i, X_i)$ là phương sai của biến X_i.
- $\text{Cov}(X_i, X_j)$ là hiệp phương sai giữa hai biến X_i và X_j.

Công thức tổng quát: Giả sử có một tập dữ liệu với n quan sát và p biến ngẫu nhiên được biểu diễn dưới dạng ma trận X có kích thước $n \times p$, với mỗi hàng là một quan sát và mỗi cột là một biến. Khi đó, ma trận hiệp phương sai được tính bằng công thức:

$$\Sigma = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})^T$$

Trong đó:

- X_i là vector giá trị của các biến tại quan sát thứ i.
- $\bar{X}$ là vector trung bình của từng biến.

Tính chất:

- Ma trận hiệp phương sai là **ma trận đối xứng**.
- Đường chéo chứa phương sai của từng biến.
- Nếu các biến không có tương quan (độc lập tuyến tính), thì các phần tử ngoài đường chéo bằng 0.

2.1.5 Hệ số tương quan Pearson (Pearson Correlation)

Độ tương quan (Correlation) là thước đo mối quan hệ tuyến tính giữa hai biến và không phụ thuộc vào các đơn vị đo lường của hai biến này.

Hệ số tương quan Pearson của 2 biến ngẫu nhiên X, Y, ký hiệu ρ_{xy}.

Công thức tính hệ số tương quan Pearson:
Đối với tổng thể:

$$\rho_{xy} = \frac{\text{cov}(X,Y)}{\sigma_x \sigma_y}$$

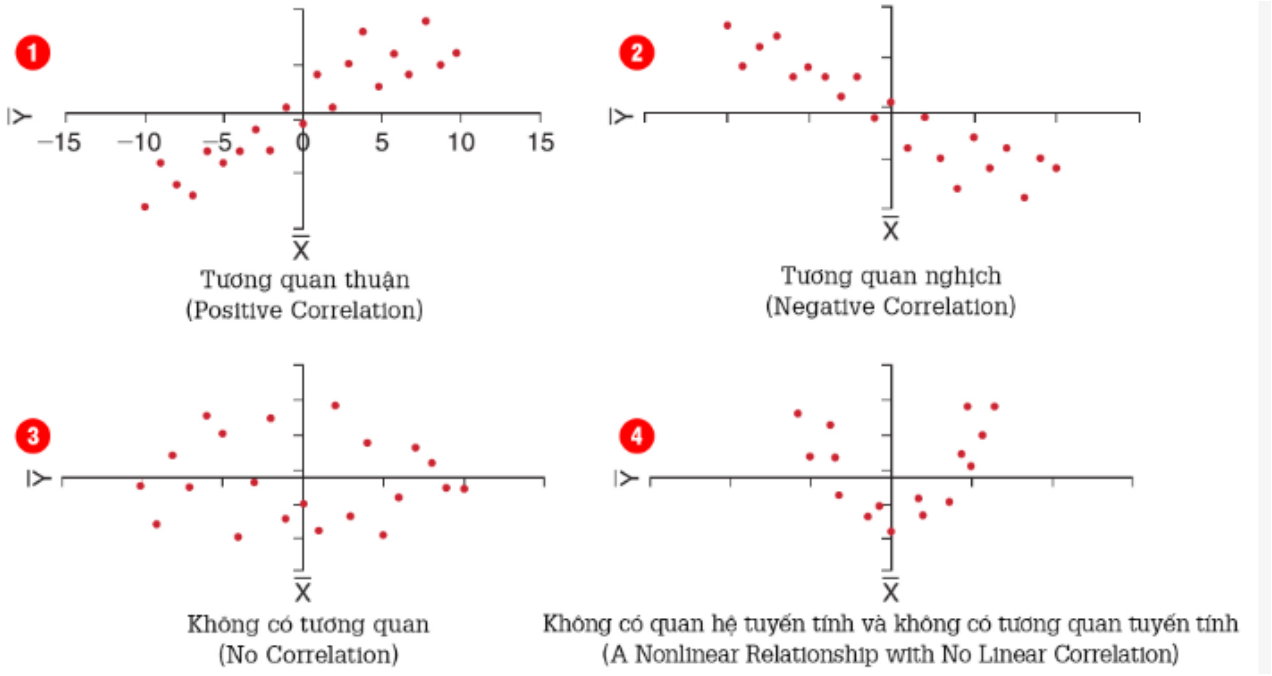
Đối với tập dữ liệu mẫu:

$$r_{xy} = \frac{\text{cov}(X,Y)}{s_x s_y}$$

Trong đó:

- $\text{cov}(X,Y)$ là hiệp phương sai của 2 biến ngẫu nhiên X, Y.
- σ_x, σ_y là độ lệch chuẩn của dữ liệu.
- s_x, s_y là độ lệch chuẩn của một phần dữ liệu.

Nhận xét: Độ lệch chuẩn đo lường độ biến thiên tuyệt đối của tập dữ liệu, do đó khi chia các giá trị hiệp phương sai cho độ lệch chuẩn, nó sẽ chia tỷ lệ giá trị xuống một phạm vi giới hạn từ -1 đến +1.



Hình 2: Minh họa sự tương quan của 2 biến ngẫu nhiên

- Tương quan thuận (Positive Correlation): $\rho_{xy} > 0$, X tăng, Y tăng.
- Tương quan nghịch (Negative Correlation): $\rho_{xy} < 0$, X tăng, Y giảm.
- Không có tương quan (No Correlation): $\rho_{xy} = 0$.
- Mối quan hệ phi tuyến tính (Nonlinear Relationship): Không có tương quan tuyến tính.

2.1.6 Ma trận tương quan

Ma trận tương quan là một ma trận vuông có kích thước $n \times n$ (với n là số biến), trong đó mỗi phần tử r_{ij} biểu diễn hệ số tương quan giữa biến X_i và X_j.

Ví dụ với n biến $X_1, X_2, \ldots, X_n$, ma trận tương quan được biểu diễn như sau:

$$R = \begin{bmatrix} 1 & r_{12} & r_{13} & \ldots & r_{1n} \\ r_{21} & 1 & r_{23} & \ldots & r_{2n} \\ r_{31} & r_{32} & 1 & \ldots & r_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & r_{n3} & \ldots & 1 \end{bmatrix}$$

Trong đó:

- $r_{ii} = 1$ vì hệ số tương quan của một biến với chính nó luôn là 1.
- $r_{ij} = r_{ji}$, tức là ma trận đối xứng.

2.2 Thuật toán Gradient Decent

Các kiến thức trong phần này được trích dẫn từ sách Giáo trình Vi tích phân 2 [4], bộ môn Giải tích, khoa Toán - Tin học, trường Đại học Khoa học tự nhiên, ĐHQG-HCM và một số blog khác.

2.2.1 Gradient Descent

Gradient Descent [5] tìm nghiệm của bài toán tối ưu:

$$\theta^* = \arg\min_\theta f(\theta)$$

Bằng cách cập nhật tham số θ theo công thức:

$$\theta_{t+1} = \theta_t - \alpha \nabla f(\theta_t)$$

Trong đó:

- θ_t là giá trị hiện tại của tham số,
- α là learning rate,
- $\nabla f(\theta_t)$ là gradient của hàm mất mát.

Nếu Gradient Descent hội tụ, ta có nghiệm tối ưu θ^* thỏa mãn:

$$\nabla f(\theta^*) = 0$$

2.2.2 Ma trận Hessian

Ma trận **Hessian** của một hàm số thực khả vi hai lần $f(x_1, x_2, \ldots, x_n)$ được định nghĩa là:

$$H_f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Phân loại điểm dừng dựa vào ma trận Hessian:

- Nếu H_f **dương xác định** $\Rightarrow$ điểm cực tiểu.
- Nếu H_f **âm xác định** $\Rightarrow$ điểm cực đại.
- Nếu H_f **không xác định** $\Rightarrow$ điểm yên ngựa.
- Nếu H_f bằng 0 hoặc có định thức bằng 0 $\Rightarrow$ không kết luận được.

2.2.3 Tốc độ hội tụ của Gradient Descent

Gradient Descent cập nhật tham số theo công thức:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla J(\theta^{(t)})$$

Với $J(\theta)$ là hàm mất mát, $\nabla J(\theta)$ là gradient và α là tốc độ học.

Mô hình hội tụ: [6]

Giả sử $J(\theta)$ là một hàm bậc hai:

$$J(\theta) = \frac{1}{2}\theta^T H \theta - b^T \theta$$

với H là ma trận Hessian đối xứng xác định dương. Khi đó, gradient là:

$$\nabla J(\theta) = H\theta - b$$

Suy ra phương trình cập nhật:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha H(\theta^{(t)} - H^{-1}b)$$

Đặt $e^{(t)} = \theta^{(t)} - \theta^*$ khi đó ta giả sử $\theta^* = H^{-1}b$ là nghiệm tối ưu, ta có:

$$e^{(t+1)} = (I - \alpha H)e^{(t)}$$

Giả sử H có các giá trị riêng $\lambda_1, \lambda_2, \ldots, \lambda_n$, ta phân rã eigen:

$$H = Q\Lambda Q^{-1}, \quad \Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

Khi đó:

$$e^{(t)} = Q(I - \alpha\Lambda)^t Q^T e^{(0)}$$

Do đó, tốc độ hội tụ phụ thuộc vào:

$$\|e^{(t)}\|_2 \leq \rho^t \|e^{(0)}\|_2$$

với $\rho = \max|1 - \alpha\lambda_i|$. Điều kiện hội tụ:

$$0 < \alpha < \frac{2}{\lambda_{\max}}$$

Tốc độ hội tụ được ước lượng bởi:

$$\frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} = \frac{\kappa(H) - 1}{\kappa(H) + 1}$$

Với $\kappa(H) = \frac{\lambda_{\max}}{\lambda_{\min}}$ là số điều kiện của H. Nếu $\kappa(H) \gg 1$, thuật toán hội tụ rất chậm.

2.2.4 Ảnh hưởng của dữ liệu chưa chuẩn hóa đối với Gradient Descent

Nếu dữ liệu chưa chuẩn hóa, ma trận Hessian $H = X^T X$ có các giá trị riêng phân bố không đều, khiến $\kappa(H)$ lớn và làm giảm tốc độ hội tụ.

Ví dụ, xét dữ liệu với hai biến: - $x_1 \in [0, 1]$ - $x_2 \in [1, 10^6]$

Khi đó, Hessian H có một giá trị riêng rất lớn và một giá trị riêng rất nhỏ, làm tăng $\kappa(H)$, khiến Gradient Descent dao động mạnh và hội tụ chậm.

Khi đưa về giá trị chuẩn (Standardization)

Dữ liệu được chuẩn hóa theo:

$$x_i' = \frac{x_i - \mu_i}{\sigma_i}$$

Ma trận Hessian mới:

$$H' = X'^T X'$$

Các giá trị riêng của H' được cân bằng hơn, giúp giảm $\kappa(H')$. Nếu chuẩn hóa tốt, ta có thể đạt $\kappa(H') \approx 1$, giúp Gradient Descent hội tụ nhanh hơn.

Kết luận: Chuẩn hóa dữ liệu giúp cải thiện tốc độ hội tụ của Gradient Descent bằng cách giảm số điều kiện $\kappa(H)$, làm thuật toán ổn định hơn và ít dao động hơn.

2.3 Hồi quy tuyến tính

Các kiến thức trong phần này được trích dẫn từ sách Mathematics for Machine Learning [7] - Đại học Cambridge và một số blog khác.

2.3.1 Hồi quy tuyến tính (Linear regression)

Mô hình toán học

Hồi quy tuyến tính [8] là một phương pháp thống kê dùng để dự báo mối quan hệ giữa biến độc lập (biến đầu vào x_i) và biến phụ thuộc (biến đầu ra y). Mô hình tuyến tính tổng quát có dạng:

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + \varepsilon \tag{1}$$

Trong đó:

- y là biến phụ thuộc (đầu ra cần dự đoán),
- $x_1, x_2, \ldots, x_n$ là các biến độc lập (đặc trưng đầu vào),
- $w_0, w_1, \ldots, w_n$ là các hệ số hồi quy (cần tìm),
- ε là nhiễu ngẫu nhiên.

Hàm mất mát

Để đo lường sai số giữa giá trị thực tế và giá trị dự đoán, ta sử dụng hàm mất mát bình phương trung bình (Mean Squared Error - MSE):

$$J(w) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 \tag{2}$$

với:

- m là số lượng mẫu dữ liệu,
- y_i là giá trị thực tế của mẫu thứ i,
- $\hat{y}_i$ là giá trị dự đoán của mẫu thứ i, được tính theo công thức:

$$\hat{y}_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_n x_{in} \tag{3}$$

Mục tiêu của mô hình là tìm các tham số w sao cho hàm mất mát $J(w)$ đạt giá trị nhỏ nhất. Có hai cách tiếp cận chính:

1. **Phương pháp đạo hàm và giải hệ phương trình**

   Khi số lượng biến không quá lớn, ta có thể tìm nghiệm bằng cách giải phương trình đạo hàm bằng không:

   $$w = (X^T X)^{-1} X^T y \tag{4}$$

   với X là ma trận đặc trưng của dữ liệu đầu vào.

2. **Gradient Descent (Hạ Gradient)**

   Khi số lượng biến lớn, ta có thể sử dụng phương pháp hạ gradient để tối ưu:

   - **Gradient của hàm mất mát theo w_j:**

   $$\frac{\partial J}{\partial w_j} = -\frac{2}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i) x_{ij} \tag{5}$$

   - **Cập nhật tham số w_j theo thuật toán Gradient Descent:**

   $$w_j := w_j - \alpha \frac{\partial J}{\partial w_j} \tag{6}$$

   trong đó:

   - α là tốc độ học (learning rate),
   - quá trình này lặp lại cho đến khi hội tụ.

2.3.2 Lý thuyết tìm cực trị có điều kiện Lagrange

Giả sử ta cần tìm cực trị của hàm số $f(x_1, x_2, ..., x_n)$ với ràng buộc:

$$g(x_1, x_2, ..., x_n) = 0$$

Phương pháp nhân tử Lagrange xây dựng hàm:

$$\mathcal{L}(x_1, x_2, ..., x_n, \lambda) = f(x_1, x_2, ..., x_n) + \lambda g(x_1, x_2, ..., x_n)$$

trong đó λ là nhân tử Lagrange.

Điều kiện cần để tìm cực trị là:

$$\nabla \mathcal{L} = 0 \Rightarrow \begin{cases} \frac{\partial \mathcal{L}}{\partial x_1} = 0 \\ \frac{\partial \mathcal{L}}{\partial x_2} = 0 \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial x_n} = 0 \\ \frac{\partial \mathcal{L}}{\partial \lambda} = g(x_1, x_2, ..., x_n) = 0 \end{cases}$$

Giải hệ phương trình trên giúp tìm nghiệm tối ưu của bài toán có ràng buộc.

2.3.3 Hồi quy Ridge

Hồi quy Ridge [9] là bài toán tối ưu có ràng buộc:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - X_i\beta)^2 \quad \text{với điều kiện} \quad \sum_{j=1}^{p} \beta_j^2 \leq C$$

Điều kiện ràng buộc $\sum_{j=1}^{p} \beta_j^2 < C$ cho thấy nghiệm tối ưu sẽ bị hạn chế về độ lớn. Trong không gian đa chiều thì điều kiện ràng buộc có miền xác định là một khối cầu có tâm là gốc tọa độ và bán kính $\sqrt{C}$. Đây chính là một cơ chế kiểm soát mà *thành phần điều chuẩn* đã áp đặt lên các biến đầu vào.

Sử dụng phương pháp Lagrange, ta xây dựng hàm:

$$\mathcal{L}(\beta, \lambda) = \sum_{i=1}^{n} (y_i - X_i\beta)^2 + \lambda \left(\sum_{j=1}^{p} \beta_j^2 - C \right)$$

Lấy đạo hàm theo β:

$$\frac{\partial \mathcal{L}}{\partial \beta} = -2X^T(y - X\beta) + 2\lambda\beta = 0$$

Giải phương trình trên, ta thu được nghiệm tối ưu:

$$\beta_{ridge} = (X^T X + \lambda I)^{-1} X^T y$$

So với hồi quy tuyến tính thông thường (OLS):

- Khi $X^T X$ gần suy biến, việc tính $\beta_{OLS} = (X^T X)^{-1} X^T y$ không ổn định, dẫn đến ước lượng β_{OLS} có phương sai lớn.
- Hồi quy Ridge thay thế $X^T X$ bằng $X^T X + \lambda I$, với $\lambda > 0$, giúp Ma trận $X^T X + \lambda I$ luôn dương xác định và Việc tính nghịch đảo trở nên ổn định hơn.Phương sai của ước lượng giảm, cho ra β_{ridge} ổn định hơn.

2.3.4 Hồi quy Lasso

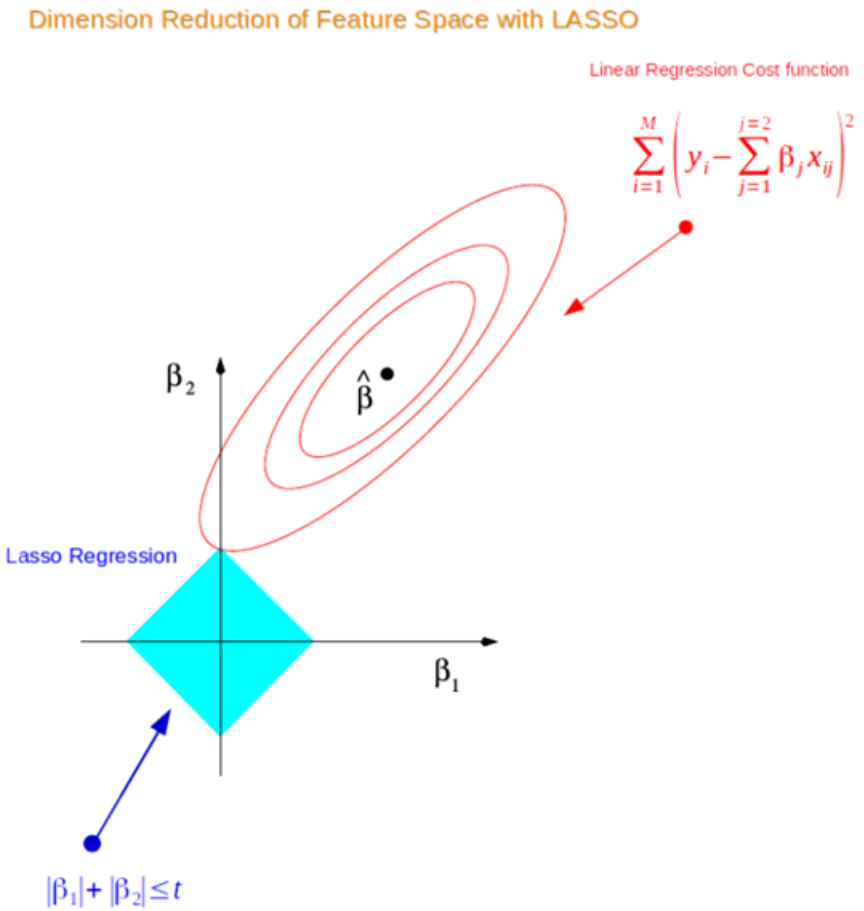
Lasso Regression [10] là một phương pháp hồi quy tuyến tính có điều chuẩn (regularization), được sử dụng để cải thiện hiệu suất của mô hình bằng cách giảm hiện tượng quá khớp (overfitting) thông qua công thức:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - X_i\beta)^2 \quad \text{với điều kiện} \quad \sum_{j=1}^{p} |\beta_j| \leq C \tag{7}$$

- C là hằng số dương, liên quan đến λ. Khi λ lớn, t nhỏ và ngược lại.

- Ràng buộc $\sum_{j=1}^{p} |\beta_j| \leq C$ giới hạn tổng giá trị tuyệt đối của các hệ số β_j, tương ứng với chuẩn L1.

Mô hình cần tối ưu hàm sai số với điều kiện là phải thỏa ràng buộc đã đặt ra.



Hình 3

Lấy ví dụ với trường hợp chỉ có 2 đặc trưng, tức là chỉ có 2 hệ số β_1, β_2. Khi đó, không gian của β là một mặt phẳng 2 chiều, với trục hoành là β_1 và trục tung là β_2. Ta tiến hành giảm sai số dự đoán sao cho tổng giá trị tuyệt đối của các hệ số β_j không được vượt quá một ngưỡng nào đó, ví dụ $|\beta_1| + |\beta_2| \leq t$. Các giá trị của β_1 và β_2 bị giới hạn trong một khu vực có dạng hình thoi và hình thoi này có các đỉnh tại các trục

Hàm mất mát của mô hình được biểu diễn bằng các vòng tròn hoặc elip (gọi là đường đồng mức), với tâm là điểm tối ưu của hồi quy thông thường (OLS). Các elip này sẽ mở rộng từ tâm cho đến khi chạm vào biên của hình thoi (vừa đủ thỏa ràng buộc nhưng vẫn gần OLS nhất). Hình thoi có các góc nhọn tại các trục (nơi $\beta_1 = 0$ hoặc $\beta_2 = 0$). Khi elip mở rộng, nó thường chạm vào các góc nhọn này trước vì các góc nhọn "nhô ra" xa hơn so với các cạnh. Đó là vì sao đôi lúc Lasso có thể loại bỏ hoàn toàn một vài đặc trưng đóng góp cho mô hình.

Sử dụng phương pháp Lagrange Ta xây dựng hàm mất mát từ (7):

$$\mathcal{L}(\beta, \lambda) = \sum_{i=1}^{n} \left(y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Các bước cập nhật tham số trong Lasso Việc tối thiểu hóa hàm mất mát của Lasso không có nghiệm dạng đóng (tính nghiệm trực tiếp) như hồi quy tuyến tính thông thường, do thành phần điều chuẩn L1 ($\lambda\|\boldsymbol{\beta}\|_1$) không khả vi tại $\beta_j = 0$. Một phương pháp phổ biến để giải bài toán này là sử dụng thuật toán *Coordinate Descent*, trong đó các tham số β_0 và $\boldsymbol{\beta}$ được cập nhật lần lượt từng thành phần.

1. **Cập nhật hệ số chặn** β_0: Coi $\boldsymbol{\beta}$ là cố định, ta tối thiểu hóa hàm mất mát theo β_0. Kết quả là:

   $$\beta_0 = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} \beta_j x_{ij} \right)$$

   Dưới dạng ma trận, điều này tương đương với:

   $$\beta_0 = \frac{1}{n} \mathbf{1}^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

   Bước này đảm bảo rằng trung bình của các giá trị dự đoán khớp với trung bình của $\mathbf{y}$.

2. **Cập nhật từng hệ số** β_j: Coi β_0 và các β_k (với $k \neq j$) là cố định, ta tối thiểu hóa hàm mất mát theo β_j. Do thành phần L1, nghiệm của β_j có dạng một hàm *soft-thresholding*:

   $$\beta_j = S\left(\frac{1}{n} \sum_{i=1}^{n} x_{ij} r_{i(j)}, \frac{\lambda}{n} \right)$$

   Trong đó:

   - $r_{i(j)} = y_i - \beta_0 - \sum_{k \neq j} \beta_k x_{ik}$ là phần dư (residual) của mẫu i khi bỏ qua đóng góp của đặc trưng j.
   - $S(z, \gamma)$ là hàm soft-thresholding, được định nghĩa như sau:

   $$S(z, \gamma) = \begin{cases} z - \gamma & \text{nếu } z > \gamma \\ 0 & \text{nếu } |z| \leq \gamma \\ z + \gamma & \text{nếu } z < -\gamma \end{cases}$$

   Dưới dạng ma trận, ta có thể tính $\mathbf{r}_{(j)} = \mathbf{y} - \beta_0 \mathbf{1} - \mathbf{X}_{-j}\boldsymbol{\beta}_{-j}$, trong đó $\mathbf{X}_{-j}$ là ma trận $\mathbf{X}$ bỏ cột j, và $\boldsymbol{\beta}_{-j}$ là vector $\boldsymbol{\beta}$ bỏ thành phần j. Khi đó:

   $$\beta_j = S\left(\frac{1}{n} \mathbf{x}_j^T \mathbf{r}_{(j)}, \frac{\lambda}{n} \right)$$

   với $\mathbf{x}_j$ là cột j của ma trận $\mathbf{X}$.

Thuật toán Coordinate Descent đảm bảo rằng hàm mất mát giảm dần qua mỗi bước, và cuối cùng hội tụ đến nghiệm tối ưu của Lasso. Điểm đặc biệt của bước cập nhật β_j là hàm soft-thresholding có thể ép β_j về 0 nếu giá trị trung gian $\frac{1}{n}\mathbf{x}_j^T\mathbf{r}_{(j)}$ nhỏ hơn ngưỡng $\frac{\lambda}{n}$, điều này giải thích tại sao Lasso có khả năng lựa chọn đặc trưng.

So sánh Lasso Regression với hồi quy tuyến tính thông thường (OLS)

- Lasso có khả năng tự động lựa chọn đặc trưng nhờ thành phần điều chuẩn L1. Trong khi đó, OLS không có cơ chế điều chuẩn, nên tất cả các đặc trưng đều được giữ lại, kể cả những đặc trưng không có ý nghĩa, điều này có thể làm tăng độ phức tạp của mô hình.
- Lasso giúp giảm hiện tượng quá khớp (overfitting) bằng cách giới hạn tổng giá trị tuyệt đối của các hệ số β_j thông qua hệ số điều chuẩn λ. Ngược lại, OLS không có điều chuẩn, nên nếu dữ liệu có nhiều đặc trưng hoặc các đặc trưng có tương quan cao, mô hình có thể học quá chi tiết trên tập huấn luyện, dẫn đến hiệu suất kém trên tập kiểm tra.
- OLS có nghiệm dạng đóng, tức là ta có thể tính trực tiếp các hệ số $\boldsymbol{\beta}$ bằng công thức $\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. Trong khi đó, Lasso không có nghiệm dạng đóng do thành phần L1 không khả vi tại 0, nên cần sử dụng các phương pháp lặp như Coordinate Descent để tìm nghiệm, dẫn đến quá trình đạo hàm phức tạp hơn.
- Trong các bài toán có số đặc trưng lớn (nhiều hơn số mẫu), OLS có thể gặp vấn đề vì ma trận $\mathbf{X}^T\mathbf{X}$ không khả nghịch (do ma trận không vuông), dẫn đến không thể tính được nghiệm. Lasso, nhờ điều chuẩn L1 nó thu nhỏ các hệ số và loại bỏ các đặc trưng không cần thiết, làm giảm chiều dữ liệu hiệu quả.

2.4 Giảm chiều dữ liệu với phương pháp PCA (Principal Component Analysis)

Phân tích thành phần chính (Principal Component Analysis - PCA) [11] là phương pháp giảm chiều dữ liệu bằng cách tìm một hệ tọa độ mới sao cho:

- Các thành phần chính (Principal Components - PCs) là các hướng có phương sai lớn nhất.
- Các PCs vuông góc (trực giao) với nhau.
- PC1 mang nhiều thông tin nhất, PC2 mang thông tin còn lại.

2.4.1 Các bước thực hiện PCA

Tính vector kỳ vọng của toàn bộ dữ liệu:

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

Tính độ lệch chuẩn của từng đặc trưng:

$$\sigma_d = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(x_{n,d} - \bar{x}_d)^2}$$

trong đó:

- $x_{n,d}$ là giá trị của đặc trưng thứ d tại điểm dữ liệu n.
- $\bar{x}_d$ là giá trị trung bình của đặc trưng d.
- σ_d là độ lệch chuẩn của đặc trưng d.

Trừ mỗi điểm dữ liệu đi vector kỳ vọng của toàn bộ dữ liệu:

$$\hat{x}_n = x_n - \bar{x}$$

Nếu các đặc trưng không có cùng miền giá trị thì ta chuẩn hóa bằng phương pháp Z-Score (xem 2.1.1)

$$\hat{x}_n = \frac{x_n - \bar{x}}{\sigma_d}$$

Tính ma trận hiệp phương sai:

$$S = \frac{1}{N}\hat{X}\hat{X}^T$$

Tính các trị riêng và vector riêng có norm bằng 1 của ma trận này, sắp xếp chúng theo thứ tự giảm dần của trị riêng.

Chọn K vector riêng với K trị riêng lớn nhất để xây dựng ma trận U_K có các cột tạo thành một hệ trực giao. K vectors này, còn được gọi là các thành phần chính, tạo thành một không gian con gần với dữ liệu ban đầu đã chuẩn hóa.

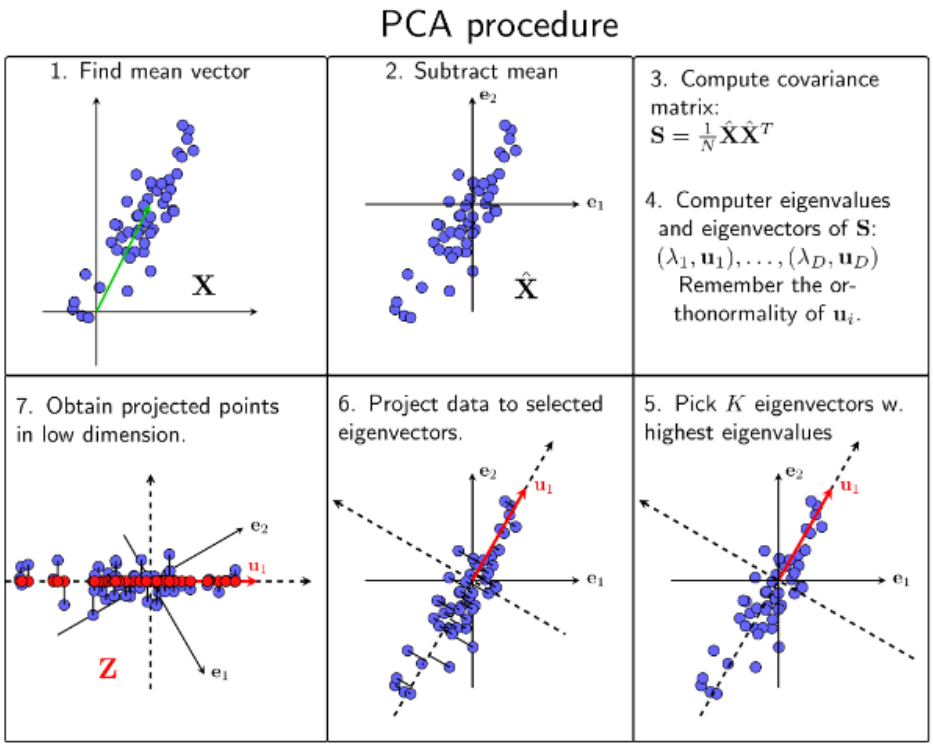
Chiếu dữ liệu ban đầu đã chuẩn hóa $\hat{X}$ xuống không gian con tìm được:

$$Z = U_K^T\hat{X}$$

Dữ liệu mới chính là tọa độ của các điểm dữ liệu trên không gian mới.

Dữ liệu ban đầu có thể tính được xấp xỉ theo dữ liệu mới như sau:

$$x \approx U_K Z + \bar{x}$$



Hình 4: Minh họa các bước thực hiện PCA

2.4.2 Ví dụ PCA từ ma trận 2 chiều về 1 chiều

Giả sử ta có ma trận 4X2

$$X = \begin{bmatrix} 2 & 3 \\ 3 & 5 \\ 5 & 8 \\ 7 & 10 \end{bmatrix}$$

Chuẩn hóa dữ liệu

Tính trung bình từng cột:

$$\mu_1 = \frac{2+3+5+7}{4} = 4.25, \quad \mu_2 = \frac{3+5+8+10}{4} = 6.5$$

Khi đó, ta có:

$$X' = X - \mu$$

$$X' = \begin{bmatrix} -2.25 & -3.5 \\ -1.25 & -1.5 \\ 0.75 & 1.5 \\ 2.75 & 3.5 \end{bmatrix}$$

Tính ma trận hiệp phương sai

$$S = \begin{bmatrix} 4.9167 & 6.8333 \\ 6.833 & 9.6667 \end{bmatrix}$$

Tính trị riêng và véc tơ riêng

Giải phương trình trị riêng:

$$\begin{vmatrix} 4.9167 - \lambda & 6.8333 \\ 6.833 & 9.6667 - \lambda \end{vmatrix} = 0$$

Giải phương trình bậc hai, ta được:

$$\lambda_1 = 14.5259648, \quad \lambda_2 = 0.0573$$

Véc tơ riêng tương ứng:

$$v_1 = \begin{bmatrix} -0.815 \\ 0.579 \end{bmatrix}, \quad v_2 = \begin{bmatrix} -0.579 \\ -0.815 \end{bmatrix}$$

Chiếu dữ liệu và trục chính

$$Z = X'V_1 = \begin{bmatrix} -2.25 & -3.5 \\ -1.25 & -1.5 \\ 0.75 & 1.5 \\ 2.75 & 3.5 \end{bmatrix} \begin{bmatrix} -0.815 \\ 0.579 \end{bmatrix} = \begin{bmatrix} -4.156 \\ -1.946 \\ 1.657 \\ 4.446 \end{bmatrix}$$

Vậy, ma trận 4x2 chiều đã được chuyển thành ma trận 4x1 chiều với giá trị mới Z.

3 Xử lý dữ liệu

3.1 Tải dữ liệu và tổng quan về dữ liệu

Dữ liệu phục vụ huấn luyện mô hình được cung cấp trong file `train.csv`, trong đó bao gồm:

- **1647 dòng**, cung cấp chi tiết về thông số kỹ thuật, đặc điểm và giá của các loại xe ô tô.
- **20 cột** dữ liệu, trong đó có 8 cột chứa dữ liệu số, 12 cột chứa dữ liệu chữ:
  - `Make`, `Model`: Hãng sản xuất xe, tên mẫu xe cụ thể.
  - `Price`: Giá niêm yết hoặc giá bán của xe.
  - `Year`: Năm sản xuất của xe.
  - `Kilometer`: Tổng số kilomet xe đã di chuyển.
  - `Fuel Type`: Loại nhiên liệu xe sử dụng (ví dụ: Xăng, Dầu Diesel, Điện).
  - `Transmission`: Loại hộp số (ví dụ: Số sàn, Số tự động).
  - `Location`: Địa điểm, thành phố hoặc khu vực bán xe.
  - `Color`: Màu sơn ngoại thất của xe.
  - `Owner`: Số lượng chủ sở hữu trước đây của xe.
  - `Seller Type`: Loại hình người bán (ví dụ: Cá nhân, Đại lý).
  - `Engine`: Dung tích xi-lanh của động cơ, thường tính bằng cc (cubic centimeters).
  - `Max Power`: Công suất cực đại mà động cơ có thể tạo ra (bhp - brake horsepower).
  - `Max Torque`: Mô-men xoắn cực đại mà động cơ có thể tạo ra (Nm - Newton-meters).
  - `Drivetrain`: Hệ thống truyền động lực từ động cơ đến bánh xe
  - `Length`, `Width`, `Height`: Chiều dài, chiều rộng, chiều cao tổng thể của xe (mm).
  - `Seating Capacity`: Số lượng chỗ ngồi tối đa trong xe (bao gồm cả lái xe).
  - `Fuel Tank Capacity`: Dung tích tối đa của bình chứa nhiên liệu, tính bằng lít.
- 149 dòng có chứa dữ liệu NaN (Not a Number).

	Make	Model	Price	Year	Kilometer	Fuel Type	Transmission	Location	Color	Owner	Seller Type	Engine	Max Power	Max Torque	Drivetrain	Length	Width
0	BMW	3-Series 320d	800000	2012	75576	Diesel	Automatic	Mumbai	White	Second	Individual	NaN	NaN	NaN	NaN	NaN	NaN
1	BMW	X1 sDrive20d xLine	2199000	2016	77000	Diesel	Automatic	Surat	Black	First	Individual	1995 cc	184 bhp @ 4000 rpm	350 Nm @ 1750 rpm	RWD	4454.0	2044.0
2	Mahindra	XUV500 W4 1.99	800000	2017	112000	Diesel	Manual	Muzaffurpur	Silver	First	Individual	1997 cc	138 bhp @ 3750 rpm	320 Nm @ 1600 rpm	FWD	4585.0	1890.0
3	Mercedes-Benz	GLS 400d 4MATIC	12900000	2021	3000	Diesel	Automatic	Delhi	White	First	Individual	2925 cc	326 bhp @ 3600 rpm	700 Nm @ 1200 rpm	AWD	5207.0	2157.0
4	Toyota	Fortuner 2.8 4x2 AT [2016-2020]	3499000	2019	73000	Diesel	Automatic	Mumbai	White	First	Individual	2755 cc	174 bhp @ 3400 rpm	450 Nm @ 1600 rpm	RWD	4795.0	1855.0

Hình 5: Các dòng đầu tiên trong bộ dữ liệu

3.2 Tiền xử lý dữ liệu - Data preprocessing

3.2.1 Chia tập dữ liệu huấn luyện - kiểm chứng

Khi xử lí dữ liệu đầu vào, các bước như làm sạch dữ liệu và mã hóa dữ liệu sẽ lấy thông tin của tổng thể như trung vị (median), giá trị có tần suất xuất hiện lớn nhất để điền vào các dòng bị thiếu thông tin. Vì vậy, để tránh hiện tượng **rò rỉ thông tin** (data leakage), ta cần chia tập dữ liệu thành **tập huấn luyện** (training set) và **tập kiểm chứng** (validation set) ngay trước khi xử lí dữ liệu.

Chia tập dữ liệu thành **tập huấn luyện** và **tập kiểm chứng** giúp ta đánh giá mô hình chính xác hơn, giúp nhận biết tốt hiện tượng **quá khớp** (overfitting) hoặc **dưới khớp** (underfitting).

Tập huấn luyện và tập kiểm chứng được chia với tỉ lệ lần lượt là 0.8 và 0.2 so với tổng dữ liệu.

3.2.2 Làm sạch dữ liệu

Ở bước làm sạch dữ liệu (Data cleaning), có 3 kĩ thuật chính được sử dụng:

- Điền vào ô NaN ở các cột có đặc trưng số với giá trị trung bình của chúng.
- Điền vào ô NaN ở các cột có đặc trưng phi số với giá trị xuất hiện thường xuyên nhất.
- Xóa các dòng dữ liệu lặp.

Tập kiểm chứng sẽ được điền vào ô NaN dựa trên thông tin (giá trị trung bình, giá trị xuất hiện thường xuyên nhất) lấy được từ tập huấn luyện.

3.2.3 Chuyển đổi dữ liệu

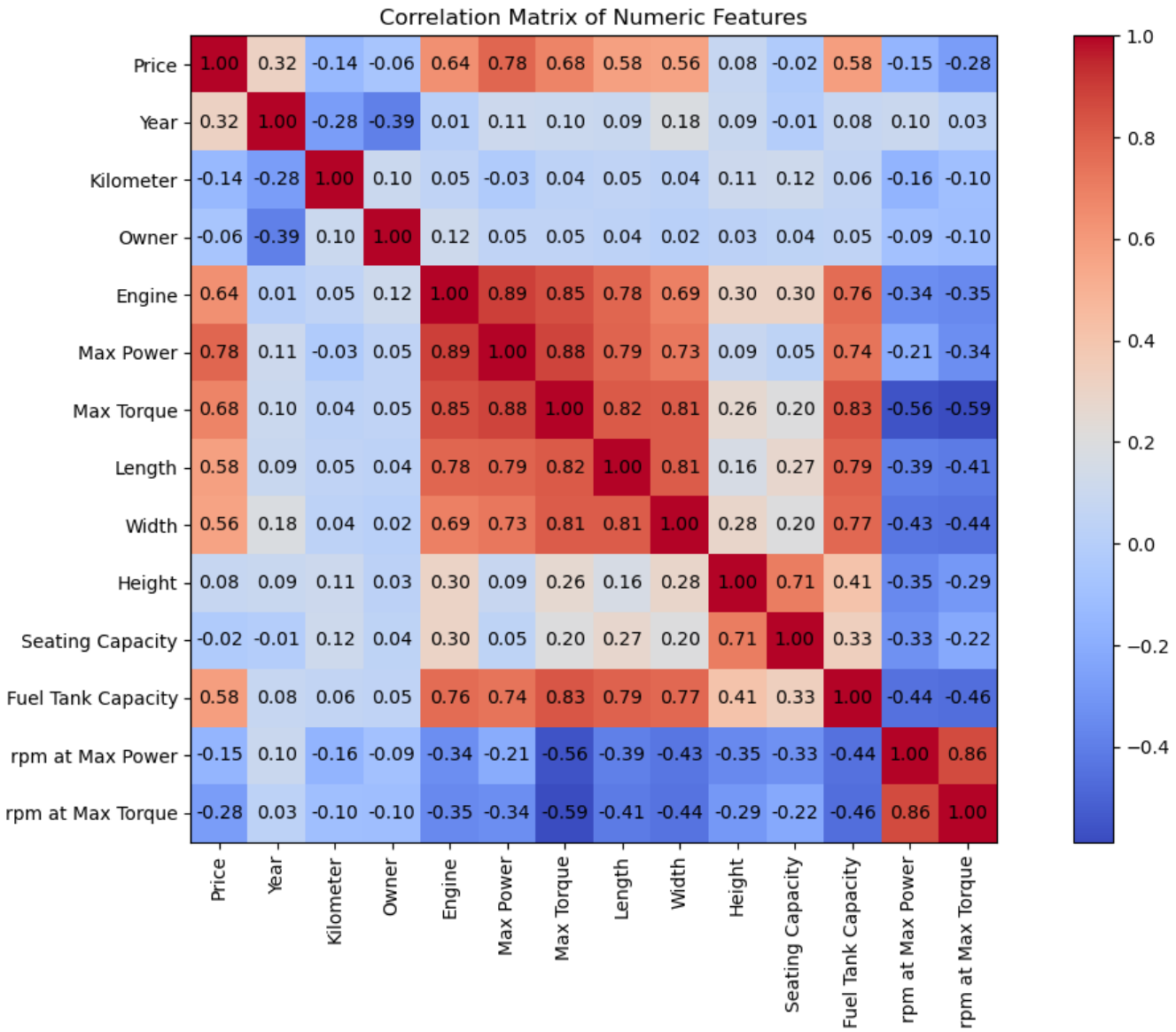
Ở bước chuyển đổi dữ liệu (Data transformation), có 3 bước:

- Thêm các cột dữ liệu: `rpm at Max Power` (vòng tua tại công suất cực đại) và `rpm at Max Torque` (vòng tua tại Mô-men xoắn cực đại).
- Cột dữ liệu `Max Power` và `Max Torque` được biến đổi, không còn các dữ liệu về vòng tua do đã được tách ra thành cột riêng.
- Các cột có đặc trưng số sẽ được đưa về dạng dữ liệu `int` (số nguyên).

3.3 Trực quan hóa dữ liệu - Data visualization

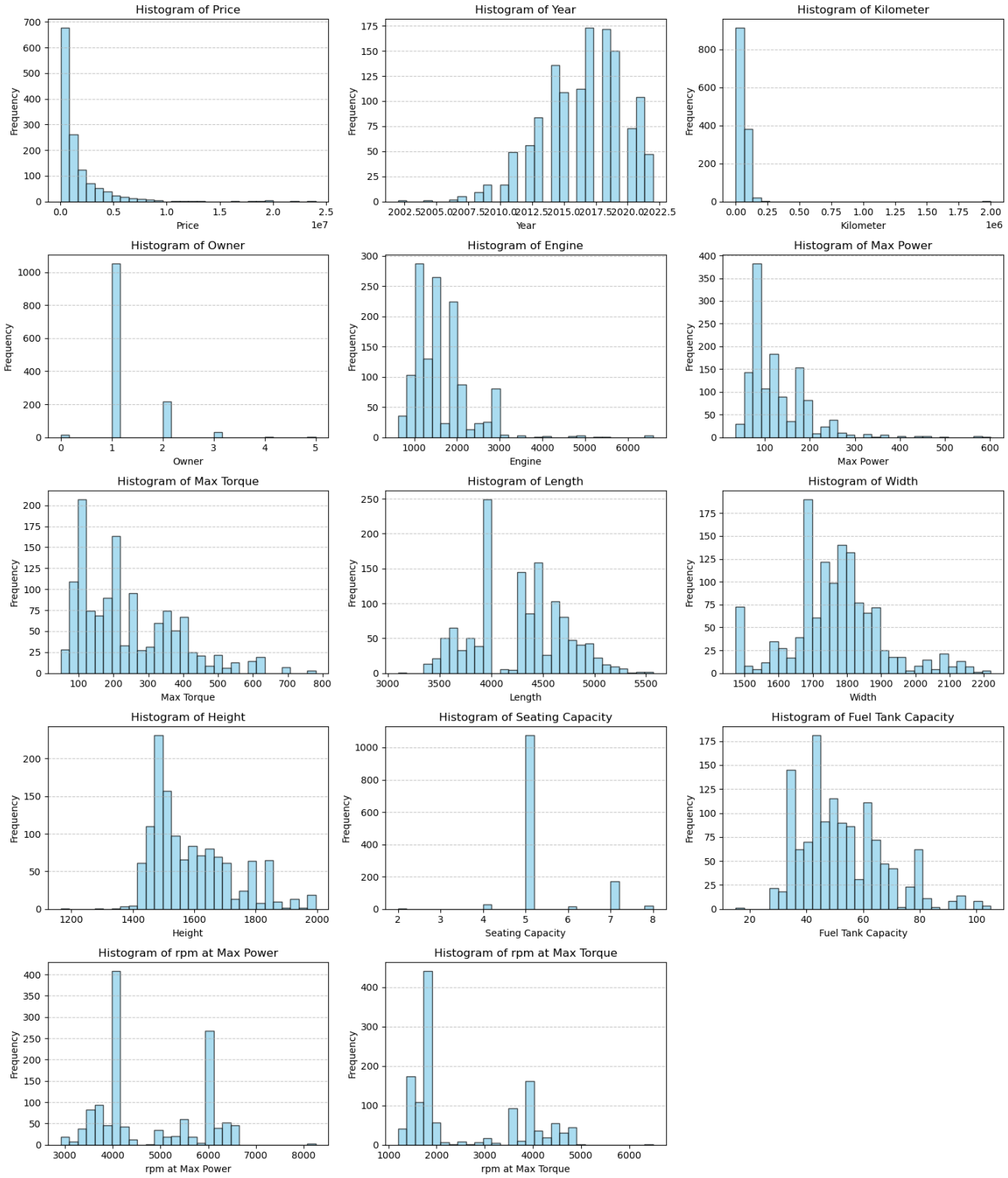
Sau đây là một vài biểu đồ thể hiện chi tiết dữ liệu (trên tập train):

3.3.1 Ma trận tương quan của các đặc trưng số



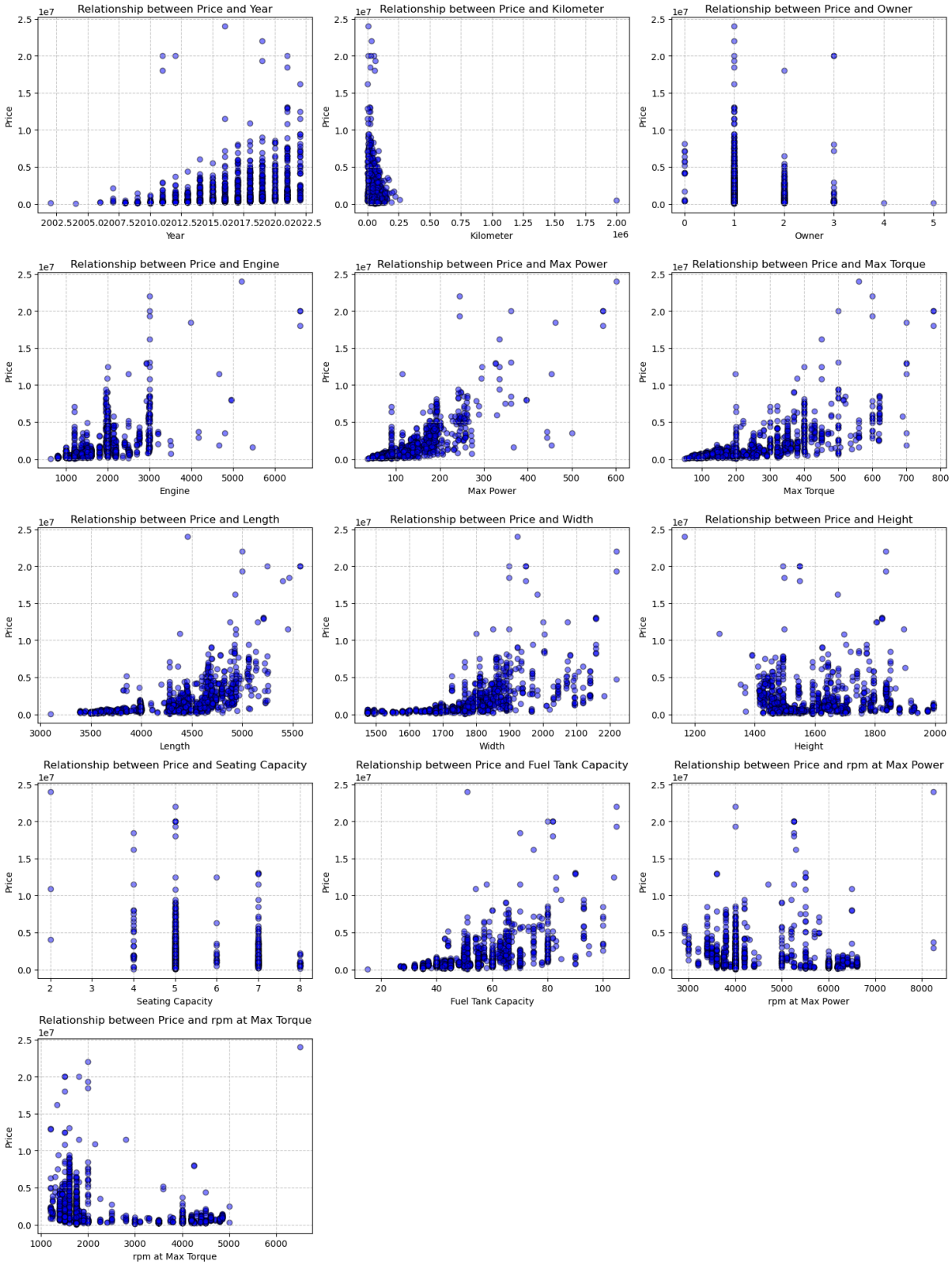
Hình 6: Ma trận tương quan

3.3.2 Biểu đồ Histogram của các đặc trưng số



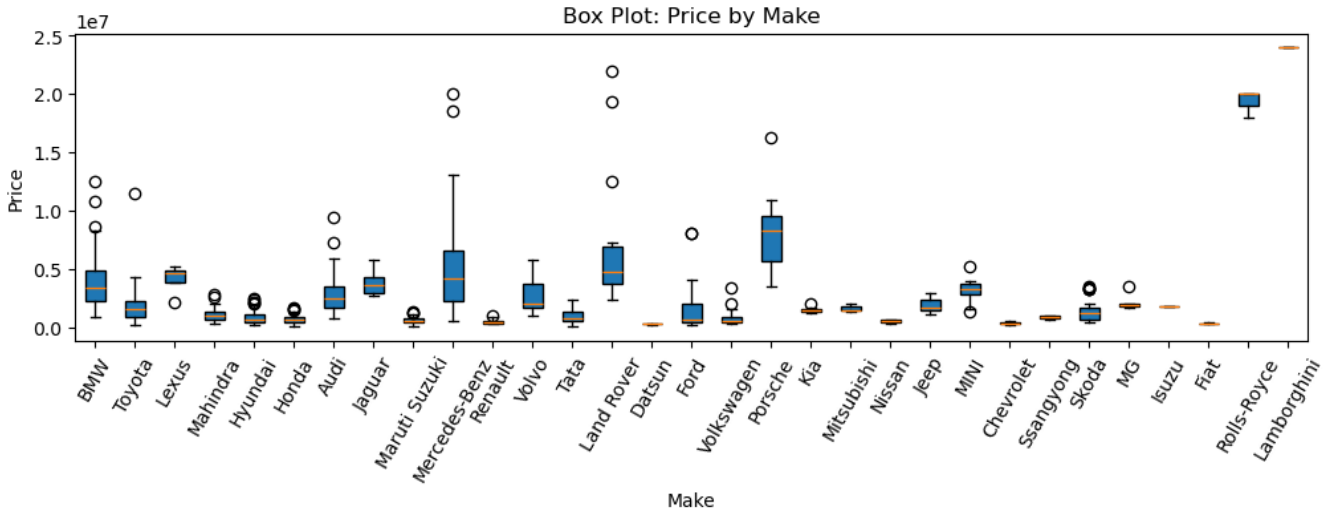
Hình 7: Biểu đồ Histogram của các đặc trưng số

3.3.3 Biểu đồ phân tán của các đặc trưng số

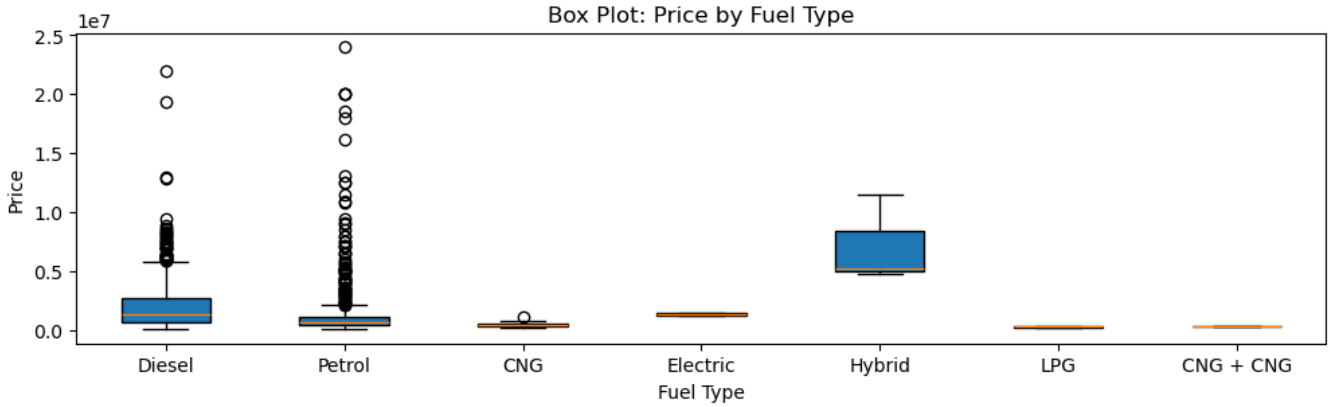


Hình 8: Biểu đồ phân tán so sánh giá xe theo các đặc trưng số

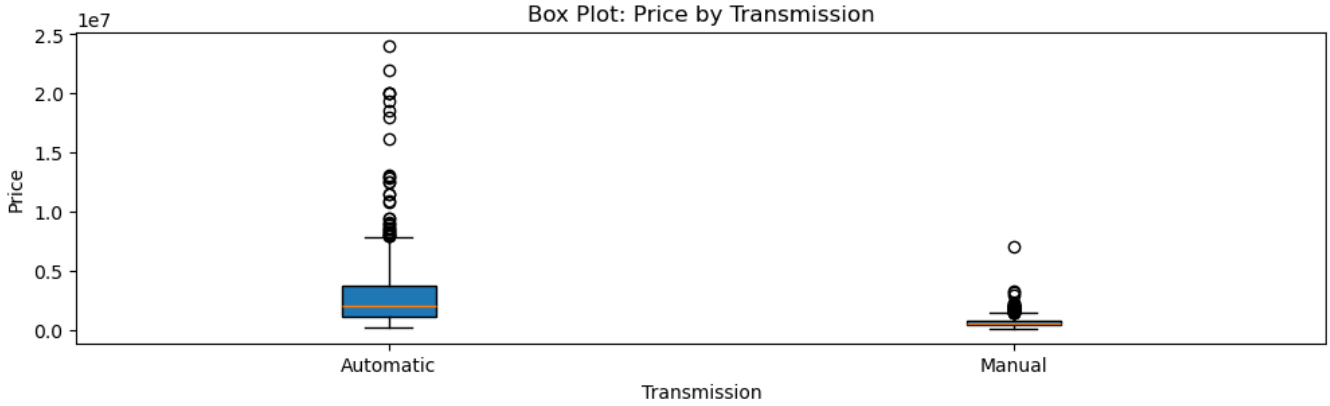
3.3.4 Biểu đồ hộp của các đặc trưng phi số



Hình 9: Biểu đồ hộp so sánh giá xe theo hãng (Make)

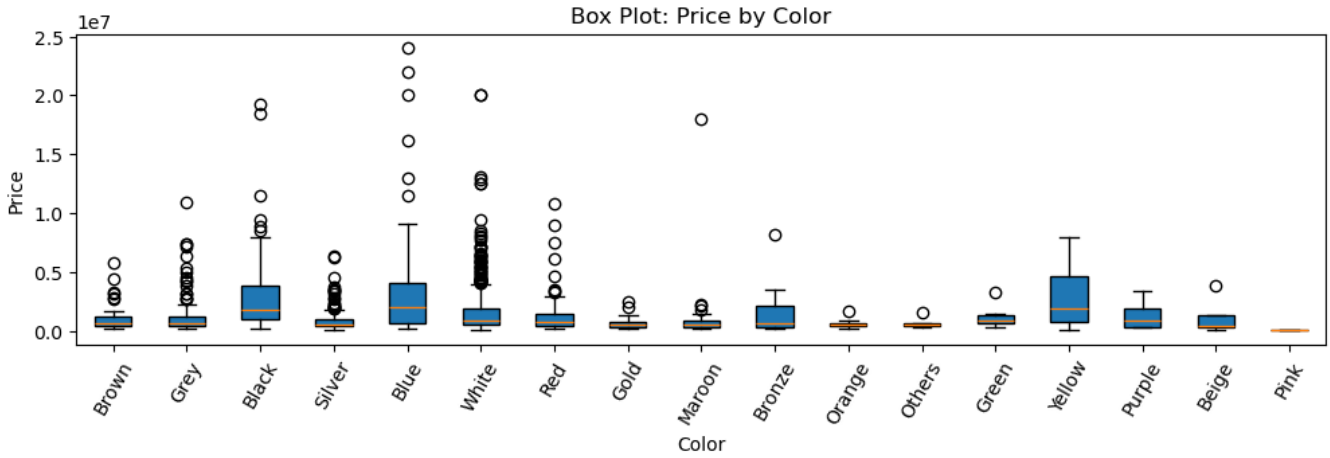


Hình 10: Biểu đồ hộp so sánh giá xe theo loại nhiên liệu sử dụng (Fuel Type)

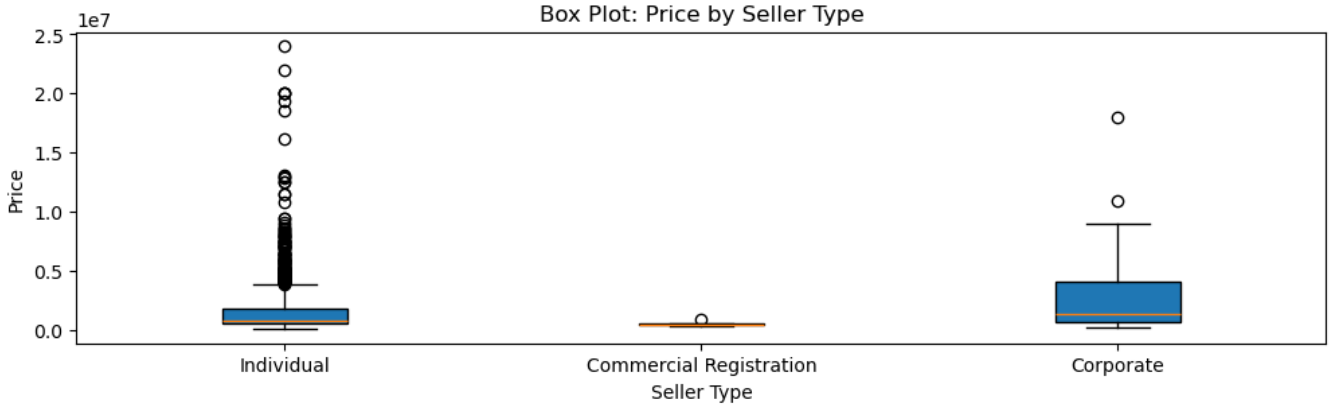


Hình 11: Biểu đồ hộp so sánh giá xe theo loại hộp số (Transmission)

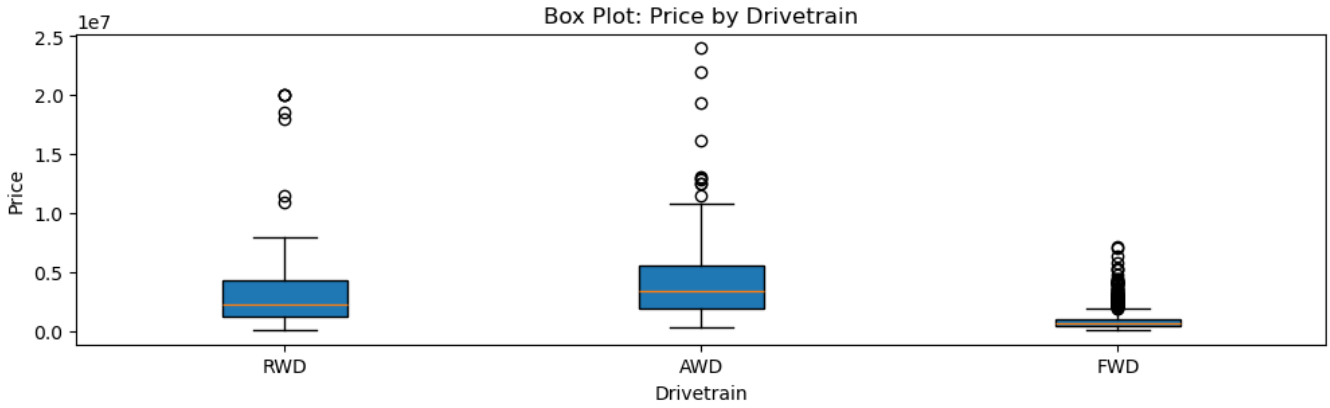
Hình 12: Biểu đồ hộp so sánh giá xe theo địa điểm (Location)



Hình 13: Biểu đồ hộp so sánh giá xe theo màu sắc (Color)



Hình 14: Biểu đồ hộp so sánh giá xe theo loại người bán (Seller type)



Hình 15: Biểu đồ hộp so sánh giá xe theo loại hệ dẫn động (Drivetrain)

3.3.5 Kết luận

Từ dữ liệu được trực quan hóa, ta rút ra được một số nhận xét sau:

- Phân bố giá trị của các đặc trưng số là gần theo **phân phối chuẩn**. Ngoài ra đặc trưng giá xe (Price) và số kilomet đi được (Kilometer) lệch phải nặng (hình 7).
- Các đặc trưng phi số có thể được **mã hóa** thành các **đặc trưng số**.
- **Khoảng giá trị** của mỗi đặc trưng số là **rất khác nhau**, hoặc là rất to, hoặc là rất nhỏ. Dữ liệu cần được chuẩn hóa để giúp mô hình học tốt hơn[1].
- Giá xe có sự khác biệt đáng kể giữa các **hãng** và các **mẫu xe** khác nhau (hình 9), ta có thể thay thế mỗi hãng và mẫu xe bằng **giá trị trung vị** tương ứng của các xe thuộc hãng/mẫu.
- Giá xe Hybrid **cao hơn**[2] so với các loại xe sử dụng nhiên liệu khác (hình 10). Tương tự, số sàn cao hơn số tự động (hình 11); các địa điểm bán cao hơn trung vị nhìn chung cao hơn các địa điểm còn lại (hình 12); màu đen, xanh dương và vàng cao hơn các màu khác (hình 13); công ty cao hơn cá nhân và đại lý (hình 14); hệ dẫn động AWD cao hơn RWD và RWD cao hơn FWD (hình 15). Các nhận định này giúp ta **mã hóa nhị phân** (loại nhiên liệu, hộp số, địa điểm bán, màu sắc, người bán) hoặc **mã hóa theo thứ tự giá trị** (hệ dẫn động) các đặc trưng phi số, giúp mô hình học một cách hiệu quả.

3.4 Mã hóa dữ liệu - Data encoding

Từ những kết luận rút ra từ việc trực quan hóa dữ liệu, ta sẽ mã hóa các đặc trưng phi số về đặc trưng số. Điều này giúp tăng các đặc trưng mà mô hình có thể học.

- **Hãng xe** (Make): Thay bằng giá trung vị của các xe thuộc hãng đó.
- **Mẫu xe** (Model): Thay bằng giá trung vị của các xe cùng mẫu.
- **Loại nhiên liệu** (Fuel Type): Xe Hybrid thay bằng 1, các loại xe còn lại thay bằng 0.

[1] Xem chứng minh ở phần cơ sở toán học
[2] Có giá trị trung vị cao hơn và khoảng IQR đủ nhỏ

- **Hộp số** (Transmission): Xe có hộp số tự động (Auto) thay bằng 1, hộp số sàn (Manual) thay bằng 0.
- **Địa điểm** (Location): Nếu giá trung vị của các xe có cùng địa điểm lớn hơn hoặc bằng giá trung vị toàn cục thì thay bằng 1; ngược lại thay bằng 0.
- **Màu sắc** (Color): Xe có màu trắng, xanh dương hoặc vàng thì thay bằng 1, còn lại là 0.
- **Loại người bán** (Seller Type): Nếu xe được bán bởi các công ty (Corporate) thì thay bằng 1, còn lại là 0.
- **Hệ dẫn động** (Drivetrain): FWD thay bằng 1, RWD thay bằng 2, AWD thay bằng 3.

Sau khi mã hóa các dữ liệu, ta có tổng cộng 23 đặc trưng số và không còn đặc trưng phi số.

3.5 Chuẩn hóa dữ liệu - Data normalization

Từ những kết luận rút ra từ việc trực quan hóa dữ liệu, ta nhận thấy khoảng giá trị của mỗi đặc trưng số là rất khác nhau. Chẳng hạn, sức chứa chỗ ngồi (Seating capacity) nằm trong khoảng $[2, 8]$ còn công suất tối đa (Max power) nằm trong khoảng $[3000, 8000]$ (hình 7). Điều đó dẫn đến một việc tất yếu là phải **chuẩn hóa dữ liệu** về một khoảng giá trị, giúp mô hình học và dự đoán tốt hơn.

3.5.1 Phương pháp chuẩn hóa dữ liệu

Từ hình 7, ta có nhận xét: phân phối của các đặc trưng nhìn chung tuân theo phân phối chuẩn, đặc trưng `Kilometer` có phân phối bị lệch phải. Ta sẽ áp dụng $log(x+1)$ Transformation [12] cho `Kilometer`, sau đó áp dụng MinMax Scaler, Standard Scaler [13] cho tất cả các đặc trưng.

1. **Log(x+1) Transformation** được sử dụng để xử lý dữ liệu bị lệch (skewed), làm cho phân phối gần với phân phối chuẩn hơn. Việc sử dụng $log(x+1)$ thay vì $log(x)$ giúp xử lý các giá trị bằng 0.

   $$x_{scaled} = \log(x + 1)$$

   Trong đó:

   - x_{scaled}: Giá trị đã được chuẩn hóa.
   - x: Giá trị ban đầu.

2. **MinMax Scaler** co giãn các giá trị về một phạm vi cụ thể, thường là từ 0 đến 1.

   $$x_{scaled} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

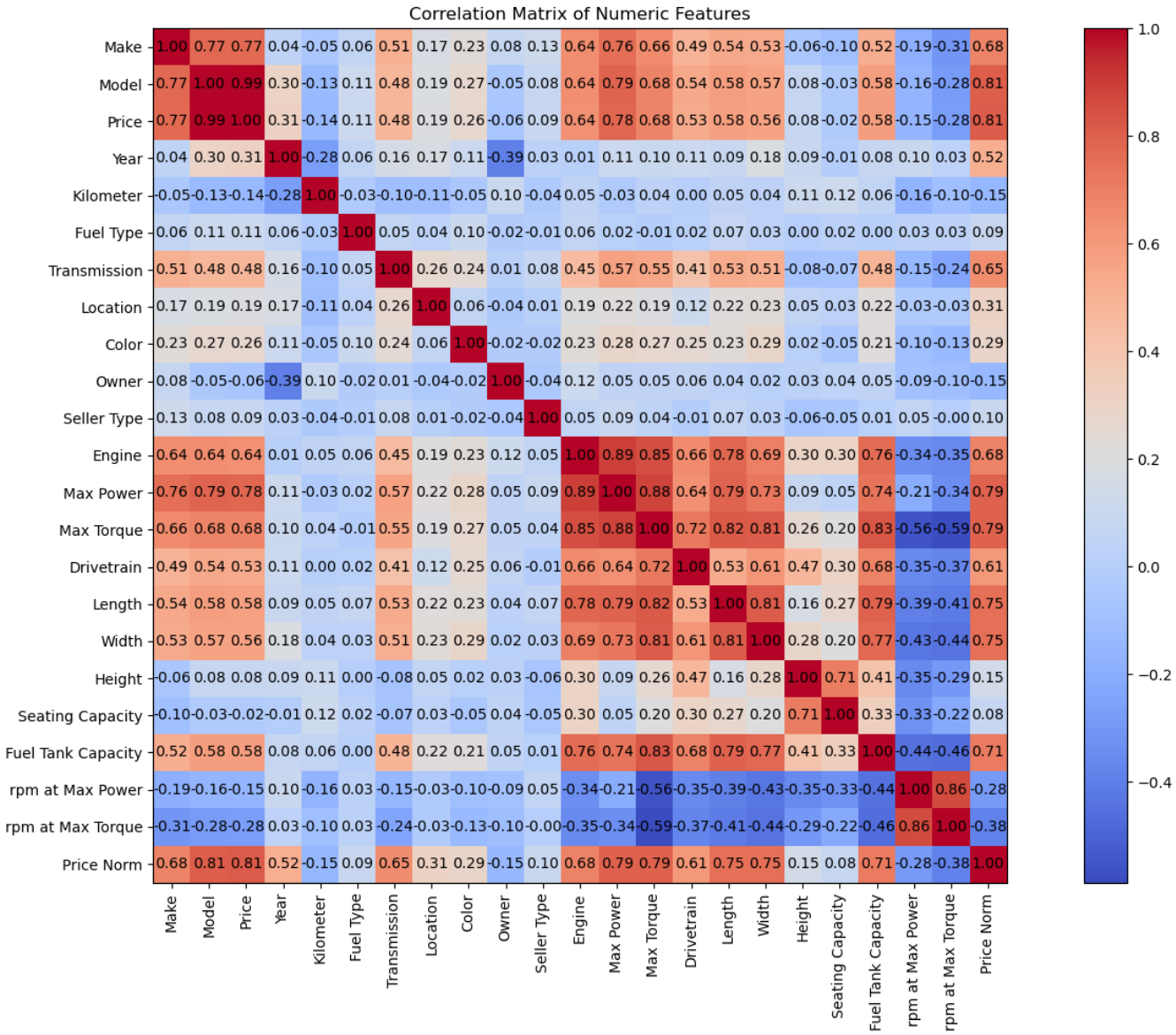
   Trong đó:

   - x_{scaled}: Giá trị đã được chuẩn hóa.

- x: Giá trị ban đầu.
- $\min(x)$: Giá trị nhỏ nhất của đặc trưng trong tập dữ liệu.
- $\max(x)$: Giá trị lớn nhất của đặc trưng trong tập dữ liệu.

3. **Standard Scaler (Chuẩn hóa Z-score)** (xem 2.1.1) được dùng cho phương pháp PCA.

3.5.2 Ma trận tương quan sau mã hóa và chuẩn hóa



Hình 16: Ma trận tương quan sau khi mã hóa và chuẩn hóa dữ liệu

Nhận xét: Sau khi mã hóa và chuẩn hóa dữ liệu, ta thấy đặc trưng `Model` có sự tương quan lớn với `Price`. Đây sẽ là cơ sở để lựa chọn đặc trưng, phục vụ các mô hình hồi quy (đặc biệt là mô hình hồi quy tuyến tính đơn).

4 Các phương pháp đánh giá mô hình

Vì kết quả dự đoán của mô hình hồi quy (Regression) là giá trị liên tục, chúng ta cần các chỉ số đánh giá (evaluation metrics) [14] để định lượng "khoảng cách" hay sai số giữa **giá trị được dự đoán** (predicted) và **giá trị thực tế** (ground truth). Ba chỉ số chính dựa trên việc tính toán sai số này là MSE, RMSE, và MAE. Bên cạnh đó, chỉ số R^2 cũng được sử dụng rộng rãi để đánh giá mức độ phù hợp của mô hình.

4.1 MSE

MSE (Mean Squared Error) hay **trung bình sai số bình phương** là giá trị trung bình của bình phương độ chênh lệch giữa giá trị mục tiêu và giá trị được dự đoán bởi mô hình hồi quy.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Trong đó:

- y_i: Giá trị thực tế.
- $\hat{y}_i$: Giá trị được dự đoán bởi mô hình hồi quy.
- N: Số lượng dữ liệu.

Ý nghĩa:

- MSE có miền giá trị từ $[0, +\infty]$.
- Trên cùng tập dữ liệu, MSE càng nhỏ thì có độ chính xác càng cao.
- Vì lấy bình phương sai số nên đơn vị của MSE khác với đơn vị của kết quả dự đoán.
- MSE nhạy cảm với các giá trị ngoại lệ (outliers) do tính bình phương. Các giá trị lớn hơn sẽ có ảnh hưởng lớn hơn đến MSE.

4.2 RMSE

RMSE (Root Mean Squared Error) hay **căn bậc hai của trung bình sai số bình phương** là giá trị căn bậc hai cho trung bình của bình phương độ chênh lệch giữa giá trị mục tiêu và giá trị được dự đoán bởi mô hình hồi quy.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Trong đó:

- y_i: Giá trị thực tế.
- $\hat{y}_i$: Giá trị được dự đoán bởi mô hình hồi quy.
- N: Số lượng dữ liệu.

Ý nghĩa:

- RMSE có miền giá trị từ $[0, +\infty]$.
- Trên cùng tập dữ liệu, RMSE càng nhỏ thì có độ chính xác càng cao.
- Việc lấy căn bậc 2 của MSE giúp RMSE có cùng đơn vị với kết quả dự đoán, đồng thời làm giá trị RMSE không quá lớn khi số lượng điểm dữ liệu lớn.
- RMSE cũng như MSE, nhạy cảm với các giá trị ngoại lệ (outliers) do tính bình phương.

4.3 MAE

MAE (Mean Absolute Error) hay **trung bình sai số tuyệt đối** là trung bình của giá trị tuyệt đối độ chênh lệch giữa giá trị mục tiêu và giá trị được dự đoán bởi mô hình hồi quy.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Trong đó:

- y_i: Giá trị thực tế.
- $\hat{y}_i$: Giá trị được dự đoán bởi mô hình hồi quy.
- N: Số lượng dữ liệu.

Ý nghĩa:

- MAE có miền giá trị từ $[0, +\infty]$.
- Trên cùng tập dữ liệu, MAE càng nhỏ thì có độ chính xác càng cao.
- MAE không nhạy cảm với giá trị ngoại lệ (outliers) do việc sử dụng giá trị tuyệt đối.
- MAE không phản ánh mức độ sai số cụ thể của mô hình. Nó không phân biệt được các lỗi dương và âm, chỉ cho ta biết lỗi trung bình.

4.4 R-squared

Hệ số xác định (Coefficient of Determination) [2] là tỷ lệ của tổng sự biến thiên trong biến phụ thuộc gây ra bởi sự biến thiên của các biến độc lập (biến giải thích) so với tổng sự biến thiên toàn phần. Hệ số xác định thường được gọi là **R - bình phương** (**R-squared**), ký hiệu là R^2.

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST} = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

Trong đó:

- y_i: Giá trị thực tế.

- $\hat{y}_i$: Giá trị được dự đoán bởi mô hình hồi quy.
- $\bar{y}$: Giá trị trung bình của tất cả các giá trị thực tế.
- N: Số lượng dữ liệu.
- SST: Tổng bình phương toàn phần (Total Sum of Squares).
- SSR: Tổng bình phương hồi quy (Regression Sum of Squares).
- SSE: Tổng bình phương sai số (Error Sum of Squares).

Ý nghĩa:

- R^2 có thường có miền giá trị từ $[0, 1]$, nhưng có thể nhận giá trị âm[3]. Giá trị R^2 càng gần 1 thì mô hình càng phù hợp.
- R^2 của một mô hình hồi quy cho phép ta đánh giá mô hình tìm được có giải thích tốt cho mối liên hệ giá trị dự đoán $\hat{y}$ và giá trị thực tế y hay không.
- Với R^2 cao (gần 1) thì trên tổng thể, các giá trị dự đoán $\hat{y}$ có xu hướng gần với giá trị thực tế y.
- R^2 không cho biết hiệu suất của từng dự đoán riêng lẻ. Một mô hình có R^2 cao vẫn có thể tạo ra một số dự đoán rất tệ cho các điểm dữ liệu cụ thể.

[3]Nếu $SSE > SST$, R^2 sẽ âm. Điều này xảy ra khi mô hình dự đoán tệ hơn cả việc chỉ dự đoán bằng giá trị $\hat{y}$.

5 Mô hình hồi quy tuyến tính đơn biến

5.1 Xây dựng mô hình

5.1.1 Phương pháp xây dựng

Ta sử dụng **phương pháp bình phương tối thiểu** (Ordinary Least Squares - OLS) (xem 2.1.2). Phương pháp này tìm cách tối thiểu hóa tổng bình phương của các sai số giữa giá trị thực tế và giá trị dự đoán. Điều này giúp MSE của mô hình là thấp nhất.

5.1.2 Công thức hồi quy

Thông qua ma trận tương quan (hình 16), ta thấy đặc trưng `Model` có sự tương quan lớn với `Price`. Qua thực nghiệm, `Model` là đặc trưng tốt nhất để xây dựng mô hình hồi quy tuyến tính đơn.

$$Price = 52041.63 + 27427303.62 * Model$$

Trong đó:

- *Price*: Giá xe được dự đoán bởi mô hình
- *Model*: Giá trị của đặc trưng `Model` sau các quá trình xử lí dữ liệu. [4]

5.2 Đánh giá mô hình

Evaluation metrics on Training Set:

	Metric	Value
0	Mean Squared Error	5.322762e+10
1	Root Mean Squared Error	2.307111e+05
2	Mean Absolute Error	7.553069e+04
3	R^2 Score	9.908667e-01

(a) Đánh giá trên tập huấn luyện

Evaluation metrics on Validation Set:

	Metric	Value
0	Mean Squared Error	3.161781e+11
1	Root Mean Squared Error	5.622972e+05
2	Mean Absolute Error	2.984018e+05
3	R^2 Score	9.302062e-01

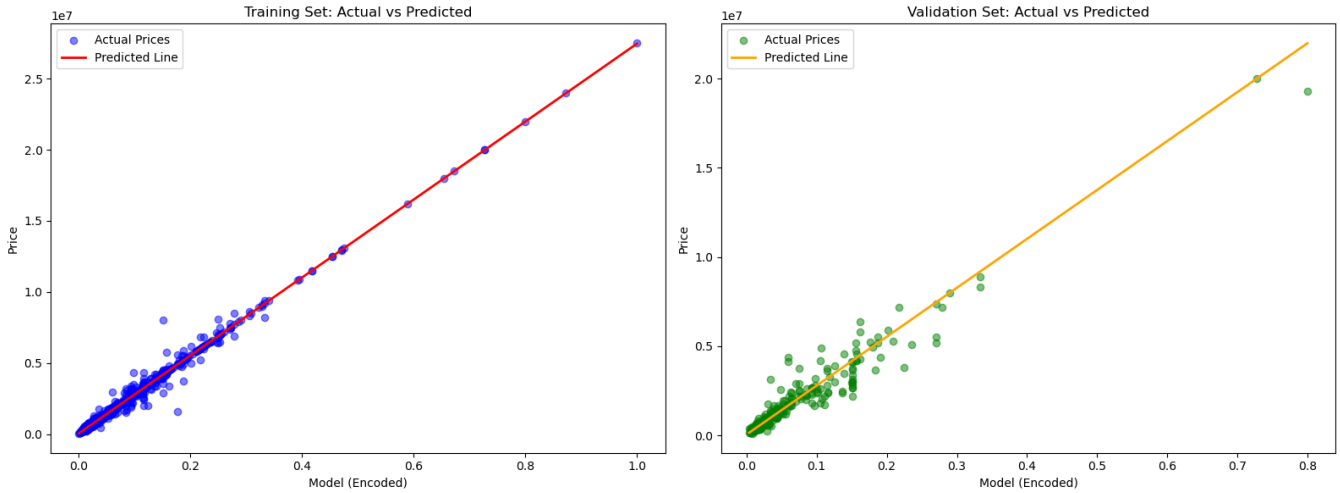
(b) Đánh giá trên tập kiểm chứng

Hình 17: So sánh đánh giá mô hình trên tập huấn luyện và kiểm chứng

Trên tập huấn luyện, mô hình đạt MSE là $5.322762 * 10^{10}$ và R^2 là 0.9908, cho thấy mô hình rất khớp với dữ liệu đã thấy. Tuy nhiên, trên tập kiểm chứng, MSE tăng lên đáng kể thành $3.161781 * 10^{11}$ và R^2 giảm xuống còn 0.9302.

[4] Sau các quá trình tiền xử lý, mã hóa và chuẩn hóa dữ liệu. Trong đó đóng góp tích cực nhất là quá trình mã hóa đặc trưng `Model` bằng các giá trị trung vị của giá xe tương ứng.

Mặc dù R^2 trên tập kiểm chứng vẫn khá cao, sự gia tăng đáng kể của các chỉ số lỗi (MSE tăng gấp 6 lần) cho thấy dấu hiệu của hiện tượng overfitting. Mô hình đã học thuộc một số đặc điểm nhiễu của tập huấn luyện và khả năng tổng quát hóa trên dữ liệu mới bị hạn chế hơn. Điều này cũng có thể quan sát thấy trong hình 18, nơi các điểm dữ liệu kiểm chứng phân tán rộng hơn quanh đường dự đoán so với tập huấn luyện. Tuy nhiên, điều này có thể chấp nhận được vì đây chỉ là mô hình hồi quy tuyến tính đơn.



Hình 18: Đường thẳng hồi quy dự đoán so với thực tế, trên tập huấn luyện và kiểm chứng

6 Mô hình hồi quy tuyến tính đa biến

6.1 Giới thiệu

6.1.1 Khái niệm hồi quy tuyến tính đa biến

Hồi quy tuyến tính đa biến [15] (Multiple Linear Regression - MLR) là một phương pháp thống kê mở rộng từ hồi quy tuyến tính đơn giản, được sử dụng để mô hình hóa mối quan hệ giữa một biến phụ thuộc và nhiều biến độc lập. Mô hình có dạng tổng quát như sau:

$$y = b + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p$$

Trong đó:

- y là biến phụ thuộc (biến kết quả)
- $x_1, x_2, \ldots, x_p$ là các biến độc lập (biến dự đoán)
- b là hệ số chặn (intercept)
- $\theta_1, \theta_2, \ldots, \theta_p$ là các hệ số hồi quy

Mục tiêu của thuật toán là tìm ra siêu phẳng phù hợp nhất có thể dự đoán các giá trị dựa trên các biến độc lập. Mô hình hồi quy học một hàm từ tập dữ liệu (với các giá trị X và Y đã biết) và sử dụng nó để dự đoán giá trị Y từ các giá trị X mới.

6.1.2 So sánh với hồi quy tuyến tính đơn giản

Hồi quy tuyến tính đa biến khác biệt với hồi quy tuyến tính đơn giản ở những điểm chính sau:

Đặc điểm	**Hồi quy đơn giản**	**Hồi quy đa biến**
Số biến độc lập	Một biến	Nhiều biến
Mô hình toán học	$Y = b + \theta_1 X$	$Y = b + \theta_1 x_1 + \cdots + \theta_p x_p$
Diễn giải hình học	Đường thẳng trong không gian 2 chiều	Siêu phẳng trong không gian $p+1$ chiều

Bảng 1: So sánh giữa hồi quy tuyến tính đơn giản và đa biến

Hồi quy tuyến tính đa biến vượt trội hơn so với hồi quy tuyến tính đơn giản trong việc:

- Cung cấp khả năng phân tích đa chiều do có thể tận dụng toàn bộ đặc điểm từ dữ liệu
- Tăng khả năng dự báo chính xác khi có nhiều yếu tố ảnh hưởng

6.2 Xây dựng mô hình

6.2.1 Ma trận hóa

Mô hình hồi quy tuyến tính đa biến đã được giới thiệu với dạng tổng quát như sau:

$$y = b + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p \tag{8}$$

Trong quá trình phân tích với bộ dữ liệu lớn, xử lý tuần tự từng mẫu trở nên kém hiệu quả. Cụ thể, với tập dữ liệu bao gồm 1647 mẫu quan sát, việc áp dụng phương pháp ma trận hóa không chỉ tối ưu hóa khả năng tính toán song song mà còn cung cấp một cách biểu diễn toán học chặt chẽ và súc tích.

Đối với một quan sát đơn lẻ với p đặc trưng (features), ta có thể biểu diễn giá trị dự đoán thông qua nhân ma trận như sau:

$$y = \begin{bmatrix} x_1 & x_2 & \dots & x_p \end{bmatrix}_{1 \times p} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_p \end{bmatrix}_{p \times 1} + b$$

Khi mở rộng phương pháp này cho toàn bộ tập dữ liệu với n quan sát, ta thu được biểu diễn ma trận tổng quát của mô hình hồi quy:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n \times 1} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}_{n \times p} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_p \end{bmatrix}_{p \times 1} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}_{n \times 1}$$

Phương trình này có thể được biểu diễn một cách súc tích hơn trong ký hiệu ma trận:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\theta} + \mathbf{b} \tag{9}$$

Trong đó:

- $\mathbf{Y} \in \mathbb{R}^{n \times 1}$ là vector cột chứa n giá trị phụ thuộc
- $\mathbf{X} \in \mathbb{R}^{n \times p}$ là ma trận chứa n mẫu quan sát với p đặc trưng cho mỗi mẫu
- $\boldsymbol{\theta} \in \mathbb{R}^{p \times 1}$ là vector tham số cần ước lượng
- $\mathbf{b} \in \mathbb{R}^{n \times 1}$ là vector hệ số chặn (intercept) với mỗi phần tử đều bằng nhau và bằng b

6.2.2 Hàm mất mát

Trong quá trình xây dựng mô hình hồi quy tuyến tính đa biến, việc lựa chọn hàm mất mát (loss function) đóng vai trò quyết định đến hiệu suất và khả năng hội tụ của mô hình. Sau khi cân nhắc các phương pháp khác nhau, Mean Squared Error (MSE) 4.1 được quyết định làm hàm mất mát chính:

Biểu diễn dưới dạng ma trận, hàm mất mát MSE có thể được viết như sau:

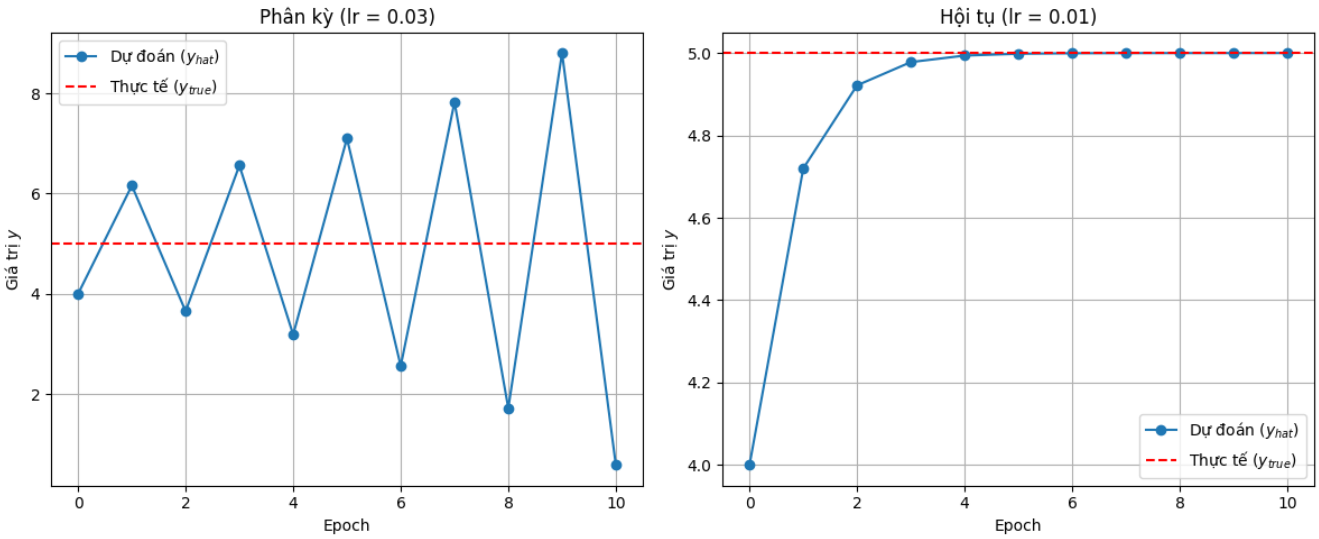
$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{b}) = \frac{1}{n} \|\mathbf{Y} - \mathbf{X}\boldsymbol{\theta} - \mathbf{b}\|_2^2$$

6.2.3 Gradient Descent

Mô hình này sẽ áp dụng Gradient Descent, là một thuật toán tối ưu hóa cơ bản nhất và quan trọng nhất trong lĩnh vực học máy, đã được đề cập ở 2.2.

Vai trò quan trọng của tốc độ học α Việc lựa chọn tốc độ học α cho thuật toán Gradient Descent có ảnh hưởng quyết định đến hiệu quả của quá trình huấn luyện:

- Nếu α có giá trị phù hợp: Mô hình sẽ hội tụ đến điểm cực tiểu của hàm mất mát sau một số lần cập nhật.
- Nếu α quá lớn: Mô hình sẽ liên tục **overshoot** (vượt lố) điểm cực tiểu, dao động qua lại giữa các điểm có giá trị mất mát lớn hơn. Sai số tích lũy sau mỗi lần cập nhật có thể dẫn đến hiện tượng phân kỳ, trong đó hàm mất mát có xu hướng tăng thay vì giảm.
- Nếu α quá nhỏ: Quá trình hội tụ sẽ diễn ra rất chậm, đòi hỏi số lượng vòng lặp lớn và tăng thời gian huấn luyện.



Hình 19: Hiện tượng phân kỳ và hội tụ tùy thuộc vào cách chọn tốc độ học

6.3 Quá trình huấn luyện

Biểu diễn ma trận của mô hình Trong mô hình hồi quy tuyến tính đa biến với n mẫu và p biến độc lập, ta có thể biểu diễn mô hình dưới dạng ma trận mở rộng để gộp cả tham số bias b vào:

- $\tilde{X} \in \mathbb{R}^{n\times(p+1)}$: Ma trận dữ liệu mở rộng, với cột đầu tiên là toàn số 1:

$$\tilde{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

- $\tilde{\theta} \in \mathbb{R}^{(p+1)\times 1}$: Vector tham số mở rộng, bao gồm cả bias như phần tử đầu tiên:

$$\tilde{\theta} = \begin{bmatrix} b \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_p \end{bmatrix}$$

Với cách biểu diễn mở rộng này, mô hình dự đoán có dạng:

$$\hat{Y} = \tilde{X}\tilde{\theta}$$

Trong đó mỗi phần tử $\hat{y}_i$ của vector $\hat{Y}$ được tính như sau:

$$\hat{y}_i = b + \sum_{j=1}^{p} \theta_j x_{ij}$$

Đạo hàm ma trận của hàm mất mát Hàm mất mát Mean Squared Error (MSE) với cách biểu diễn mở rộng có dạng:

$$\mathcal{L}(\tilde{\theta}) = \frac{1}{n}(Y - \tilde{X}\tilde{\theta})^T(Y - \tilde{X}\tilde{\theta})$$

Gradient của hàm mất mát theo vector tham số mở rộng $\tilde{\theta}$ là:

$$\frac{\partial \mathcal{L}}{\partial \tilde{\theta}} = -\frac{2}{n}\tilde{X}^T(Y - \hat{Y})$$

Khi phân tích thành các thành phần:

$$\frac{\partial \mathcal{L}}{\partial \tilde{\theta}} = -\frac{2}{n}\begin{bmatrix} \sum_{i=1}^{n}(y_i - \hat{y}_i) \\ \sum_{i=1}^{n}(y_i - \hat{y}_i)x_{i1} \\ \sum_{i=1}^{n}(y_i - \hat{y}_i)x_{i2} \\ \vdots \\ \sum_{i=1}^{n}(y_i - \hat{y}_i)x_{ip} \end{bmatrix}$$

Cập nhật tham số thông qua ma trận Sử dụng thuật toán Gradient Descent, ta cập nhật vector tham số mở rộng $\tilde{\theta}$ như sau:

$$\tilde{\theta}^{(t+1)} = \tilde{\theta}^{(t)} + \frac{2\alpha}{n}\tilde{X}^T(Y - \hat{Y}^{(t)})$$

Cụ thể:

$$\begin{bmatrix} b^{(t+1)} \\ \theta_1^{(t+1)} \\ \theta_2^{(t+1)} \\ \vdots \\ \theta_p^{(t+1)} \end{bmatrix} = \begin{bmatrix} b^{(t)} \\ \theta_1^{(t)} \\ \theta_2^{(t)} \\ \vdots \\ \theta_p^{(t)} \end{bmatrix} + \frac{2\alpha}{n} \begin{bmatrix} \sum_{i=1}^{n}(y_i - \hat{y}_i^{(t)}) \\ \sum_{i=1}^{n}(y_i - \hat{y}_i^{(t)})x_{i1} \\ \sum_{i=1}^{n}(y_i - \hat{y}_i^{(t)})x_{i2} \\ \vdots \\ \sum_{i=1}^{n}(y_i - \hat{y}_i^{(t)})x_{ip} \end{bmatrix}$$

Với α là tốc độ học và t là chỉ số của bước lặp trong quá trình huấn luyện.

6.4 Thử nghiệm mô hình

Để đảm bảo hiệu năng tối ưu, ta tiến hành thử nghiệm mô hình thông qua nhiều phương pháp huấn luyện khác nhau nhằm xác định cấu hình nào mang lại kết quả tốt nhất.

6.4.1 Kiến trúc mô hình

Mô hình cơ sở (base model) Mô hình cơ sở chính là mô hình tuyến tính đa biến chuẩn mà ta đã thảo luận ở (9).

Hình 20: Giá trị hàm mất mát qua từng epoch (base model)

	Metric	Value
0	Mean Squared Error	1.807090e+11
1	Root Mean Squared Error	4.250989e+05
2	Mean Absolute Error	2.425312e+05
3	R^2 Score	9.689923e-01

(a) Đánh giá trên tập huấn luyện

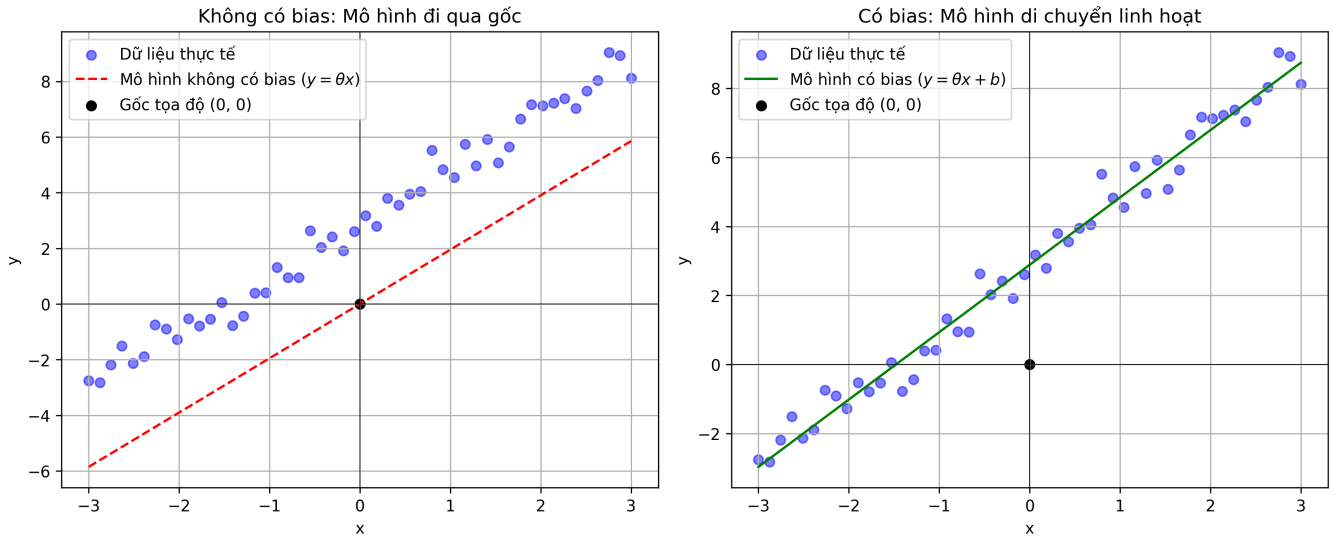
	Metric	Value
0	Mean Squared Error	2.212506e+11
1	Root Mean Squared Error	4.703728e+05
2	Mean Absolute Error	2.992231e+05
3	R^2 Score	9.511607e-01

(b) Đánh giá trên tập kiểm chứng

Hình 21: So sánh đánh giá mô hình trên tập huấn luyện và kiểm chứng (base model)

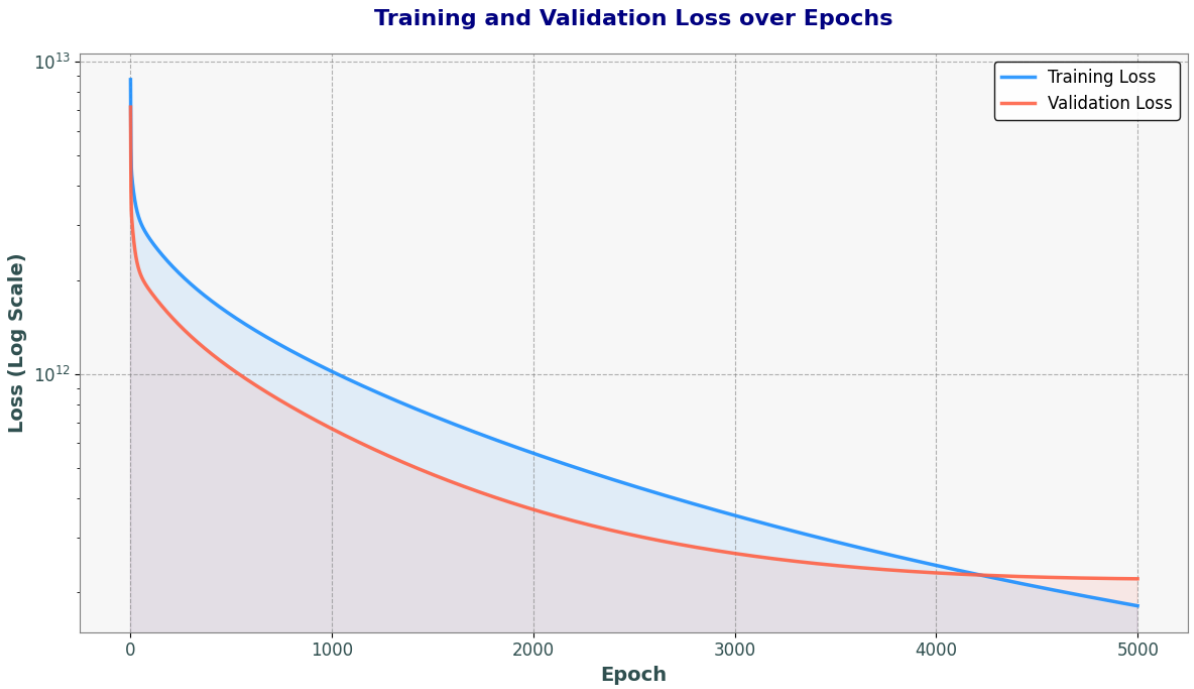
Nhận xét: Mô hình cơ sở chưa thông qua các phương pháp gì đặc biệt nhưng vẫn đạt kết quả khá tốt.

Mô hình không thành phần bias Theo lý thuyết, việc loại bỏ thành phần bias thường không được khuyến khích vì nó làm giảm đáng kể khả năng biểu diễn và độ phức tạp của mô hình. Để minh họa rõ hơn vấn đề này, ta xét một trường hợp đơn giản với mô hình hồi quy một biến:



Hình 22: So sánh hiệu năng giữa mô hình có và không có thành phần bias

Ở mô hình không có bias, mặc dù mô hình học được xu hướng của dữ liệu nhưng do bị giới hạn ở gốc tọa độ, nó không thể nắm bắt được chính xác các mối quan hệ của chúng.



Hình 23: Giá trị hàm mất mát qua từng epoch (no bias)

	Metric	Value
0	Mean Squared Error	1.807259e+11
1	Root Mean Squared Error	4.251187e+05
2	Mean Absolute Error	2.416191e+05
3	R^2 Score	9.689894e-01

(a) Đánh giá trên tập huấn luyện

	Metric	Value
0	Mean Squared Error	2.207125e+11
1	Root Mean Squared Error	4.698005e+05
2	Mean Absolute Error	2.980952e+05
3	R^2 Score	9.512795e-01

(b) Đánh giá trên tập kiểm chứng

Hình 24: So sánh đánh giá mô hình trên tập huấn luyện và kiểm chứng (no bias)

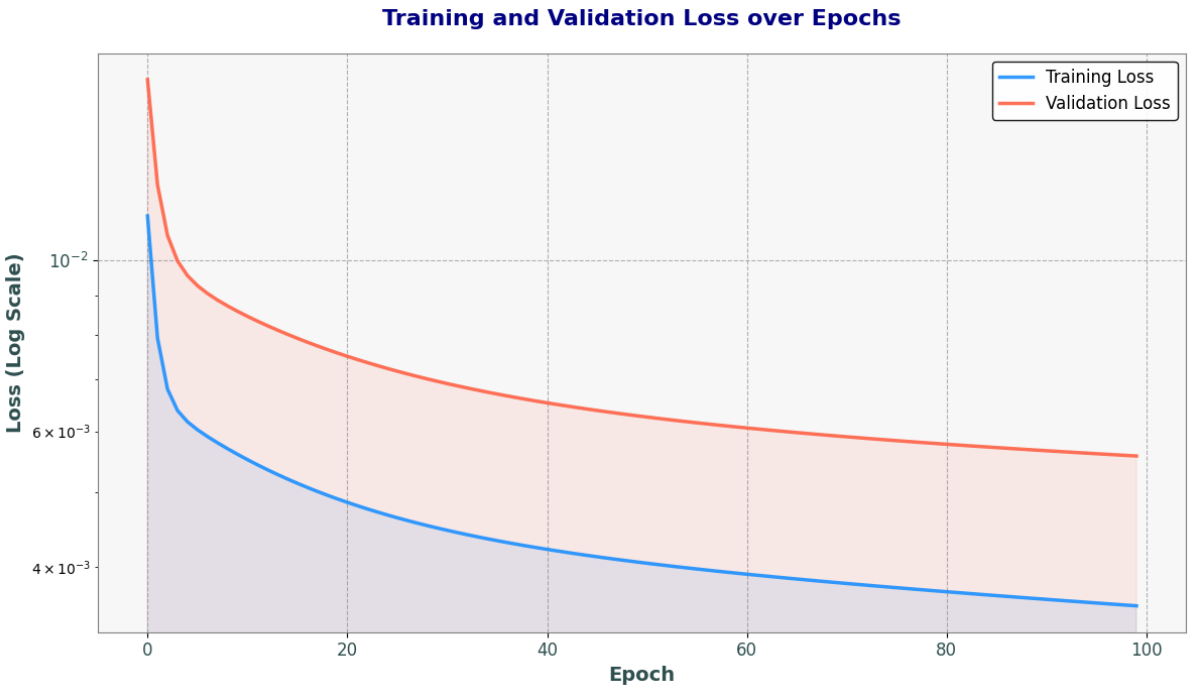
6.4.2 Chuyển hóa biến mục tiêu

Trong các thử nghiệm trước, ta đã sử dụng các đặc trưng đã chuẩn hóa để dự đoán giá nhà ở giá trị gốc. Phần này trình bày các phương pháp chuyển hóa biến mục tiêu nhằm đưa giá trị về phạm vi nhỏ hơn, tạo điều kiện thuận lợi cho quá trình huấn luyện và cải thiện hiệu suất của mô hình.

Phương pháp Min-Max Scaling Phép biến đổi Min-Max Scaling được áp dụng theo công thức sau:

$$y_{\text{transformed}} = \frac{y_{\text{original}} - y_{\min}}{y_{\max} - y_{\min}}$$

Phương pháp này chuẩn hóa dữ liệu về khoảng [0, 1], giữ nguyên hình dạng phân phối gốc nhưng thu hẹp phạm vi giá trị.



Hình 25: Giá trị hàm mất mát qua từng epoch (Min-Max Scaling)

	Metric	Value
0	Mean Squared Error	2.677557e+12
1	Root Mean Squared Error	1.636324e+06
2	Mean Absolute Error	8.271154e+05
3	R^2 Score	5.405598e-01

(a) Đánh giá trên tập huấn luyện

	Metric	Value
0	Mean Squared Error	2.198734e+12
1	Root Mean Squared Error	1.482813e+06
2	Mean Absolute Error	6.473916e+05
3	R^2 Score	5.146469e-01

(b) Đánh giá trên tập kiểm chứng

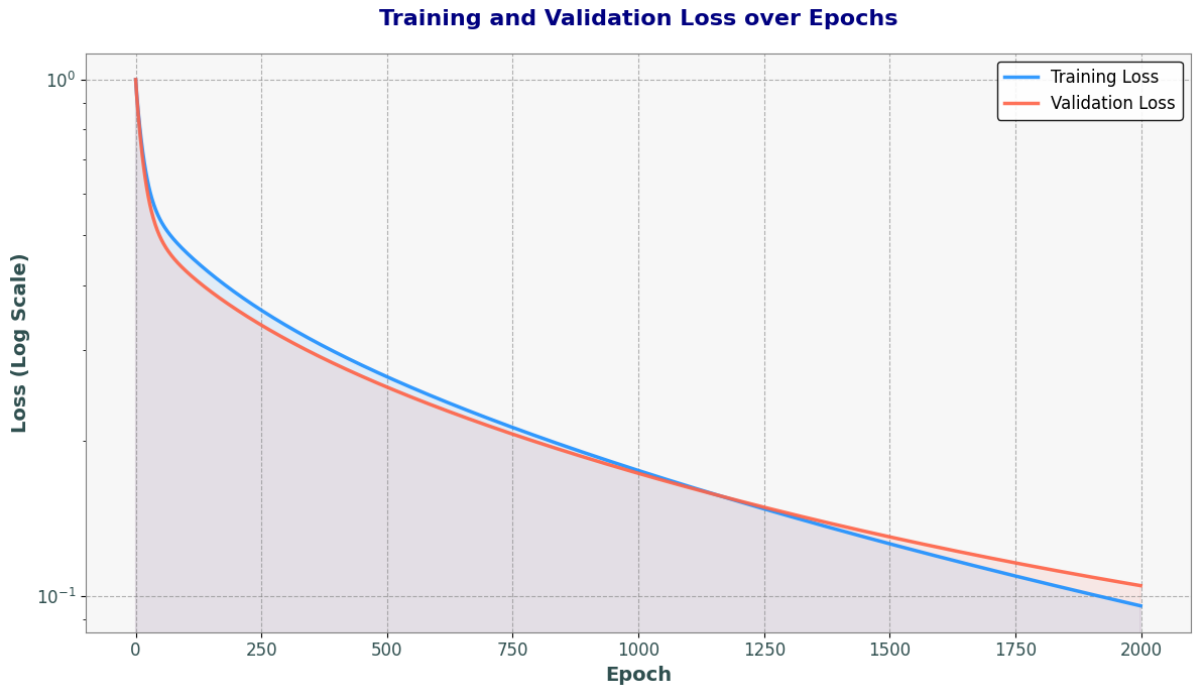
Hình 26: So sánh đánh giá mô hình trên tập huấn luyện và kiểm chứng (Min-Max Scaling)

Nhận xét: Sau khi chuyển hóa, tốc độ hội tụ của mô hình tăng nhanh đáng kể (chỉ sau khoảng 10 epoch). Hàm mất mát giảm mạnh ở một vài epoch đầu và sườn dốc bắt đầu thoải hơn (giảm chậm) ở các epoch sau chứng tỏ mô hình đã hội tụ.

Phương pháp chuẩn hóa (Standardization) [16], [17]: Phương pháp chuẩn hóa áp dụng công thức:

$$y_{\text{transformed}} = \frac{y_{\text{original}} - \mu_y}{\sigma_y}$$

Trong đó μ_y là giá trị trung bình và σ_y là độ lệch chuẩn của biến mục tiêu. Phương pháp này chuyển đổi dữ liệu về phân phối có trung bình bằng 0 và phương sai bằng 1.



Hình 27: Giá trị hàm mất mát qua từng epoch (Standardization)

	Metric	Value
0	Mean Squared Error	5.571901e+11
1	Root Mean Squared Error	7.464517e+05
2	Mean Absolute Error	4.132061e+05
3	R^2 Score	9.043921e-01

(a) Đánh giá trên tập huấn luyện

	Metric	Value
0	Mean Squared Error	4.740211e+11
1	Root Mean Squared Error	6.884919e+05
2	Mean Absolute Error	3.682402e+05
3	R^2 Score	8.953636e-01

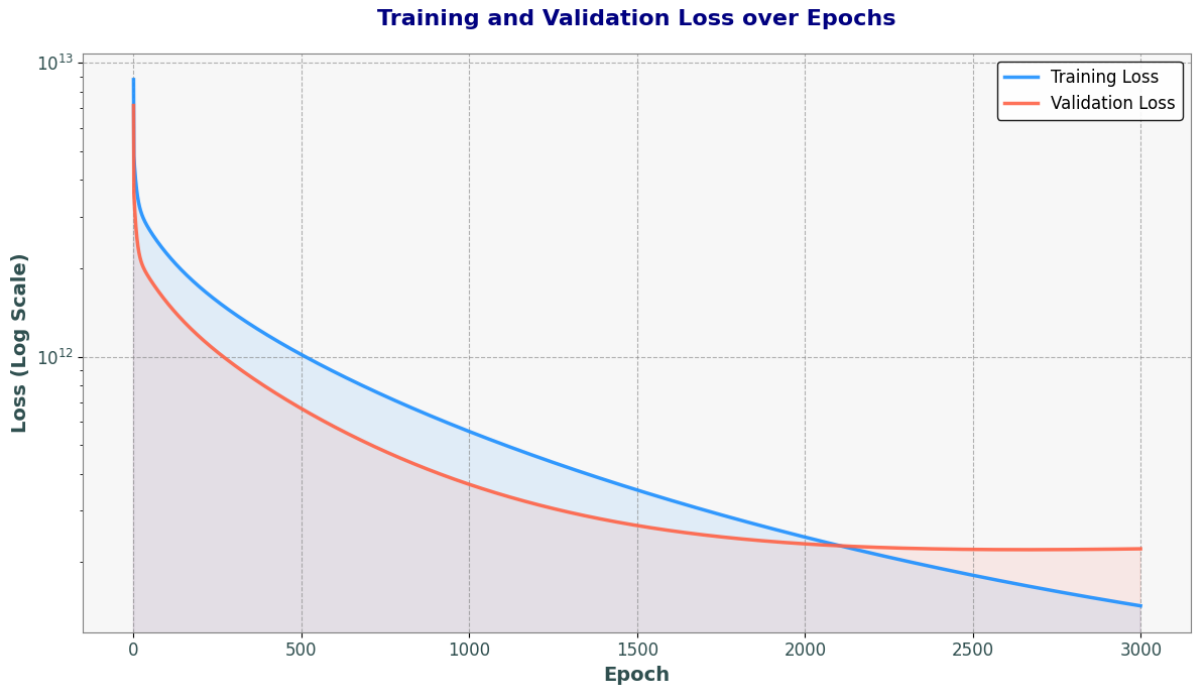
(b) Đánh giá trên tập kiểm chứng

Hình 28: So sánh đánh giá mô hình trên tập huấn luyện và kiểm chứng (Standardization)

6.4.3 Regularization

Regularization là phương pháp kiểm soát độ phức tạp của mô hình thông qua việc thêm các ràng buộc vào hàm mục tiêu trong quá trình tối ưu hóa. Phương pháp này đóng vai trò quan trọng trong việc giảm thiểu hiện tượng *quá khớp* (overfitting) trên tập huấn luyện, đồng thời tăng cường khả năng *tổng quát hóa* (generalization) của mô hình. Kết quả là mô hình có thể duy trì độ chính xác cao khi dự đoán trên dữ liệu mới chưa từng được tiếp xúc trong quá trình huấn luyện.

Hồi quy Lasso (Lasso Regression) Phương pháp hồi quy Lasso áp dụng kỹ thuật điều chuẩn L1 thông qua việc bổ sung thành phần phạt vào hàm mất mát, đã được giới thiệu ở 2.3.4.



Hình 29: Giá trị hàm mất mát qua từng epoch (Lasso)

	Metric	Value
0	Mean Squared Error	1.421503e+11
1	Root Mean Squared Error	3.770283e+05
2	Mean Absolute Error	2.147435e+05
3	R^2 Score	9.756085e-01

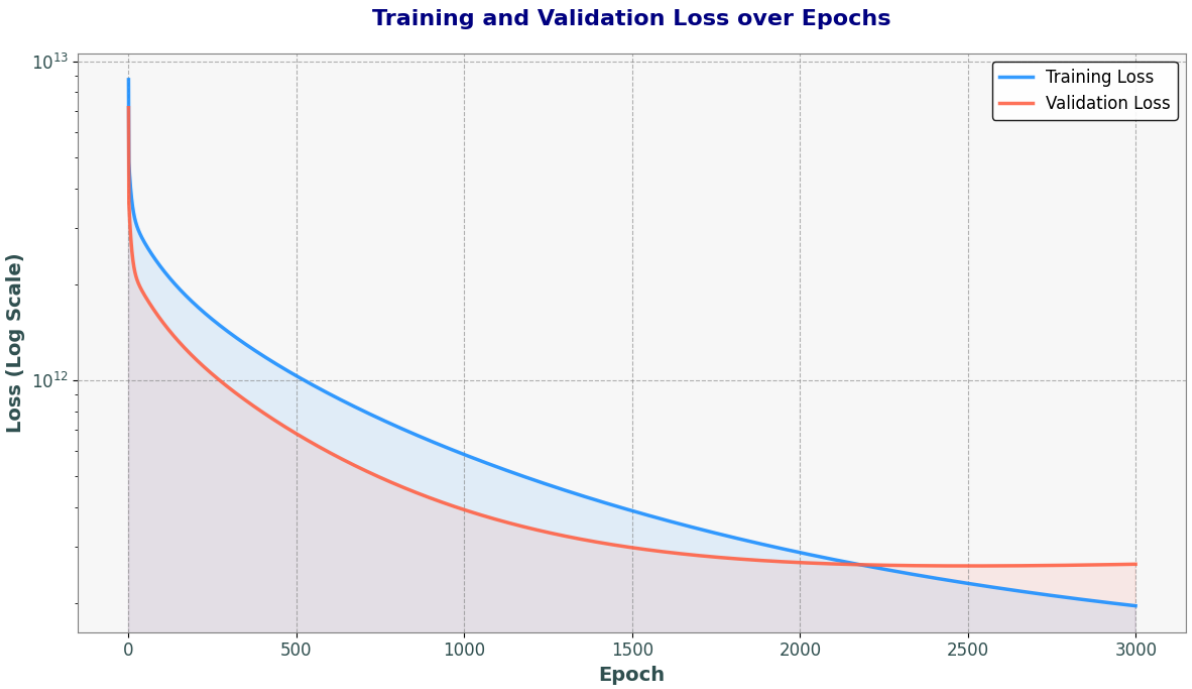
(a) Đánh giá trên tập huấn luyện

	Metric	Value
0	Mean Squared Error	2.223322e+11
1	Root Mean Squared Error	4.715212e+05
2	Mean Absolute Error	2.901655e+05
3	R^2 Score	9.509219e-01

(b) Đánh giá trên tập kiểm chứng

Hình 30: So sánh đánh giá mô hình trên tập huấn luyện và kiểm chứng (Lasso)

Hồi quy Ridge (Ridge Regression) Phương pháp hồi quy Ridge sử dụng kỹ thuật điều chuẩn L2, đã được giới thiệu ở 2.3.3



Hình 31: Giá trị hàm mất mát qua từng epoch (Ridge)

	Metric	Value
0	Mean Squared Error	1.519305e+11
1	Root Mean Squared Error	3.897827e+05
2	Mean Absolute Error	2.203223e+05
3	R^2 Score	9.739303e-01

(a) Đánh giá trên tập huấn luyện

	Metric	Value
0	Mean Squared Error	2.208482e+11
1	Root Mean Squared Error	4.699449e+05
2	Mean Absolute Error	2.903846e+05
3	R^2 Score	9.512495e-01

(b) Đánh giá trên tập kiểm chứng

Hình 32: So sánh đánh giá mô hình trên tập huấn luyện và kiểm chứng (Ridge)

6.5 Nhận xét và đánh giá

6.5.1 Đánh giá tổng quan hiệu năng mô hình

Bảng dưới đây trình bày kết quả đánh giá hiệu năng của các mô hình và các phương pháp khác nhau:

	Base	No bias	Min-Max scaler	Standardization	Lasso	Ridge
Mean Squared Error	2.212506e+11	2.207125e+11	2.198734e+12	4.740211e+11	2.223322e+11	2.208482e+11
Root Mean Squared Error	4.703728e+05	4.698005e+05	1.482813e+06	6.884919e+05	4.715212e+05	4.699449e+05
Mean Absolute Error	2.992231e+05	2.980952e+05	6.473916e+05	3.682402e+05	2.901655e+05	2.903846e+05
R^2 Score	9.511607e-01	9.512795e-01	5.146469e-01	8.953636e-01	9.509219e-01	9.512495e-01

Hình 33: Tổng kết mức hiệu quả của từng phương pháp

Phân tích kết quả cho thấy một số điểm đáng chú ý sau:

1. **Về giá trị mất mát:** Các chỉ số MSE (Mean Squared Error) ghi nhận giá trị tương đối cao, lên đến hàng trăm tỷ. Tuy nhiên, điều này không đáng lo ngại vì biến mục tiêu Price vốn có phạm vi giá trị rất lớn. Đặc biệt là với hàm MSE, độ chênh lệch lên cả hàng trăm tỷ vì hàm có bậc bình phương, dẫn đến sẽ đẩy cao giá trị mất mát lên nhiều lần.

2. **Hiệu suất mô hình cơ sở:** Mô hình cơ sở (Base) và mô hình không thành phần bias (No bias) đều đạt hiệu suất tương đương nhau với chỉ số R^2 xấp xỉ 0.951. Điều này cho thấy rằng trong bài toán cụ thể này, thành phần bias không đóng vai trò quan trọng đến hiệu năng của mô hình.

3. **Tác động của các phương pháp chuyển hóa biến mục tiêu:** Min-Max scaling làm giảm hiệu suất của mô hình đáng kể, với chỉ số $R^2 \approx 0.514$, trong khi đó Standardization đem lại kết quả tương đối tích cực, với R^2 đạt gần 0.9

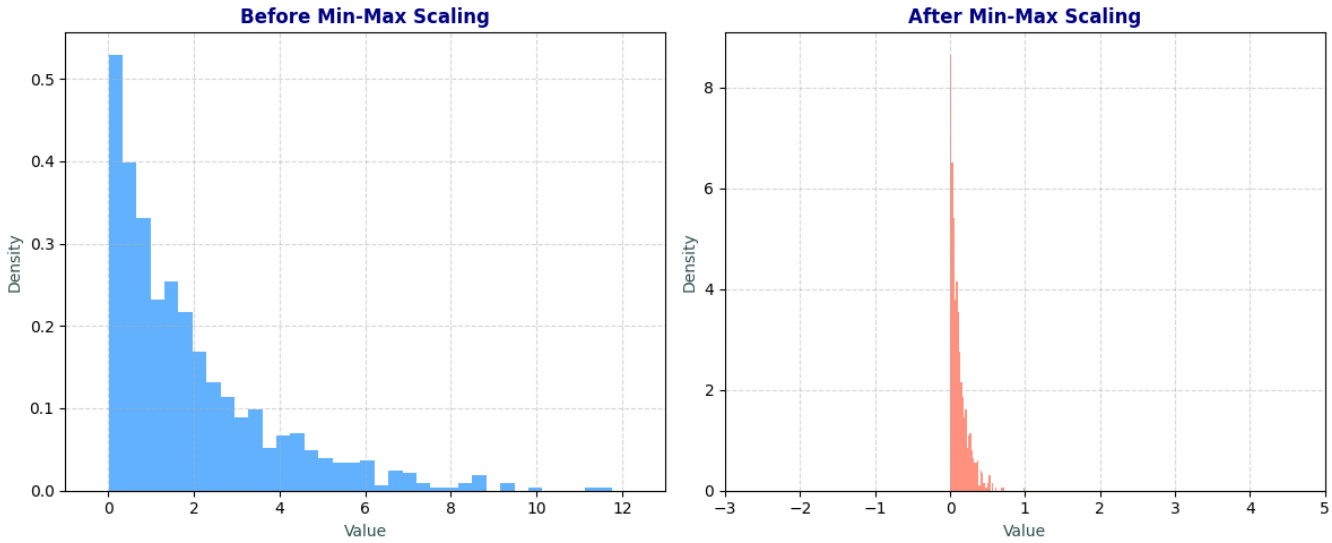
4. **Hiệu quả của Regularization:** Các phương pháp điều chuẩn (Lasso và Ridge) cho hiệu suất tương đương với mô hình cơ sở ($R^2 \approx 0.951$), chứng tỏ các kỹ thuật regularization có khả năng kiểm soát độ phức tạp của mô hình mà không làm ảnh hưởng đến hiệu suất dự đoán.

Kết luận: Mô hình cơ sở, không bias và các mô hình có áp dụng regularization (Lasso, Ridge) cho hiệu suất xuất sắc với chỉ số R^2 đạt 0.951, nghĩa là các mô hình này có khả năng giải thích được 95.1% phương sai trong dữ liệu giá nhà. Các phương pháp chuyển hóa biến mục tiêu không mang lại cải thiện đáng kể về hiệu suất trong bài toán này, thậm chí còn làm giảm khả năng dự đoán của mô hình.

6.5.2 Vấn đề với việc chuyển hóa biến mục tiêu

Tùy thuộc vào bài toán, việc scale biến mục tiêu có thể gia tăng hiệu suất mô hình lên đáng kể (tốc độ học nhanh hơn, tiết kiệm chi phí,...). Theo như thống kê [33], Min-Max scaling đem lại kết quả khá tệ. Lý do có thể là một vài nguyên nhân chính sau đây:

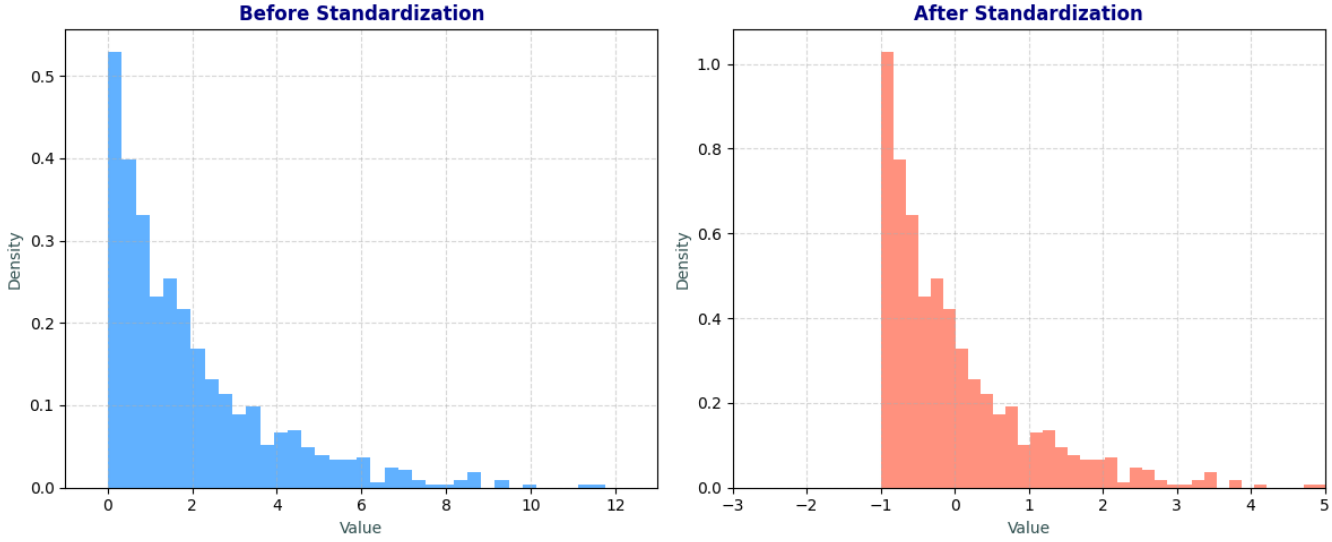
- **Min-Max Scaling**: Phương pháp này đưa biến mục tiêu về khoảng [0,1], giữ nguyên dạng phân phối nhưng nén phạm vi giá trị. Việc nén này có thể làm mất thông tin về không gian giữa các đặc trưng, dẫn đến mô hình học sai các mối quan hệ.



Hình 34: Minh họa phân phối dữ liệu trước và sau Min-Max Scaling

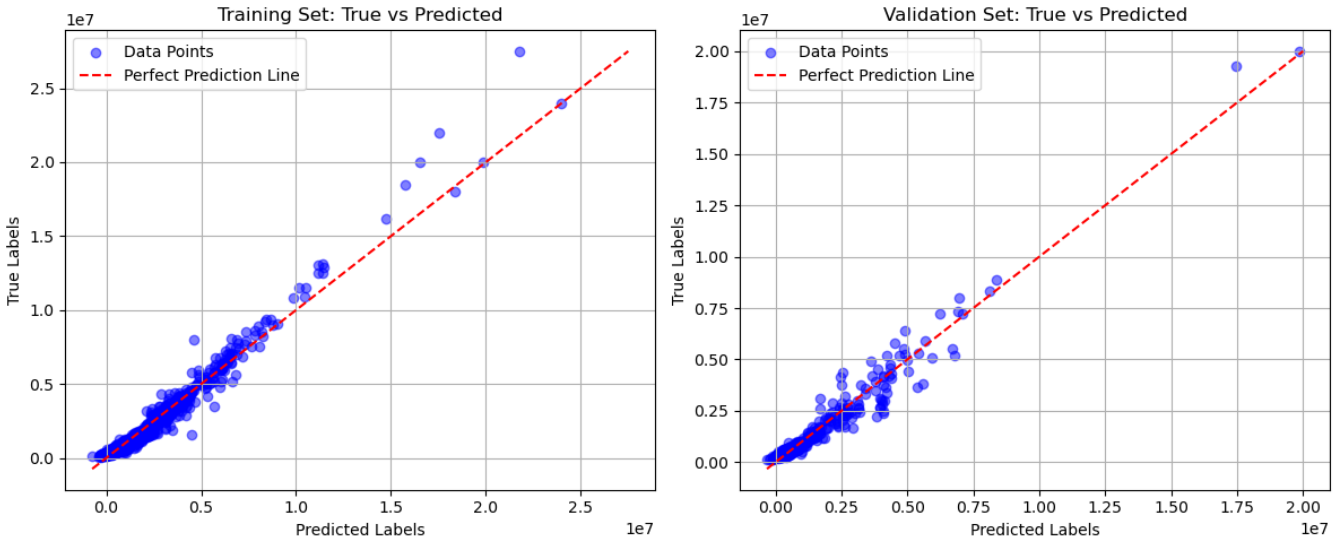
Nhận xét: Dễ thấy độ phân tán của các điểm dữ liệu thấp do bị ép nén về một phạm vi nhỏ.

- **So sánh với Standardization**: Ngược lại, Standardization hoạt động hiệu quả hơn vì đây là một phép biến đổi tuyến tính, nó không làm thay đổi mối quan hệ tuyến tính giữa biến đầu vào và đầu ra, đồng thời, nó vẫn đưa biến mục tiêu về cùng thang đo với các đặc trưng đã chuẩn hóa, giúp cải thiện hiệu suất.



Hình 35: Minh họa phân phối dữ liệu trước và sau Standardization

6.5.3 Kết quả dự đoán của mô hình tốt nhất:



Hình 36: Đường thẳng hồi quy dự đoán so với thực tế, trên tập huấn luyện và kiểm chứng

7 Mô hình hồi quy tuyến tính đa thức

Hồi quy tuyến tính đa thức [18] (Polynomial regression) là một dạng hồi quy mở rộng của hồi quy tuyến tính, trong đó mối quan hệ giữa biến độc lập và biến phụ thuộc không phải là một đường thẳng mà là một đa thức bậc cao hơn. Ngoài ra, hồi quy tuyến tính đa thức còn là mô hình hiệu quả để đánh giá các mối quan hệ phi tuyến tính giữa các giá trị đầu vào và đầu ra, đặc biệt hữu dụng khi phân loại các dữ liệu cần đường hồi quy phức tạp.

Với hồi quy tuyến tính đa thức, mô hình được mở rộng thành các bậc cao hơn của biến độc lập X. Mô hình hồi quy đa thức đơn biến bậc n có dạng:

$$Y = \beta_0 + \beta_1 * X + \beta_2 * X^2 + ... + \beta_k * X^k + \epsilon$$

Trong đó:

- k: Bậc của đa thức
- X: Biến độc lập
- Y: Biến phụ thuộc
- ϵ: Thành phần sai số
- β_0: Hệ số chặn (intercept)
- $\beta_1, \beta_2..., \beta_k$: Các hệ số hồi quy (coefficient)

7.1 Xây dựng mô hình

7.1.1 Phương pháp xây dựng

Nếu coi mỗi lũy thừa của biến độc lập như một biến độc lập mới: $X_1 = X, X_2 = X^2, X_3 = X^3, ...$ Ta nhận thấy rằng mô hình hồi quy đa thức trở thành:

$$Y = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + ... + \beta_k * X_k + \epsilon$$

Nhận xét: Qua mô hình trên thấy được hồi quy tuyến tính đa thức là một dạng đặc biệt của hồi quy tuyến tính đa biến (Multiple Linear Regression) (xem ở 2.3). Vì vậy để xây dựng mô hình hồi quy tuyến tính đa thức, ta có thể xây dựng dựa vào mô hình hồi quy đa biến với các đặc trưng đa thức $[X, X^1, X^2, X^3, ..., X^k]$.

Ta sử dụng **phương pháp bình phương tối thiểu** (Ordinary Least Squares - OLS) (xem 2.1.2) hoặc **phương pháp Gradient Descent** (xem 2.2) để giải quyết bài toán. Ngoài ra, ta có thể sử dụng thêm kỹ thuật **Chính quy hóa** (Regularization) với L1 (Lasso) (xem 2.3.4) hoặc L2 (Ridge) (xem 2.3.3) để giảm tình trạng overfitting hoặc giải được bài toán hồi quy đa thức bậc cao với nhiều đặc trưng (đa biến) có bao gồm sự tương tác giữa các đặc trưng với nhau.

7.1.2 Xây dựng đặc trưng đa thức

a) Giới thiệu: Đặc trưng đa thức là các đặc trưng mới được xây dựng từ các đặc trưng đầu vào gốc bằng cách nâng bậc chúng lên. Ví dụ:

Với một biến đầu vào x:

- Bậc 1: x (đặc trưng gốc)
- Bậc 2: x, x^2
- Bậc 3: x, x^2, x^3
- ...
- Bậc k: $x, x^2, x^3, ..., x^k$

Với hai biến đầu vào x_1 và x_2 (Bao gồm sự tương tác giữa các biến):

- Bậc 1: x_1, x_2 (đặc trưng gốc)
- Bậc 2: $x_1, x_2, x_1^2, x_2^2, x_1 * x_2$
- Bậc 3: $x_1, x_2, x_1^2, x_2^2, x_1 * x_2, x_1^3, x_2^3, x_1^2 * x_2, x_1 * x_2^2$
- ...

Tương tự với các trường hợp biến đầu vào nhiều hơn 2. Vì vậy sau khi áp dụng biến đổi đa thức với bậc d, chúng ta sẽ thu được ma trận đặc trưng đa thức X_{poly} với kích thước $n * m$, trong đó m là số lượng đặc trưng đa thức được tạo ra tính bằng công thức dưới.

b) Công thức tính tổng quát số lượng đặc trưng đa thức:

$$C(k+d, d) = \binom{k+d}{d} = \frac{(k+d)!}{d! * k!}$$

Trong đó:

- $C(k+d, d)$: tổ hợp chập d của $k+d$.
- k: số lượng biến đầu vào (features) ban đầu.
- d: bậc đa thức tối đa.

c) Ma trận hóa: Quá trình ma trận hóa đặc trưng đa thức thực hiện biến đổi trên từng hàng (mẫu) của ma trận đầu vào $\mathbf{X}$ để tạo ra một hàng mới trong ma trận $\mathbf{X_{poly}}$.

Ví dụ minh họa với 2 biến đầu vào (k=2) và bậc đa thức d=2:

Giả sử chúng ta có một mẫu dữ liệu duy nhất $\mathbf{x} = \begin{pmatrix} x_1 & x_2 \end{pmatrix}$ và chúng ta muốn tạo đặc trưng đa thức bậc 2. Các đặc trưng đa thức được tạo ra là:

1. Bậc 0: 1 (bias term)
2. Bậc 1: x_1, x_2
3. Bậc 2: x_1^2, x_1x_2, x_2^2

Như vậy, vector đặc trưng đa thức cho mẫu $\mathbf{x}$ sẽ là:

$$\mathbf{x_{poly}} = \begin{pmatrix} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 \end{pmatrix}$$

Tổng quát hóa cho ma trận X:

Để tạo ma trận $\mathbf{X_{poly}}$ từ ma trận $\mathbf{X}$, chúng ta sẽ áp dụng biến đổi trên cho từng hàng của $\mathbf{X}$.

Ví dụ, nếu ma trận đầu vào $\mathbf{X}$ là:

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{pmatrix}$$

Thì ma trận đặc trưng đa thức $\mathbf{X_{poly}}$ bậc 2 sẽ là:

$$\mathbf{X_{poly}} = \begin{pmatrix} 1 & x_{11} & x_{12} & x_{11}^2 & x_{11}x_{12} & x_{12}^2 \\ 1 & x_{21} & x_{22} & x_{21}^2 & x_{21}x_{22} & x_{22}^2 \\ 1 & x_{31} & x_{32} & x_{31}^2 & x_{31}x_{32} & x_{32}^2 \end{pmatrix}$$

Dựa trên ma trận hóa của mô hình **hồi quy đa biến (Multiple Regression)** (xem 2.1.2). Đối với toàn tập dữ liệu có n quan sát và m đặc trưng đa thức (features), ta có thể biểu diễn tổng quát ma trận hồi quy:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n \times 1} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{11}^2 & x_{12}^2 & \dots & x_{11} \cdot x_{12} & \dots \\ 1 & x_{21} & x_{22} & \dots & x_{21}^2 & x_{22}^2 & \dots & x_{21} \cdot x_{22} & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots \\ 1 & x_{n1} & x_{n2} & \dots & x_{n1}^2 & x_{n2}^2 & \dots & x_{n1} \cdot x_{n2} & \dots \end{bmatrix}_{n \times m} \begin{bmatrix} \beta 0 \\ \beta 1 \\ \vdots \\ \beta m \end{bmatrix}_{m \times 1} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}_{n \times 1} \tag{10}$$

Phương trình này có thể được biểu diễn một cách ngắn gọn hơn trong ký hiệu ma trận:

$$\mathbf{Y} = \mathbf{X_{poly}}\boldsymbol{\beta} + \epsilon \tag{11}$$

Trong đó:

- $\mathbf{Y} \in \mathbb{R}^{n \times 1}$ là vector cột chứa n giá trị phụ thuộc

- $\mathbf{X} \in \mathbb{R}^{n \times m}$ là ma trận chứa n mẫu quan sát với m đặc trưng đa thức cho mỗi mẫu
- $\boldsymbol{\beta} \in \mathbb{R}^{m \times 1}$ là vector hệ số hồi quy cần ước lượng
- $\epsilon \in \mathbb{R}^{n \times 1}$ là giá trị thành phần sai số ngẫu nhiên (the random error component)

d) Nhận xét: Bậc của đa thức là một hyperparameter quan trọng trong hồi quy đa thức. Việc lựa chọn bậc phù hợp có ảnh hưởng lớn đến hiệu suất của mô hình:

- Bậc quá thấp (Underfitting): Mô hình quá đơn giản, không thể nắm bắt được các mẫu phức tạp trong dữ liệu. Dẫn đến sai số cao trên cả dữ liệu huấn luyện và dữ liệu kiểm tra.
- Bậc vừa phải (Good Fit): Mô hình có độ phức tạp vừa đủ để nắm bắt được các mẫu quan trọng trong dữ liệu mà không quá khớp. Đạt được hiệu suất tốt trên cả dữ liệu huấn luyện và dữ liệu kiểm tra.
- Bậc quá cao (Overfitting): Mô hình quá phức tạp, khớp quá sát với dữ liệu huấn luyện, bao gồm cả nhiễu. Dẫn đến sai số thấp trên dữ liệu huấn luyện nhưng sai số cao trên dữ liệu kiểm tra (khả năng tổng quát hóa kém).

7.1.3 Huấn luyện mô hình

Để huấn luyện mô hình **hồi quy đa thức (Polynomial Regression)** ta có thể sử dụng quá trình huấn luyện mô hình bằng phương pháp Gradient Descent của **hồi quy đa biến (Multiple Regression)** (xem 2.2) dựa trên đặc trưng đa thức được xây dựng ở trên.

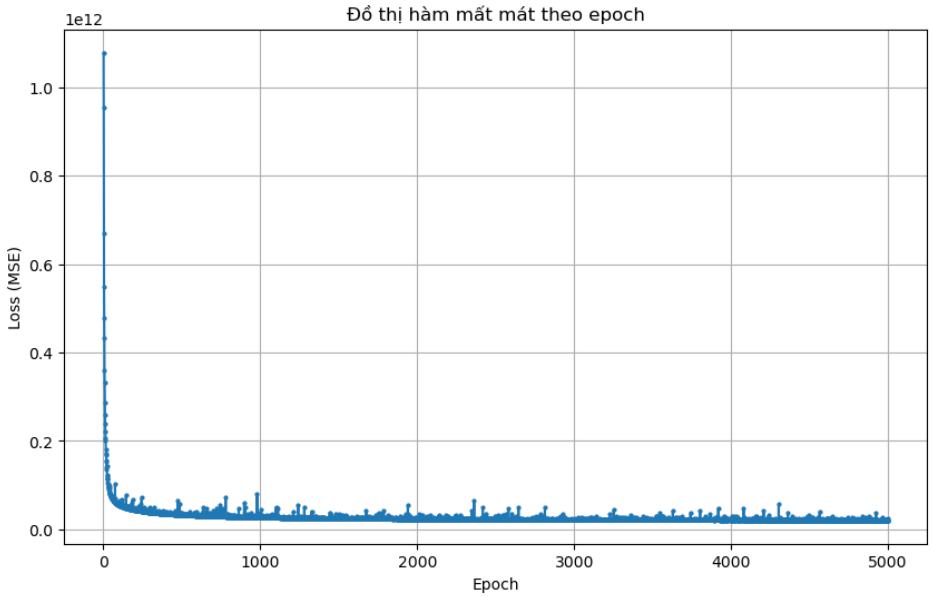
Các tham số:

- Learning rate: 0.01
- Số epoch: 5000
- Batch size: 32
- Epsilon: 10^{-8} (Ngưỡng hội tụ)

Triển khai:

- Dữ liệu huấn luyện được sắp xếp ngẫu nhiên trước mỗi epoch để giảm thiểu phương sai trong quá trình cập nhật trọng số.
- Batch Gradient Descent được sử dụng để cập nhật trọng số mô hình.
- Lưu lại bộ trọng số tốt nhất trong quá trình huấn luyện.

Quá trình huấn luyện:



Hình 37: Biểu đồ Loss qua các Epoch.

- Biểu đồ cho thấy giá trị hàm mất mát (Loss) giảm dần theo số epoch, chứng tỏ mô hình đang học được các mẫu từ dữ liệu huấn luyện.
- Ban đầu, loss rất lớn (khoảng hơn 10^{12}), nhưng sau 5000 epoch, nó giảm xuống đáng kể còn khoảng $2 * 10^{10}$.
- Mô hình đạt được giá trị loss tối ưu (trên tập huấn luyện là 20342161463.87)

7.1.4 Công thức hồi quy

Hiển thị top 10 số hạng dựa trên hệ số hồi quy lớn nhất:

$$Price = 1680790.7703 \cdot Model + 1559994.1677 \cdot Model \cdot Year + 1464639.3648 \cdot Model \cdot Year^2 + 1046459.7736 \cdot Model \cdot Transmission + 1046459.7736 \cdot Model \cdot Transmission^2 + 1009863.5419 \cdot Model \cdot Kilometer + 946888.0919 \cdot Model \cdot Year \cdot Transmission + 938621.1008 \cdot Model \cdot Location + 938621.1008 \cdot Model \cdot Location^2 + 935093.6800 \cdot Model \cdot Year \cdot Kilometer + ...$$

Trong đó:

- $Price$: Giá xe được dự đoán bởi mô hình
- $Model, Year, Transmission, ...$: Giá trị của các đặc trưng trong tập dữ liệu sau các quá trình xử lí dữ liệu.

Nhận xét: Dựa vào công thức hồi quy, ta thấy `Model` là đặc trưng tốt nhất có khả năng tương quan thuận với `Price`. Ngoài ra, các đặc trưng như `Year`, `Tranmission` cũng có sự ảnh hưởng lớn đến giá trị `Price`.

7.2 Đánh giá mô hình

Evaluate metrics on Training Set:

4 rows | 4 rows × 2 columns

	Metric	Value
0	Mean Squared Error	40398195546.67
1	Root Mean Squared Error	200993.02
2	Mean Absolute Error	112892.66
3	R^2 Score	0.99

(a) Đánh giá trên tập huấn luyện

Evaluate metrics on Validation Set:

4 rows | 4 rows × 2 columns

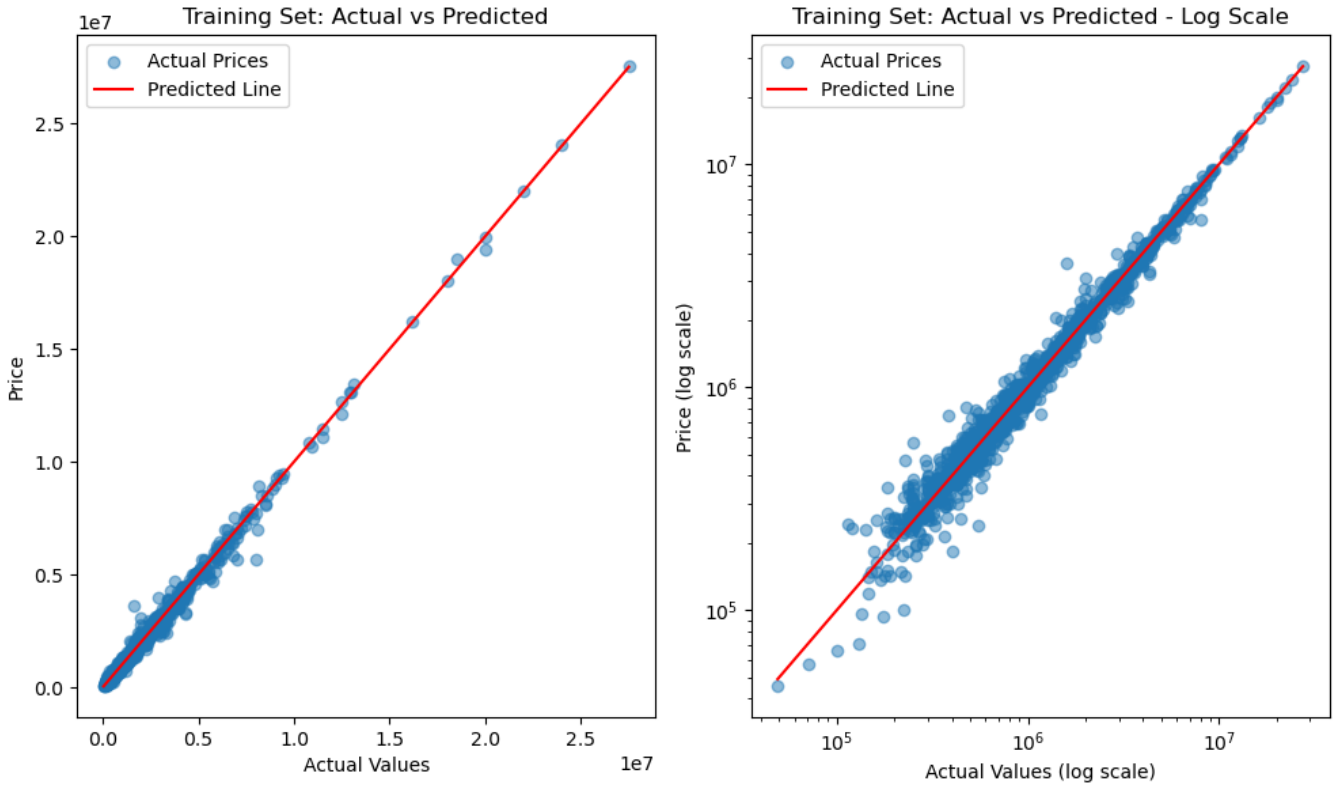
	Metric	Value
0	Mean Squared Error	189811437470.50
1	Root Mean Squared Error	435673.54
2	Mean Absolute Error	228969.98
3	R^2 Score	0.96

(b) Đánh giá trên tập kiểm chứng

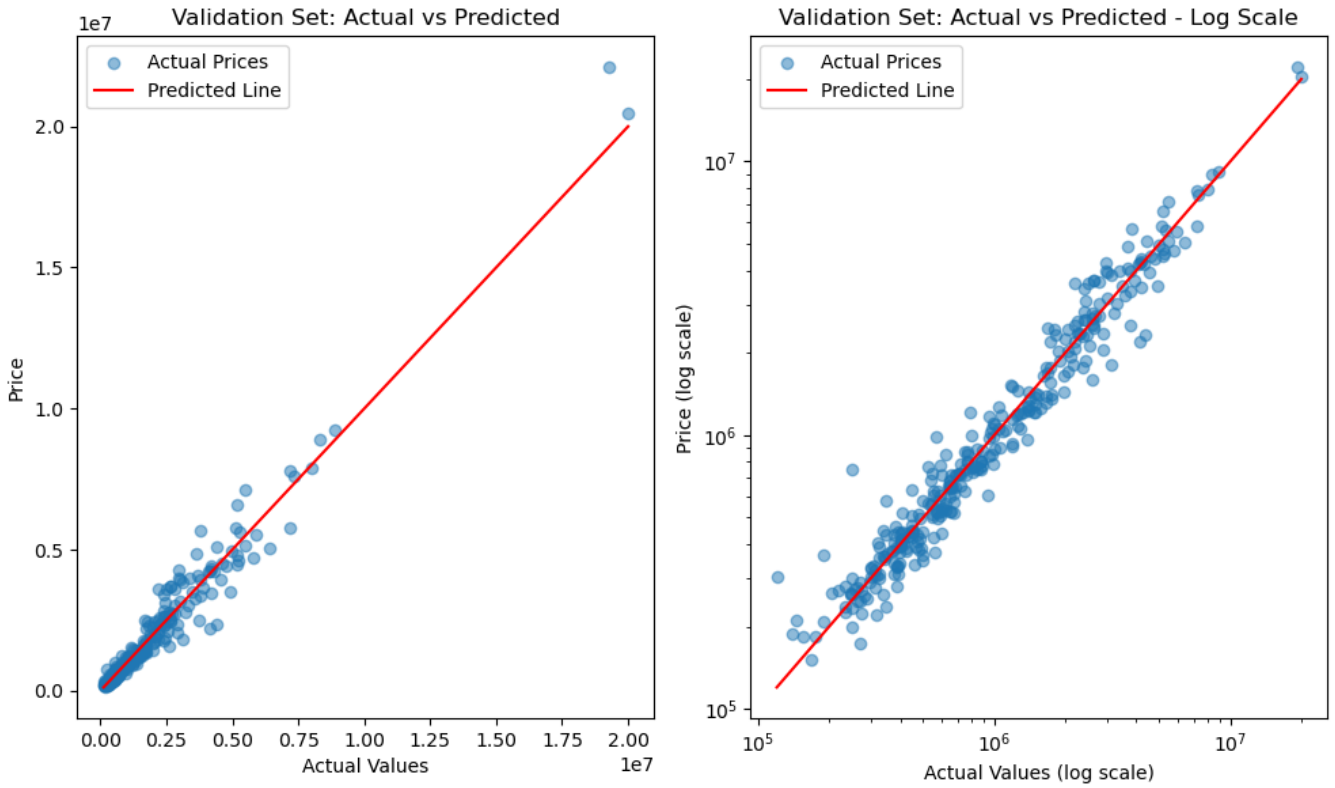
Hình 38: So sánh đánh giá mô hình hồi quy đa thức bậc 3 trên tập huấn luyện và kiểm chứng

Trên tập huấn luyện, mô hình đạt MSE là $40398195546.67 \approx 4.0398 * 10^{10}$ và R^2 là 0.99, cho thấy mô hình rất khớp với dữ liệu đã thấy. Tuy nhiên, trên tập kiểm chứng, MSE tăng lên đáng kể thành $189811437470.50 \approx 1.8981 * 10^{11}$ và R^2 giảm xuống còn 0.96.

Mặc dù giá trị R^2 trên tập kiểm chứng vẫn khá cao, sự gia tăng đáng kể của các chỉ số lỗi (MSE tăng gấp 5 lần, RMSE và MAE tăng gấp 2 lần) cho thấy dấu hiệu của hiện tượng overfitting. Mô hình đã học thuộc một số đặc điểm nhiễu của tập huấn luyện và khả năng tổng quát hóa trên dữ liệu mới bị hạn chế hơn. Điều này cũng có thể quan sát thấy trong hình 38, nơi các điểm dữ liệu kiểm chứng phân tán rộng hơn quanh đường dự đoán so với tập huấn luyện.



(a) Đánh giá trên tập huấn luyện



(b) Đánh giá trên tập kiểm chứng

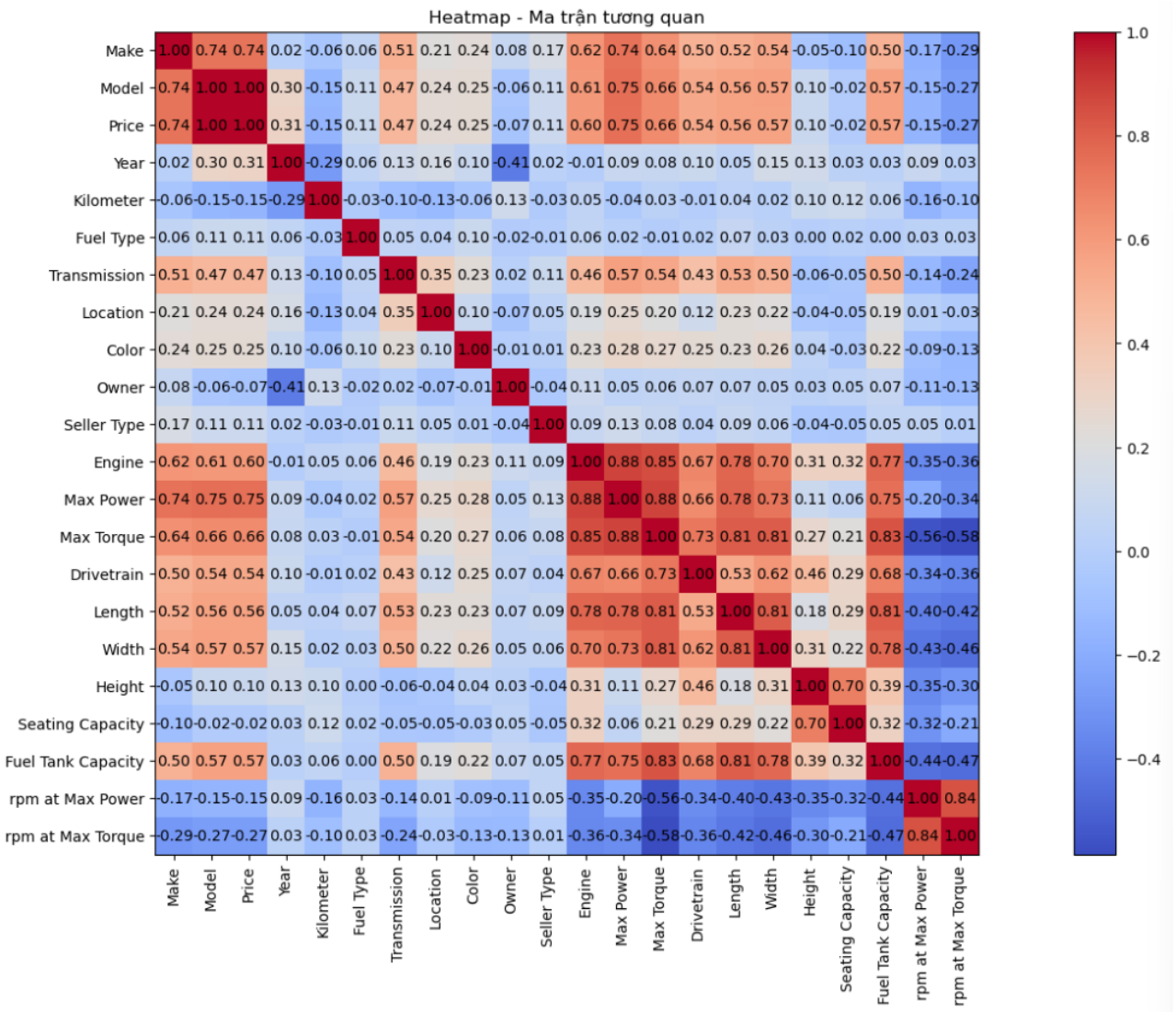
Hình 39: So sánh đánh giá mô hình hồi quy đa thức bậc 3 trên tập huấn luyện và kiểm chứng

8 Mô hình hồi quy tuyến tính kết hợp với các phương pháp PCA(Principal Component Analysis)

PCA (Principal Component Analysis) giúp giảm chiều dữ liệu bằng cách tìm ra các **thành phần chính (Principal Components - PCs)** mà vẫn giữ được nhiều thông tin nhất. Trong tập dữ liệu này, ta có nhiều biến liên quan đến nhau (tương quan cao), nên việc dùng PCA giúp:

- **Giảm đa cộng tuyến** (nhiều biến có tương quan cao sẽ gây khó khăn cho mô hình dự đoán).
- **Tăng hiệu suất tính toán** (mô hình không cần xử lý quá nhiều biến).
- **Loại bỏ nhiễu** và **giữ lại thông tin quan trọng** nhất.

8.1 Phân tích tương quan và chọn nhóm biến



Hình 40: Ma trận tương quan giữa các biến.

Phân tích ma trận tương quan cho thấy các nhóm biến có mối liên hệ chặt chẽ:

- **Nhóm kích thước xe** (`Seating Capacity`, `Length`, `Height`, `Width`, `Fuel Tank Capacity`) có mối tương quan cao, đặc biệt giữa `Length` - `Width` (0.81), `Height` - `Seating Capacity` (0.7) và `Fuel Tank Capacity` - `Width` (0.78), cho thấy khả năng giảm chiều dữ liệu.
- **Nhóm công suất động cơ** (`Engine`, `Max Power`, `Max Torque`, `Drivetrain`) có quan hệ chặt chẽ, với `Engine` - `Max Power` (0.88) và `Max Power` - `Max Torque` (0.88). `Drivetrain` - `Max Torque` (0.72) cho thấy khả năng giảm chiều dữ liệu.
- **Nhóm vòng tua động cơ** (`rpm at Max Power`, `rpm at Max Torque`) có tương quan 0.84, thể hiện mối quan hệ tuyến tính mạnh, phù hợp để giảm số chiều dữ liệu.

8.2 Áp dụng phương pháp PCA

PCA được áp dụng trên ba nhóm biến có tương quan cao nhằm giảm chiều dữ liệu mà vẫn bảo toàn thông tin quan trọng. Quá trình thực hiện gồm các bước:

1. **Áp dụng PCA trên tập train**:
   - Áp dụng phương pháp PCA (xem các bước thực hiện ở 2.4.1)
   - Kết quả thu được ba đặc trưng mới: `PCA_hs`, `PCA_cs`, `PCA_kt`.
   - Các cột ban đầu được loại bỏ khỏi tập dữ liệu.
2. **Áp dụng PCA trên tập validation**:
   - Sử dụng giá trị trung bình và vector thành phần chính từ tập train để biến đổi dữ liệu validation, đảm bảo nhất quán.
   - Thay thế các biến gốc bằng các đặc trưng PCA tương ứng.

Sau khi áp dụng **PCA**, ta thu được các 13 đặc trưng gồm `PCA_hs`, `PCA_cs`, `PCA_kt`, cùng với các đặc trưng ban đầu như `Make`, `Model`, `Year`, `Kilometer`, `Fuel Type`, `Transmission`, `Location`, `Color`, `Owner`, `Seller Type` dùng để dự đoán giá xe *Price*

8.3 Chuẩn hóa dữ liệu và chia tập dữ liệu để huấn luyện

- **Chuẩn hóa dữ liệu** bằng phương pháp Z-score (xem 2.1.1). Việc này giúp đưa dữ liệu về cùng một thang đo, giảm ảnh hưởng của đơn vị đo khác nhau và cải thiện tốc độ hội tụ của thuật toán tối ưu (xem 2.2.3).
- **Chia tập dữ liệu**:
  - $X_{\text{train\_v2}}$, $X_{\text{val\_v2}}$: Đặc trưng đầu vào (sau chuẩn hóa và PCA).
  - $y_{\text{train\_v2}}$, $y_{\text{val\_v2}}$: Biến mục tiêu (`Price`).

8.4 Xây dựng mô hình

8.4.1 Phương pháp xây dựng

Mô hình hồi quy tuyến tính đa biến (Multiple Linear Regression) (xem ở 2.3) được xây dựng để dự đoán giá xe dựa trên các đặc trưng đã xử lý (bao gồm các thành phần từ PCA và các biến khác).

8.4.2 Huấn luyện mô hình

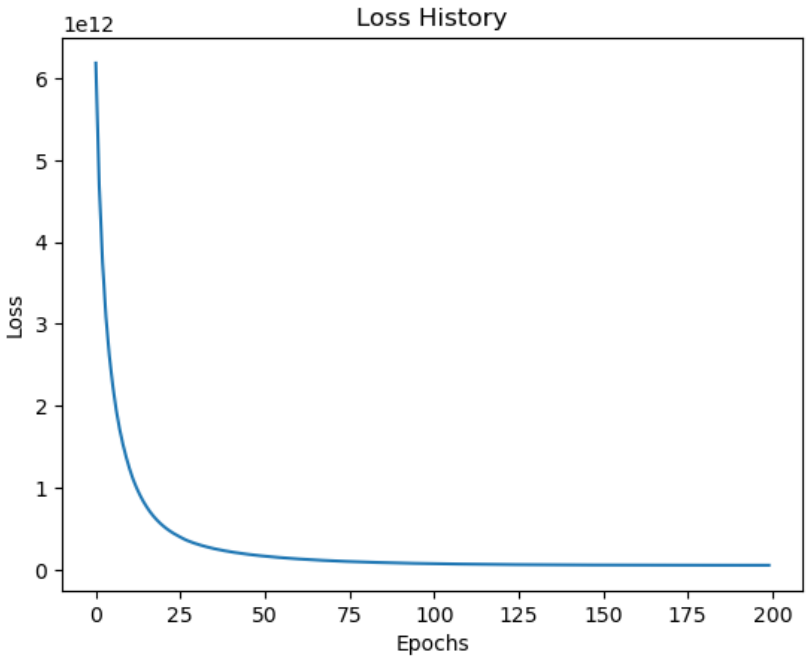
Các siêu tham số:

- Learning rate: 0.001
- Số epoch: 200
- Batch size: 32

Triển khai:

- Dữ liệu huấn luyện được xáo trộn trước mỗi epoch để giảm thiểu phương sai trong quá trình cập nhật trọng số.
- Batch Gradient Descent được sử dụng để cập nhật trọng số mô hình.
- Lưu lại bộ trọng số tốt nhất trong quá trình huấn luyện.

Quá trình huấn luyện:



Hình 41: Biểu đồ Loss qua các Epoch.

- Biểu đồ cho thấy giá trị hàm mất mát (Loss) giảm dần theo số epoch, chứng tỏ mô hình đang học được các mẫu từ dữ liệu huấn luyện.
- Ban đầu, loss rất lớn (khoảng 1369526006662), nhưng sau 200 epoch, nó giảm xuống đáng kể còn khoảng 552622539599.0718.
- Mô hình đạt được giá trị loss tối ưu (trên tập huấn luyện là 552622539599.0718)

8.4.3 Công thức hồi quy

$$\begin{aligned} Price = 1715141.18 &+ 57577.71x_1 + 2332775.74x_2 + 39443.65x_3 \\ &- 7834.82x_4 + 1770.64x_5 - 11278.79x_6 \\ &- 3295.49x_7 - 4708.11x_8 - 18789.88x_9 \\ &+ 7566.50x_{10} + 11094.97x_{11} - 7151.10x_{12} \\ &- 4041.87x_{13} \end{aligned}$$

Trong đó:

- $Price$: Giá xe được dự đoán bởi mô hình
- x_i với $i \in \{1, 2, \ldots, 13\}$: Giá trị của các đặc trưng sau các quá trình xử lí dữ liệu và quá trình PCA (xem 8.2)

8.5 Đánh giá mô hình

Evaluation metrics on Training Set:

	Metric	Value
0	Mean Squared Error	5.262254e+10
1	Root Mean Squared Error	2.293960e+05
2	Mean Absolute Error	9.658851e+04
3	R^2 Score	9.909705e-01

(a) Đánh giá trên tập huấn luyện

Evaluation metrics on Validation Set:

	Metric	Value
0	Mean Squared Error	2.952896e+11
1	Root Mean Squared Error	5.434056e+05
2	Mean Absolute Error	2.913129e+05
3	R^2 Score	9.348171e-01

(b) Đánh giá trên tập kiểm chứng

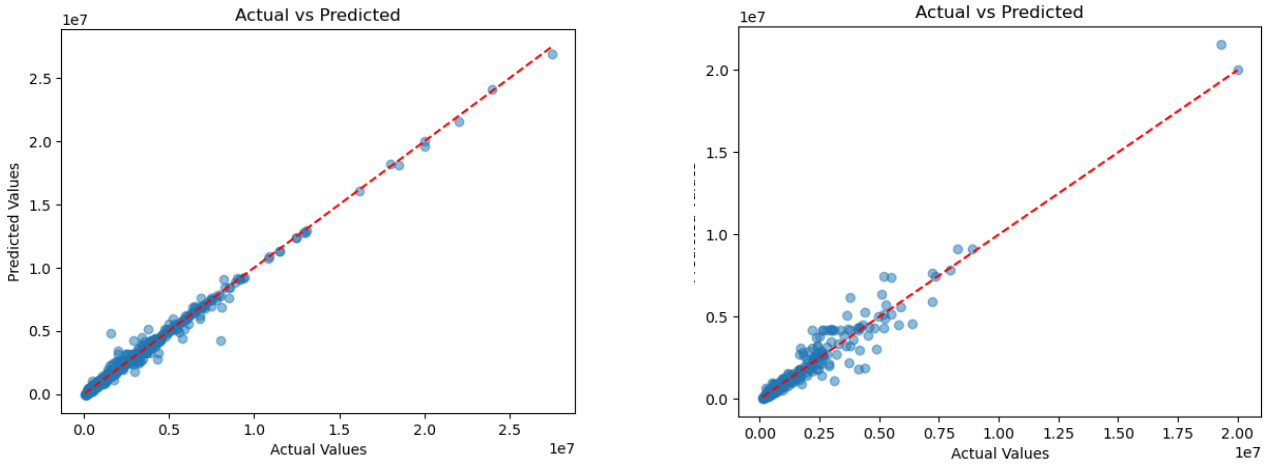
Hình 42: So sánh đánh giá mô hình trên tập huấn luyện và kiểm chứng

Trên tập huấn luyện, mô hình đạt MSE là 5.262254×10^{10} và R^2 là 0.9910 (từ $9.909705e-01$), cho thấy mô hình đã học tốt giữa biến đầu vào biến đầu ra. Tuy nhiên, trên tập kiểm chứng, MSE tăng lên đáng kể thành 2.952896×10^{11} và R^2 giảm xuống còn 0.9348 (từ $9.348171e-01$).

Mặc dù R^2 trên tập kiểm chứng vẫn khá cao, sự gia tăng đáng kể của các chỉ số lỗi (MSE tăng từ 5.26×10^{10} lên 2.95×10^{11}) cho thấy dấu hiệu của hiện tượng overfitting. Điều này cho thấy rằng mô hình đã học quá tốt trên bộ dữ liệu, làm giảm khả năng tổng quát hóa trên bộ dữ liệu mới.

Biểu đồ "Actual vs Predicted" minh họa sự so sánh giữa giá trị thực tế và giá trị dự đoán của mô hình (quan sát thấy trong hình 43). Nhìn chung trên tập huấn luyện và kiểm chứng, các dữ liệu tập trung gần đường thẳng hồi quy cho thấy rằng mô hình dự đoán tương đối tốt xu hướng chung. Tuy nhiên, trên tập kiểm chứng, các điểm dữ liệu phân tán rộng xung quanh đường hồi quy (Có thể là kết quả của các giá trị ngoại lai hoặc những trường hợp mô hình dự đoán chưa chính xác).

Từ những nhận xét trên, ta thấy rằng mô hình hoạt động khá tốt với dữ liệu khi được áp dụng phương pháp PCA (Principal Component Analysis) và áp dụng phương pháp chuẩn hóa Z-score. Đúng với cơ sở lý thuyết toán học đã viết ở trên (2.2.3, 2.4.1).



(a) Đánh giá trên tập huấn luyện

(b) Đánh giá trên tập kiểm chứng

Hình 43: Đường thẳng hồi quy dự đoán so với thực tế, trên tập huấn luyện và kiểm chứng

9 Kết luận

9.1 Hiệu suất các mô hình

Mô hình	Phương pháp đánh giá							
	Tập huấn luyện				Tập kiểm chứng			
Tên	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**	R^2
Simple	5.32×10^{10}	2.30×10^5	7.55×10^4	0.99	3.16×10^{11}	5.62×10^5	2.98×10^5	0.930
Multiple	1.80×10^{11}	4.25×10^5	2.41×10^5	0.96	2.20×10^{11}	4.69×10^5	2.98×10^5	0.951
Poly	4.03×10^{10}	2.00×10^5	1.12×10^5	0.99	1.89×10^{11}	4.35×10^5	2.28×10^5	0.960
PCA	5.26×10^{10}	2.29×10^5	9.65×10^4	0.99	2.95×10^{11}	5.43×10^5	2.91×10^5	0.934

Bảng 2: Các đánh giá cho từng mô hình

Nhận xét:

- Mô hình Simple đạt R^2 cao do sự tương quan lớn giữa đặc trưng `Model` và `Price`.
- Mô hình Multiple cho hiệu suất không chênh lệch nhiều[5] trên tập huấn luyện và kiểm chứng.
- Mô hình Polynomial cho hiệu suất tốt nhất trên cả tập huấn luyện và tập kiểm chứng.
- Mô hình PCA đã cải thiện hiệu suất hơn so với mô hình Simple.

9.2 So sánh các loại mô hình

Đặc điểm	**Hồi quy đơn giản**	**Hồi quy đa biến**	**Hồi quy đa thức**
Số biến độc lập	Một biến	Nhiều biến	Một biến/Nhiều biến có bậc
Mô hình toán học	$Y = b + \theta_1 X$	$Y = b + \theta_1 x_1 + \cdots + \theta_p x_p$	$y = \sum_{\lvert\mathbf{j}\rvert \le d} \beta_{\mathbf{j}} \mathbf{x}^{\mathbf{j}} + \epsilon$
Diễn giải hình học	Đường thẳng trong không gian 2 chiều	Siêu phẳng trong không gian $p+1$ chiều	Siêu mặt cong trong không gian nhiều chiều
Ưu điểm	Đơn giản, dễ đánh giá, tránh được một số vấn đề đa cộng tuyến.	Mô hình hóa tuyến tính, đơn giản, ít bị đa cộng tuyến.	Mô hình hóa mối quan hệ phi tuyến tính, dự đoán hiệu quả dữ liệu phức tạp.
Nhược điểm	Hạn chế khả năng mô hình hóa tương tác.	Hạn chế mô hình hóa phi tuyến tính.	Phức tạp, dễ bị cộng đa tuyến, overfitting.

Bảng 3: So sánh giữa các mô hình hồi quy tuyến tính

Nhận xét: Hồi quy tuyến tính đa thức vượt trội hơn so với hồi quy tuyến tính đơn giản và hồi quy đa biến trong việc:

[5] Mô hình Multiple có sử dụng Cross-validation trong quá trình huấn luyện

- Mô hình hóa các mối quan hệ phi tuyến tính.
- Cung cấp khả năng phân tích sự tương tác giữa các đặc trưng của bộ dữ liệu.
- Tăng khả năng dự báo chính xác khi có nhiều yếu tố ảnh hưởng, bộ dữ liệu phức tạp.

Qua thực nghiệm, mô hình hồi quy đa thức (polynomial regression) là mô hình tốt nhất để giải quyết bài toán dự đoán giá xe.

Tài liệu

[1] Ha Duong, Hoang Trung, Duc Tai, and Minh Trung. Math for AI - MTH00056, AI23@HCMUS. https://github.com/ductai05/Math-For-AI/, 2025.

[2] NTM Ngoc, NV Thin, NTH Nhung, and ND Minh. *Giáo trình bài tập Xác suất thống kê*. Nhà xuất bản Đại học Quốc gia TP.HCM, HCMC, 1st edition, 2022.

[3] Trung tâm Thông tin Khoa học và Công nghệ TP.HCM. Thống kê mô tả trong nghiên cứu. https://thongke.cesti.gov.vn/dich-vu-thong-ke/tai-lieu-phan-tich-thong-ke/861-thong-ke-mo-ta-trong-nghien-cuu-dai-luong-tuong-quan.

[4] Lý Kim Hà, Nguyễn Vũ Huy, Bùi Lê Trọng Thanh, Nguyễn Thị Thu Vân, Huỳnh Quang Vũ, Lê Văn Chánh, Hồ Thị Kim Vân, and Nguyễn Hoàng Hải. *Giáo trình Vi tích phân 2*. Bộ môn Giải tích, khoa Toán - Tin học, trường Đại học Khoa học tự nhiên - ĐHQG.HCM, HCMC, 18-3-2025 edition, 2025. https://drive.google.com/file/d/1Td7-zDZYFdop6f1IXvsPO0S4Cxc7ccd3.

[5] Machine Learning Cơ Bản. Gradient descent. https://machinelearningcoban.com/2017/01/12/gradientdescent/, 2017.

[6] CS Study Fun. Tốc độ hội tụ. https://csstudyfun.wordpress.com/tag/tốc-dộ-hội-tụ/.

[7] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press, 2020.

[8] Machine Learning Cơ Bản. Hồi quy tuyến tính. https://machinelearningcoban.com/2016/12/28/linearregression/, 2016.

[9] Phạm Đình Khánh. Ridge regression. https://phamdinhkhanh.github.io/deepai-book/ch_ml/RidgedRegression.html.

[10] Xavier Bourret Sicotte. Coordinate descent soft-thresholding update operator for lasso. https://stats.stackexchange.com/questions/123672/coordinate-descent-soft-thresholding-update-operator-for-lasso.

[11] Machine Learning Cơ Bản. Phân tích thành phần chính (PCA). https://machinelearningcoban.com/2017/06/15/pca/, 2017.

[12] Kenneth Benoit. Linear regression models with logarithmic transformations. *London School of Economics, London*, 22(1):23–36, 2011.

[13] Jason Brownlee. How to use standardscaler and minmaxscaler transforms in python. *Machine Learning Mastery*, 10:10, 2020.

[14] GeeksforGeeks. Regression metrics. https://www.geeksforgeeks.org/regression-metrics, 2025.

[15] GeeksforGeeks. Multiple linear regression using python. https://www.geeksforgeeks.org/ml-multiple-linear-regression-using-python/.

[16] ARYAN GULATI. Standardization (what is and why to standardize data. https://www.kaggle.com/discussions/general/221956.

[17] Normalized Nerd. Standardization vs normalization clearly explained! https://youtu.be/sxEqtjLC0aM?si=fn54TjtX-FM9U6MQ.

[18] Eva Ostertagová. Modelling using polynomial regression. *Procedia engineering*, 48:500–506, 2012.