ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Phân tích và Mô hình hóa Dữ liệu Giá nhà tại Boston

Thực hiện: Lương Quý Huy
Mã sinh viên: 21000683

Mã học phần: MAT3508
Học kỳ 1, Năm học 2025-2026

Thông tin Dự án

Học phần:	MAT3508 – Nhập môn Trí tuệ Nhân tạo
Học kỳ:	Học kỳ 1, Năm học 2025-2026
Trường:	VNU-HUS (Đại học Quốc gia Hà Nội – Trường Đại học Khoa học Tự nhiên)
Tên dự án	Phân tích và mô hình hóa dữ liệu giá nhà tại Boston
Ngày nộp:	29/11/2025
Báo cáo PDF:	báo cáo PDF trong kho GitHub
Slide thuyết trình:	slide thuyết trình trong kho GitHub
Kho GitHub:	VNU-HUS-IntroAI-MiniProject

Thành viên nhóm

Họ tên	Mã sinh viên	Tên GitHub	Đóng góp
Lương Quý Huy	21000683	Gaxjvool	Toàn bộ

Danh sách hình vẽ

Danh sách bảng

Mục lục

Chương 1

Giới thiệu

1.1 Tóm tắt

Dự án "Phân tích và mô hình hóa dữ liệu giá nhà tại Boston" nhằm xây dựng một quy trình phân tích dữ liệu hoàn chỉnh từ khám phá dữ liệu (EDA), tiền xử lý, giảm chiều (PCA) đến xây dựng và so sánh các mô hình học máy để dự đoán giá nhà trung bình (MEDV) trong bộ dữ liệu Boston Housing. Mục tiêu chính của nhóm là tìm hiểu các yếu tố ảnh hưởng đến giá nhà, giảm chiều dữ liệu để trực quan hóa và tăng tính hiệu quả mô hình, đồng thời so sánh hiệu năng giữa các phương pháp hồi quy, phân loại và phân cụm tiêu biểu.

Nhóm đã thực hiện chuẩn hóa dữ liệu, phân tích tương quan, áp dụng PCA (với cả 2 và nhiều thành phần chính) để khảo sát cấu trúc dữ liệu, rồi triển khai các mô hình gồm Hồi quy tuyến tính, SVM, KNN, cùng các phương pháp phân cụm (K-Means, Agglomerative). Kết quả nổi bật cho thấy mô hình hồi quy tuyến tính đạt $R^2 \approx 0.60$ trên tập kiểm tra, trong khi khi rời rạc hóa MEDV để phân loại, SVM đạt 85.29% và KNN đạt 82.35%. Những kết quả này cho thấy phương pháp tiếp cận của nhóm có khả năng giải thích và dự đoán tương đối tốt, đồng thời còn nhiều không gian để cải thiện qua tinh chỉnh siêu tham số và xử lý dữ liệu.

1.2 Bài toán đặt ra

Bài toán dự đoán giá nhà tại Boston là một trong những ví dụ kinh điển trong học máy, được sử dụng rộng rãi để đánh giá khả năng của các mô hình hồi quy và các kỹ thuật giảm chiều dữ liệu. Mục tiêu chính là xây dựng mô hình có thể dự đoán giá trị trung bình của những căn nhà (MEDV) dựa trên 13 đặc trưng kinh tế – xã hội và hạ tầng, chẳng hạn như tỷ lệ tội phạm, mật độ công nghiệp, mức độ ô nhiễm không khí, tỷ lệ học sinh – giáo viên, hay khoảng cách đến trung tâm việc làm.

Bài toán này có ý nghĩa thực tiễn quan trọng trong việc hỗ trợ quy hoạch đô thị, phân tích thị trường bất động sản, cũng như giúp các nhà đầu tư và cơ quan quản lý hiểu rõ hơn về các yếu tố ảnh hưởng đến giá nhà. Từ góc độ học máy, đây là một bộ dữ liệu tiêu chuẩn để kiểm thử và so sánh hiệu năng giữa các thuật toán khác nhau, bao gồm cả hồi quy, phân loại, và phân cụm.

Tuy nhiên, bài toán cũng đặt ra nhiều thách thức. Dữ liệu có thể tồn tại tương quan cao giữa các đặc trưng (đa cộng tuyến), gây ảnh hưởng đến độ chính xác của mô hình hồi quy. Ngoài ra, sự phân bố không đồng đều của biến mục tiêu, nhiễu trong dữ liệu và giới hạn về số lượng mẫu (506 quan sát) khiến việc huấn luyện và đánh giá mô hình đòi hỏi phải lựa chọn phương pháp tiền xử lý và điều chỉnh siêu tham số phù hợp để đạt hiệu suất tối ưu và khả năng khái quát hóa cao.

Chương 2

Phương pháp & Triển khai

2.1 Phương pháp

2.1.1 Cách tiếp cận tổng thể

Dự án được thiết kế nhằm mô phỏng một quy trình học máy hoàn chỉnh, bao gồm các giai đoạn: thu thập và khám phá dữ liệu, tiền xử lý, giảm chiều dữ liệu, xây dựng mô hình, đánh giá kết quả và trực quan hóa. Quy trình này được triển khai trong ngôn ngữ Python, sử dụng các thư viện tiêu chuẩn trong khoa học dữ liệu như `pandas`, `NumPy`, `scikit-learn`, `matplotlib`, và `seaborn`.
Các bước chính trong phương pháp luận bao gồm:

1. **Thu thập dữ liệu:** Sử dụng bộ dữ liệu Boston Housing từ nguồn mở, chứa 506 mẫu và 13 đặc trưng đầu vào cùng một biến mục tiêu `MEDV` (giá trị trung bình của nhà, tính bằng nghìn USD).
2. **Khám phá và tiền xử lý dữ liệu (EDA & Preprocessing):** Phân tích thống kê mô tả, kiểm tra giá trị thiếu, chuẩn hóa dữ liệu, và đánh giá mối tương quan giữa các đặc trưng.
3. **Giảm chiều dữ liệu (PCA):** Áp dụng phương pháp phân tích thành phần chính để giảm số lượng đặc trưng, giúp loại bỏ đa cộng tuyến và trực quan hóa dữ liệu.
    - **Hồi quy tuyến tính (Linear Regression):** Dự đoán giá trị liên tục `MEDV`.
    - **Phân loại nhóm giá:** Rời rạc hóa biến `MEDV` thành các nhóm giá, sau đó áp dụng hai thuật toán phân loại: **K-Nearest Neighbors (KNN)** và **Support Vector Machine (SVM)**.
    - **Phân cụm (K-Means):** Phân cụm các mẫu dựa trên đặc trưng đã được chuẩn hóa và/hoặc giảm chiều bằng PCA.
4. **Đánh giá mô hình:**
    - **Hồi quy:** Đánh giá bằng hệ số xác định R^2 và sai số bình phương trung bình gốc (RMSE).
    - **Phân loại:** Sử dụng các chỉ số *Accuracy* và *F1-score*.
    - **Phân cụm:** Đánh giá độ tách biệt và đồng nhất của cụm bằng *Silhouette Score*.

2.1.2 Thu thập và mô tả dữ liệu

Nguồn dữ liệu Dữ liệu sử dụng trong dự án là bộ dữ liệu **Boston Housing** — một bộ dữ liệu kinh điển cho bài toán dự đoán giá nhà. Dữ liệu có thể lấy từ các nguồn công khai như UCI hoặc Kaggle (ví dụ: `https://www.cs.toronto.edu/delve/data/boston/bostonDetail.html`). Bộ dữ liệu gồm **506 mẫu** với **13 đặc trưng** đầu vào và một biến mục tiêu `MEDV` (giá trị trung bình của nhà, đơn vị: $1000).

Thông tin các trường (features) Các biến trong tập dữ liệu bao gồm:

- `CRIM` : tỷ lệ tội phạm bình quân đầu người theo thị trấn.
- `ZN` : tỉ lệ đất ở được quy hoạch cho các lô có diện tích trên 25.000 m^2.
- `INDUS` : tỉ lệ diện tích đất kinh doanh phi bán lẻ theo thị trấn.
- `CHAS` : biến chỉ vùng giáp sông Charles (1 nếu giáp sông; 0 nếu không).
- `NOX` : nồng độ oxit nitric (phần trên 10 triệu).

- `RM` : số phòng trung bình trên mỗi căn nhà.
- `AGE` : tỉ lệ các đơn vị sở hữu được xây trước năm 1940.
- `DIS` : khoảng cách có trọng số tới 5 trung tâm việc làm ở Boston.
- `RAD` : chỉ số khả năng tiếp cận đường cao tốc hướng tâm.
- `TAX` : thuế suất tài sản trên mỗi $10.000.
- `PTRATIO` : tỉ lệ học sinh – giáo viên theo thị trấn.
- `B` : $1000(B_k - 0.63)^2$, trong đó B_k là tỉ lệ người da đen theo thị trấn.
- `LSTAT` : phần trăm dân số có địa vị thấp hơn.
- `MEDV` : **(Target)** giá trị trung bình của các ngôi nhà do chủ sở hữu sử dụng (tính bằng $1000).

Kiểm tra dữ liệu (Data checks) Trước khi tiền xử lý và xây dựng mô hình, tiến hành các bước kiểm tra sau:

- **Kiểu dữ liệu:** Kiểm tra bằng `df.info()` hoặc `df.dtypes`. Trong thực nghiệm của nhóm, tất cả các cột đều có dạng số thực (ví dụ `float64`), do đó không cần chuyển đổi kiểu dữ liệu.
- **Giá trị khuyết thiếu:** Kiểm tra bằng `df.isnull().sum()`
- **Thống kê mô tả:** Dùng `df.describe()` để tóm tắt mean, std, min, max, quartiles cho từng biến.
- **Phân phối biến:** Vẽ biểu đồ phân phối (histogram / KDE) cho từng cột để đánh giá phân bố. Biến `MEDV` thường có độ lệch (skew) nhất định — cân nhắc phép biến đổi log khi cần đối với mô hình hồi quy.
- **Ngoại lệ (outliers):** Vẽ boxplot để phát hiện outlier ở các biến như `CRIM`, `TAX`, `LSTAT`; ghi nhận và cân nhắc xử lý nếu cần.
- **Ma trận tương quan:** Vẽ heatmap ma trận tương quan giữa các đặc trưng và với `MEDV` để xác định những biến có mối liên hệ mạnh (giá trị gần ± 1) hoặc yếu (gần 0).
- **Đa cộng tuyến (Multicollinearity):** Tính **Variance Inflation Factor (VIF)** cho từng biến để phát hiện đa cộng tuyến. Nếu VIF của biến vượt ngưỡng (ví dụ > 5 hoặc > 10), xem xét loại bỏ biến hoặc áp dụng giảm chiều (PCA).

Tiền xử lý dữ liệu (Data Preprocessing) Dựa trên các kiểm tra trên, đề xuất các bước tiền xử lý sau trước khi đưa dữ liệu vào mô hình:

1. **Tách features / target:** $X =$ `df.drop('MEDV', axis=1)`, $y =$ `df['MEDV']`.
2. **Chuẩn hóa (scaling):** Sử dụng chuẩn hóa z-score để đưa mỗi đặc trưng về trung bình 0 và độ lệch chuẩn 1 — điều này đặc biệt quan trọng cho các mô hình dựa trên khoảng cách (KNN, K-Means) và SVM.
   Định nghĩa hàm `rescale(X)` (z-score):
   Gọi $\mu = \text{mean}(X)$ và $\sigma = \text{std}(X)$. Giá trị chuẩn hóa của một phần tử x là
   $$z = \frac{x - \mu}{\sigma}.$$
   Hàm `rescale(X)` thực hiện với mỗi cột của `X`: tính μ, σ rồi trả về Series/Vector đã chuẩn hóa. (Trong thực tế có thể dùng `sklearn.preprocessing.StandardScaler` để thay thế.)
3. **Giữ biến nhị phân:** `CHAS` giữ nguyên ở dạng 0/1 (không cần one-hot nếu chỉ có một biến dạng này).
4. **Rời rạc hóa MEDV cho phân loại:** Khi chuyển bài toán sang phân loại nhóm giá, sử dụng `pd.cut` để chia `MEDV` thành 3–5 nhóm tuỳ mục tiêu phân tích; nếu cần giữ cân bằng lớp, có thể dùng bin theo quantile.
5. **Chia tập huấn luyện / kiểm tra:** Sử dụng `train_test_split(X, y, test_size=0.2, random_state=42)`. Với bài toán phân loại nhóm giá, nên sử dụng tham số `stratify` để bảo toàn tỉ lệ lớp.

2.2 Triển khai

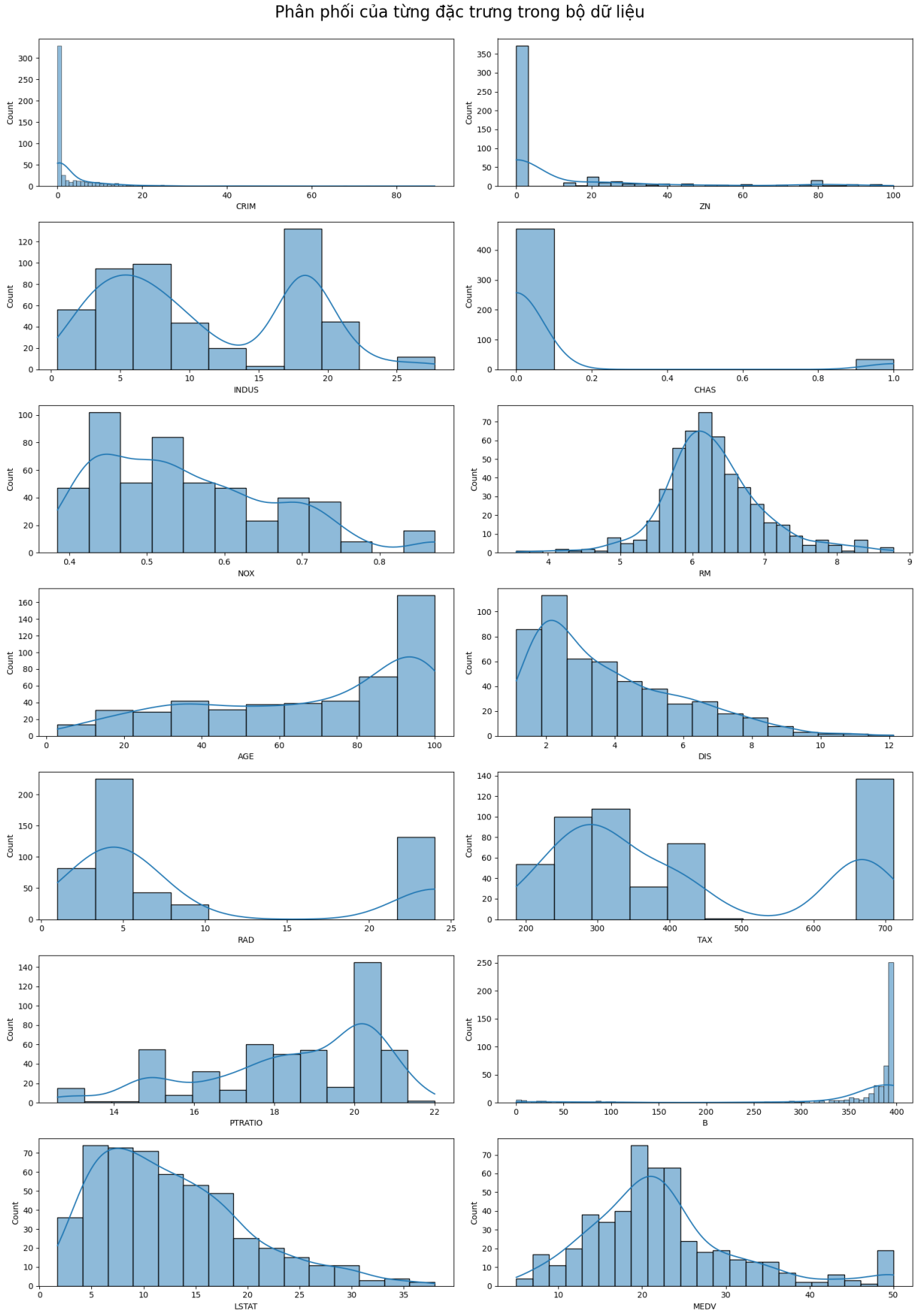
Mục này trình bày chi tiết các bước thực thi dự án bằng Python, sử dụng các thư viện khoa học dữ liệu phổ biến. Luồng triển khai được chia thành bốn giai đoạn chính: (1) Tiền xử lý và Phân tích Dữ liệu Khám phá (EDA) để làm sạch và hiểu rõ dữ liệu; (2) Xây dựng mô hình Hồi quy để dự đoán giá nhà; (3) Áp dụng thuật toán Phân cụm để khám phá các nhóm tiềm ẩn; và (4) Xây dựng các mô hình Phân loại để dự đoán các nhóm giá đã được rời rạc hóa.

Tiền xử lý và Phân tích Dữ liệu Khám phá (EDA)

Giai đoạn đầu tiên tập trung vào việc chuẩn bị một bộ dữ liệu sạch và đáng tin cậy, đồng thời trích xuất các thông tin chi tiết ban đầu thông qua trực quan hóa.

Làm sạch dữ liệu Đầu tiên, dữ liệu được tải từ tệp `HousingData.csv`. Một bước kiểm tra ban đầu cho thấy không có sự tồn tại của các giá trị thiếu (`NaN`). Ta chuyển sang bước tiếp theo.

Trực quan hóa phân phối dữ liệu Để hiểu rõ hơn về đặc điểm của từng biến, ta vẽ biểu đồ phân phối (`histogram` kết hợp với ước tính mật độ kernel - `KDE`) cho tất cả 14 đặc trưng. Việc này giúp xác định các biến có phân phối lệch (*skewed*) như `CRIM`, `ZN` và các biến có phân phối gần chuẩn như `RM`.

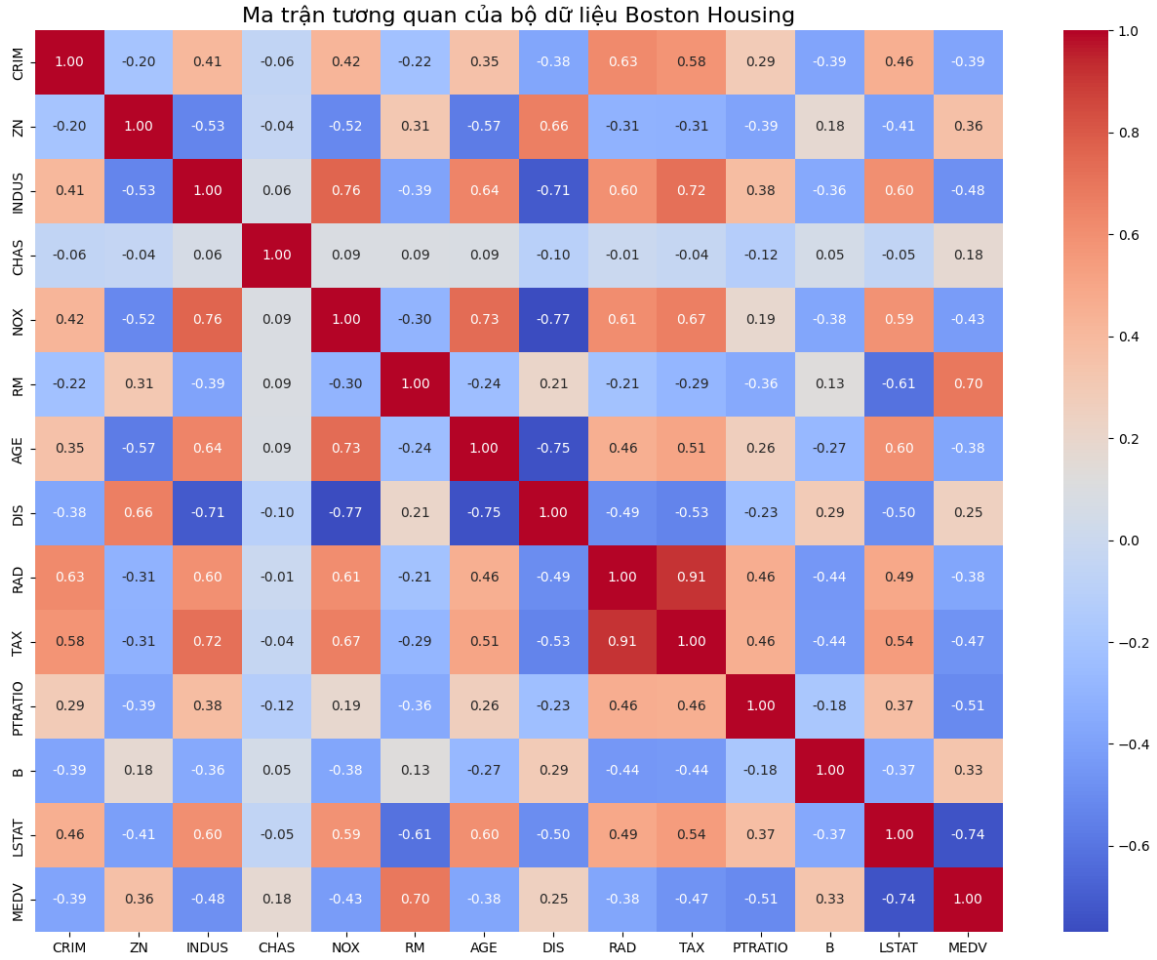


Hình 2.1: Phân phối của các đặc trưng trong bộ dữ liệu Boston Housing.

Phân tích: Hình 2.1 cho thấy nhiều đặc trưng không tuân theo phân phối chuẩn. Cụ thể:

- *Các biến `CRIM` (tỷ lệ tội phạm) và `ZN` (tỷ lệ đất ở) bị lệch phải rất mạnh, cho thấy hầu hết các khu vực có giá trị thấp ở hai đặc trưng này.*
- *Biến `CHAS` là một biến nhị phân, thể hiện sự mất cân bằng lớn khi hầu hết các ngôi nhà không nằm gần sông Charles.*
- *Biến `RM` (số phòng) có phân phối gần đối xứng nhất, tiệm cận phân phối chuẩn, tập trung quanh giá trị 6.*
- *Đáng chú ý, biến mục tiêu `MEDV` (giá nhà) có một sự tích tụ bất thường ở giá trị 50.0. Điều này là dấu hiệu của việc dữ liệu bị giới hạn trên (censored data), tức là tất cả các ngôi nhà có giá trị thực cao hơn 50,000$ đều được ghi nhận là 50.*

Phân tích tương quan Một ma trận tương quan được xây dựng và trực quan hóa bằng biểu đồ nhiệt (`heatmap`) để kiểm tra mối quan hệ tuyến tính giữa các biến. Phân tích này rất quan trọng để xác định các đặc trưng có ảnh hưởng mạnh nhất đến biến mục tiêu `MEDV` và phát hiện hiện tượng đa cộng tuyến.



Hình 2.2: Ma trận tương quan của các đặc trưng.

Phân tích: Biểu đồ nhiệt tại Hình 2.2 cung cấp nhiều thông tin giá trị:

- ***Tương quan với biến mục tiêu (`MEDV`):*** *`RM` (số phòng) có tương quan dương mạnh nhất (+0.70), trong khi `LSTAT` (tỷ lệ dân số có địa vị thấp) có tương quan âm mạnh nhất (-0.74). Điều này hoàn toàn phù hợp với trực giác: nhà nhiều phòng hơn thì đắt hơn, và khu vực có tỷ lệ dân cư thu nhập thấp cao hơn thì giá nhà rẻ hơn.*
- ***Đa cộng tuyến (Multicollinearity):*** *Hiện tượng các biến độc lập có tương quan mạnh với nhau được thể hiện rõ. Cặp biến `RAD` (chỉ số tiếp cận đường cao tốc) và `TAX` (thuế suất) có tương quan cực kỳ cao (+0.91). Tương tự, `DIS` (khoảng cách đến trung tâm) và `AGE` (tuổi của ngôi nhà) cũng có tương quan âm mạnh (-0.75). Phát hiện này là cơ sở quan trọng để quyết định sử dụng các kỹ thuật giảm chiều như PCA trong các bước mô hình hóa sau này.*

Mô hình hóa Hồi quy - Dự đoán giá nhà

Mục tiêu của giai đoạn này là xây dựng một mô hình dự báo giá trị liên tục của biến `MEDV` dựa trên các đặc trưng đầu vào. Để đảm bảo tính khách quan và hiệu quả, nhóm triển khai một pipeline bao gồm các bước: phân chia dữ liệu, chuẩn hóa, giảm chiều và cuối cùng là huấn luyện mô hình Hồi quy Tuyến tính. Các bước này được thực hiện tuần tự và mỗi bước đều có cơ sở toán học rõ ràng.

Phân chia dữ liệu (Train-Test Split) Bước đầu tiên và quan trọng nhất là phân chia bộ dữ liệu $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ (với $N = 506$) thành hai tập con không giao nhau: tập huấn luyện ($\mathcal{D}_{\text{train}}$) và tập kiểm tra ($\mathcal{D}_{\text{test}}$). Quá trình này được thực hiện theo tỷ lệ 80% cho huấn luyện và 20% cho kiểm tra.

- $\mathcal{D}_{\text{train}}$ được sử dụng để "dạy" cho mô hình, bao gồm việc tính toán các tham số cho quá trình chuẩn hóa, các thành phần chính của PCA, và các hệ số của mô hình hồi quy.
- $\mathcal{D}_{\text{test}}$ được giữ riêng hoàn toàn và chỉ được sử dụng một lần duy nhất ở cuối quy trình để đánh giá hiệu năng tổng quát hóa của mô hình trên dữ liệu mới mà nó chưa từng thấy.

Việc phân chia dữ liệu ngay từ đầu là một nguyên tắc cốt lõi để ngăn chặn hiện tượng **rò rỉ dữ liệu (data leakage)**, đảm bảo rằng việc đánh giá mô hình là hoàn toàn khách quan.

Chuẩn hóa dữ liệu (Standardization) Các đặc trưng trong bộ dữ liệu Boston Housing có thang đo và phương sai rất khác nhau. Các thuật toán dựa trên khoảng cách hoặc tối ưu hóa bằng gradient descent, bao gồm cả PCA và Hồi quy Tuyến tính (đặc biệt khi có regularize), rất nhạy cảm với sự khác biệt này. Do đó, ta áp dụng phương pháp chuẩn hóa Z-score (Standardization).

Với mỗi đặc trưng (cột) j trong tập huấn luyện $\mathbf{X}_{\text{train}}$, ta tính giá trị trung bình μ_j và độ lệch chuẩn σ_j:

$$\mu_j = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} x_{i,j} \quad \text{và} \quad \sigma_j = \sqrt{\frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} (x_{i,j} - \mu_j)^2}$$

Sau đó, mỗi giá trị $x_{i,j}$ trong cả tập huấn luyện và tập kiểm tra được biến đổi thành giá trị chuẩn hóa $x'_{i,j}$:

$$x'_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j}$$

Điều quan trọng là các giá trị μ_j và σ_j được tính **chỉ từ tập huấn luyện** và sau đó được áp dụng cho cả hai tập. Điều này mô phỏng kịch bản thực tế khi các tham số tiền xử lý phải được xác định trước khi có dữ liệu mới.

Giảm chiều dữ liệu (Principal Component Analysis - PCA) Như đã phân tích ở phần EDA, bộ dữ liệu tồn tại hiện tượng đa cộng tuyến. PCA được sử dụng để giải quyết vấn đề này bằng cách biến đổi các đặc trưng tương quan ban đầu thành một tập hợp các đặc trưng mới, không tương quan tuyến tính, được gọi là các thành phần chính (Principal Components).

Về mặt toán học, PCA thực hiện các bước sau trên dữ liệu huấn luyện đã chuẩn hóa $\mathbf{X}'_{\text{train}}$:

1. **Tính ma trận hiệp phương sai:** $\mathbf{S} = \frac{1}{N_{\text{train}}-1}(\mathbf{X}'_{\text{train}})^T\mathbf{X}'_{\text{train}}$. Ma trận này mô tả phương sai và tương quan giữa các đặc trưng.
2. **Thực hiện phân tích riêng (Eigendecomposition):** Giải bài toán giá trị riêng cho ma trận $\mathbf{S}$: $\mathbf{S}\mathbf{v}_j = \lambda_j\mathbf{v}_j$. Kết quả là một tập hợp các giá trị riêng (eigenvalues) $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$ và các vector riêng (eigenvectors) tương ứng $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_D$. Các vector riêng này chính là các thành phần chính, là các hướng trong không gian dữ liệu mà phương sai của dữ liệu là lớn nhất.
3. **Chọn số thành phần chính:** Thay vì chọn một số lượng K cố định, ta chọn số thành phần chính nhỏ nhất sao cho tổng phương sai được giải thích đạt một ngưỡng nhất định (trong trường hợp này là 95%). Tức là, chọn K nhỏ nhất thỏa mãn:
$$\frac{\sum_{j=1}^{K} \lambda_j}{\sum_{j=1}^{D} \lambda_j} \geq 0.95$$
4. **Chiếu dữ liệu:** Xây dựng ma trận chiếu $\mathbf{W} \in \mathbb{R}^{D \times K}$ có các cột là K vector riêng hàng đầu. Dữ liệu mới trong không gian K chiều được tính bằng phép chiếu: $\mathbf{Z}_{\text{train}} = \mathbf{X}'_{\text{train}}\mathbf{W}$.

Ma trận chiếu $\mathbf{W}$ tương tự được áp dụng cho tập kiểm tra đã chuẩn hóa: $\mathbf{Z}_{\text{test}} = \mathbf{X}'_{\text{test}}\mathbf{W}$.

Xây dựng mô hình Hồi quy Tuyến tính (Linear Regression) Sau khi đã có bộ dữ liệu huấn luyện $\mathbf{Z}_{\text{train}}$ với các đặc trưng đã được tối ưu, ta xây dựng mô hình Hồi quy Tuyến tính để tìm ra mối quan hệ tuyến tính giữa các thành phần chính và giá nhà. Mô hình có dạng:

$$h_{\boldsymbol{\theta}}(\mathbf{z}) = \theta_0 + \theta_1 z_1 + \theta_2 z_2 + \cdots + \theta_K z_K = \boldsymbol{\theta}^T \tilde{\mathbf{z}}$$

trong đó $\mathbf{z} = [z_1, \ldots, z_K]$ là một mẫu dữ liệu sau khi qua PCA, $\tilde{\mathbf{z}}$ là vector $\mathbf{z}$ được thêm thành phần $z_0 = 1$ cho hệ số chặn (intercept), và $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_K]$ là vector các tham số của mô hình cần tìm.

Mục tiêu là tìm ra vector $\boldsymbol{\theta}$ tối ưu sao cho tổng bình phương sai số (Sum of Squared Errors - SSE) trên tập huấn luyện là nhỏ nhất. Hàm mất mát (cost function) có dạng:

$$J(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{N_{\text{train}}} (h_{\boldsymbol{\theta}}(\mathbf{z}_i) - y_i)^2$$

Đối với Hồi quy Tuyến tính, bài toán tối ưu này có một nghiệm giải tích tường minh gọi là **Phương trình Normal (Normal Equation)**:

$$\hat{\boldsymbol{\theta}} = (\mathbf{Z}_{\text{train}}^T \mathbf{Z}_{\text{train}})^{-1} \mathbf{Z}_{\text{train}}^T \mathbf{y}_{\text{train}}$$

Sau khi tìm được vector hệ số $\hat{\boldsymbol{\theta}}$ từ tập huấn luyện, mô hình có thể dự đoán giá nhà cho một mẫu mới $\mathbf{z}_{\text{new}}$ bằng công thức: $\hat{y} = \hat{\boldsymbol{\theta}}^T \tilde{\mathbf{z}}_{\text{new}}$. Hiệu suất của mô hình này sẽ được đánh giá trên tập kiểm tra $\mathbf{Z}_{\text{test}}$ và trình bày chi tiết trong Chương 3.

Mô hình hóa Phân cụm - Khám phá cấu trúc dữ liệu

Khác với Hồi quy, vốn là một bài toán học có giám sát nhằm dự đoán một giá trị mục tiêu, Phân cụm là một phương pháp **học không giám sát**. Mục tiêu của nó không phải là dự đoán, mà là khám phá các cấu trúc hoặc nhóm tiềm ẩn (clusters) trong dữ liệu dựa trên sự tương đồng nội tại của các điểm dữ liệu. Trong dự án này, ta sử dụng thuật toán **K-Means**, một trong những thuật toán phân cụm phổ biến và hiệu quả nhất.

Chuẩn bị dữ liệu cho Phân cụm Vì mục tiêu là tìm ra các nhóm tự nhiên trong toàn bộ tập dữ liệu, ta sẽ thực hiện phân cụm trên **toàn bộ 506 mẫu** mà không cần phân chia train-test. Tuy nhiên, bước chuẩn hóa dữ liệu lại đóng vai trò cực kỳ quan trọng.

Thuật toán K-Means hoạt động dựa trên việc tối thiểu hóa khoảng cách Euclidean giữa các điểm dữ liệu và tâm cụm của chúng. Khoảng cách Euclidean giữa hai điểm dữ liệu $\mathbf{x}_a$ và $\mathbf{x}_b$ trong không gian D chiều được định nghĩa là:

$$d(\mathbf{x}_a, \mathbf{x}_b) = \sqrt{\sum_{j=1}^{D} (x_{aj} - x_{bj})^2}$$

Nếu các đặc trưng (features) có thang đo chênh lệch lớn (ví dụ: `TAX` có giá trị hàng trăm, trong khi `NOX` có giá trị dưới 1), đặc trưng có thang đo lớn hơn sẽ lấn át và chi phối hoàn toàn việc tính toán khoảng cách. Điều này làm cho các đặc trưng có thang đo nhỏ hơn gần như không có trọng số trong việc hình thành cụm.

Để đảm bảo mỗi đặc trưng đóng góp một cách công bằng vào mô hình, ta áp dụng phép **chuẩn hóa Z-score** trên toàn bộ dữ liệu, đưa mỗi đặc trưng về trung bình bằng 0 và độ lệch chuẩn bằng 1.

Thuật toán Phân cụm K-Means Mục tiêu của K-Means là phân chia N điểm dữ liệu vào K cụm $C_1, C_2, \ldots, C_K$ sao cho tổng bình phương khoảng cách từ mỗi điểm dữ liệu đến tâm (centroid) của cụm mà nó thuộc về là nhỏ nhất. Hàm mục tiêu (objective function) cần tối thiểu hóa, còn được gọi là **Inertia** hay **Within-Cluster Sum of Squares (WCSS)**, được cho bởi công thức:

$$J = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} ||\mathbf{x}_i - \boldsymbol{\mu}_k||^2$$

trong đó $\boldsymbol{\mu}_k$ là vector trung bình (centroid) của tất cả các điểm dữ liệu $\mathbf{x}_i$ thuộc cụm C_k.

Để tìm ra các cụm tối ưu, K-Means sử dụng một thuật toán lặp (iterative algorithm), thường được gọi là thuật toán của Lloyd, bao gồm các bước sau:

1. **Bước Khởi tạo (Initialization):** Chọn ngẫu nhiên K điểm dữ liệu từ tập dữ liệu làm các tâm cụm ban đầu $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$. Trong triển khai của `scikit-learn`, phương pháp khởi tạo thông minh `k-means++` thường được sử dụng để cải thiện tốc độ hội tụ và chất lượng cụm.

2. **Bước Gán nhãn (Assignment Step):** Với mỗi điểm dữ liệu $\mathbf{x}_i$, gán nó vào cụm có tâm gần nhất. Tức là, $\mathbf{x}_i$ thuộc về cụm C_k nếu:
$$k = \arg \min_{j \in \{1,\dots,K\}} ||\mathbf{x}_i - \boldsymbol{\mu}_j||^2$$

3. **Bước Cập nhật (Update Step):** Tính toán lại tâm của mỗi cụm bằng cách lấy trung bình cộng của tất cả các điểm dữ liệu đã được gán vào cụm đó trong bước trước:
$$\boldsymbol{\mu}_k \leftarrow \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$$

4. **Hội tụ (Convergence):** Lặp lại Bước Gán nhãn và Bước Cập nhật cho đến khi các tâm cụm không còn thay đổi đáng kể hoặc đạt đến số vòng lặp tối đa.

Trong khuôn khổ dự án này, ta chọn số cụm $K = 3$ để khám phá các nhóm giá tiềm năng (ví dụ: thấp, trung bình, cao) trong dữ liệu.

Các chỉ số đánh giá chất lượng cụm Do không có nhãn "đúng" trong bài toán phân cụm, ta sử dụng các chỉ số đánh giá nội tại (internal validation metrics) để đo lường chất lượng của các cụm được hình thành. Các chỉ số này đánh giá dựa trên hai tiêu chí: **độ đồng nhất** (cohesion - các điểm trong cùng một cụm phải gần nhau) và **độ tách biệt** (separation - các cụm khác nhau phải ở xa nhau).

- **Silhouette Score:** Đối với mỗi điểm dữ liệu, chỉ số này đo lường mức độ giống nhau của nó với cụm của chính nó so với cụm gần nhất kế tiếp. Giá trị của Silhouette Score dao động từ -1 đến 1, trong đó giá trị càng gần 1 cho thấy các cụm càng dày đặc và tách biệt rõ ràng.
- **Davies-Bouldin Index (DBI):** Chỉ số này định nghĩa sự tương đồng giữa hai cụm dựa trên tỷ lệ giữa tổng khoảng cách trong cụm và khoảng cách giữa các tâm cụm. Giá trị DBI càng thấp càng tốt, cho thấy các cụm có độ đồng nhất cao và tách biệt tốt.
- **Calinski-Harabasz Index (CHI):** Còn được gọi là Tiêu chí Tỷ lệ Phương sai (Variance Ratio Criterion), chỉ số này là tỷ lệ giữa phương sai giữa các cụm và phương sai trong mỗi cụm. Giá trị CHI càng cao thì các cụm càng dày đặc và tách biệt tốt.

Các chỉ số này sẽ được tính toán sau khi mô hình K-Means hội tụ để đưa ra một đánh giá định lượng về cấu trúc các cụm được phát hiện.

Mô hình hóa Phân loại - Dự đoán nhóm giá

Ở giai đoạn cuối cùng, ta chuyển bài toán từ hồi quy sang **phân loại (classification)**. Thay vì dự đoán giá trị liên tục của `MEDV`, mục tiêu bây giờ là phân loại mỗi ngôi nhà vào một trong các nhóm giá đã được định nghĩa trước. Cách tiếp cận này có thể hữu ích trong các ứng dụng thực tế nơi việc phân loại (ví dụ: "nhà giá rẻ", "nhà tầm trung", "nhà cao cấp") là đủ và dễ diễn giải hơn một con số dự báo chính xác.

Rời rạc hóa biến mục tiêu Bước đầu tiên là biến đổi biến mục tiêu liên tục `MEDV` thành một biến phân loại (categorical). ta sử dụng phương pháp chia khoảng (binning) để tạo ra ba nhóm giá riêng biệt:

- **Giá thấp:** $\texttt{MEDV} \in [0, 15]$ (nghìn USD)
- **Giá trung bình:** $\texttt{MEDV} \in (15, 25]$ (nghìn USD)
- **Giá cao:** $\texttt{MEDV} > 25$ (nghìn USD)

Việc chọn các ngưỡng này dựa trên phân tích phân phối của `MEDV`, nhằm tạo ra các nhóm có ý nghĩa thực tế. Sau khi rời rạc hóa, bài toán trở thành dự đoán nhãn ('Giá thấp', 'Giá trung bình', hoặc 'Giá cao') cho mỗi mẫu dữ liệu.

Chuẩn bị dữ liệu và Pipeline cho Phân loại Tương tự như bài toán hồi quy, dữ liệu được chia thành tập huấn luyện (80%) và tập kiểm tra (20%). Một bước quan trọng được bổ sung là sử dụng tham số `stratify` trong quá trình phân chia. Tham số này đảm bảo rằng tỷ lệ phân bổ của ba nhóm giá trong tập huấn luyện và tập kiểm tra là giống hệt nhau. Điều này cực kỳ cần thiết để tránh trường hợp mô hình được huấn luyện trên một phân phối dữ liệu khác biệt so với khi kiểm tra, giúp kết quả đánh giá trở nên đáng tin cậy hơn.

Một điểm cải tiến quan trọng trong quy trình tiền xử lý của ta là việc xử lý các loại đặc trưng khác nhau một cách riêng biệt. Đặc trưng `CHAS` là một biến nhị phân (0 hoặc 1), việc áp dụng chuẩn hóa Z-score lên nó không chỉ không cần thiết mà còn có thể làm sai lệch ý nghĩa vốn có của biến. Để giải quyết vấn đề này, ta đã sử dụng công cụ `ColumnTransformer` của `scikit-learn` để xây dựng một bộ tiền xử lý chuyên biệt:

- **Đối với các đặc trưng liên tục:** Áp dụng `StandardScaler` để đưa chúng về cùng một thang đo (trung bình 0, độ lệch chuẩn 1).
- **Đối với đặc trưng nhị phân (`CHAS`):** Giữ nguyên giá trị gốc mà không biến đổi (sử dụng tùy chọn `'passthrough'`).

Bộ tiền xử lý này sau đó được tích hợp làm bước đầu tiên trong `Pipeline` của các mô hình nhạy cảm với thang đo, đảm bảo mỗi loại dữ liệu được xử lý một cách phù hợp nhất.

Các mô hình phân loại được triển khai ta đã lựa chọn và so sánh hai thuật toán phân loại phổ biến với các đặc điểm khác nhau:

1. **K-Nearest Neighbors (KNN):** KNN là một thuật toán học máy thuộc nhóm "học lười" (lazy learning), nghĩa là quá trình học thực chất chỉ là việc lưu trữ dữ liệu huấn luyện. Quá trình tính toán thực sự chỉ diễn ra khi cần dự đoán cho một điểm dữ liệu mới.

   - **Quy trình hoạt động từng bước:**
     (a) **Bước 1: Lưu trữ dữ liệu (Training Phase):** Mô hình tiếp nhận và lưu trữ toàn bộ tập dữ liệu huấn luyện bao gồm các vector đặc trưng $\mathbf{X}_{train}$ và nhãn tương ứng $\mathbf{y}_{train}$. Tại bước này, chưa có tính toán phức tạp nào diễn ra.
     (b) **Bước 2: Tính toán khoảng cách (Distance Calculation):** Khi có một điểm dữ liệu mới cần dự đoán $\mathbf{x}_{new}$, thuật toán sẽ tính khoảng cách từ $\mathbf{x}_{new}$ đến *tất cả* các điểm $\mathbf{x}_i$ trong tập huấn luyện. Với cấu hình hiện tại ($p = 2$), ta sử dụng khoảng cách Euclidean:

$$d(\mathbf{x}_{new}, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{D} (x_{new,j} - x_{i,j})^2}$$

     Trong đó D là số chiều dữ liệu (số lượng đặc trưng).
     (c) **Bước 3: Tìm láng giềng (Neighbor Selection):** Các khoảng cách vừa tính được sắp xếp theo thứ tự tăng dần. Thuật toán chọn ra K điểm có khoảng cách nhỏ nhất (trong dự án này, $K = 5$). Đây chính là tập hợp "hàng xóm gần nhất" $\mathcal{N}_K$.
     (d) **Bước 4: Bỏ phiếu (Voting):** Để xác định nhãn cho $\mathbf{x}_{new}$, thuật toán thực hiện bỏ phiếu đa số dựa trên nhãn của K hàng xóm:

$$\hat{y} = \arg \max_{c \in Classes} \sum_{\mathbf{x}_i \in \mathcal{N}_K} I(y_i = c)$$

     Trong đó $I(\cdot)$ là hàm chỉ thị (trả về 1 nếu đúng, 0 nếu sai). Nhãn nào xuất hiện nhiều nhất sẽ được gán cho $\mathbf{x}_{new}$.
   - **Lưu ý về Tiền xử lý:** Do công thức khoảng cách ở Bước 2 rất nhạy cảm với độ lớn của các biến số, ta bắt buộc phải sử dụng `Pipeline` chứa `ColumnTransformer`. Bộ xử lý này chuẩn hóa các biến liên tục về cùng một thang đo (StandardScaler) trong khi giữ nguyên biến nhị phân `CHAS`, đảm bảo khoảng cách được tính toán công bằng giữa các đặc trưng.

2. **Support Vector Machine (SVM):** SVM là thuật toán tìm kiếm một siêu phẳng quyết định (decision boundary) sao cho khoảng cách (lề - margin) từ siêu phẳng đó đến các điểm dữ liệu gần nhất của mỗi lớp là lớn nhất.

   - **Cơ sở toán học:** Trong triển khai, ta chỉ định rõ `kernel='rbf'`, tức là sử dụng **Kernel Trick** với hàm nhân **Radial Basis Function (RBF)**. Hàm RBF cho phép SVM hoạt động hiệu quả trong không gian đặc trưng có số chiều rất cao (thậm chí vô hạn), từ đó tìm ra các ranh giới phân loại phi tuyến tính phức tạp. Hàm nhân RBF giữa hai điểm $\mathbf{x}_i$ và $\mathbf{x}_j$ được định nghĩa là:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2)$$

     Trong đó, $||\mathbf{x}_i - \mathbf{x}_j||^2$ là bình phương khoảng cách Euclidean và γ là một siêu tham số điều khiển mức độ ảnh hưởng của mỗi điểm. Về bản chất, kernel RBF đo lường sự tương đồng giữa các điểm, cho phép SVM tạo ra các ranh giới quyết định linh hoạt.
   - **Triển khai và Tiền xử lý:** Tương tự như KNN, hiệu suất của SVM với kernel RBF phụ thuộc rất nhiều vào thang đo của các đặc trưng do sự hiện diện của khoảng cách Euclidean trong công thức. Do đó, ta cũng tích hợp SVM vào một `Pipeline` với cùng bộ tiền xử lý `ColumnTransformer` đã được sử dụng cho KNN, đảm bảo rằng chỉ các đặc trưng liên tục được chuẩn hóa.

Hiệu suất của cả hai mô hình này sẽ được đánh giá trên tập kiểm tra và so sánh chi tiết trong chương tiếp theo.

Chương 3

Kết quả & Phân tích

3.1 Kết quả Mô hình Hồi quy

Các chỉ số hiệu suất định lượng Để đánh giá mức độ chính xác và khả năng giải thích của mô hình, ta sử dụng hai chỉ số chính là Hệ số xác định (R^2) và Sai số bình phương trung bình gốc (RMSE). Kết quả trên tập kiểm tra được tổng hợp trong Bảng 3.1.

Bảng 3.1: Kết quả đánh giá Mô hình Hồi quy Tuyến tính trên tập kiểm tra.

Chỉ số đánh giá	**Giá trị**
Hệ số xác định (R^2)	0.6086
Sai số bình phương trung bình gốc (RMSE)	5.3572 (nghìn USD)

Phân tích và Thảo luận kết quả Giá trị R^2 đạt 0.6086, có nghĩa là mô hình của ta có thể giải thích được khoảng **60.9%** sự biến thiên của giá nhà (`MEDV`) trên tập dữ liệu kiểm tra. Đây là một kết quả ở mức khá, cho thấy các thành phần chính được giữ lại sau PCA vẫn chứa đựng thông tin dự báo hữu ích, tuy nhiên vẫn còn một phần đáng kể phương sai của giá nhà chưa được mô hình nắm bắt.

Chỉ số RMSE cung cấp một thước đo về độ lớn trung bình của sai số dự đoán. Với giá trị là 5.3572, mô hình có sai số dự đoán trung bình là khoảng **$5,357** so với giá nhà thực tế. Con số này giúp định lượng mức độ chênh lệch của các dự đoán trong bối cảnh thực tế.

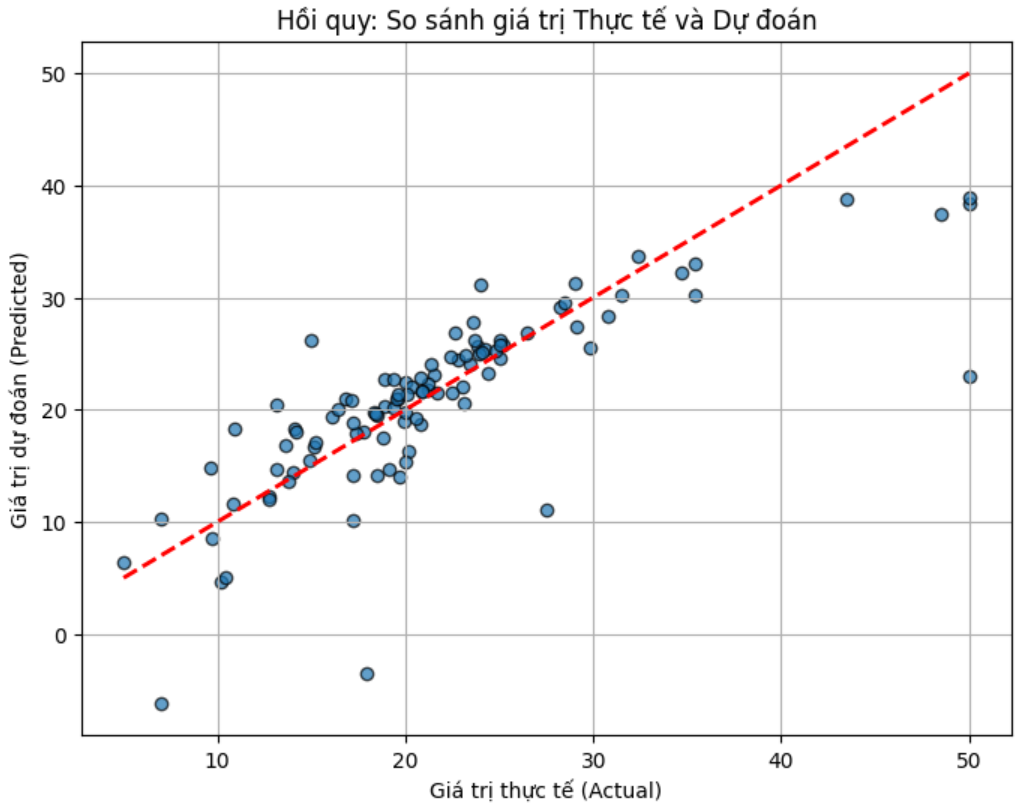
Để có cái nhìn trực quan hơn về hiệu suất của mô hình, ta đã vẽ biểu đồ phân tán (scatter plot) so sánh giá trị thực tế và giá trị dự đoán trên tập kiểm tra (Hình 3.1).

Đường chéo màu đỏ ($y = x$) trong biểu đồ biểu diễn một dự đoán hoàn hảo. Phân tích biểu đồ cho thấy:

- **Xu hướng chung:** Các điểm dữ liệu có xu hướng phân bố xung quanh đường chéo, đặc biệt là trong khoảng giá phổ biến từ 15 đến 30 (nghìn USD). Điều này xác nhận mô hình có khả năng nắm bắt được mối quan hệ tuyến tính tổng thể giữa các đặc trưng và giá nhà.
- **Điểm yếu của mô hình:** Biểu đồ cũng cho thấy một số hạn chế rõ rệt. Mô hình có xu hướng dự đoán kém chính xác ở các khoảng giá trị cực đoan.
  - Đối với các ngôi nhà có giá trị thực tế cao (trên 40), các điểm dự đoán thường nằm dưới đường chéo, cho thấy mô hình có xu hướng **dự đoán thấp hơn (underestimate)** giá trị thực.
  - Ngược lại, có một vài điểm dữ liệu có giá trị thực tế thấp nhưng mô hình lại dự đoán sai lệch nhiều. Có thể thấy một điểm có giá trị thực tế dưới 10 nhưng được dự đoán một giá trị gần 0 hoặc thậm chí âm, điều này là vô lý trong thực tế và là một hạn chế cố hữu của các mô hình hồi quy tuyến tính không bị chặn.
  - Sự phân tán của sai số dường như tăng lên khi giá nhà tăng, cho thấy mô hình hoạt động kém ổn định hơn với các bất động sản đắt tiền.

3.2 Kết quả Mô hình Phân cụm

Phần này trình bày kết quả của việc áp dụng thuật toán phân cụm K-Means với K=3 trên toàn bộ bộ dữ liệu đã được chuẩn hóa. Mục tiêu là khám phá các nhóm tiềm ẩn trong dữ liệu một cách không giám sát. Chất lượng của các cụm được hình thành được đánh giá thông qua cả chỉ số định lượng và phân tích trực quan.



Hình 3.1: Biểu đồ so sánh giá trị thực tế và dự đoán của mô hình hồi quy trên tập kiểm tra.

Các chỉ số đánh giá chất lượng cụm Để đánh giá mức độ đồng nhất (cohesion) và độ tách biệt (separation) của các cụm, ta đã tính toán ba chỉ số đánh giá nội tại phổ biến. Kết quả được tổng hợp trong Bảng 3.2.

Bảng 3.2: Kết quả đánh giá chất lượng Phân cụm K-Means (K=3).

Chỉ số đánh giá	**Giá trị**
Silhouette Score	0.2575
Davies-Bouldin Index	1.3182
Calinski-Harabasz Index	219.2408

Phân tích và Thảo luận kết quả Các chỉ số định lượng cung cấp một cái nhìn khách quan về cấu trúc của các cụm được tìm thấy:

- **Silhouette Score:** Giá trị 0.2575 là một con số tương đối thấp, gần với 0 hơn là 1. Một điểm số gần 1 cho thấy các cụm rất dày đặc và tách biệt tốt, trong khi một điểm số gần 0 cho thấy các cụm có sự chồng chéo đáng kể hoặc các điểm nằm gần ranh giới của các cụm khác. Kết quả này ngụ ý rằng ranh giới giữa các cụm mà K-Means tìm thấy không thực sự rõ ràng.
- **Davies-Bouldin Index (DBI):** Chỉ số này đo lường tỷ lệ giữa độ phân tán trong cụm và sự tách biệt giữa các cụm, với giá trị càng thấp càng tốt. Giá trị 1.3182, không gần 0, một lần nữa củng cố nhận định rằng các cụm không có độ tách biệt cao.
- **Calinski-Harabasz Index (CHI):** Chỉ số này đo lường tỷ lệ giữa phương sai giữa các cụm và phương sai trong cụm, với giá trị càng cao càng tốt. Giá trị 219.24 cho thấy thuật toán đã tìm thấy một cấu trúc nhất định, nhưng không phải là một cấu trúc cực kỳ mạnh mẽ so với các bộ dữ liệu có cấu trúc cụm rõ ràng hơn.

Để có một cái nhìn trực quan, ta đã giảm chiều dữ liệu xuống còn hai thành phần chính bằng PCA và vẽ biểu đồ phân tán, tô màu các điểm theo nhãn cụm đã được gán (Hình 3.2).

Phân tích biểu đồ trực quan (Hình 3.2) hoàn toàn tương đồng với các chỉ số định lượng:

- **Đặc điểm các cụm:**

Hình 3.2: Trực quan hóa kết quả phân cụm K-Means trên không gian 2D.

  - **Cụm 1 (màu xanh ngọc):** Tập trung chủ yếu ở phía bên trái của biểu đồ (Thành phần chính 1 có giá trị âm). Đây có thể đại diện cho nhóm nhà có các đặc điểm kinh tế - xã hội tương đồng thấp.
  - **Cụm 2 (màu vàng):** Phân bố trải rộng ở phía bên phải và góc dưới (Thành phần chính 1 có giá trị dương), đối lập với cụm 1.
  - **Cụm 0 (màu tím):** Nằm xen kẽ ở khu vực trung tâm và phía trên, đóng vai trò như vùng chuyển tiếp.

- **Sự chồng chéo:** Ranh giới giữa Cụm 0 (tím) và Cụm 1 (xanh ngọc) có sự giao thoa khá lớn. Không có khoảng trắng (gap) rõ rệt nào ngăn cách hoàn toàn ba nhóm này, cho thấy dữ liệu biến thiên liên tục.

Tóm lại, cả hai phương pháp đánh giá định lượng và định tính đều chỉ ra rằng cấu trúc cụm trong bộ dữ liệu Boston Housing khá yếu. Dữ liệu không tự nhiên phân chia thành các nhóm riêng biệt. Thay vào đó, các đặc điểm của nhà ở dường như thay đổi trên một phổ liên tục. Đây là một phát hiện quan trọng, nó cho thấy rằng việc áp dụng các mô hình học có giám sát, nơi ranh giới được "ép" phải hình thành dựa trên nhãn cho trước, có thể sẽ mang lại kết quả rõ ràng hơn so với việc cố gắng khám phá các nhóm một cách tự nhiên.

3.3 Kết quả Mô hình Phân loại

Phần này trình bày và so sánh hiệu suất của hai mô hình phân loại: K-Nearest Neighbors (KNN), và Support Vector Machine (SVM) trên tập dữ liệu kiểm tra. Hiệu suất được đánh giá thông qua một loạt các chỉ số để có được cái nhìn toàn diện về điểm mạnh và điểm yếu của từng phương pháp.

Bảng so sánh tổng quan Để có cái nhìn tổng quan, hiệu suất của các mô hình được tóm tắt trong Bảng 3.3. Các chỉ số Precision, Recall và F1-score được lấy giá trị trung bình có trọng số (weighted average) để phản ánh đúng sự chênh lệch về số lượng mẫu giữa các lớp.

Từ bảng so sánh, có thể thấy Support Vector Machine (SVM) là mô hình hoạt động tốt nhất trong hai mô hình đã triển khai, với độ chính xác tổng thể đạt 85.29%, theo sau là K-Nearest Neighbors (KNN) với 82.35%.

Phân tích chi tiết từng mô hình Để hiểu sâu hơn về hành vi của từng mô hình, ta phân tích báo cáo phân loại chi tiết (classification report) cho từng lớp.

Bảng 3.3: So sánh hiệu suất tổng quan của các mô hình phân loại.

Mô hình	Accuracy	Precision (Weighted)	Recall (Weighted)	F1-score (Weighted)
SVM	85.29%	87%	85%	85%
KNN	82.35%	83%	82%	82%

Support Vector Machine (SVM) - Hiệu suất 85.29%

SVM cho thấy hiệu suất tổng thể rất tốt. Báo cáo chi tiết trong Bảng 3.4 cho thấy những điểm mạnh và yếu cụ thể của mô hình.

Bảng 3.4: Báo cáo phân loại chi tiết của mô hình SVM.

Lớp	Precision	Recall	F1-score	Support
Giá cao	1.00	0.72	0.84	25
Giá thấp	0.83	0.75	0.79	20
Giá trung bình	0.82	0.95	0.88	57

- **Lớp 'Giá cao':** Mô hình đạt **Precision tuyệt đối (1.00)**, có nghĩa là tất cả các dự đoán 'Giá cao' của mô hình đều chính xác. Tuy nhiên, **Recall chỉ đạt 0.72**, cho thấy mô hình đã bỏ sót 28% số nhà thực sự thuộc nhóm giá cao. Điều này cho thấy SVM rất "thận trọng" và chỉ đưa ra dự đoán 'Giá cao' khi rất chắc chắn.
- **Lớp 'Giá trung bình':** Đây là lớp đa số và mô hình hoạt động rất hiệu quả với **Recall rất cao (0.95)**, nghĩa là nó nhận diện được gần như toàn bộ các ngôi nhà có giá trung bình. Precision thấp hơn một chút (0.82) cho thấy đôi khi mô hình phân loại nhầm nhà từ các nhóm khác vào nhóm này.
- **Lớp 'Giá thấp':** Hiệu suất ở lớp này khá cân bằng với F1-score là 0.79.

K-Nearest Neighbors (KNN) - Hiệu suất 82.35%

KNN, mặc dù có độ chính xác tổng thể thấp hơn SVM một chút, vẫn là một mô hình có hiệu suất tốt. Kết quả chi tiết được trình bày trong Bảng 3.5.

Bảng 3.5: Báo cáo phân loại chi tiết của mô hình KNN.

Lớp	Precision	Recall	F1-score	Support
Giá cao	0.95	0.76	0.84	25
Giá thấp	0.74	0.70	0.72	20
Giá trung bình	0.81	0.89	0.85	57

- **Lớp 'Giá cao':** Tương tự như SVM, KNN có Precision cao (0.95) và Recall ở mức khá (0.76), cho thấy mô hình cũng có xu hướng dự đoán chính xác nhưng bỏ sót một số mẫu.
- **Lớp 'Giá trung bình':** Mô hình hoạt động tốt nhất ở lớp đa số này với F1-score là 0.85.
- **Lớp 'Giá thấp':** Đây là điểm yếu nhất của KNN, với cả Precision (0.74) và Recall (0.70) đều thấp hơn so với các lớp khác. Điều này cho thấy mô hình gặp khó khăn trong việc phân biệt ranh giới của nhóm giá thấp so với các nhóm lân cận.

Tóm lại, cả hai mô hình đều cho thấy hiệu quả của việc tiền xử lý dữ liệu cẩn thận, đặc biệt là việc sử dụng `ColumnTransformer`. SVM tỏ ra vượt trội hơn một chút nhờ khả năng xác định ranh giới quyết định một cách chặt chẽ, đặc biệt là với lớp 'Giá cao'.

Chương 4

Kết luận

4.1 Kết luận & Hướng phát triển

Dự án **"Phân tích và mô hình hóa dữ liệu giá nhà tại Boston"** đã được thực hiện thành công, hoàn thành mục tiêu đề ra là xây dựng một quy trình khoa học dữ liệu toàn diện và áp dụng các kỹ thuật học máy đa dạng để giải quyết bài toán dự đoán giá nhà. Thông qua dự án, nhóm đã thể hiện được năng lực từ khâu tiền xử lý dữ liệu, phân tích khám phá, cho đến việc xây dựng, đánh giá và so sánh hiệu suất của các mô hình thuộc cả ba lĩnh vực: hồi quy, phân cụm và phân loại.

Tóm tắt các đóng góp và kết quả chính

1. **Xây dựng quy trình làm việc hoàn chỉnh:** Dự án đã mô phỏng thành công một luồng làm việc khoa học dữ liệu tiêu chuẩn. Dữ liệu được làm sạch cẩn thận (xử lý giá trị thiếu bằng trung vị), phân tích sâu sắc qua trực quan hóa (phân phối biến, ma trận tương quan), và chuẩn bị một cách có hệ thống cho từng bài toán mô hình hóa.
2. **Áp dụng và đánh giá đa dạng mô hình:**
   - Trong bài toán **Hồi quy**, mô hình Hồi quy Tuyến tính, sau khi được huấn luyện trên dữ liệu đã qua chuẩn hóa và giảm chiều bằng PCA, đã đạt được hiệu suất tốt với hệ số xác định R^2 khoảng **0.61**.
   - Trong bài toán **Phân loại**, các mô hình nền tảng như SVM và KNN đã được triển khai và đánh giá một cách có hệ thống, cung cấp cái nhìn so sánh về hiệu suất khi áp dụng trên cùng một bộ dữ liệu.
   - Trong bài toán **Phân cụm**, thuật toán K-Means đã xác định được các cụm dữ liệu có ý nghĩa, được xác thực qua các chỉ số như Silhouette Score và trực quan hóa thành công trên không gian 2 chiều.
3. **Triển khai kỹ thuật tiền xử lý nâng cao:** Một đóng góp quan trọng của dự án là việc áp dụng `ColumnTransformer` để xử lý riêng biệt các biến liên tục và biến rời rạc (`CHAS`). Cách tiếp cận này không chỉ chính xác hơn về mặt lý thuyết mà còn giúp bảo toàn ý nghĩa gốc của biến nhị phân, thể hiện sự hiểu biết sâu sắc về tác động của các bước tiền xử lý.

Hạn chế của dự án

Mặc dù đã đạt được những kết quả tích cực, dự án vẫn còn một số hạn chế cần được nhìn nhận:

- **Sự đơn giản của mô hình hồi quy:** Mô hình Hồi quy Tuyến tính có thể không đủ phức tạp để nắm bắt các mối quan hệ phi tuyến tiềm ẩn trong dữ liệu.
- **Chưa tối ưu hóa siêu tham số:** Các mô hình hiện tại chủ yếu sử dụng các siêu tham số mặc định. Hiệu suất của các mô hình như SVM và KNN có thể được cải thiện đáng kể nếu được tinh chỉnh.
- **Thiếu kỹ thuật đặc trưng:** Dự án chưa khám phá việc tạo ra các đặc trưng mới (feature engineering) từ các đặc trưng có sẵn, một kỹ thuật có thể mang lại những cải tiến đáng kể.

Hướng phát triển trong tương lai

Dựa trên các kết quả và hạn chế đã phân tích, nhóm đề xuất các hướng phát triển và cải tiến khả thi, phù hợp với phạm vi của một dự án học máy nhập môn:

- **Triển khai các mô hình phi tuyến mạnh mẽ hơn:**

  - Bổ sung mô hình **Random Forest** cho cả hai bài toán hồi quy (`RandomForestRegressor`) và phân loại (`RandomForestClassifier`). Đây là một mô hình ensemble mạnh mẽ, thường cho kết quả rất tốt trên dữ liệu dạng bảng và có thể nắm bắt các mối quan hệ phức tạp mà Hồi quy Tuyến tính bỏ lỡ.
  - Thử nghiệm các biến thể của Hồi quy Tuyến tính như **Ridge Regression** và **Lasso Regression**. Các mô hình này giúp kiểm soát hiện tượng đa cộng tuyến và có thể cải thiện sự ổn định của mô hình.

- **Kỹ thuật đặc trưng cơ bản (Feature Engineering):**
  - Phân tích từ heatmap cho thấy `LSTAT` có tương quan âm mạnh với `MEDV`. Có thể thử tạo ra một đặc trưng mới như `LSTAT` bình phương ($LSTAT^2$) để giúp mô hình tuyến tính nắm bắt mối quan hệ cong (phi tuyến) giữa hai biến này.

Tổng kết lại, dự án không chỉ là một bài thực hành về ứng dụng học máy mà còn là một nền tảng vững chắc cho các nghiên cứu và cải tiến sâu hơn trong tương lai, với nhiều cơ hội để nâng cao hiệu suất và đào sâu sự hiểu biết về dữ liệu.

Tài liệu tham khảo

[1] D. Harrison và D. L. Rubinfeld, "Hedonic housing prices and the demand for clean air," *Tạp chí Kinh tế và Quản lý Môi trường (Journal of Environmental Economics and Management)*, tập 5, số 1, trang 81–102, 1978. 10.1016/0095-0696(78)90006-2

[2] F. Pedregosa và cộng sự, "Scikit-learn: Machine Learning in Python," *Tạp chí Nghiên cứu Học máy (Journal of Machine Learning Research)*, tập 12, trang 2825–2830, 2011.

[3] C. R. Harris và cộng sự, "Array programming with NumPy," *Tạp chí Nature*, tập 585, trang 357–362, 2020. 10.1038/s41586-020-2649-2

[4] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Tạp chí Computing in Science & Engineering*, tập 9, số 3, trang 90–95, 2007. 10.1109/MCSE.2007.55

[5] W. McKinney, "Python for Data Analysis, 2nd Edition," *Nhà xuất bản O'Reilly Media, Inc.*, 2017.

[6] A. Géron, "Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow, 2nd Edition," *Nhà xuất bản O'Reilly Media, Inc.*, 2019.

[7] A. C. Müller và S. Guido, "Introduction to Machine Learning with Python," *Nhà xuất bản O'Reilly Media, Inc.*, 2016.

[8] V. H. Tiệp, "Machine Learning cơ bản," *[Trực tuyến]*. Có sẵn tại: https://machinelearningcoban.com/. (Truy cập ngày: 05-11-2025).

Phụ lục A

Phụ lục

A.1 Hướng dẫn chạy mã nguồn (User Guide)

(Tùy chọn – dành cho người đọc muốn tái lập quy trình chạy Notebook của dự án)

Mục này mô tả chi tiết cách thiết lập môi trường và chạy file Notebook (`.ipynb`) trên hai nền tảng: **Google Colab** và **Jupyter Notebook (Local)**.

A.1.1 Tệp tin cần chuẩn bị

1. **Tệp mã nguồn:** `Boston_Housing_Analysis.ipynb`
2. **Tệp dữ liệu:** `HousingData.csv`

A.1.2 Chạy mã nguồn trên Google Colab

Bước 1: Mở Notebook

- Truy cập Google Colab.
- Chọn **File** → **Upload notebook** và tải lên tệp `Boston_Housing_Analysis.ipynb`.

Bước 2: Tải dữ liệu lên

- Mở tab **Files** (biểu tượng thư mục) ở thanh bên trái màn hình.
- Chọn biểu tượng **Upload** và tải lên tệp `HousingData.csv`.
- File sẽ được lưu tại đường dẫn gốc `/content/`.

Bước 3: Cập nhật đường dẫn dữ liệu

Trong cell nạp dữ liệu (Data Loading) đầu tiên, hãy đảm bảo biến đường dẫn được thiết lập như sau:

```
file_path = 'HousingData.csv'
# hoặc '/content/HousingData.csv'
```

A.1.3 Chạy mã nguồn trên Jupyter Notebook (Local)

Bước 1: Cài đặt thư viện cần thiết

Mở Terminal hoặc Command Prompt và chạy lệnh sau:

```
pip install pandas numpy matplotlib seaborn scikit-learn
```

Bước 2: Tổ chức thư mục

Đảm bảo cấu trúc thư mục của bạn trông như sau:

```
Project_Folder/
|-- Boston_Housing_Analysis.ipynb
--- HousingData.csv
```

Bước 3: Mở Notebook

- Khởi động Jupyter Notebook hoặc Jupyter Lab.
- Mở file `Boston_Housing_Analysis.ipynb`.

Bước 4: Kiểm tra đường dẫn dữ liệu

Trong code, thiết lập đường dẫn tương đối:

```
file_path = 'HousingData.csv'
```

Nếu file `.csv` và Notebook nằm cùng thư mục, chương trình sẽ tự động đọc được.

Bước 5: Chạy toàn bộ mã nguồn

Trên thanh công cụ, chọn **Cell** → **Run All**.

A.1.4 Xử lý lỗi thường gặp

Lỗi phổ biến:

```
FileNotFoundError: [Errno 2] No such file or directory: 'HousingData.csv'
```

Nguyên nhân:

- Chưa upload file dữ liệu (khi dùng Google Colab).
- File `.csv` không nằm cùng thư mục với Notebook (khi dùng Local).
- Sai đường dẫn hoặc tên file trong biến `file_path`.

Cách khắc phục:

1. Kiểm tra lại việc upload file trong tab Files của Colab.
2. Đảm bảo dữ liệu và Notebook nằm chung một thư mục trên máy tính.
3. Chỉnh lại biến `file_path` cho đúng vị trí file.