ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN CHẤT LƯỢNG CAO

ĐỒ ÁN THỰC HÀNH
TOÁN ỨNG DỤNG VÀ THỐNG KÊ – MTH00051

ĐỒ ÁN THỰC HÀNH SỐ 3: LINEAR REGRESSION

THẦY (CÔ) HƯỚNG DẪN

CÔ. PHAN THỊ PHƯƠNG UYÊN
THẦY. NGUYỄN NGỌC TOÀN
THẦY. VŨ QUỐC HOÀNG
THẦY. NGUYỄN VĂN QUANG HUY

—o0o—

THÔNG TIN SINH VIÊN THỰC HIỆN

HỌ VÀ TÊN: LÊ PHƯỚC PHÁT
LỚP: 22CLC10
MSSV: 22127322
EMAIL: lpphat22@clc.fitus.edu.vn

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 08 NĂM 2024

MỤC LỤC

NHẬN XÉT CỦA GIÁO VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Ngày …, tháng …, năm 2024
Giáo viên nhận xét và chấm điểm

LỜI CAM KẾT

Tôi cam đoan rằng nghiên cứu này là do tôi thực hiện, dưới sự giám sát và hướng dẫn của các thầy cô bộ môn **Toán Ứng Dụng và Thống Kê – MTH00051**: cô **Phan Thị Phương Uyên** và thầy **Nguyễn Ngọc Toàn**. Kết quả của nghiên cứu này là hợp pháp và chưa được công bố ở bất kỳ hình thức nào trước đây. Tất cả các tài liệu được sử dụng trong nghiên cứu này được tôi thu thập bằng cách tự mình và từ các nguồn khác nhau, và được liệt kê đầy đủ trong phần tài liệu tham khảo. Ngoài ra, chúng tôi cũng sử dụng kết quả của một số tác giả và tổ chức khác. Tất cả đều được trích dẫn đúng đắn. Trong trường hợp có phạm bản quyền, chúng tôi chịu trách nhiệm cho hành động đó. Do đó, **Trường Đại học Khoa học Tự nhiên TP.HCM** không chịu trách nhiệm về bất kỳ vi phạm bản quyền nào được thực hiện trong nghiên cứu của tôi.

NỘI DUNG BÁO CÁO ĐỒ ÁN

I. Đánh giá mức độ hoàn thành của đồ án

Yêu cầu	Chi tiết các yêu cầu thực hiện	Mức độ hoàn thành
1	Thực hiện phân tích khám phá dữ liệu (1 điểm).	100 %
2a	Mô hình sử dụng toàn bộ 5 đặc trưng (2 điểm).	100 %
2b	Xây dựng mô hình sử dụng duy nhất 1 đặc trưng và tìm mô hình cho kết quả tốt nhất (2 điểm).	100 %
2c	Xây dựng / Thiết kế m mô hình và tìm mô hình cho kết quả tốt nhất (2 điểm).	100 %
3	Viết báo cáo và nhận xét kết quả (1 + (5 + 1) + (m + 1)) mô hình (3 điểm).	100 %

II. Giới thiệu

Trong đồ án này, chúng ta sẽ dùng một thuật toán học có giám sát (supervised learning), là một loại của học máy mà thuật toán học từ dữ liệu được gán nhãn sẵn. Dữ liệu dán nhãn sẵn là bộ dữ liệu mà giá trị target cũng đã biết sẵn. Và nhiệm vụ của thuật toán học có giám sát sẽ phát triển một mô hình dự đoán (predictive model) dựa trên cả dữ liệu đầu vào và đầu ra.

- Classification (Phân loại): là bài toán phân loại dữ liệu, tức là thuật toán này sẽ dự đoán các lớp dữ liệu dựa trên các biến đầu vào độc lập (biến không phụ thuộc). Lớp phân loại sẽ là những giá trị rời rạc (discrete data) hoặc những giá trị phân loại (categorical data).
    - Ví dụ: bài toán phân loại chữ số viết tay, bài toán phân loại các loài hoa iris, …
- Regression (Hồi quy): là bài toán dự báo ra một hoặc nhiều số thực, tức là những biến đầu ra liên tục dựa trên những biến đầu vào độc lập.

- Ví dụ: bài toán dự báo giá nhà dựa trên tuổi thọ của nhà, khoảng cách từ đường chính, địa điểm, khu vực, …, hay bài toán dự đoán cân nặng hay bài toán dựa vào chiều cao, dự đoán doanh thu và lợi nhuận dựa vào chi phí sản xuất, …

1. Mô hình hồi quy tuyến tính (Linear Regression)

Hồi quy tuyến tính (Linear Regression) là một nhánh của thuật toán học máy (machine learning algorithm), đặc biệt là thuật toán học máy có giám sát (supervised machine-learning algorithm) rằng thuật toán này sẽ học tử bộ dữ liệu có dán nhãn (labelled dataset) và ánh xạ (map) những điểm dữ liệu lên những hàm tuyến tính tối ưu nhất, và những hàm tuyến tính này được dùng để dự đoán những bộ dữ liệu mới.

Tập dữ liệu huấn luyện (training set) là bộ dữ liệu theo cặp $\{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})\}$ với $x^{(i)} \in \mathcal{X}$ và $y^{(i)} \in \mathcal{Y}$. Trong đó:

- $x^{(i)}$: vector đầu vào chứa những biến độc lập đầu vào ("input" variables) với mỗi biến sẽ là một giá trị độc lập hay còn gọi là biến chứa những giá trị đặc trưng đầu vào ("input" features)
- $y^{(i)}$: biến phụ thuộc đầu ra ("output" variables) hay còn gọi là biến mục tiêu (target variables).
- $\mathcal{X}$: không gian của những biến độc lập đầu vào.
- $\mathcal{Y}$: không gian của những biến phụ thuộc đầu ra.
- Một cặp $(x^{(i)}, y^{(i)})$ là một tập thử huấn luyện thứ (i) trong bộ dữ liệu huấn luyện. Hay nói cách khác, (x, y) là một tập thử huấn luyện (training example).

Một hàm giả thuyết (hypothesis function) hay một hàm dự đoán (predictive function) là hàm $\hbar: \mathcal{X} \in \mathbb{R} \longrightarrow \mathcal{Y} \in \mathbb{R}$.

- Ví dụ: đối với bài toán dự đoán giá nhà, $\mathcal{X}$ là tập không gian chứa những dữ liệu đặc trưng về ngôi nhà (như tuổi thọ căn nhà, khoảng cách từ nhà đến trung tâm, …) và $\mathcal{Y}$ là tập chứa những giá nhà ứng với từng tập đặc trưng trong $\mathcal{X}$.
- Lưu ý:
  - Nếu $\mathcal{Y}$ là tập các giá trị liên tục thì đây sẽ là bài toán hồi quy (regression).
  - Nếu $\mathcal{Y}$ là tập các giá trị rời rạc thì đây sẽ là bài toán phân loại (classification).

Công thức hồi quy tuyến tính:

$$\hbar_\theta(x) = y^{(1)} = \theta_0 + \theta_1 x_1^{(1)} + \theta_2 x_2^{(1)} + \dots + \theta_d x_d^{(1)} = \theta^T . x^{(1)}$$

Trong đó:

$$\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{pmatrix} \text{được gọi là } weights\ (parameters) \text{ và } x^{(1)} = \begin{pmatrix} x_0^{(1)} \\ x_1^{(1)} \\ \vdots \\ x_d^{(1)} \end{pmatrix} = \begin{pmatrix} 1 \\ x_1^{(1)} \\ \vdots \\ x_d^{(1)} \end{pmatrix}$$

Công thức tổng quát với $x_0 = 1$ thì ta có:

$$\mathcal{y} \approx h(x) = \sum_{i=0}^{d} \theta_i x_i = \theta^T . X$$

Trong đó:

- $\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{pmatrix}$: bộ trọng số để tối ưu hóa hàm hồi quy (mô hình - model) có dạng ((d + 1) x 1) với d là số đặc trưng của mỗi $x^{(i)}$ (vì $x_0 = 1$ nên ta có θ_0 là $bias$).
- X: ma trận thiết kế gồm các training examples $x^{(i)}$ sau khi đã được tiền xử lý có dạng (n x (d + 1)) với n là số training examples có trong training dataset và d + 1 là số cột của X với d là số đặc trưng ban đầu của mỗi $x^{(i)}$ (vì $x_0 = 1$ nên ta phải thêm vào vector 1 vào đầu cột ma trận nhằm tính intercept term ➔ đây còn gọi là phương pháp bias trick[1]) .
- $\mathcal{y}$: bộ dữ liệu đã gắn nhãn hay vector chứa những biến phụ thuộc đầu ra có dạng (n x 1) với n là số training examples có trong training dataset.

Khi chúng tra mới bắt đầu huấn luyện mô hình thì ta sẽ được bộ dữ liệu huấn luyện gồm X và $\mathcal{y}$. Mô hình này sẽ cố gắng huấn luyện bằng cách tìm ra đường hồi quy tốt nhất nhằm ánh xạ những điểm dữ liệu lên đường tuyến tính để dự đoán giá trị y dựa trên những dữ liệu X. Để mô hình tìm được đường hồi quy tuyến tính tốt nhất, chúng ta cần phải điều chỉnh bộ tham số θ sao cho h(x) $\approx$ y.

Bằng cách đạt được mô hình hồi quy tuyến tính tots nhất, mục tiêu của mô hình là dự đoán giá trị $\mathcal{y}$ bằng cách so sánh độ lỗi chênh lệch giữa giá trị thực tế (ground-truth) và giá trị dự đoán (predicted scores) là nhỏ nhất. Vì thế, nó rất quan trọng trong việc điều chỉnh các giá trị của bộ tham số θ nhằm tối thiểu hóa độ lỗi giữa giá trị dự đoán và giá trị thực tế. Tuy nhiên, trong thực tế, có rất nhiều hàm đo và tính đoán độ lỗi của các mô hình hồi quy như RMSE (Root Mean Squared Error), MSE (Mean Squared Error), …. Trong khuôn khổ đồ án này, em sẽ sử dụng hàm đo độ lỗi là MAE (Mean Absolute Error) dùng để ước lượng trung bình sai số (độ lỗi) tuyệt đối.

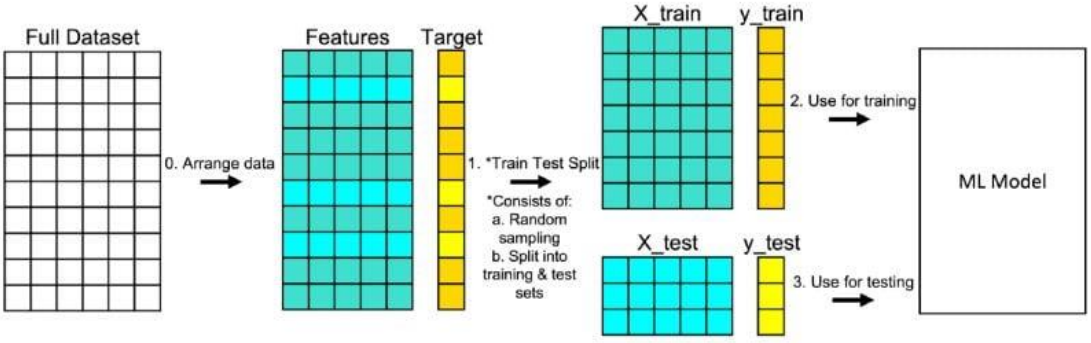
$$J = MAE = \frac{1}{n}\sum_{n}^{i=1}|y_i - \hat{y}_i|$$

2. Phương pháp chia K-Fold (K-fold Cross Validation)
   Trong lĩnh vực học máy (Machine Learning), việc đánh giá độ chính xác của mô hình là một bước quan trọng để đảm bảo mô hình hoạt động tốt trên dữ liệu mới, chưa từng thấy. Để làm được điều này, chúng ta phải đảm bảo mô hình nhận được các mẫu dữ liệu chính xác và ít bị nhiễu đi. Một trong những kỹ thuật phổ biến nhất để thực hiện điều này là K-Fold Cross Validation, giúp chúng ta đánh giá mô hình một cách khách quan và giảm thiểu vấn đề overfitting.

   - *Phương pháp chia tách dữ liệu Train-Test Split*

     Mục tiêu quan trọng khi phát triển bất kỳ mô hình học máy nào là làm thế nào để xây dựng mô hình biểu hiện tốt trên tập dữ liệu mới thực tế. Nhưng nếu chúng ta không có dữ liệu mới để kiểm tra và bộ dữ liệu hiện tại đủ lớn, cách tốt nhất để phân chia bộ dữ liệu ban đầu là dùng phương pháp train-test split.

     Train test split là một quá trình điều chỉnh mô hình cho phép chúng ta có thể mô phỏng mô hình sẽ hoạt động như thế nào trên những bộ dữ liệu mới hay chưa được tìm thấy.



Hình 1. Train test split procedure. | Image: Michael Galarnyk

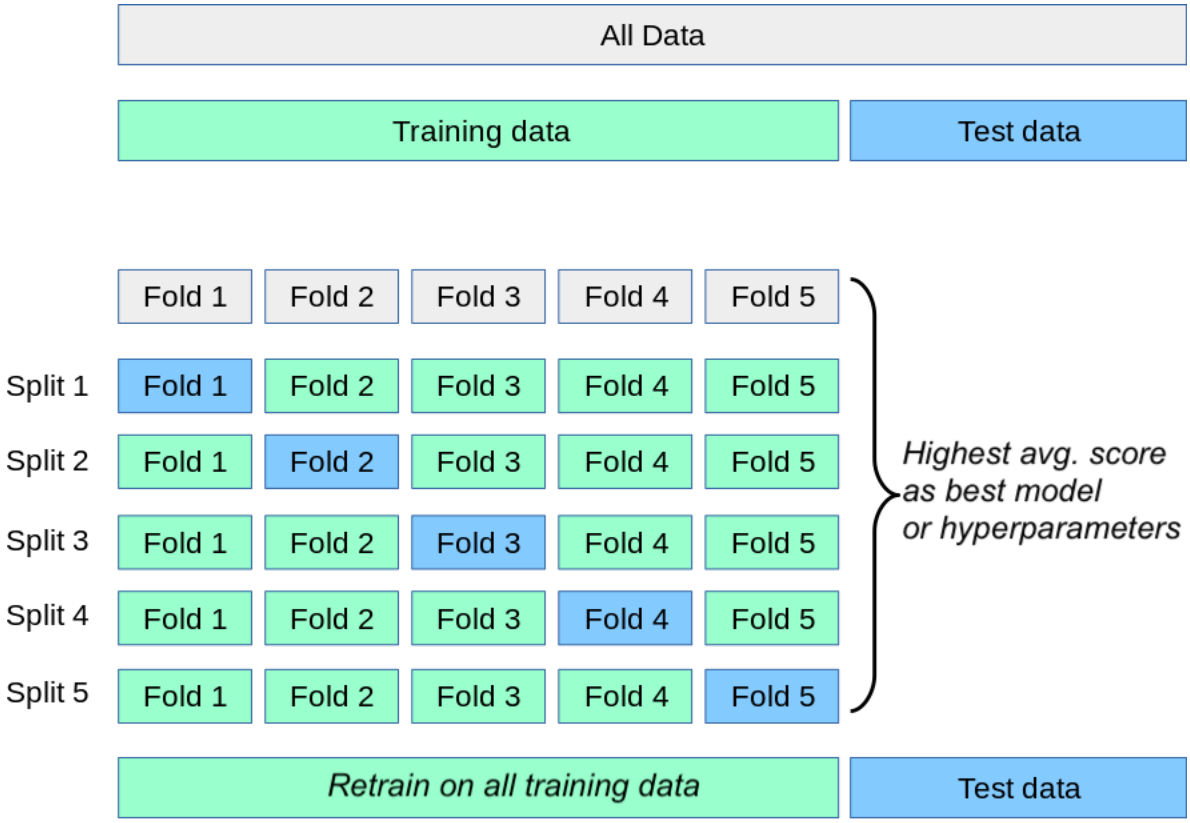
Chúng ta sẽ chia bộ dữ liệu ban đầu thành hai tập dữ liệu: tập huấn luyện (training set) và tập kiểm thử (testing set), bao gồm mỗi mẫu sẽ được sắp đặt ngẫu nhiên không có sự thay thế với 75% hay 80% cho bộ huấn luyện và 25% hay 20 % còn lại cho bộ kiểm thử.

Sau đó, chúng ta sẽ huấn luyện mô hình dựa trên tập huấn luyện để tìm ra bộ tham số tối ưu nhất cho mô hình và ta sẽ dùng mô hình đã huấn luyện được kiểm thử trên tập kiểm thử và đánh giá độ lỗi.

- *K-Fold Cross Validation (Xác thực chéo K-Fold)*

  Thông thường, ngoài tập train và test, chúng ta sẽ tách tập train thành hai tập nhỏ train và validation, trong đó tập train sẽ gồm 80% dữ liệu của tập train ban đầu (là dữ liệu để train model thực sự) và tập validation gồm 20% dữ liệu còn lại của tập train ban đầu (làm dữ liệu để kiểm tra model trong quá trình train). Việc chia này sẽ gặp bất lợi làm cho mô hình hoạt động cực kỳ kém khi bộ dữ liệu ban đầu quá nhỏ. Vì một số điểm dữ liệu có ích trong quá trình huấn luyện sẽ bị đưa vào tập validation, tập test và mô hình sẽ không có cơ hội học điểm dữ liệu đó. Thậm chí, có một vài đặc trưng chỉ có trong tập validation và test trong khi đó tập train lại không có (do việc chia dữ liệu cho tập train và validation là hoàn toàn ngẫu nhiên) nên dẫn đến mô hình không thể học các đặc trưng đó mà khi kiểm thử và đánh giá mô hình sẽ làm tăng độ lỗi lên mức đáng kể.
  Để giải quyết vấn đề đó, ta sẽ dùng phương pháp K-Fold Cross Validation để chia tách tập train ban đầu thành hai tập train và validation để đánh giá mô hình.
  - *Ý tưởng thực hiện:* Phương pháp này sẽ chia tập train thành k phần (k folds). Mỗi lần lặp, một phần (fold) sẽ được sử dụng để làm tập đánh giá (validation set), trong khi k – 1 phần còn lại được sử dụng để làm tập huấn luyện (training set). Quá trình này được lặp đi lắp lại k lần, và kết quả cuốn cùng là giá trị trung bình của các độ lỗi đánh giá trên mỗi lần lặp.
  - *Các bước thực hiện*
    - Xáo trộn tập train một cách ngẫu nhiên.
    - Chia tập train thành k nhóm
    - Với mỗi nhóm:
      - Sử dụng nhóm hiện tại để đánh giá hiệu quả mô hình
      - Các nhóm còn lại được sử dụng để huấn luyện mô hình
      - Huấn luyện mô hình
      - Đánh giá và sau đó hủy mô hình
    - Tổng hợp hiệu quả của mô hình dựa từ các số liệu đánh giá bằng cách lấy trung bình độ lỗi k nhóm.

  Tuy nhiên, trong thực tế có rất nhiều kỹ thuật cross-validation được sử dụng để xác định cách chia các tập dữ liệu thành nhiều phần nhỏ.



Hình 2. K-Fold Cross Validation

3. Vấn đề quá khớp (overfitting) và chưa khớp (Underfitting)

- *Overfitting*

  - Overfitting xảy ra khi mô hình học quá chi tiết và phức tạp từ dữ liệu huấn luyện (thường là các mô hình phi tham số hay phi tuyến), đến mức nó không thể tổng quát hóa tốt cho dữ liệu mới. Điều này thường xảy ra khi mô hình cố gắng "ghi nhớ" các đặc điểm nhỏ nhặt hoặc nhiễu (noise) trong dữ liệu huấn luyện, thay vì học các mối quan hệ tổng quát.
  - Nguyên nhân
    - Mô hình quá phức tạp (số lượng tham số quá lớn).
    - Dữ liệu huấn luyện chứa nhiều nhiễu hoặc không đủ đa dạng.
    - Thời gian huấn luyện quá lâu.

- *Underfitting*

  - Underfitting xảy ra khi mô hình quá đơn giản để có thể nắm bắt các mối quan hệ trong dữ liệu. Điều này dẫn đến hiệu suất kém trên cả dữ liệu huấn luyện và dữ liệu kiểm tra, vì mô hình không thể học được các mẫu (patterns) quan trọng từ dữ liệu.

- Nguyên nhân
  - Mô hình quá đơn giản (ví dụ như đường thẳng trong mô hình hồi quy tuyến tính khi dữ liệu phức tạp hơn).
  - Thiếu đặc trưng quan trọng hoặc đặc trưng không được biến đổi đúng cách.
  - Huấn luyện mô hình không đủ lâu.

4. Bộ dữ liệu Thành tích Sinh viên (Student Performance)

- Bộ dữ liệu Thành tích Sinh viên (Student Performance) là bộ dữ liệu được thiết kế để kiểm tra những tác nhân ảnh hưởng đến thành tích học sinh trung học.
- Bộ dữ liệu gồm 10,000 mẫu, với mỗi mẫu chứa thông tin về những tác nhân dự báo đa dạng và thứ tự của performance.
- Ý nghĩa của từng đặc trưng và kiểu dữ liệu

STT	Thuộc tính	Mô tả	Kiểu dữ liệu
1	Hour Studied	Tổng số giờ học của mỗi học sinh.	Số nguyên
2	Previous Score	Điểm số học sinh đạt được trong các bài kiểm tra trước đó.	Số nguyên
3	Extracurricular Activities	Sinh viên có tham gia hoạt động ngoại khóa không (Có hay Không)	Boolean
4	Sleep Hours	Số giờ ngủ trung bình mỗi ngày của sinh viên.	Số nguyên
5	Sample Question Papers Practiced	Số bài kiểm tra mẫu mà học sinh đã luyện tập	Số nguyên
6	Performance Index	Thước đo thành tích tổng thể của mỗi học sinh. Chỉ số thể hiện thành tích học tập nằm trong đoạn [10, 100]. Chỉ số này tỉ lệ thuận với thành tích.	Số thực

- Bộ dữ liệu này cung cấp cái nhìn tổng quan về mối quan hệ giữa nhưngx biến dự đoán với chỉ số thành tích tổng thể của từng học sinh. Nhà nghiên cứu và nhà phân tích dữ liệu có thể sử dụng bộ dữ liệu này để khám phá các khía cạnh ảnh hưởng của Hours Studied, Previous Score, Extracurricular Activities, Sleep Hours, Sample Question Papers Practiced trên thành tích của học sinh (Student Performance).
- Chú ý rằng tập dữ liệu này là tổng hợp và được tạo ra nhằm mục đích minh họa. Mối quan hệ giữa các biến số và chỉ số hiệu suất có thể không phản ánh các tình huống thực tế.
- Trong đồ án này, dữ liệu trên đã được thực hiện tiền xử lý chuyển đổi kiểu dữ liệu cho thuộc tính Extracurricular Activities. Sau khi tiền xử lý, bộ dữ liệu

được chia ngẫu nhiên thành 2 tập với tỉ lệ 9:1. Trong đó 9 phần cho tập huấn luyện (training set) và 1 phân cho tập kiểm tra (testing set).

- train.csv: chứa 9000 mẫu dùng để huấn luyện mô hình.
- test.csv: chứa 1000 mẫu dùng để kiểm tra mô hình.

III. Mô tả tất cả các hàm và các chức năng quan trong

1. Tất cả các thư viện và hàm của thư viện đã sử dụng

i. Thư viện đã sử dụng

- *Thư viện Pandas*

  Pandas là một thư viện mạnh mẽ và phổ biến trong ngôn ngữ Python, được sử dụng rộng rãi trong lĩnh vực khoa học dữ liệu và phân tích dữ liệu. Pandas cung cấp các cấu trúc dữ liệu như DataFrame (2D) và Series (1D) để lưu trữ và thao tác với dữ liệu một cách linh hoạt. Đồng thời, nó còn hỗ trợ cho việc đọc, ghi dữ liệu từ nhiều định dạng khác nhau như CSV, Excel, SQL, JSON và nhiều định dạng khác. Nó cũng hỗ trợ việc lọc, nhóm, tổng hợp, biến đổi, và sắp xếp dữ liệu, giúp dễ dàng chuẩn bị dữ liệu cho các bước phân tích tiếp theo. Ngoài ra, nó còn xử lý các dữ liệu thiếu và làm sạch dữ liệu, cũng như trực quan hóa dữ liệu.
  Trong đồ án này, chúng ta sử dụng thư viện này nhằm mục đích:
  - Đọc dữ liệu từ tệp CSV: cung cấp hàm pd.read_csv() giúp đọc dữ liệu từ tệp CSV vào DataFrame.
  - Trích xuất các đặc trưng và giá trị target: hỗ trợ trích xuất các cột cụ thể DataFrame và Series, giúp dễ phân tích và trích xuất các đặc trưng đầu vào và giá trị mục tiêu cho huấn luyện mô hình.
  - Thống kê và phân tích dữ liệu: sử dụng hàm info(), describe(), isnull().sum(), value_counts(), corr() giúp phân tích cấu trúc và thống kê mô tả cơ bản của dữ liệu.

- *Thư viện NumPy*

  NumPy (Numerical Python) là một thư viện được sử dụng để tính toán khoa học và kỹ thuật, cung cấp một đối tượng mảng đa chiều hiệu quả cao, cùng với nhiều hàm hỗ trợ thực hiện các thao tác số học trên các mảng này.
  Trong đồ án này, chúng ta sử dụng thư viện này nhằm mục đích:

- Xử lý mảng số liệu: bởi lẽ Numpy cung cấp các chức năng để tạo, thao tác và tính toán trên các mảng đa chiều một cách nhanh chóng và dễ dàng.
- Đại Số Tuyến Tính: Thư viện NumPy hỗ trợ các phép toán đại số tuyến tính như nhân ma trận, nghịch đảo ma trận, và tính toán giá trị riêng và vector riêng.
- Thống Kê: NumPy cung cấp các công cụ để tính toán các chỉ số thống kê cơ bản như trung bình, trung vị, phương sai, và nhiều hơn nữa.
- Tích Hợp Với Thư Viện Khác: NumPy là cơ sở của nhiều thư viện khác.

- *Thư viện Plotly*

  Plotly là một thư viện mạnh mẽ dành cho việc trực quan hóa dữ liệu trong Python. Không giống như Matplotlib, Plotly cho phép tạo ra các biểu đồ tương tác, điều này làm cho nó trở nên đặc biệt hữu ích trong các ứng dụng web và các môi trường phân tích dữ liệu. Các biểu đồ được tạo bởi Plotly có thể được tùy chỉnh sâu sắc và dễ dàng chia sẻ trực tuyến.
  Plotly được phát triển nhằm cung cấp một công cụ dễ sử dụng cho việc tạo ra các biểu đồ trực quan tương tác, chất lượng cao mà không cần phải có nhiều kiến thức về đồ họa hay lập trình. Nó phục vụ nhiều đối tượng từ các nhà khoa học dữ liệu, nhà phân tích tài chính, kỹ sư, cho đến các nhà phát triển ứng dụng web.
  Trong đồ án này, chúng ta sử dụng thư viện này nhằm mục đích:
  - Trực Quan Hóa Dữ Liệu Tương Tác: Plotly cho phép tạo ra các biểu đồ mà người dùng có thể tương tác, như phóng to, thu nhỏ, di chuyển, và lựa chọn các phần tử trên biểu đồ.
  - Học Máy và Khoa Học Dữ Liệu: Trong học máy và khoa học dữ liệu, Plotly giúp trực quan hóa các kết quả mô hình, phân phối dữ liệu, và nhiều dạng phân tích dữ liệu khác.

ii. Hàm đã sử dụng từ thư viện

1. Hàm từ thư viện Pandas

- Hàm: pd.read_csv()

  - *Tham số đầu vào*
    - Đường dẫn đến file CSV filepath_or_buffer. Đây là tham số bắt buộc.

- Các tham số tùy chọn khác: sep, delimiter, header, names, index_col, ...

▪ *Kết quả đầu ra*

- Hàm trả về một DataFrame, đối tượng chính trong Pandas để lưu trữ và thao tác với dữ liệu dạng bảng. DataFrame chứa các dữ liệu từ file CSV và có thể được sử dụng để thực hiện các thao tác phân tích dữ liệu, lọc, nhóm, và nhiều hơn nữa.

▪ *Mô tả hoạt động*

- Hàm read_csv() mở và đọc file CSV từ đường dẫn hoặc đối tượng file-like.
- Dữ liệu được phân tách thành các trường dựa trên ký tự phân cách.
- Dữ liệu được tổ chức thành các cột và hàng, sau đó được chuyển thành một DataFrame.
- Các tùy chọn như tiêu đề cột, kiểu dữ liệu, giá trị thiếu, và các tùy chọn khác được áp dụng trong quá trình đọc dữ liệu.
- Cuối cùng, hàm trả về DataFrame chứa dữ liệu đã đọc và được sẵn sàng để sử dụng trong phân tích dữ liệu.

o Hàm: pd.DataFrame.iloc[]

▪ *Tham số đầu vào*

- [row_indexer, column_indexer]
  - row_indexer: Chỉ số hoặc dải chỉ số hàng mà bạn muốn truy cập.
  - column_indexer: Chỉ số hoặc dải chỉ số cột mà bạn muốn truy cập.
- Các tham số này có thể là:
  - Số nguyên: Chỉ số của hàng hoặc cột cụ thể.
  - Danh sách các số nguyên: Danh sách các chỉ số hàng hoặc cột.
  - Dải chỉ số: Ví dụ, 0:5 để chọn từ hàng hoặc cột 0 đến 4.
  - Boolean array: Mảng boolean có kích thước giống với số lượng hàng hoặc cột, dùng để chọn các hàng hoặc cột thỏa mãn điều kiện.

▪ *Kết quả đầu ra*

- Hàm trả về một phần của DataFrame hoặc Series theo chỉ số vị trí được chỉ định
  - Một phần DataFrame: nếu chỉ định cả cột và dòng.
  - Một Series: nếu chỉ định dòng hoặc cột.
  - Một giá trị đơn: nếu chỉ định một dòng và một cột cụ thể.

- *Mô tả hoạt động*
  - Sử dụng chỉ số vị trí (integer) để truy cập dữ liệu, thay vì tên cột hoặc hàng. Điều này có nghĩa là bạn có thể chỉ định hàng hoặc cột bằng chỉ số số nguyên, giúp dễ dàng thao tác với dữ liệu mà không cần phải biết tên cột hoặc hàng cụ thể.
  - Hỗ trợ cắt (slicing) dữ liệu, cho phép bạn chọn các dải hàng hoặc cột.
  - Có thể sử dụng mảng boolean để chọn các hàng hoặc cột thỏa mãn điều kiện

- Hàm: pd.DataFrame.info()

  - *Tham số đầu vào*
    - verbose: (mặc định là None) Xác định xem có hiển thị toàn bộ thông tin chi tiết hay không.
      - True: Hiển thị toàn bộ thông tin, không thiếu bao gồm cột tên, dữ liệu loại, giá trị số lượng.
      - False: Hiển thị các tập tin tắt thông tin, bao gồm chỉ loại dữ liệu và cột số lượng.
    - buf: (mặc định là None) Giống như tệp đối tượng để ghi đầu ra. Default is sys.stdout, cụ thể là hiển thị thông tin trên bảng điều khiển màn hình. Nếu muốn ghi thông tin vào một file, bạn có thể chỉ định file đối tượng tại đây.
    - max_cols: (mặc định là None) Hiển thị tối đa số lượng cột. Nếu số cột lớn hơn max_cols, một phần của DataFrame sẽ được hiển thị thay vì toàn bộ. If for None, toàn bộ cột sẽ được hiển thị (if verbose=True).

- memory_usage: (mặc định là None) Xác định xem có hiển thị thông tin về dung lượng bộ nhớ mà DataFrame sử dụng hay không.
  - True: Hiển thị dung lượng bộ nhớ.
  - False: Không hiển thị dung lượng bộ nhớ.
  - "deep": Tính toán chi tiết bộ nhớ, bao gồm tất cả các đối tượng được tham chiếu trong các kiểu cột object.
- null_counts: (mặc định là None) Xác định chế độ xem hiển thị thiếu giá trị (null) trong mỗi cột hoặc không. Nếu DataFrame có nhiều hơn 100.000 hàng và null_countskhông được cung cấp, giá trị mặc định là False.

▪ *Kết quả đầu ra*
- In thông tin tổng quan về DataFrame.

▪ *Mô tả hoạt động*
- Hàm cung cấp thông tin tổng quan về cấu trúc và nội dung của DataFrame giúp người dùng hiểu đôi nét về dữ liệu như: số lượng cột, tiêu đề từng cột, số lượng dữ liệu non-null (không trống) và kiểu dữ liệu của từng cột, lượng tài nguyên bộ nhớ được sử dụng.

o Hàm: pd.DataFrame.describe()

▪ *Tham số đầu vào*
- Không bắt buộc.

▪ *Kết quả đầu ra*
- Hàm trả về một DataFrame mới chứa các thống kê tóm tắt cho từng cột được chọn trong DataFrame gốc.
  - count: Số lượng giá trị không bị thiếu (non-null).
  - mean: Giá trị trung bình.
  - std: Độ lệch chuẩn.
  - min: Giá trị nhỏ nhất.
  - 25%: Percentile 25%.
  - 50%: Median (percentile 50%).
  - 75%: Percentile 75%.
  - max: Giá trị lớn nhất.

- *Mô tả hoạt động*
  - Thống kê cơ bản: Khi được gọi, hàm describe() sẽ tự động tính toán các thống kê mô tả cho các cột số trong DataFrame. Nếu bạn muốn thống kê cả các cột không phải số (e.g., kiểu object hoặc category), bạn cần chỉ định thêm tham số include.
  - Lựa chọn cột: Bạn có thể kiểm soát những cột nào được tính toán bằng cách sử dụng các tham số include và exclude. Điều này giúp linh hoạt trong việc chỉ xem các thông tin cần thiết.
  - Percentiles tùy chỉnh: Mặc định, describe() tính toán các percentiles 25%, 50%, và 75%. Bạn có thể chỉ định thêm hoặc thay đổi các percentiles này bằng cách cung cấp một danh sách mới cho tham số percentiles.

2. *Hàm từ thư viện Numpy*
   - Hàm: np.hstack()
     - Tham số đầu vào
       - tup: Tuple hoặc danh sách các mảng mà bạn muốn ghép theo chiều ngang. Tất cả các mảng phải có số hàng (hoặc chiều tương ứng) giống nhau. Nếu không, NumPy sẽ ném ra một lỗi.
     - Kết quả đầu ra
       - out: Mảng mới được tạo ra bằng cách ghép các mảng trong tup theo chiều ngang. Kích thước của mảng đầu ra là kết hợp số cột của các mảng đầu vào, trong khi số hàng giữ nguyên.
     - Mô tả hoạt động
       - Sử dụng để kết hợp (hay ghép) các mảng theo chiều ngang (theo cột).
   - Hàm: np.ravel()
     - Tham số đầu vào
       - • a: Mảng NumPy cần được làm phẳng.
       - • order: (Tùy chọn) Chuỗi chỉ định cách sắp xếp các phần tử trong mảng đầu ra:
         - 'C': Theo thứ tự C (hàng chính) - sắp xếp các phần tử từ trái sang phải, từ trên xuống dưới.

        - 'F': Theo thứ tự Fortran (cột chính) - sắp xếp các phần tử từ trên xuống dưới, từ trái sang phải.
        - 'A': Theo thứ tự của mảng gốc. Nếu mảng gốc là theo thứ tự C, thì 'A' giống như 'C'. Nếu mảng gốc là theo thứ tự Fortran, thì 'A' giống như 'F'.
    - Kết quả đầu ra
        - out: Mảng một chiều, có cùng phần tử như mảng đầu vào, nhưng được sắp xếp theo thứ tự nhất định.
    - Mô tả hoạt động
        - sử dụng để làm phẳng (flatten) một mảng nhiều chiều thành một mảng một chiều. Hàm này trả về một bản sao của mảng, hoặc nếu có thể, nó sẽ trả về một mảng phẳng của các phần tử mà không sao chép dữ liệu.
- Hàm: np.column_stack()
    - Tham số đầu vào
    - Kết quả đầu ra
    - Mô tả hoạt động
- Hàm: np.arange()
    - Tham số đầu vào
        - start: (Tùy chọn) Giá trị bắt đầu của dãy. Mặc định là 0 nếu không được chỉ định.
        - stop: Giá trị kết thúc của dãy (không bao gồm giá trị này).
        - step: (Tùy chọn) Bước nhảy giữa các giá trị trong dãy. Mặc định là 1 nếu không được chỉ định.
        - dtype: (Tùy chọn) Kiểu dữ liệu của mảng đầu ra. Nếu không được chỉ định, kiểu dữ liệu được suy ra từ các tham số đầu vào.
    - Kết quả đầu ra
        - out: Mảng NumPy chứa các giá trị đều nhau từ start đến stop, với bước nhảy là step.
    - Mô tả hoạt động
        - Sử dụng để tạo ra một mảng NumPy với các giá trị đều nhau trong một khoảng nhất định.
- Hàm: np.random.shuffle()
    - Tham số đầu vào
        - x: Mảng NumPy cần được xáo trộn. Mảng này sẽ bị thay đổi tại chỗ, tức là các phần tử

trong mảng sẽ được xáo trộn mà không tạo ra một mảng mới.

- Kết quả đầu ra
  - None: Hàm không trả về giá trị nào. Thay vào đó, nó thực hiện việc xáo trộn mảng x tại chỗ.
- Mô tả hoạt động
  - Sử dụng để xáo trộn (hoặc hoán đổi) các phần tử của một mảng dọc theo trục chính.

- Hàm: np.concatenate()
  - Tham số đầu vào
    - a1, a2, ...: Tuple hoặc danh sách các mảng NumPy mà bạn muốn kết hợp. Tất cả các mảng phải có cùng kích thước trong các chiều không phải trục ghép nối.
    - axis: (Tùy chọn) Trục dọc theo đó các mảng sẽ được kết hợp. Mặc định là 0, có nghĩa là kết hợp các mảng dọc theo trục hàng (dọc theo trục chính).
    - out: (Tùy chọn) Mảng đầu ra nơi kết quả sẽ được lưu trữ. Nếu không được cung cấp, hàm sẽ tạo ra một mảng mới để chứa kết quả.
  - Kết quả đầu ra
    - out: Mảng mới được tạo ra bằng cách kết hợp các mảng đầu vào theo trục được chỉ định
  - Mô tả hoạt động
    - Sử dụng để nối (hoặc kết hợp) nhiều mảng dọc theo một trục cụ thể.

Lưu ý: các hàm tính toán numpy như np.ones(), np.linalg.inv(), np.mean(), np.abs() có thể được lược bỏ vì chủ yếu là các hàm dùng để tính toán, xử lý và thao tác mảng đơn giản.

3. *Hàm từ thư viện Plotly*
   - Hàm: plotly.graph_objects.Pie()
     - Tham số đầu vào
       - labels: Danh sách hoặc mảng các nhãn (labels) tương ứng với từng phần của biểu đồ tròn. Các nhãn này thường là tên của các danh mục mà bạn muốn biểu diễn.

- values: Danh sách hoặc mảng các giá trị số tương ứng với các nhãn, xác định kích thước của từng phần trong biểu đồ tròn.
- hole: Giá trị số từ 0 đến 1 để xác định tỷ lệ lỗ hổng ở giữa biểu đồ tròn, tạo ra biểu đồ donut nếu giá trị này lớn hơn 0.
- textinfo: Xác định thông tin nào sẽ được hiển thị trên biểu đồ. Các lựa chọn có thể bao gồm 'label', 'value', 'percent', và 'none'.
- hoverinfo: Xác định thông tin nào sẽ được hiển thị khi người dùng di chuột qua các phần của biểu đồ. Tương tự như textinfo, nó có thể bao gồm 'label', 'value', 'percent', 'name', và 'none'.
- colors: Danh sách các mã màu cho từng phần của biểu đồ. Nếu không được cung cấp, Plotly sẽ tự động chọn màu sắc.
- pull: Danh sách hoặc mảng các giá trị số từ 0 đến 1, xác định mức độ "kéo" của từng phần ra khỏi biểu đồ tròn (tạo hiệu ứng phần nổi bật ra ngoài).
- title: Tiêu đề của biểu đồ tròn.
- domain: Xác định vị trí và kích thước của biểu đồ tròn trong không gian tọa độ của đồ thị (ví dụ: x, y, row, column).
- name: Tên của biểu đồ tròn, hữu ích khi kết hợp nhiều biểu đồ trong một đồ thị lớn hơn.
- direction: Định hướng của các phần trong biểu đồ ('clockwise' hoặc 'counterclockwise').
- showlegend: Boolean xác định xem có hiển thị chú thích (legend) hay không.
- opacity: Giá trị số từ 0 đến 1 xác định độ mờ của biểu đồ tròn.

▪ Kết quả đầu ra
- Pie object: Đối tượng biểu đồ tròn có thể được sử dụng trong một đồ thị (figure) của Plotly bằng cách thêm nó vào danh sách dữ liệu của đồ thị.

▪ Mô tả hoạt động
- Hàm plotly.graph_objects.Pie() tạo một đối tượng biểu đồ tròn dựa trên các tham số đầu

vào. Mỗi phần của biểu đồ sẽ được xác định bởi các giá trị trong tham số values, và các nhãn của từng phần sẽ được lấy từ labels.

- Đối tượng Pie được tạo ra có thể được thêm vào một đồ thị Plotly lớn hơn bằng cách sử dụng plotly.graph_objects.Figure để hiển thị biểu đồ tròn cùng với các biểu đồ khác hoặc riêng lẻ.

- Hàm: plotly.graph_objects.Histogram()
  - Tham số đầu vào
    - x: Mảng hoặc danh sách các giá trị dữ liệu cần được biểu diễn trên trục x. Đây là dữ liệu đầu vào chính cho biểu đồ histogram.
    - y: Mảng hoặc danh sách các giá trị dữ liệu để xác định chiều cao của các thanh. Sử dụng nếu bạn muốn histogram đại diện cho dữ liệu khác ngoài số đếm cơ bản.
    - nbinsx: Số lượng khoảng giá trị (bins) dọc theo trục x. Xác định cách dữ liệu được chia nhỏ trên trục x.
    - autobinx: Boolean xác định xem các khoảng giá trị (bins) có được tính tự động dựa trên dữ liệu hay không. Nếu là True, Plotly sẽ tự động tính toán số lượng và kích thước các khoảng.
    - histnorm: Phương pháp chuẩn hóa dữ liệu histogram. Các tùy chọn bao gồm 'count', 'percent', 'density', và 'probability'.
      - 'count': Hiển thị số lượng các giá trị trong mỗi khoảng (mặc định).
      - 'percent': Hiển thị phần trăm của tổng số.
      - 'density': Hiển thị mật độ xác suất.
      - 'probability': Hiển thị xác suất.
    - histfunc: Hàm để xác định giá trị của các thanh (bars). Tùy chọn bao gồm 'count', 'sum', 'avg', 'min', và 'max'. Điều này xác định cách dữ liệu y được tổng hợp nếu có nhiều giá trị cho một bin.
    - cumulative: Boolean xác định xem biểu đồ có là dạng cộng dồn hay không (tích lũy các giá trị từ trái sang phải).

- orientation: Hướng của các thanh trong biểu đồ ('v' cho cột đứng và 'h' cho cột ngang).
- text: Văn bản hiển thị trên các thanh của histogram, có thể là một danh sách các chuỗi hoặc mảng.
- hovertext: Văn bản hiển thị khi di chuột qua các thanh của biểu đồ.
- marker: Tham số để tùy chỉnh thuộc tính của các thanh như màu sắc, đường viền (border), độ rộng (width), và nhiều thuộc tính khác.
- xbins: Một dictionary chứa các thông số tùy chỉnh về khoảng (bin) trên trục x như start, end, và size.
    - start: Giá trị bắt đầu của khoảng giá trị đầu tiên.
    - end: Giá trị kết thúc của khoảng giá trị cuối cùng.
    - size: Kích thước của mỗi khoảng.
- name: Tên của biểu đồ, hiển thị trong chú thích (legend) khi có nhiều biểu đồ trong một figure.
- opacity: Giá trị từ 0 đến 1 để xác định độ mờ của các thanh.
- barmode: Kiểu hiển thị các thanh nếu có nhiều biểu đồ histogram cùng lúc (ví dụ: 'overlay', 'stack', 'group').
- x_axis và y_axis: Tham chiếu đến các trục x và y nếu bạn muốn đặt histogram này trên một trục cụ thể trong một đồ thị phức hợp.

▪ Kết quả đầu ra
- Histogram object: Đối tượng biểu đồ histogram có thể được sử dụng và thêm vào một đồ thị (figure) trong Plotly.

▪ Mô tả hoạt động
- Hàm này tạo ra một đối tượng biểu đồ histogram dựa trên dữ liệu đầu vào. Histogram chia dữ liệu thành các khoảng giá trị (bins) và sau đó đếm hoặc tổng hợp dữ liệu trong mỗi khoảng, biểu diễn kết quả bằng các thanh cột.
- Đối tượng Histogram được tạo ra có thể được thêm vào một đồ thị Plotly lớn hơn

bằng cách sử dụng plotly.graph_objects.Figure, cho phép hiển thị cùng với các biểu đồ khác hoặc riêng lẻ.

- Hàm: plotly.graph_objects.Heatmap()
  - Tham số đầu vào
    - z: Ma trận 2D hoặc mảng 2D chứa các giá trị cần được hiển thị, quyết định màu sắc của các ô trong heatmap.
    - x: Danh sách hoặc mảng các nhãn trục x.
    - y: Danh sách hoặc mảng các nhãn trục y.
    - colorscale: Thang màu (colorscale) dùng để ánh xạ các giá trị trong z sang các màu tương ứng.
    - colorbar: Dictionary chứa các tùy chỉnh cho thanh màu (color bar).
    - zmin và zmax: Giá trị tối thiểu và tối đa của thang màu.
    - text: Mảng 2D chứa chuỗi văn bản hiển thị trong các ô của heatmap.
    - hoverinfo: Thông tin hiển thị khi di chuột qua các ô của heatmap.
    - reversescale: Boolean đảo ngược thang màu.
    - showscale: Boolean xác định hiển thị thanh màu hay không.
    - x_gap và y_gap: Khoảng cách giữa các ô theo trục x và y.
    - opacity: Độ mờ của heatmap.
  - Kết quả đầu ra
    - Đối tượng Heatmap có thể được thêm vào một đồ thị Plotly.
  - Mô tả hoạt động
    - Tạo biểu đồ heatmap dựa trên dữ liệu đầu vào, trực quan hóa phân bố và sự biến đổi của dữ liệu thông qua màu sắc trên không gian 2D.
- Hàm: plotly.graph_objects.Figure()
  - Tham số đầu vào
    - data: Danh sách các đối tượng đồ thị (ví dụ: Scatter, Bar, Heatmap) sẽ được hiển thị.
    - layout: Tham số layout (cấu trúc) để tùy chỉnh các thuộc tính chung của đồ thị như

tiêu đề, nhãn trục, kích thước, và cách hiển thị các thành phần khác.
- frames: Dùng để tạo đồ thị động (animation) bằng cách tạo ra các khung hình (frames) khác nhau.

- Kết quả đầu ra
  - Đối tượng Figure chứa toàn bộ thông tin của biểu đồ, có thể được hiển thị trực tiếp hoặc kết hợp với các đối tượng đồ thị khác.
- Mô tả hoạt động
  - Tạo ra một đối tượng biểu đồ hoàn chỉnh từ các thành phần khác nhau (dữ liệu, layout, frames). Đối tượng Figure này có thể chứa một hoặc nhiều loại biểu đồ, và có thể tùy chỉnh để tạo ra các đồ thị phức tạp hoặc trực quan hóa dữ liệu theo các cách khác nhau.

- Hàm: plotly.express.scatter()
  - Tham số đầu vào
    - data_frame: DataFrame hoặc cấu trúc dữ liệu tương tự chứa dữ liệu cần vẽ.
    - x: Tên cột hoặc chuỗi giá trị cho trục x.
    - y: Tên cột hoặc chuỗi giá trị cho trục y.
    - color: Cột dùng để mã hóa màu sắc các điểm dữ liệu dựa trên giá trị.
    - size: Cột dùng để mã hóa kích thước các điểm dữ liệu.
    - symbol: Cột dùng để mã hóa kiểu dấu hiệu (symbols) của các điểm dữ liệu.
    - hover_name: Cột chứa thông tin hiển thị khi di chuột qua các điểm dữ liệu.
    - facet_row và facet_col: Cột dùng để tạo các subplot dựa trên giá trị của cột này.
    - trendline: Tên phương pháp vẽ đường xu hướng (ols, lowess,...).
    - title: Tiêu đề của biểu đồ.
  - Kết quả đầu ra
    - Đối tượng Figure chứa biểu đồ scatter (biểu đồ phân tán).
  - Mô tả hoạt động
    - Tạo biểu đồ scatter giúp trực quan hóa mối quan hệ giữa hai biến số thông qua vị trí của các điểm dữ liệu trên đồ thị. Các tham số

như màu sắc, kích thước, kiểu biểu tượng có thể mã hóa thêm thông tin giúp người dùng nhận diện các đặc tính khác của dữ liệu.

- Hàm: plotly.express.box()
  - Tham số đầu vào
    - data_frame: DataFrame hoặc cấu trúc dữ liệu tương tự chứa dữ liệu cần vẽ.
    - x: Tên cột hoặc chuỗi giá trị cho trục x.
    - y: Tên cột hoặc chuỗi giá trị cho trục y.
    - color: Cột dùng để mã hóa màu sắc các hộp dựa trên giá trị.
    - points: Xác định cách hiển thị các điểm dữ liệu trong biểu đồ boxplot (outliers, all, suspectedoutliers, hoặc None).
    - facet_row và facet_col: Cột dùng để tạo các subplot dựa trên giá trị của cột này.
    - title: Tiêu đề của biểu đồ.
    - labels: Dictionary dùng để gán nhãn cho các trục và giá trị.
  - Kết quả đầu ra
    - Đối tượng Figure chứa biểu đồ boxplot.
  - Mô tả hoạt động
    - Tạo biểu đồ boxplot (hộp) dùng để trực quan hóa sự phân phối, sự biến đổi, và phát hiện các điểm dữ liệu ngoại lệ trong một tập dữ liệu. Biểu đồ này hiển thị các phần tử như median, interquartile range (IQR), và các điểm dữ liệu ngoại lệ một cách trực quan.

2. Hàm preprocess (…)
   - *Tham số đầu vào:* một đối tượng mảng numpy, là dữ liệu đầu vào muốn xử lý có dạng (n_samples, n_features) với n_samples là số lượng mẫu và n_features là số lượng đặc trưng.
   - *Kết quả đầu ra:* một mảng numpy được được xử lý, đây là dữ liệu đầu vào của mảng "X" với một cột bổ sung của các giá trị 1 được them vào ở vị trí cột đầu tiên. Kích thước của "X" là (n_samples, n_features + 1)
   - *Mô tả hoạt động*
     - Hàm sử dụng np.hstack() để nối hai mảng dọc theo trục cột.
     - np.ones((x.shape[0], 1)) tạo ra một mảng numpy có kích thước (n_samples, 1) với tất cả các giá trị là 1. Đây là cột bổ sung sẽ được them vào dữ liệu đầu vào "x".

- np.hstack([np.ones((x.shape[0], 1)), x]) nối cột của các giá trị 1 này với mảng x dọc theo trục cột, tạo ra một mảng mới X có kích thước (n_samples, n_features + 1).
- Cuối cùng, hàm trả về mảng X, trong đó cột đầu tiên chứa các giá trị 1 và các cột tiếp theo chứa dữ liệu từ mảng x.

3. Hàm standard_scaler (…)

- Tham số đầu vào: một đối tượng mảng numpy, đây là dữ liệu dầu vào cần được chuẩn hóa. Thường là mảng 2 chiều với kích thước (n_samples, n_features), trong đó n_samples là số lượng mẫu và n_features là số lượng đặc trưng (features).
- Kết quả đầu ra
  - X_scaled: mảng numpy đã đuojc chuẩn hóa, đây là dữ liệu đầu vào của "X" sau khi đã được chuẩn hóa, với mỗi đặc trưng có giá trị trung bình là 0 và độ lệch chuẩn là 1. Kích thước của "X_scaled" là giống với kích thước của "X".
  - "mean": một đối tượng mảng numpy chứa giá trị trung bình của mỗi đặc trưng trong dữ liệu đầu vào X. Kích thước của mean là (n_features,).
  - std: Một đối tượng np.array chứa độX. Kích thước của std là (n_features,).
- Mô tả hoạt động
  - Đầu tiên, ta tính giá trị trung bình của X. Kết quả là một mảng 1.
  - std = np.std(X, axis=0) tính toán độ lệch chuẩn của mỗi đặc trưng dọc theo trục cột trong mảng X. Kết quả là một mảng 1 chiều chứa độ lệch chuẩn của từng đặc trưng.
  - X_scaled = (X - mean) / std chuẩn hóa từng đặc trưng trong X bằng cách trừ đi giá trị trung bình và chia cho độ lệch chuẩn. Kết quả là một mảng X_scaled mà mỗi đặc trưng có giá trị trung bình là 0 và độ lệch chuẩn là 1.
  - X_scaled = (X - mean) / std chuẩn hóa từng đặc trưng trong X bằng cách trừ đi giá trị trung bình và chia cho độ lệch chuẩn. Kết quả là một mảng X_scaled mà mỗi đặc trưng có giá trị trung bình là 0 và độ lệch chuẩn là 1.

4. Lớp OLSLinearRegression (…)

Class này được sử dụng để thực hiện hồi quy tuyến tính bằng phương pháp bình phương tối thiểu thông thường (Ordinary Least Squares - OLS), bao gồm các phương thức để huấn luyện mô hình, lấy các tham số đã ước lượng, và dự đoán đầu ra dựa trên dữ liệu đầu vào.

i. Phương thức fit (self, X, y)
   Sử dụng để huấn luyện mô hình hồi quy tuyến tính bằng cách sử dụng phương pháp bình phương tối thiểu thông thường. Hàm tính toán và lưu trữ các tham số tối ưu dựa trên dữ liệu đầu vào.
   - Tham số đầu vào

- "X": mảng numpy có kích thước (n_samples, n_features) là ma trận dữ liệu đầu vào, trong đó mỗi hàng là một mẫu và mỗi cột là một đặc trưng (feature).
- "y": mảng numpy có kích thước (n_samples,), là vector đầu ra tương ứng với các mẫu trong "X".

- Kết quả đầu ra
  - Trả về đối tượng của lớp với các tham số tối ưu được lưu trữ trong thuộc tính.
- Mô tả hoạt động
  - Đầu tiên, ta sẽ tính bộ trọng số tối ưu theo công thức sau:

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

  - X.T @ X: Tính tích vô hướng của ma trận chuyển vị của X và chính nó.
  - np.linalg.inv(...): Tính ma trận nghịch đảo.
  - Tính toán tham số tối ưu w bằng cách nhân pseudo-inverse X_pinv với vector đầu ra y.
  - self.w sẽ lưu trữ các tham số tối ưu.

ii. Phương thức get_params (self)
Sử dụng để truy xuất các tham số đã được ước lượng sau khi mô hình được huấn luyện.
- Tham số đầu vào: không có
- Kết quả đầu ra
  - self.w: mảng NumPy (np.array) chứa các tham số tối ưu của mô hình dưới dạng vector cột.
- Mô tả hoạt động

iii. Phương thức predicts (self, X)
Sử dụng để dự đoán đầu ra dựa trên mô hình đã huấn luyện và dữ liệu đầu vào mới.
- Tham số đầu vào
  - "X": Mảng NumPy (np.array) có kích thước (n_samples, n_features). Đây là dữ liệu đầu vào mới muốn dự đoán.
- Kết quả đầu ra
  - X @ self.w: mảng NumPy (np.array) chứa các giá trị dự đoán tương ứng với các mẫu trong X.
- Mô tả hoạt động
  - Thực hiện phép nhân ma trận giữa X và vector tham số self.w để tính toán các giá trị dự đoán.
  - Trả về các giá trị dự đoán này.

5. Hàm mae (…)
- Tham số đầu vào
  - y: Mảng NumPy (np.array) chứa các giá trị thực tế của dữ liệu đầu ra. Đây là dữ liệu mà mô hình cần dự đoán.

- y_hat: Mảng NumPy (np.array) chứa các giá trị dự đoán được tạo ra bởi mô hình.

- Kết quả đầu ra: giá trị lỗi trung bình tuyệt đối (MAE), là một số thực (float). Đây là giá trị trung bình của độ lớn tuyệt đối của các sai lệch giữa giá trị thực tế y và giá trị dự đoán y_hat.
- Mô tả hoạt động
  - Công thức ước lượng trung bình của sai số (độ lỗi) tuyệt đối

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

  - Làm phẳng các mảng: y.ravel() và y_hat.ravel() làm phẳng các mảng đầu vào y và y_hat thành các mảng một chiều (1D). Điều này giúp đảm bảo rằng dù y và y_hat ban đầu có nhiều chiều (multi-dimensional), phép trừ giữa chúng vẫn sẽ được thực hiện đúng cách.
  - Tính độ lệch tuyệt đối
    - Phép tính np.abs(y.ravel() - y_hat.ravel()) sẽ trả về một mảng mới chứa các giá trị độ lệch tuyệt đối giữa y và y_hat cho từng phần tử tương ứng.
  - Tính giá trị trung bình
    - Cuối cùng, np.mean(...) tính toán giá trị trung bình của các độ lệch tuyệt đối này, trả về giá trị lỗi trung bình tuyệt đối (MAE).

6. Hàm process_2a (…)
   Hàm này dùng xử lý yêu cầu câu 2a bao gồm các bước để tiền xử lý dữ liệu, huấn luyện mô hình hồi quy tuyến tính (Ordinary Least Squares - OLS) và dự đoán các giá trị trên dữ liệu kiểm tra.
   - Tham số đầu vào
     - X_train: Mảng NumPy (np.ndarray) hoặc DataFrame của Pandas (pd.DataFrame) chứa ma trận đặc trưng (feature matrix) cho dữ liệu huấn luyện.
     - y_train: Mảng NumPy (np.ndarray) hoặc Series của Pandas (pd.Series) chứa biến mục tiêu (target variable) cho dữ liệu huấn luyện.
     - X_test: Mảng NumPy (np.ndarray) hoặc DataFrame của Pandas (pd.DataFrame) chứa ma trận đặc trưng (feature matrix) cho dữ liệu kiểm tra.
   - Kết quả đầu ra
     - coefficients: Mảng NumPy (np.ndarray) chứa các hệ số (weights) của mô hình hồi quy tuyến tính đã được huấn luyện.
     - y_hat: Mảng NumPy (np.ndarray) chứa các giá trị dự đoán cho dữ liệu kiểm tra.
   - Mô tả hoạt động
     - Đầu tiên, ta sẽ tiền xử lý dữ liệu huấn luyện và kiểm tra.

- Gọi hàm preprocess() để tiền xử lý cả dữ liệu huấn luyện X_train và dữ liệu kiểm tra X_test. Hàm preprocess() có thể bao gồm các bước như thêm cột một (constant term) và bình phương dữ liệu.

  - Tiếp theo, ta khởi tạo mô hình hồi quy tuyến tính OLSLinearRegression().
  - Sau đó, ta sẽ huấn luyện mô hình bằng cách sử dụng phương thức fit() của đối tượng mô hình để huấn luyện mô hình với dữ liệu huấn luyện đã được tiền xử lý (X_train và y_train).
  - Lấy các hệ số (weights) của mô hình từ phương thức get_params().
  - Dự đoán các giá trị mục tiêu cho dữ liệu kiểm tra (X_test) bằng phương thức predict() của mô hình.
  - Trả về một tuple chứa các hệ số của mô hình và các giá trị dự đoán cho dữ liệu kiểm tra.

7. Hàm shuffle_and_split_folds (…)
   Hàm shuffle_and_split_folds giúp chia dữ liệu thành các phần nhỏ hơn (folds) sau khi đã xáo trộn ngẫu nhiên, giúp thực hiện kiểm tra chéo (cross-validation).
   - Tham số đầu vào
     - "X": mảng numpy np.ndarray(), là ma trận đặc trưng của tập dữ liệu. Mỗi hàng đại diện cho một mẫu, và mỗi cột đại diện cho một đặc trưng.
     - "y": mảng numpy chứa giá trị mục tiêu (target) tương ứng với các đặc trưng trong X.
     - "k": là số lượng phần (folds) cho kiểm tra chéo. Mặc định là 5 phần.
   - Kết quả đầu ra
     - "folds_X": danh sách chứa dữ liệu đặc trưng cho mỗi phần (fold). Mỗi phần là một mảng con của X.
     - "folds_y": danh sách chứa dữ liệu mục tiêu cho mỗi phần (fold). Mỗi phần là một mảng con của y.
   - Mô tả hoạt động
     - Tính toán số mẫu: n = X.shape[0] tính toán số lượng mẫu trong tập dữ liệu.
     - Tạo chỉ số và xáo trộn: indices = np.arange(n) tạo một mảng chứa các chỉ số của mẫu, sau đó np.random.shuffle(indices) xáo trộn các chỉ số để ngẫu nhiên hóa dữ liệu.
     - Xáo trộn dữ liệu: Sử dụng các chỉ số đã xáo trộn để xáo trộn dữ liệu đặc trưng và mục tiêu: X_shuffle = X[indices] và y_shuffle = y[indices].
     - Tính toán kích thước phần: fold_size = n // k tính toán kích thước của mỗi phần.
     - Chia dữ liệu thành các phần: Tạo các danh sách chứa các phần dữ liệu đặc trưng và mục tiêu cho mỗi phần dựa trên kích thước của phần.
8. Hàm finding_best_feature (…)
   - Tham số đầu vào
     - X_train (np.ndarray): Dữ liệu huấn luyện chứa các đặc trưng, dạng mảng numpy.

- y_train (np.ndarray): Các giá trị mục tiêu (target) tương ứng với dữ liệu huấn luyện, dạng mảng numpy.
- features (pd.Index hoặc list-like): Tên hoặc chỉ số của các đặc trưng trong X_train.
- k (int, mặc định = 5): Số lượng lần kiểm tra chéo (số phần dữ liệu được chia).

- Kết quả đầu ra
  - best_feature: Tên hoặc chỉ số của đặc trưng có MAE trung bình thấp nhất.
  - best_mae: Giá trị MAE trung bình thấp nhất tương ứng với đặc trưng tốt nhất.
- Mô tả hoạt động
  - Đầu tiên, ta sẽ tạo một từ điển mae_results để lưu trữ giá trị MAE trung bình cho mỗi đặc trưng.
  - Dữ liệu huấn luyện (X_train và y_train) được xáo trộn và chia thành k phần thông qua hàm shuffle_and_split_folds()
  - Kiểm tra chéo cho từng đặc trưng
    - Lặp qua từng đặc trưng trong tập dữ liệu huấn luyện.
    - Với mỗi đặc trưng, thực hiện kiểm tra chéo k lần:
      - Trong mỗi lần, chọn một phần dữ liệu làm tập kiểm tra và phần còn lại làm tập huấn luyện.
      - Tách dữ liệu liên quan đến đặc trưng đang xét.
      - Tiền xử lý dữ liệu bằng cách thêm cột 1 vào đầu bằng hàm preprocess().
      - Huấn luyện mô hình hồi quy tuyến tính (OLSLinearRegression) với dữ liệu huấn luyện.
      - Dự đoán giá trị trên tập kiểm tra.
      - Tính toán MAE cho lần kiểm tra đó và lưu lại giá trị này.
    - Tính giá trị MAE trung bình cho đặc trưng đó và lưu vào từ điển mae_results.
  - **Tìm đặc trưng tốt nhất**: Đặc trưng có MAE trung bình thấp nhất được coi là đặc trưng tốt nhất.

9. Hàm train_best_feature_model (…)

   Hàm train_best_feature_model thực hiện việc huấn luyện một mô hình hồi quy tuyến tính dựa trên đặc trưng tốt nhất đã được chọn trước đó.
   - Tham số đầu vào
     - X_best_feature (numpy.ndarray): Dữ liệu đặc trưng tốt nhất đã được chọn để huấn luyện, dạng mảng numpy.
     - y_train (numpy.ndarray): Dữ liệu mục tiêu tương ứng với dữ liệu đặc trưng, dạng mảng numpy.
   - Kết quả đầu ra
     - best_feature_model (OLSLinearRegression): Mô hình hồi quy tuyến tính đã được huấn luyện.

- best_feature_params (dict): Thông số của mô hình đã được huấn luyện, bao gồm trọng số (weights) và hệ số chặn (intercept).

- Mô tả hoạt động
  - Khởi tạo mô hình hồi quy tuyến tính: Hàm tạo ra một mô hình hồi quy tuyến tính (sử dụng lớp OLSLinearRegression).
  - Huấn luyện mô hình: Sử dụng dữ liệu của đặc trưng tốt nhất (X_best_feature) và dữ liệu mục tiêu (y_train), hàm tiến hành huấn luyện mô hình.
  - Lấy thông số của mô hình: Sau khi huấn luyện, hàm lấy các thông số của mô hình (bao gồm trọng số và hệ số chặn - intercept) bằng cách gọi phương thức get_params() của đối tượng mô hình.
  - Trả về kết quả: Hàm trả về mô hình đã được huấn luyện và các thông số của nó.
  -

10. Hàm finding_best_m_model (…)

    Hàm finding_best_m_model thực hiện việc kiểm tra chéo k lần (k-fold cross-validation) để đánh giá và tìm mô hình tốt nhất từ danh sách các mô hình đã được cung cấp, dựa trên lỗi tuyệt đối trung bình (MAE)

    - Tham số đầu vào
      - X_train (numpy.ndarray): Dữ liệu huấn luyện chứa các đặc trưng, dạng mảng numpy.
      - y_train (numpy.ndarray): Các giá trị mục tiêu tương ứng với dữ liệu huấn luyện, dạng mảng numpy.
      - models (dict): Một từ điển chứa các mô hình và hàm tương ứng để biến đổi dữ liệu. Các khóa là tên mô hình, và các giá trị là hàm biến đổi dữ liệu.
      - k (int, mặc định = 5): Số lượng lần kiểm tra chéo (số phần dữ liệu được chia).
    - Kết quả đầu ra
      - best_model: Tên của mô hình có MAE trung bình thấp nhất.
      - best_mae_model: Giá trị MAE trung bình thấp nhất tương ứng với mô hình tốt nhất.
    - Mô tả hoạt động
      - Tạo một từ điển mae_results để lưu trữ giá trị MAE trung bình cho mỗi mô hình.
      - Dữ liệu huấn luyện (X_train và y_train) được xáo trộn và chia thành k phần bằng cách sử dụng hàm shuffle_and_split_folds.
      - Kiểm tra chéo cho từng mô hình:
        - Lặp qua từng mô hình trong danh sách models.
        - Với mỗi mô hình, thực hiện kiểm tra chéo k lần:
          - Chia dữ liệu thành tập kiểm tra và tập huấn luyện.
          - Chuẩn hóa dữ liệu bằng hàm standard_scaler, trả về dữ liệu chuẩn hóa cùng với giá trị trung bình và độ lệch chuẩn (để chuẩn hóa dữ liệu kiểm tra).

- Áp dụng hàm xử lý tương ứng với mỗi mô hình (model_fn) để biến đổi dữ liệu.
- Huấn luyện mô hình hồi quy tuyến tính (OLSLinearRegression) với dữ liệu đã chuẩn hóa.
- Dự đoán trên tập kiểm tra và tính toán MAE cho lần kiểm tra đó.

- Tính giá trị MAE trung bình cho mô hình và lưu vào từ điển mae_results.

- Tìm mô hình tốt nhất: Mô hình có MAE trung bình thấp nhất được coi là mô hình tốt nhất.

IV. Ý tưởng thực hiện các yêu cầu

1. Yêu cầu 1 – Phân tích khám phá dữ liệu (EDA)

Đối với yêu cầu này, chúng ta cần phải thực hiện được phân tích EDA để khám phá dữ liệu bằng cách như sau:

- Thống kê và mô tả chi tiết các dữ liệu của đặc trưng và rút ra kết luận về dữ từng đặc trưng phân bố như thế nào trong bộ dữ liệu.
- Phân tích và quan sát sự phân bố của các đặc trưng và mối quan hệ giữa các đặc trưng đến thành tích học tập tổng thể của từng học sinh. Từ đó, rút ra kết luận rằng đặc trưng nào ảnh hưởng nhiều nhất đến thành tích học tập của học sinh.

Phần này sẽ được báo cáo rõ hơn trong mục V. Báo cáo và phân tích khám phá bộ dữ liệu.

2. Yêu cầu 2a – Mô hình hồi quy tuyến tính sử dụng toàn 5 đặc trưng

Đối với yêu cầu này, ta sẽ phải huấn luyện mô hình hồi quy tuyến tính sử dụng cả 5 đặc trưng cho toàn bộ tập huấn luyện. Sau đó, báo cáo độ lỗi của mô hình trên tập test và biểu diễn công thức hồi quy tính toán y (Student Performance) theo 5 đặc trưng.

Các bước thực hiện

Bước 1: trích xuất dữ liệu toàn bộ đặc trưng (X_train) và dữ liệu target Performance Index (y_train) từ tập training set nhằm huấn luyện mô hình. Cũng như dữ liệu toàn bộ đặc trưng kiểm thử (X_test) và Performance Index (y_test) từ tập test set nhằm kiểm thử và đánh giá mô hình.

Bước 2: Tiền xử lý tập X_train và X_test.

Bước 3: Khởi tạo và huấn luyện mô hình hồi quy tuyến tính sử dụng tập X_train và y_train

Bước 4: Lấy hệ số chặn và các hệ số tương ứng với từng đặc trưng.

Bước 5: Dự đoán Performance Index mới bằng cách sử dụng mô hình vừa huấn luyện trên tập test.

Bước 6: Tính ước lượng sai số trung bình của mô hình trên tập test (MAE)

Bước 7: Ghi công thức hồi quy tuyến tính Student Performance đánh giá thành tích tổng thể của học sinh.

3. **Yêu cầu 2b – Mô hình hồi quy tuyến tính sử dụng một đặc trưng tốt nhất tìm được**

Đối với yêu cầu này, chúng ta sẽ phải huấn luyện lần lượt 5 mô hình sử dụng duy nhất một đặc trưng có trong bộ dữ liệu huấn luyện. Đồng thời, khi huấn luyện từng mô hình sử dụng 1 đặc trưng trong 5 đặc trưng ban đầu thì ta sẽ sử dụng phương pháp K-Fold Cross Validation với thuật toán như phần giới thiệu đã mô tả. Sau đó, dự đoán mô hình trên tập validation và đánh giá độ lỗi của từng mô hình và chọn ra mô hình có độ lỗi thấp nhất, tức là mô hình tốt nhất với đặc trưng tốt nhất.
Sau đó, ta sẽ phải huấn luyện lại mô hình sử dụng đặc trưng tốt nhất với bộ train và dự đoán mô hình trên tập test và lấy bộ trọng số của mô hình cũng như đánh giá độ lỗi mô hình.

4. **Yêu cầu 2c – Mô hình hồi quy tuyến tính sử dụng các đặc trưng tự thiết kế**
Đối với yêu cầu này, sinh viên cần phải tự thiết kế và xây dựng các mô hình theo từng phương pháp sau đây:
Mô hình 1: mô hình sử dụng đặc trưng mới đã được kết hợp với nhau bằng cách nhân.
Mô hình 2: mô hình sử dụng các đặc trưng đã được biến đổi bằng cách bình phương lên.
Mô hình 3: mô hình sử dụng nhiều đặc trưng đã được chuẩn hóa.

V. Báo cáo và phân tích khám phá bộ dữ liệu

1. **Thống kê và mô tả dữ liệu các đặc trưng**
Nhìn vào thông tin tổng quan về dữ liệu, ta có thể thấy số lượng mẫu (entries) là 9000 mẫu (từ 0 đến 8999), gồm 6 cột, trong đó:
- Có 5 cột đầu tiên (Hour Studied, Previous Scores, Extracurricular Activities, Sleep Hours, Sample Question Papers Practiced) với kiểu dữ liệu int64, đại diện cho các biến số nguyên.
- Có 1 cột cuối cùng (Performance Index) với kiểu dữ liệu float64, đại diện cho một biến số thực (biến liên tục).
- Các cột dữ liệu:
- Hours Studied: Tổng số giờ học của mỗi sinh viên.
- Previous Scores: Điểm số học sinh đạt được trong các bài kiểm tra trước đó.

- Extracurricular Activities: Sinh viên có tham gia hoạt động ngoại khóa không (Có hoặc Không).
- Sleep Hours: Số giờ ngủ trung bình mỗi ngày của sinh viên.
- Sample Question Papers Practiced: Số bài kiểm tra mẫu mà học sinh đã luyện tập.
- Performance Index: Thước đo thành tích tổng thể cho mỗi sinh viên. Chỉ số thể hiện thành tích học tập, nằm trong đoạn [10, 100]. Chỉ số này tỉ lệ thuận với thành tích.

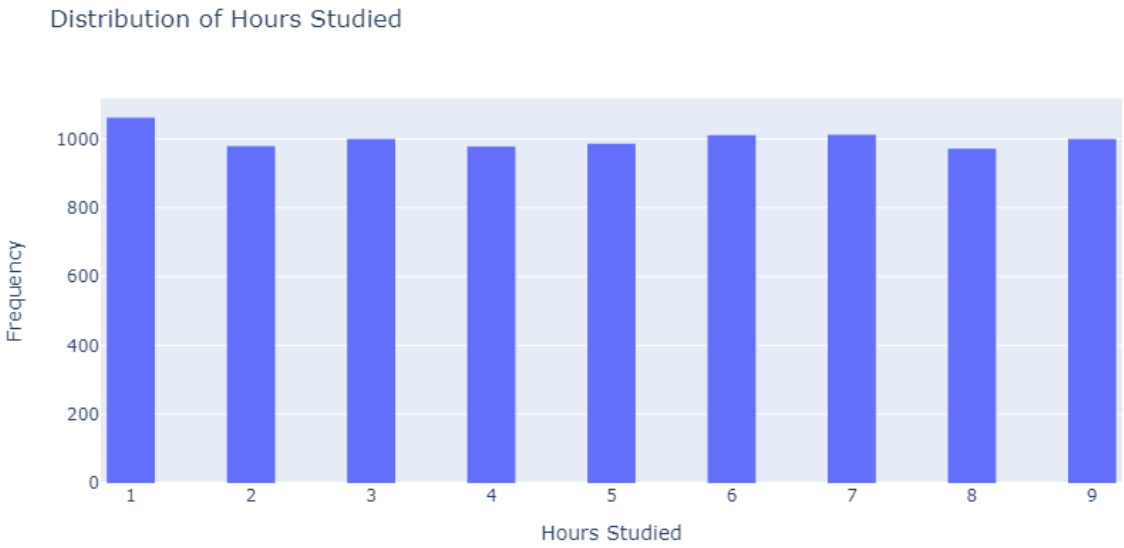
- Tất cả các cột đều có 9000 giá trị không thiếu (non-null), điều này có nghĩa là không có giá trị nào bị thiếu trong tập dữ liệu này.

Các dữ liệu đã được làm sạch kỹ càng.

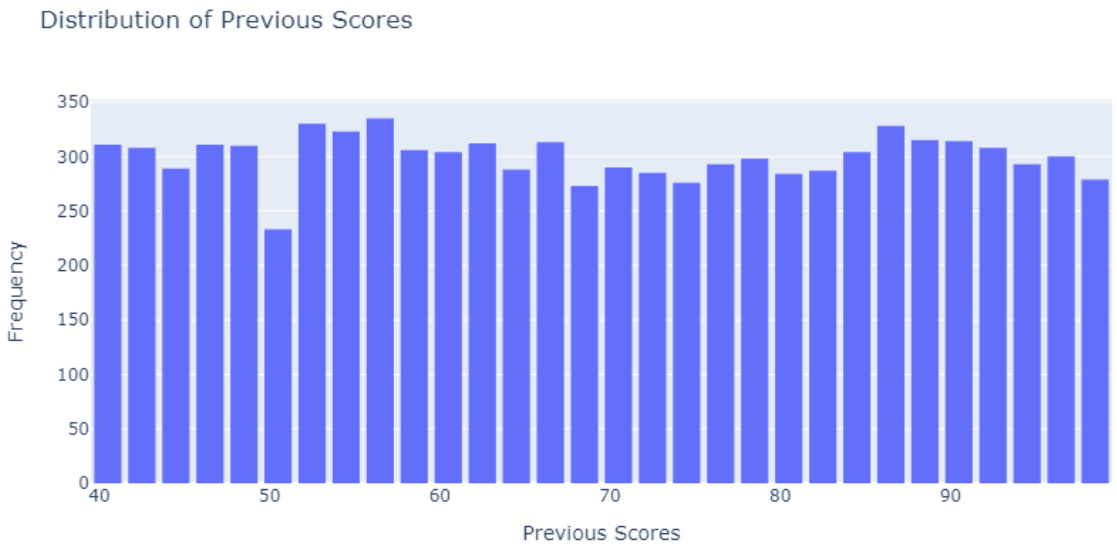
- Nhận xét chung
- Hầu hết các biến đều có phân phối rộng, với giá trị min và max khác biệt, đặc biệt là ở Performance Index, Previous Scores, và Hours Studied.
- Các biến như Hours Studied, Sleep Hours, và Sample Question Papers Practiced** có độ phân tán cao, được thể hiện qua độ lệch chuẩn.
- Các biến như Performance Index và Previous Scores có thể có phân phối lệch hoặc có các giá trị cực trị.
- Extracurricular Activities có tính chất nhị phân (0 hoặc 1), trong đó phần lớn giá trị là 0.

2. **Phân tích và quan sát các đặc trưng**

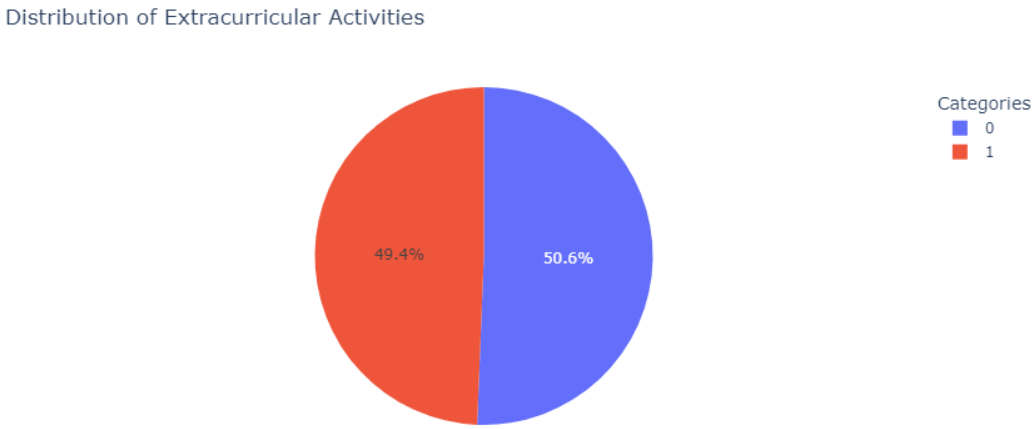
Hours Studied



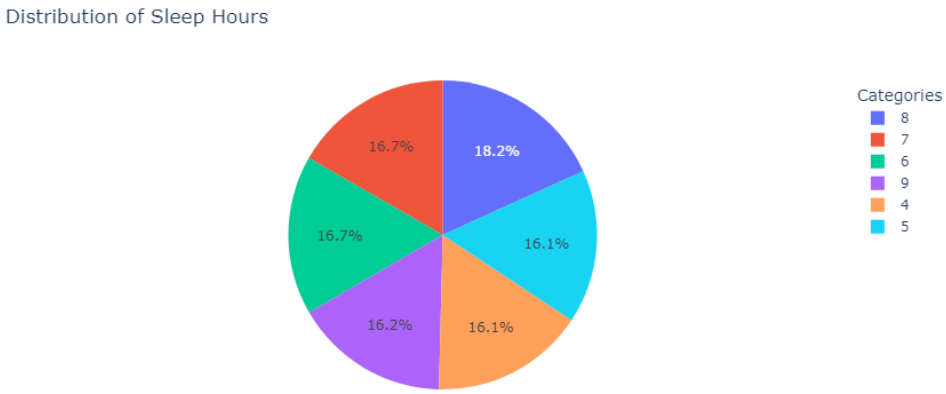
Previous Scores



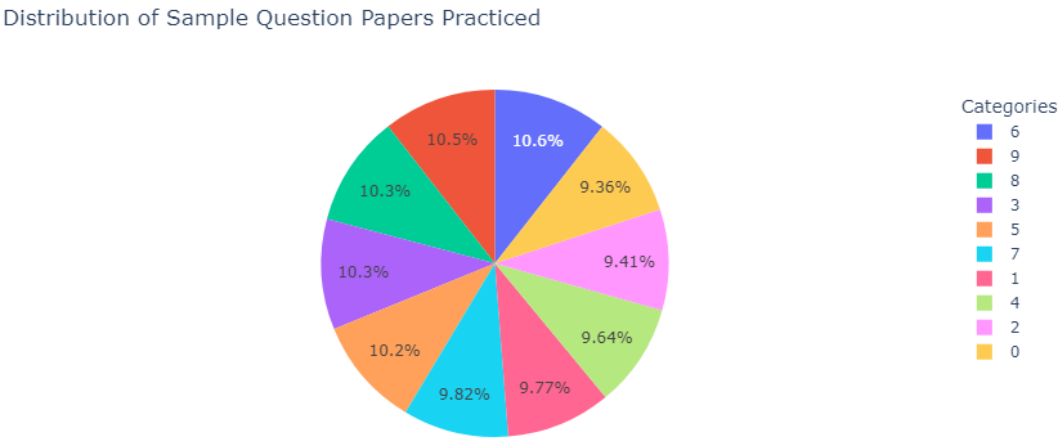
Extracurricular Activities



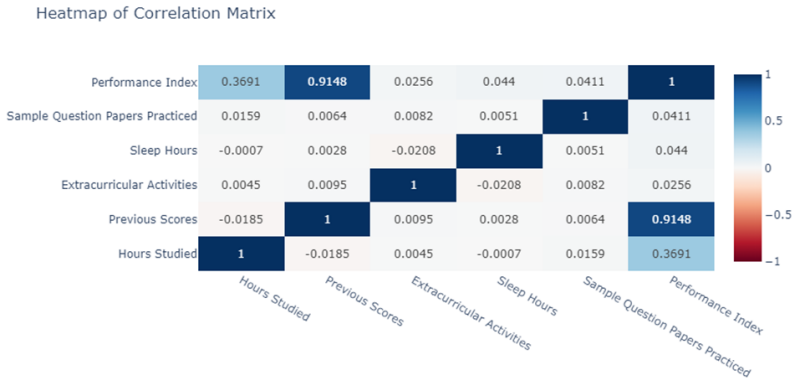
Sleep Hours



Sample Question Papers Practiced



Heatmap



VI. Báo cáo và nhận xét các mô hình

1. **Mô hình sử dụng toàn bộ 5 đặc trưng (yêu cầu 2a)**
   i. Báo cáo kết quả mô hình
      - Bộ trọng số bao gồm
        - Hệ số chặn: -33.969
        - Hệ số của Hours Studied: 2.852
        - Hệ số của Previous Score: 1.018
        - Hệ số của Extracurricular Activities: 0.604
        - Hệ số Sleep Hours: 0.474
        - Hệ số Sample Questions Papers Practice: 0.192
      - Công thức hồi quy tuyến tính của mô hình:

$$SP = -33.969 + 2.852x_1 + 1.018x_2 + 0.604x_3 + 0.474x_4 + 0.192x_5$$

        - Trong đó:
          - x_1: đặc trưng Hours Studied chỉ tổng số giờ học của học sinh
          - x_2: đặc trưng Previous Score chỉ điểm số học sinh đạt được trong các bài kiểm tra trước đó.
          - x_3: đặc trưng Extracurricular Activities cho biết học sinh có tham gia hoạt động ngoại khóa.
          - x_4: đặc trưng Sleep Hours chỉ số giờ ngủ trung bình mỗi ngày của sinh viên.
          - x_5: đặc trưng Sample Question Papers Practiced cho biết số bài kiểm tra mẫu mà học sinh đã luyện tập.
        - Ước lượng trung bình của sai số tuyệt đối của mô hình trên tập kiểm thử: **MAE ≈ 1.596**.

   ii. Nhận xét mô hình
      - Mô hình hồi quy tuyến tính sử dụng 5 đặc trưng có kết quả đạt tốt, với độ lỗi tuyệt đối là 1.596 trên tập kiểm tra, cho thấy sự chênh lệch giữa mô hình với các giá trị thực tế sai lệch 1.596 đơn vị.
      - Ta có thể thấy hệ số của Hours Studied là lớn nhất (2.852) sau đó là hệ số của Previous Score (1.018) trong các đặc trưng đề cập cho thấy các yếu tố về tổng số giờ học của mỗi sinh viên có ảnh hưởng lớn nhất đến thành tích tổng thể của mỗi sinh viên sau đó là yêu tố các điểm số đạt được trong các bài kiểm tra trước đó.
      - Các đặc trưng còn lại có hệ số vô cùng nhỏ bé hơn 1cho thấy các yếu tố như số giờ ngủ trung bình của mỗi học sinh, số bài kiểm tra mẫu hay hoạt động ngoại khóa không ảnh hưởng đến thành tích của từng học sinh.
      - Mô hình này có độ lỗi thấp nhất trong tất cả các mô hình được huấn luyện trong đồ án này cho thấy tầm quan trọng của cả 5 đặc trưng chứ không chỉ riêng từng đặc trưng cụ thể.
      - Hệ số chặn âm (-33.969) không có ý nghĩa thực tế vì nếu các đặc trưng trong mô hình đều là 0 thì thước đo thành tích tổng thể là -33,969, điều này vi phạm và sai với giá trị thực tế.

2. Mô hình sử dụng duy nhất 1 đặc trưng (yêu cầu 2b)

i. Báo cáo kết quả từng mô hình

STT	Mô hình với một đặc trưng	MAE
1	Hours Studied	15.449
2	Previous Scores	6.618
3	Extracurricular Activities	16.195
4	Sleep Hours	16.186
5	Sample Question Papers Practiced	16.184

Như vậy, mô hình với đặc trưng tốt nhất là Previous Scores vì độ lỗi tuyệt đối trung bình MAE xấp xỉ 6.618, nhỏ nhất trong tất cả mô hình.
Bộ trọng số cho mô hình với đặc trưng tốt nhất Previous Scores:

- Hệ số chặn: -14.989
- Hệ số của Previous Score: 1.011

Công thức hồi quy tuyến tính cho mô hình có đặc trưng tốt nhất:

$$SP = -14.989 + 1.011x$$

Trong đó:
x: đặc trưng Previous Scores cho biết điểm số học sinh đạt được trong các bài kiêm tra trước đó.
Ước lượng trung bình của sai số tuyệt đối của mô hình trên tập test là: MAE ≈ 6.544

ii. Nhận xét mô hình

Giải thích vì sao Previous Scores là đặc trưng tốt nhất.

Previous Scores là đặc trưng ảnh hưởng đến quá trình học tập của học sinh một cách sâu sắc bởi lẽ thành tích học tập của học sinh là không dựa trên bài kiểm tra hiện tại mà còn phải dựa trên tất cả các bài kiểm tra trong quá khứ thì mới đánh giá được học sinh đó có thành tích như thế nào. Điều này cho thấy sự quan trọng của quá trình học tập và tích lũy kiến thức trong thời gian dài. Chính điều này đã giúp cho mô hình hồi quy tuyến tính sử dụng đặc trưng này có khả năng dự báo tốt hơn so với các đặc trưng khác.
Đồng thời các yếu tố khác như Hours Studied, Extracurricular Activities, Sleep Hours, và Sample Question Papers Practiced cũng đóng vai trò quan trọng trong quá trình học tập nhưng chúng không có sự ràng buộc chặt chẽ như Previous Scores. Điều này do những yếu tố này học sinh không bắt buộc thực hiện và mang tính tạm thời, và biến động theo từng thời kỳ, thời điểm cụ thể. Ví dụ, học sinh có thể thức trắng đêm để học bài học chơi

game thì Sleep Hours vẫn mang tính chất đánh giá chung với giá trị nhỏ, thì điều này không chắc học sinh sẽ đạt được điểm tốt trong kỳ thi hoặc có thể đạt điểm tốt.

Cuối cùng, nếu thành tích học tập học sinh của sinh viên càng tốt thì chứng tỏ học sinh đó đã học tập chăm chỉ và nền tảng kiến thức vững chắc.

Tuy nhiên, mô hình này độ lỗi vẫn khá cao cho thấy khi xem xét thành tích học tập của sinh viên thì không nên chỉ xem xét một khía cạnh.

3. Mô hình tự thiết kết và xây dựng (Yêu cầu 2c)

i. Báo cáo kết quả từng mô hình

STT	Mô hình	MAE
1	Sử dụng 2 đặc trưng (Hours Studied, Previous Scores)	11.082
2	Sử dụng 4 đặc trưng (“Hours Studied”, “Previous Scores”, “Sleep Hours”, “Sample Question Papers Practiced”)	2.657
3	Sử dụng 3 đặc trưng (“Hours Studied”, “Previous Scores”, “Sleep Hours”)	1.702

Mô hình tốt nhất là mô hình số chuẩn hóa dữ liệu số 3 vì MAE nhỏ nhất là 1.702

Bộ trọng số cho mô hình tốt nhất:

Hệ số chặn: -32.82

Hệ số Hours Studied: 2.856

Hệ số Previous Score: 1.018

Hệ số Sleep Hours: 0.472

Công thức hồi quy tuyến tính:

$$SP = -32.82 + 2.856x_1 + 1.018x_2 + 0.472x_3$$

Trong đó:

x_1: đặc trưng Hours Studied chỉ tổng số giờ học của học sinh

x_2: đặc trưng Previous Score chỉ điểm số học sinh đạt được trong các bài kiểm tra trước đó.

x_3: đặc trưng Sleep Hours chỉ số giờ ngủ trung bình mỗi ngày của sinh viên.

Độ lỗi MAE xấp xỉ 1.702.

ii. Nhận xét mô hình

Giải thích vì sao mô hình 3 tốt nhất ?

Việc chuẩn hóa các đặc trưng giúp mô hình hồi quy tuyến tính hoạt động hiệu quả hơn, đặc biệt khi các đặc trưng có đơn vị hoặc phạm vi giá trị khác nhau. Điều này giúp tránh tình trạng một đặc trưng áp đảo các đặc trưng khác trong quá trình huấn luyện, làm tăng độ chính xác của mô hình.

TÀI LIỆU THAM KHẢO

Trong quá trình thực hiện **đồ án thực hành số 3 – Linear Processing**, nhằm phục vụ cho quá trình thực hiện được suôn sẻ và tăng hiệu suất cũng như độ chính xác trong quá trình báo cáo, em đã tham khảo và sử dụng một số tài liệu mở và điện tử sau đây:

Tài liệu tham khảo về Báo cáo

[1] Vấn đề overfitting và underfitting *(ngày truy cập: 12/08/2024)*

[5] K-Fold Cross Validation *(ngày truy cập: 13/08/2024)*