TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG – HCM
KHOA CÔNG NGHỆ THÔNG TIN

----- ☙ 🕮 ❧ -----

BÁO CÁO ĐỒ ÁN
PROJECT 3: LINEAR REGRESSION

Môn học: Toán ứng dụng và thống kê cho Công nghệ thông tin

Giảng viên: Vũ Quốc Hoàng
Nguyễn Văn Quang Huy
Nguyễn Ngọc Toàn
Phan Thị Phương Uyên

Sinh viên thực hiện: Dương Trung Nghĩa
MSSV: 22127293
Lớp: 22CLC05

Thành phố Hồ Chí Minh, năm 2024

MỤC LỤC

NHẬN XÉT CỦA GIẢNG VIÊN

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Giảng viên

MÔ TẢ THƯ VIỆN

Các thư viện sử dụng trong đồ án là:

1. “Pandas” là một module Python phổ biến được sử dụng chủ yếu cho thao tác phân tích dữ liệu. Nó cung cấp các cấu trúc dữ liệu như “DataFrame” và “Series”, giúp dễ dàng thao tác với dữ liệu theo cách tương tự như trong “Excel” nhưng với khả năng xử lý tốt hơn. Trong đồ án này, “Pandas” được sử dụng để xử lý và tiền xử lý dữ liệu, đặc biệt là trong việc chia nhỏ và tổ chức tập dữ liệu thành các tập huấn luyện và kiểm tra. Cấu trúc “DataFrame” của Pandas cho phép quản lý và thao tác với các tập dữ liệu lớn một cách hiệu quả, giúp dễ dàng áp dụng các mô hình học máy. Module “Pandas” cũng hỗ trợ việc gộp và kết hợp dữ liệu, điều này rất quan trọng khi cần kết hợp các phần dữ liệu sau khi chia chúng để kiểm tra chéo.
2. “Numpy” (Numerical Python) là một thư viện cơ bản cho tính toán khoa học trong Python, cung cấp một đối tượng mảng N-dimensional mạnh mẽ. Nó rất hiệu quả trong việc thực hiện các phép toán số học trên các mảng lớn, hỗ trợ các phép toán ma trận, đại số tuyến tính, và các hàm toán học ngẫu nhiên. Ở đồ án này, “Numpy” được sử dụng cho các thao tác như chia nhỏ tập dữ liệu thành nhiều phần để kiểm tra chéo và tính toán các chỉ số như MAE.
3. “Matplotlib” là một thư viện trực quan hóa dữ liệu. Nó cho phép tạo ra các biểu đồ và đồ thị với mức độ tùy chỉnh cao, từ các biểu đồ đơn giản như biểu đồ cột và biểu đồ đường, đến các biểu đồ phức tạp như đồ thị 3D. Matplotlib rất linh hoạt, cho phép kiểm soát chi tiết về hình dáng, màu sắc, và kiểu dáng của các biểu đồ. Trong đồ án, được sử dụng để trực quan hóa kết quả phân tích dữ liệu, giúp hiểu rõ hơn về dữ liệu thông qua các hình ảnh biểu đồ.
4. “Seaborn” là một thư viện trực quan hóa dữ liệu được xây dựng trên nền tảng của “Matplotlib”, cung cấp một giao diện cấp cao hơn để tạo ra các biểu đồ đẹp mắt. “Seaborn” tích hợp tốt với “Pandas”, với dự án này, module “Seaborn” được sử dụng để tạo ra các biểu đồ nâng cao như biểu đồ nhiệt độ tương quan giúp hiểu rõ hơn về mối quan hệ giữa các biến trong tập dữ liệu và hiệu suất của các mô hình. Sự tích hợp chặt chẽ với “Pandas DataFrames” khiến “Seaborn” đặc biệt tiện lợi khi trực quan hóa dữ liệu phức tạp.

MÔ TẢ CÁC HÀM

Đọc dữ liệu:

1. Hàm pd.read_csv()

- **Chức năng**: Hàm "read_csv()" của "Pandas" được sử dụng để đọc dữ liệu từ tệp CSV và lưu trữ nó dưới dạng DataFrame. CSV (Comma-Separated Values) là một định dạng tệp phổ biến cho việc lưu trữ dữ liệu dạng bảng.
- **Cú pháp**: pd.read_csv(filepath_or_buffer, sep=',', ...)
- **Đầu vào**:
  - "filepath_or_buffer": Đường dẫn đến tệp CSV hoặc một buffer. Đây là tham số bắt buộc. (ở đồ án này truyền vào file train.csv và test.csv)
  - "sep": Ký tự phân tách giữa các giá trị, mặc định là dấu phẩy (,). Nếu dữ liệu của file được phân tách bằng dấu khác, có thể chỉ định nó tại đây.
- **Đầu ra**: Trả về một DataFrame chứa dữ liệu được đọc từ tệp CSV.

2. Hàm iloc[]

- **Chức năng**: Hàm iloc[] của Pandas được sử dụng để truy cập các hàng và cột trong DataFrame bằng vị trí (index) thay vì tên cột hoặc chỉ số. Ở đây, được sử dụng để lấy tất cả hàng cột trừ cột cuối cùng (X_train, X_test) và lấy tất cả các hàng và chỉ cột cuối (y_train, y_test)
- **Cú pháp:** DataFrame.iloc[row_indexer, column_indexer]
- **Đầu vào**:
  - row_indexer: Vị trí (hoặc các vị trí) của hàng cần truy cập. Có thể là một số nguyên, một danh sách các số nguyên, hoặc một lát cắt (slice).
  - column_indexer: Vị trí (hoặc các vị trí) của cột cần truy cập. Cũng có thể là một số nguyên, một danh sách các số nguyên, hoặc một lát cắt.
- **Đầu ra**: Trả về một phần của DataFrame (hoặc Series) tương ứng với các hàng và cột đã chọn.

Cài đặt hàm hỗ trợ:

3. Class OLSLinearRegression [6][7]

- **Chức năng**: Được thiết kế để thực hiện hồi quy tuyến tính bằng phương pháp Ordinary Least Squares (OLS) - phương pháp tối ưu hóa đơn giản và phổ biến để dự đoán giá trị mục tiêu dựa trên các đặc trưng đầu vào.
- **Phương thức:** Gồm fit(self, X, y), get_params(self), predict(self, X)

3.1. Hàm fit(self, X, y)

- **Chức năng**: Hàm này thực hiện việc huấn luyện mô hình hồi quy tuyến tính bằng cách tính toán trọng số (w) thông qua phương pháp giả nghịch đảo của ma trận đặc trưng.
- **Đầu vào**:
  - "X": Ma trận đặc trưng đầu vào, là một mảng "NumPy" hai chiều với kích thước (n_samples, n_features). Mỗi hàng là một mẫu dữ liệu, và mỗi cột là một đặc trưng ở đây có 5 đặc trưng là (Hours Studied, Previous Scores, Extracurricular Activities, Sleep Hours, Sample Question Papers Practiced).
  - "y": Giá trị mục tiêu, là một mảng NumPy một chiều với kích thước (n_samples,). Mỗi phần tử trong mảng là giá trị mục tiêu tương ứng với mẫu dữ liệu trong X (Performance Index).
- **Đầu ra**: Trả về đối tượng self với các trọng số (w) đã được tính toán và lưu trong thuộc tính self.w.
- **Mô tả**:
  - Tính ma trận giả nghịch đảo của ma trận truyền vào bằng hàm np.linalg.pinv(X) của thư viện "Numpy" kể cả ma trận không vuông hoặc không nghịch đảo. Hàm này sử dụng phân tích giá trị suy phân (SVD), trong đó ma trận gốc "X" được phân tích thành tích ba ma trận: "U", "Σ", và "$V\hat{}T$". Giả nghịch đảo của A được tính bằng cách:

$$A\hat{}+= V\Sigma\hat{} + U\hat{}T$$

  - Công thức hồi quy tuyến tính đơn giản có dạng:

$$\hat{y} = Xw$$

    Với "w" được tính như sau:

$$w = X\hat{} + y$$

  - Sau đó, nhân ma trận giả nghịch đảo với "y" truyền vào để tìm trọng số tối ưu. Trọng số này sẽ được sử dụng trong mô hình hồi quy tuyến tính để dự đoán các giá trị mới dựa trên dữ liệu đầu vào. Cuối cùng, lưu trọng số "w" trong "self.w".

3.2. Hàm get_params(self)

- **Chức năng**: Trả về các trọng số (w) của mô hình sau khi đã được huấn luyện.
- **Đầu vào**: Không có.
- **Đầu ra**: Trả về một mảng "NumPy" chứa các trọng số (w) của mô hình hồi quy tuyến tính.

- **Mô tả**: Hàm này đơn giản trả về các trọng số đã được tính toán trong quá trình huấn luyện hàm fit.

3.3. Hàm predict(self, X)

- **Chức năng**: Sử dụng các trọng số đã được tính toán để dự đoán giá trị mục tiêu dựa trên dữ liệu đầu vào (X).
- **Đầu vào**: "X" - Ma trận đặc trưng đầu vào, là một mảng NumPy hai chiều với kích thước (n_samples, n_features). Mỗi hàng là một mẫu dữ liệu, và mỗi cột là một đặc trưng.
- **Đầu ra**: Trả về một mảng NumPy chứa các giá trị mục tiêu dự đoán, tương ứng với từng mẫu dữ liệu trong "X".
- **Mô tả**: Như đã đề cập công thức hồi quy tuyến tính đơn giản, hàm này thực hiện dự đoán bằng cách nhân ma trận đặc trưng "X" với mảng trọng số "w".

4. Hàm mae_cal(y, y_hat)

- **Chức năng**: Tính toán giá trị lỗi trung bình tuyệt đối (MAE) giữa giá trị thực (y) và giá trị dự đoán (y_hat). MAE là một trong những chỉ số phổ biến nhất để đánh giá hiệu suất của mô hình hồi quy, phản ánh sai số trung bình giữa giá trị dự đoán và giá trị thực tế.
- **Đầu vào**:
  - "y" - Mảng NumPy chứa các giá trị thực tế, có kích thước (n_samples,).
  - "y_hat" - Mảng NumPy chứa các giá trị dự đoán, có kích thước (n_samples,).
- **Đầu ra**: Trả về một giá trị số (float) là giá trị lỗi trung bình tuyệt đối (MAE).
- **Mô tả**:
  - y.ravel() - y_hat.ravel(): Trừ các giá trị dự đoán khỏi các giá trị thực để tính sai số cho từng mẫu dữ liệu. ravel() là hàm của NumPy dùng để chuyển đổi mảng sang một chiều.
  - np.abs(...): Tính giá trị tuyệt đối của sai số để đảm bảo rằng các sai số âm không ảnh hưởng đến kết quả.
  - np.mean(...): Tính giá trị trung bình của tất cả các sai số tuyệt đối để ra được giá trị MAE.

5. Hàm preprocess(X)

- **Chức năng**: Hàm này thêm một cột các giá trị 1 vào ma trận đặc trưng (X) để bao gồm hệ số chặn (intercept) trong mô hình hồi quy tuyến tính. Việc thêm cột này là cần thiết để đảm bảo mô hình có thể dự đoán chính xác ngay cả khi không có mối quan hệ tuyến tính rõ ràng giữa các biến đầu vào và đầu ra.

- **Đầu vào**: "X" - Ma trận đặc trưng đầu vào, là một mảng NumPy hai chiều với kích thước (n_samples, n_features). Mỗi hàng là một mẫu dữ liệu, và mỗi cột là một đặc trưng.
- **Đầu ra**: Trả về một mảng NumPy hai chiều mới với kích thước (n_samples, n_features + 1), trong đó cột đầu tiên là các giá trị 1 và các cột còn lại là các đặc trưng ban đầu.
- **Mô tả**:
  - np.ones((X.shape[0], 1)): Tạo ra một mảng NumPy với kích thước (n_samples, 1) chứa toàn các giá trị 1. Điều này được thực hiện để thêm hệ số chặn (intercept) vào mô hình.
  - np.hstack((..., X)): Kết hợp mảng 1 vừa tạo với ma trận đặc trưng X bằng cách ghép theo chiều ngang, tạo ra một ma trận mới với cột đầu tiên là các giá trị 1.

Yêu cầu 1: [8][11]

6. Hàm pd.DataFrame.describe()

- **Chức năng**: Hàm describe() trong "Pandas" được sử dụng để tính toán các thống kê tóm tắt cho các cột số hoặc cột dạng phân loại trong một DataFrame.
- **Cú pháp**: DataFrame.describe(percentiles=None, include=None, exclude=None, datetime_is_numeric=False)
- **Đầu vào**:
  - "percentiles": Một danh sách các phần trăm (ví dụ: [0.25, 0.5, 0.75] cho các phần tư). Mặc định là [0.25, 0.5, 0.75]. Cho phép xác định các phần trăm tùy chỉnh ngoài giá trị mặc định.
  - "include": Loại dữ liệu sẽ được bao gồm trong bảng thống kê. Mặc định là chỉ bao gồm các cột số. Có thể sử dụng include='all' để bao gồm tất cả các cột hoặc cung cấp danh sách kiểu dữ liệu cụ thể như ['float64', 'int64'].
  - "exclude": Loại dữ liệu sẽ bị loại trừ khỏi bảng thống kê. Có thể chỉ định các kiểu dữ liệu không muốn đưa vào.
  - "datetime_is_numeric": Nếu được đặt là True, các cột kiểu datetime sẽ được coi như dữ liệu số để tính toán các thống kê. Mặc định là False.
- **Đầu ra**: Trả về một DataFrame chứa các thống kê tóm tắt cho mỗi cột.
- **Cách hoạt động**:
  - Đối với các cột số, hoạt động như một tổ hợp, kết hợp nhiều phương pháp nội bộ của "Pandas" gọi các hàm như count() – số lượng phần tử không rỗng, mean() – giá trị trung bình, std() – độ lệch chuẩn, min() – gia trị nhỏ nhất, max() – giá trị lớn nhất, "25,50,75%" được tính toán bằng cách sử dụng

phương pháp phân vị, không phải trực tiếp từ các hàm riêng lẻ, nhưng dựa trên việc sắp xếp dữ liệu và tìm giá trị tại các vị trí này trên từng cột của DataFrame để tạo ra bảng thống kê.

- Đối với các cột phân loại hoặc chuỗi ký tự gọi các hàm count() – đếm số lượng các giá trị không rỗng, nunique() – đếm số lượng các giá trị duy nhất trong cột, mode() – tìm giá trị xuất hiện nhiều nhất trong cột, value_counts() – đếm số lần xuất hiện của giá trị phổ biến nhất trong cột, bằng cách sử dụng các phương pháp nội bộ trên từng cột.

7. Hàm pd.DataFrame.info()

- **Chức năng**: Hàm info() trong "Pandas" cung cấp một cái nhìn tổng quan về cấu trúc của một DataFrame, bao gồm số lượng các hàng, cột, tên cột, kiểu dữ liệu của các cột, số lượng giá trị không rỗng (non-null) trong mỗi cột, và lượng bộ nhớ được sử dụng.
- **Cú pháp**: DataFrame.info(verbose=None, buf=None, max_cols=None, memory_usage=None, null_counts=None)
- **Đầu vào**:
  - verbose:
    - True (mặc định): Hiển thị đầy đủ thông tin về DataFrame.
    - False: Chỉ hiển thị số lượng cột và kiểu dữ liệu, bỏ qua các thông tin chi tiết.
  - buf: Đối tượng ghi (writeable) để in thông tin (ví dụ: sys.stdout). Mặc định là in ra màn hình.
  - max_cols: Số lượng cột tối đa để hiển thị. Nếu DataFrame có nhiều cột hơn số này, chỉ một phần thông tin sẽ được hiển thị.
  - memory_usage:
    - True (mặc định): Hiển thị lượng bộ nhớ mà DataFrame sử dụng.
    - False: Không hiển thị thông tin về bộ nhớ.
    - "deep": Tính toán chi tiết hơn về bộ nhớ, bao gồm các đối tượng nội tại như chuỗi ký tự.
  - null_counts:
    - True: Hiển thị số lượng giá trị không rỗng (non-null).
    - False hoặc None: Ẩn thông tin về số lượng giá trị không rỗng.
- **Đầu ra**: Trả về thông tin chi tiết về DataFrame, bao gồm số lượng hàng và cột, kiểu dữ liệu của các cột, số lượng giá trị không rỗng trong mỗi cột, và lượng bộ nhớ sử dụng.

- **Cách hoạt động**:
  - Hàm lấy thông tin về số lượng hàng, số lượng cột, và chỉ số DataFrame.
  - Từng cột được duyệt qua để xác định số lượng giá trị không rỗng và kiểu dữ liệu.
  - Các phép tính cơ bản như đếm số lượng giá trị không rỗng count(), xác định kiểu dữ liệu (dtype), và ước lượng bộ nhớ (memory_usage) được thực hiện. Sau đó hiển thị.

8. Hàm pd.DataFrame.corr()

- **Chức năng:** train.corr() được sử dụng để tính toán ma trận tương quan giữa các cột số liệu trong DataFrame train. Tương quan (correlation) đo lường mức độ và hướng của mối quan hệ tuyến tính giữa hai biến. Giá trị của tương quan nằm trong khoảng từ -1 đến 1.
- **Cú pháp:** DataFrame.corr(method='pearson', min_periods=1)
- **Đầu vào:**
  - method: Phương pháp tính tương quan.
    - 'pearson' (mặc định): Tính tương quan Pearson.
    - 'kendall': Tính tương quan Kendall Tau.
    - 'spearman': Tính tương quan Spearman.
  - min_periods: Số lượng tối thiểu các quan sát cần thiết để có thể tính toán tương quan giữa hai biến.
- **Đầu ra**: Trả về một DataFrame chứa ma trận tương quan giữa các cột số liệu (correlation_matrix).
- **Cách hoạt động**: Sử dụng phương pháp mặc định (thường là pearson), hàm này tính toán tương quan giữa từng cặp cột bằng cách đo lường mối quan hệ tuyến tính giữa chúng. Công thức cho tương quan Pearson là:

$$r = (\sum\_(i = 1)\^n\blacksquare\ 〖(x\_i - \bar{x})(y\_i - \bar{y})〗\ )/(\sqrt{(\sum\_(i = 1)\^n\blacksquare\ 〖(x\_i - \bar{x}\ )\^2\ \sum\_(i = 1)\^n\blacksquare(y\_i - \bar{y}\ )\^2〗\ )\ )}$$

9. Hàm plt.figure()

- Chức năng: Tạo một cửa sổ hình ảnh mới với kích thước truyền vào
- Cú pháp: plt.figure(figsize=(width, height))
- Đầu vào: "**figsize"**: Một tuple xác định kích thước (chiều rộng, chiều cao) của hình ảnh.
- Đầu ra: Tạo một cửa sổ hình ảnh mới để chứa các biểu đồ.
- Cách hoạt động:

- Hàm này khởi tạo một đối tượng Figure mới, là một không gian trong đó các biểu đồ sẽ được vẽ.
- Sử dụng tham số figsize, hàm xác định kích thước của cửa sổ, trong trường hợp này là 10 hoặc 12 đơn vị chiều rộng và 6 hoặc 8 đơn vị chiều cao.

10.Hàm sns.heatmap()

- Chức năng: Được sử dụng để vẽ một biểu đồ heatmap, hiển thị ma trận tương quan dưới dạng các ô màu sắc.
- Cú pháp: sns.heatmap(data, annot=None, cmap=None, fmt='.2g', ...)
- Đầu vào:
  - data: Ma trận dữ liệu để vẽ heatmap (ở đây là correlation_matrix).
  - annot: Nếu True, các giá trị sẽ được hiển thị trực tiếp trên các ô của heatmap.
  - cmap: Bảng màu được sử dụng cho heatmap. Ở đây, "coolwarm" là bảng màu chuyển từ xanh dương (mối tương quan âm) sang đỏ (mối tương quan dương).
  - fmt: Định dạng hiển thị các giá trị trên ô. Ở đây, ".2f" định dạng số với 2 chữ số thập phân.
- Đầu ra: Vẽ một biểu đồ heatmap trên cửa sổ hình ảnh hiện tại, hiển thị ma trận tương quan với các giá trị và màu sắc tương ứng.
- Cách hoạt động:
  - Các giá trị trong ma trận tương quan được chuyển thành các ô vuông trong biểu đồ, với màu sắc đại diện cho độ mạnh của tương quan.
  - Sử dụng bảng màu coolwarm, các giá trị âm sẽ được hiển thị bằng màu xanh dương, và các giá trị dương sẽ được hiển thị bằng màu đỏ.
  - Các giá trị trên ô được định dạng với hai chữ số thập phân (fmt=".2f") và hiển thị trực tiếp lên biểu đồ nhờ tham số annot=True.

11.Hàm DataFrame.hist()

- Chức năng: train.hist() được sử dụng để vẽ biểu đồ histogram cho mỗi cột số liệu trong DataFrame train. Mỗi biểu đồ histogram hiển thị phân bố của dữ liệu trong một cột cụ thể, giúp hiểu rõ hơn về sự phân phối của các giá trị trong cột đó.
- Cú pháp: DataFrame.hist(column=None, by=None, grid=True, xlabelsize=None, xrot=None, ylabelsize=None, yrot=None, ax=None, sharex=False, sharey=False, figsize=None, layout=None, bins=10, backend=None, legend=False, **kwargs)
- Đầu vào:
  - **bins**: Số lượng các "thùng" (bins) trong mỗi histogram. Mỗi "thùng" đại diện cho một khoảng giá trị trên trục x, và chiều cao của mỗi "thùng" đại diện cho số lượng mẫu dữ liệu nằm trong khoảng đó.

- **color**: Màu sắc của các "thùng" trong biểu đồ histogram. Trong trường hợp đồ án, màu của các "thùng" được đặt là 'skyblue'.
- **edgecolor**: Màu của đường viền xung quanh mỗi "thùng". Trong trường hợp này, đường viền của mỗi "thùng" sẽ có màu đen ('black').
- **figsize**: Kích thước của toàn bộ hình ảnh chứa các biểu đồ. Đây là một tuple (chiều rộng, chiều cao) thể hiện kích thước hình ảnh theo đơn vị inch. Trong trường hợp ở đây, kích thước hình ảnh được đặt là (12, 8).

➢ Đầu ra: Trả về một **matplotlib.axes._subplots.AxesSubplot** chứa các biểu đồ histogram cho mỗi cột số liệu trong DataFrame train. Biểu đồ này sẽ hiển thị trong cửa sổ hình ảnh của matplotlib nếu chưa có một hình ảnh cụ thể nào được tạo ra.

➢ Cách hoạt động:
- Đầu tiên, hàm hist() sẽ tự động lựa chọn các cột trong DataFrame train có kiểu dữ liệu số (như int64, float64). Các cột không phải kiểu số sẽ bị bỏ qua.
- Với mỗi cột số liệu, hàm hist() sẽ chia các giá trị thành các khoảng giá trị (bins) dựa trên tham số bins=9.
- Sau đó, hàm đếm số lượng mẫu dữ liệu rơi vào từng khoảng giá trị và vẽ các "thùng" histogram tương ứng. Mỗi "thùng" sẽ có chiều cao tương ứng với số lượng mẫu dữ liệu trong khoảng giá trị đó.
- Màu sắc của các "thùng" được đặt thành màu xanh da trời ('skyblue') và có đường viền màu đen ('black').

12. Hàm sns.scatterplot() (Tương tự cho “Sleep Hours” và “Sample Question Papers Practiced”)

➢ Chức năng: sns.scatterplot() trong thư viện Seaborn được sử dụng để vẽ biểu đồ phân tán (scatter plot), hiển thị mối quan hệ giữa hai biến số trên trục x và trục y. Trong trường hợp này, biến Hours Studied sẽ được đặt trên trục x, và biến Performance Index sẽ được đặt trên trục y.

➢ Cú pháp: sns.scatterplot(x=None, y=None, hue=None, style=None, size=None, data=None, palette=None, markers=True, legend='auto', ax=None, **kwargs)

➢ Đầu vào:
- x: Tên của cột trong DataFrame muốn sử dụng làm trục x (ở đây là 'Hours Studied').
- y: Tên của cột trong DataFrame muốn sử dụng làm trục y (ở đây là 'Performance Index').
- data: DataFrame chứa dữ liệu cần vẽ (ở đây là train).

- color: Màu sắc của các điểm trên biểu đồ. Trong trường hợp này, màu sắc được đặt là màu xanh dương ('blue').

- Đầu ra: Trả về một **matplotlib.axes._subplots.AxesSubplot** chứa biểu đồ phân tán (scatter plot) cho hai biến được chỉ định.
- Cách hoạt động:
  - Hàm sns.scatterplot() sẽ tìm cột 'Hours Studied' và cột 'Performance Index' trong DataFrame train.
  - Với mỗi cặp giá trị (x, y) tương ứng từ hai cột, hàm sẽ vẽ một điểm (marker) trên biểu đồ.

13. Hàm sns.boxplot()

- Chức năng: sns.boxplot() trong thư viện Seaborn được sử dụng để vẽ biểu đồ hộp (box plot). Biểu đồ hộp là một cách trực quan để hiển thị sự phân bố của dữ liệu, bao gồm các thông tin về trung vị (median), tứ phân vị (quartiles), và các giá trị ngoại lai (outliers). Trong trường hợp ở đây, biểu đồ hộp sẽ cho thấy phân bố của Performance Index dựa trên các mức độ khác nhau của Extracurricular Activities.
- Cú pháp: sns.boxplot(x=None, y=None, hue=None, data=None, order=None, hue_order=None, orient=None, color=None, palette=None, saturation=0.75, width=0.8, dodge=True, fliersize=5, linewidth=None, whis=1.5, ax=None, **kwargs)
- Đầu vào:
  - x: Tên của cột trong DataFrame được sử dụng làm trục x. Trong trường hợp này, cột 'Extracurricular Activities' sẽ được sử dụng.
  - y: Tên của cột trong DataFrame muốn sử dụng làm trục y (ở đây là 'Performance Index').
  - data: DataFrame chứa dữ liệu cần vẽ (ở đây là train).
- Đầu ra: Trả về một **matplotlib.axes._subplots.AxesSubplot** chứa biểu đồ hộp (box plot) hiển thị sự phân bố của Performance Index theo các mức độ của Extracurricular Activities.
- Cách hoạt động:
  - Hàm sns.boxplot() sẽ tìm cột 'Extracurricular Activities' và cột 'Performance Index' trong DataFrame train.
  - **Trục y (y='Performance Index')**: Các giá trị của Performance Index sẽ được sử dụng để vẽ hộp, hiển thị trung vị (median), tứ phân vị (25%, 75%), và các giá trị ngoại lai nếu có.

14. Hàm sns.regplot()

- Chức năng: sns.regplot() trong thư viện Seaborn được sử dụng để vẽ biểu đồ phân tán (scatter plot) và thêm một đường hồi quy tuyến tính (regression line) để hiển thị mối quan hệ tuyến tính giữa hai biến số trên trục x và trục y. Trong trường hợp này, hiển thị mối quan hệ giữa Previous Scores và Performance Index, đồng thời thêm một đường hồi quy màu đỏ (line_kws={"color": "red"}).
- Cú pháp: sns.regplot(x=None, y=None, data=None, x_estimator=None, x_bins=None, x_ci="ci", scatter=True, fit_reg=True, ci=95, n_boot=1000, markers='o', color=None, scatter_kws=None, line_kws=None, ax=None)
- Đầu vào:
  - x: Tên của cột trong DataFrame muốn sử dụng làm trục x. Trong trường hợp này, cột 'Previous Scores' sẽ được sử dụng.
  - y: Tên của cột trong DataFrame muốn sử dụng làm trục y (ở đây là 'Performance Index').
  - data: DataFrame chứa dữ liệu cần vẽ (ở đây là train).
  - Line_kws: Một từ điển các tham số cho đường hồi quy. Ở đây, {"color": "red"} chỉ định màu của đường hồi quy là màu đỏ.
- Đầu ra: Trả về một **matplotlib.axes._subplots.AxesSubplot** chứa biểu đồ phân tán (scatter plot) hiển thị mối quan hệ giữa Previous Scores và Performance Index, kèm theo một đường hồi quy tuyến tính.
- Cách hoạt động:
  - Hàm sns.regplot() sẽ tìm cột 'Previous Scores' và cột 'Performance Index' trong DataFrame train.
  - Với mỗi cặp giá trị (x, y) tương ứng từ hai cột, hàm sẽ vẽ một điểm (marker) trên biểu đồ, giúp trực quan hóa mối quan hệ giữa Previous Scores và Performance Index.
  - Hàm sẽ tính toán đường hồi quy tuyến tính phù hợp nhất dựa trên dữ liệu đã cho và vẽ đường này trên biểu đồ. Đường hồi quy được vẽ với màu đỏ như được chỉ định trong line_kws={"color": "red"}.

Yêu cầu 2: [9][10]

15. Hàm DataFrame.columns.values()

- Chức năng: sử dụng Pandas để truy xuất danh sách tên các cột trong DataFrame X_train.
- Đầu vào: “X_train” - một DataFrame. Trong trường hợp này, nó là DataFrame chứa các đặc trưng của tập huấn luyện.

- Đầu ra: Trả về một mảng NumPy (dtype = object) chứa các tên cột trong DataFrame X_train.
- Cách hoạt động:
  - "X_train.columns()" truy xuất đối tượng "Index" chứa các tên cột của DataFrame "X_train".
  - "X_train.columns.values()" truy xuất các tên cột dưới dạng một mảng NumPy.

16. Hàm np.arange(X_train.shape[0])

- Chức năng: np.arange() là một hàm trong thư viện NumPy, được sử dụng để tạo ra một mảng NumPy chứa các số nguyên liên tiếp, bắt đầu từ một giá trị cho trước và kết thúc trước một giá trị khác. Nó tương tự như hàm range() trong Python, nhưng trả về một mảng NumPy thay vì một đối tượng range. Ở đây, sử dụng để tạo ra một mảng các chỉ số (indices) từ 0 đến số lượng hàng trong DataFrame "X_train".
- Cú pháp: np.arange([start,] stop, [step,] dtype=None, *, like=None)
- Đầu vào: "X_train.shape[0]" – số lượng hàng của DataFrame
- Đầu ra: Một mảng NumPy với các giá trị từ 0 đến số lượng hàng của X_train - 1.

17. Hàm np.random.shuffle()

- Chức năng: "np.random.shuffle()" được sử dụng để xáo trộn ngẫu nhiên thứ tự các phần tử trong mảng hoặc danh sách được cung cấp. Thay vì tạo ra một mảng mới, hàm này xáo trộn mảng tại chỗ, tức là mảng ban đầu sẽ bị thay đổi.
- Cú pháp: np.random.shuffle(x)
- Đầu vào: "x" – mảng muốn xáo trộn ở đây là mảng với giá trị "indices" từ 0 đến số lượng hàng của X_train - 1
- Đầu ra: Không có trả về chỉ thực hiện chức năng xáo trộn phần tử.
- Cách hoạt động: Hàm np.random.shuffle() sử dụng một thuật toán ngẫu nhiên để thay đổi thứ tự của các phần tử trong mảng ban đầu. Các phần tử sẽ được hoán đổi vị trí một cách ngẫu nhiên chứ không phải tạo mảng mới.

18. Hàm reset_index()

- Chức năng: Thiết lập lại chỉ số của DataFrame sau khi sắp xếp lại các hàng, và bỏ đi chỉ số cũ.
- Cú pháp: DataFrame.reset_index(level=None, drop=False, inplace=False, col_level=0, col_fill='')

- Đầu vào: **drop=True** - Không giữ lại chỉ số cũ. Nếu là False thì giữ lại chỉ số cũ.
- Đầu ra: Trả về một DataFrame với chỉ số đã được thiết lập lại từ 0.

19. Hàm np.array_split()

- Chức năng: np.array_split() được sử dụng để chia một mảng NumPy hoặc DataFrame thành nhiều phần nhỏ hơn, có thể có kích thước không đồng đều. Điều này hữu ích khi cần phân chia dữ liệu thành các tập con để xử lý riêng biệt hoặc khi chia dữ liệu thành các tập huấn luyện và kiểm tra.
- Cú pháp: np.array_split(ary, indices_or_sections, axis=0)
- Đầu vào:
  - Ary: Mảng numpy hoặc DataFrame muốn chia nhỏ
  - Indices_or_sections: số lượng phần cần chia hoặc danh sách các chỉ số xác định nơi cắt
  - Axis: xác định trục chia mảng
- Đầu ra: Trả về một danh sách chứa các mảng NumPy hoặc DataFrame nhỏ hơn sau khi đã chia.

20. Hàm pd.concat([X_splits[j][[name]] for j in range(n_splits) if j != i])

- Chức năng: Mục đích của câu lệnh này là kết hợp lại các phần của DataFrame từ danh sách X_splits, nhưng loại trừ một phần cụ thể (được chỉ định bởi i), và chỉ giữ lại các cột được chỉ định bởi name.
- Cú pháp: pd.concat(objs, axis=0, ignore_index=False, ...)
- Đầu vào:
  - X_splits[j][[name]]: Truy cập và chọn cột name từ phần thứ j trong danh sách X_splits.
  - for j in range(n_splits) if j != i: Vòng lặp tạo danh sách các phần từ X_splits, trừ phần thứ i.
  - pd.concat([...]): Kết hợp các DataFrame đã chọn từ vòng lặp thành một DataFrame duy nhất.
- Đầu ra: Trả về một DataFrame mới được tạo ra bằng cách kết hợp tất cả các phần của X_splits, ngoại trừ phần thứ i, và chỉ chứa cột được chỉ định bởi name.

21. Hàm model1(X)

- Chức năng: Tạo một ma trận thiết kế cho mô hình hồi quy bằng cách thêm một cột các giá trị 1 (đại diện cho hệ số chặn) vào ma trận đặc trưng, và loại bỏ cột "Extracurricular Activities" khỏi DataFrame "X".
- Đầu vào: "**X**" - DataFrame chứa các đặc trưng của tập dữ liệu (Previous Scores, Hours Studied, Sleep Hours, Extracurricular Activities, Sample Question Papers Practiced).
- Đầu ra: Trả về một mảng NumPy 2D với cột đầu tiên toàn giá trị 1 để đại diện cho hệ số chặn và các cột còn lại là các đặc trưng từ DataFrame "X" trừ cột "Extracurricular Activities".
- Cách hoạt động:
  - np.ones((X.shape[0], 1)): Tạo một mảng có X.shape[0] hàng và 1 cột, tất cả các giá trị là 1.
  - X.drop("Extracurricular Activities", axis=1): Loại bỏ cột "Extracurricular Activities" từ DataFrame X.
  - np.hstack(...): Kết hợp cột toàn bộ giá trị 1 với các cột còn lại sau khi đã loại bỏ cột "Extracurricular Activities".

22. Hàm model2(X)

- Chức năng: Tạo một ma trận thiết kế với cột hệ số chặn, các đặc trưng "Previous Scores" và "Hours Studied", cùng với một cột tương tác (interaction term) giữa hai đặc trưng này.
- Đầu vào: "**X**" - DataFrame chứa các đặc trưng của tập dữ liệu (Previous Scores, Hours Studied, Sleep Hours, Extracurricular Activities, Sample Question Papers Practiced).
- Đầu ra: Trả về một mảng NumPy 2D với cột đầu tiên toàn giá trị 1 để đại diện cho hệ số chặn, cột "Previous Scores", cột "Hours Studied" và một cột mới chứa tích của "Previous Scores" và "Hours Studied" (cột tương tác).
- Cách hoạt động:
  - np.ones((X.shape[0], 1)): Tạo một mảng có X.shape[0] hàng và 1 cột, tất cả các giá trị là 1.
  - X["Previous Scores"].to_frame(): Chuyển cột "Previous Scores" thành một DataFrame.
  - X["Hours Studied"].to_frame(): Chuyển cột "Hours Studied" thành một DataFrame.

- (X["Previous Scores"] * X["Hours Studied"]).to_frame(name="Interaction"): Tạo cột tương tác bằng cách nhân "Previous Scores" với "Hours Studied" và đặt tên cho cột là "Interaction".
- np.hstack(...): Kết hợp các cột vừa tạo thành một mảng NumPy 2D.

23. Hàm model3(X)

- Chức năng: Tạo một ma trận thiết kế với cột hệ số chặn, các đặc trưng "Previous Scores", "Hours Studied", cùng với các cột tương tác và biến đổi phi tuyến tính của một số đặc trưng khác.
- Đầu vào: "**X**" - DataFrame chứa các đặc trưng của tập dữ liệu (Previous Scores, Hours Studied, Sleep Hours, Extracurricular Activities, Sample Question Papers Practiced).
- Đầu ra: Trả về một mảng NumPy 2D với cột đầu tiên toàn giá trị 1 để đại diện cho hệ số chặn, cột "Previous Scores", cột "Hours Studied", một cột mới chứa tích của "Previous Scores" và "Hours Studied" (cột tương tác), cột căn bậc hai "Extracurricular Activities", cột căn bậc hai của "Sleep Hours" và cột căn bậc hai của "Sample Question Papers Practiced".
- Cách hoạt động:
  - np.ones((X.shape[0], 1)): Tạo một mảng có X.shape[0] hàng và 1 cột, tất cả các giá trị là 1.
  - interaction_term = (X["Previous Scores"] * X["Hours Studied"]).to_frame(name="Interaction"): Tạo cột tương tác bằng cách nhân "Previous Scores" với "Hours Studied" và đặt tên cho cột là "Interaction".
  - np.sqrt(X["Sleep Hours"].to_frame()): Tạo cột mới bằng cách lấy căn bậc hai của "Sleep Hours".
  - np.sqrt(X["Extracurricular Activities "].to_frame()): Tạo cột mới bằng cách lấy căn bậc hai của " Extracurricular Activities".
  - np.sqrt(X["Sample Question Papers Practiced"].to_frame()): Tạo cột mới bằng cách lấy căn bậc hai của "Sample Question Papers Practiced".
  - np.hstack(...): Kết hợp tất cả các cột thành một mảng NumPy 2D.

PHÂN TÍCH DỮ LIỆU

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
count	9000.000000	9000.000000	9000.000000	9000.000000	9000.000000	9000.000000
mean	4.976444	69.396111	0.493667	6.535556	4.590889	55.136333
std	2.594647	17.369957	0.499988	1.695533	2.864570	19.187669
min	1.000000	40.000000	0.000000	4.000000	0.000000	10.000000
25%	3.000000	54.000000	0.000000	5.000000	2.000000	40.000000
50%	5.000000	69.000000	0.000000	7.000000	5.000000	55.000000
75%	7.000000	85.000000	1.000000	8.000000	7.000000	70.000000
max	9.000000	99.000000	1.000000	9.000000	9.000000	100.000000

Phân tích:

- Có thể thấy count của tất cả đặc trưng đều bằng 9000 => không có dữ liệu không bị thiếu null trong bất kì biến nào
- Hours Studied (Số giờ học):
  - Trung bình số giờ học là 4.98 giờ với độ lệch chuẩn là 2.59. Điều này cho thấy mức độ phân tán của thời gian học tập khá lớn.
  - Mức trung vị (50%) là 5 giờ, cho thấy một nửa số học sinh dành ít hơn hoặc bằng 5 giờ để học.
  - Khoảng giữa của 50% học sinh (khoảng giữa từ 25% đến 75%) là từ 3 đến 7 giờ.
- Previous Scores (Điểm trước đây):
  - Điểm trung bình là 69.4 với độ lệch chuẩn là 17.37. Điều này cho thấy mức độ biến thiên của điểm số khá cao.
  - Trung vị cũng là 69, cho thấy phân phối điểm số có thể gần như đối xứng.
  - Phân vị thứ 25 và 75 là 54 và 85, cho thấy phần lớn học sinh có điểm số nằm trong khoảng này.
- Extracurricular Activities (Hoạt động ngoại khóa):
  - Trung bình là 0.49, gần với 0.5, có nghĩa là số lượng học sinh tham gia và không tham gia hoạt động ngoại khóa gần như bằng nhau.
  - Vì biến này có giá trị nhị phân (0 hoặc 1), độ lệch chuẩn cao cho thấy có sự khác biệt rõ ràng giữa hai nhóm.
- Sleep Hours (Số giờ ngủ):

- Trung bình số giờ ngủ là 6.54 giờ với độ lệch chuẩn là 1.7 giờ, cho thấy học sinh có giấc ngủ tương đối ổn định.
- Trung vị là 7 giờ, cho thấy một nửa số học sinh ngủ ít hơn hoặc bằng 7 giờ.
- Khoảng 50% học sinh ngủ từ 5 đến 8 giờ.

➢ Sample Question Papers Practiced (Số bài thi mẫu được làm):
- Trung bình là 4.59 bài với độ lệch chuẩn là 2.86, cho thấy số lượng bài thi mẫu mà học sinh làm có sự biến động lớn.
- Trung vị là 5 bài, cho thấy phần lớn học sinh làm ít hơn hoặc bằng 5 bài thi mẫu.
- Khoảng giữa của 50% học sinh là từ 2 đến 7 bài.

➢ Performance Index (Chỉ số hiệu suất):
- Trung bình là 55.14 với độ lệch chuẩn là 19.19, cho thấy sự khác biệt lớn trong hiệu suất của học sinh.
- Trung vị là 55, cho thấy phân phối của chỉ số hiệu suất có thể gần như đối xứng.
- Phân vị thứ 25 và 75 là 40 và 70, nghĩa là 50% học sinh có chỉ số hiệu suất nằm trong khoảng này.

Nhận xét:

➢ Các biến như "Previous Scores", "Sleep Hours", và "Performance Index" có sự phân phối khá đồng đều với trung bình gần trung vị, cho thấy phân phối gần với phân phối chuẩn.

➢ Các biến như "Extracurricular Activities" cho thấy rõ ràng sự phân tách giữa hai nhóm (tham gia và không tham gia).

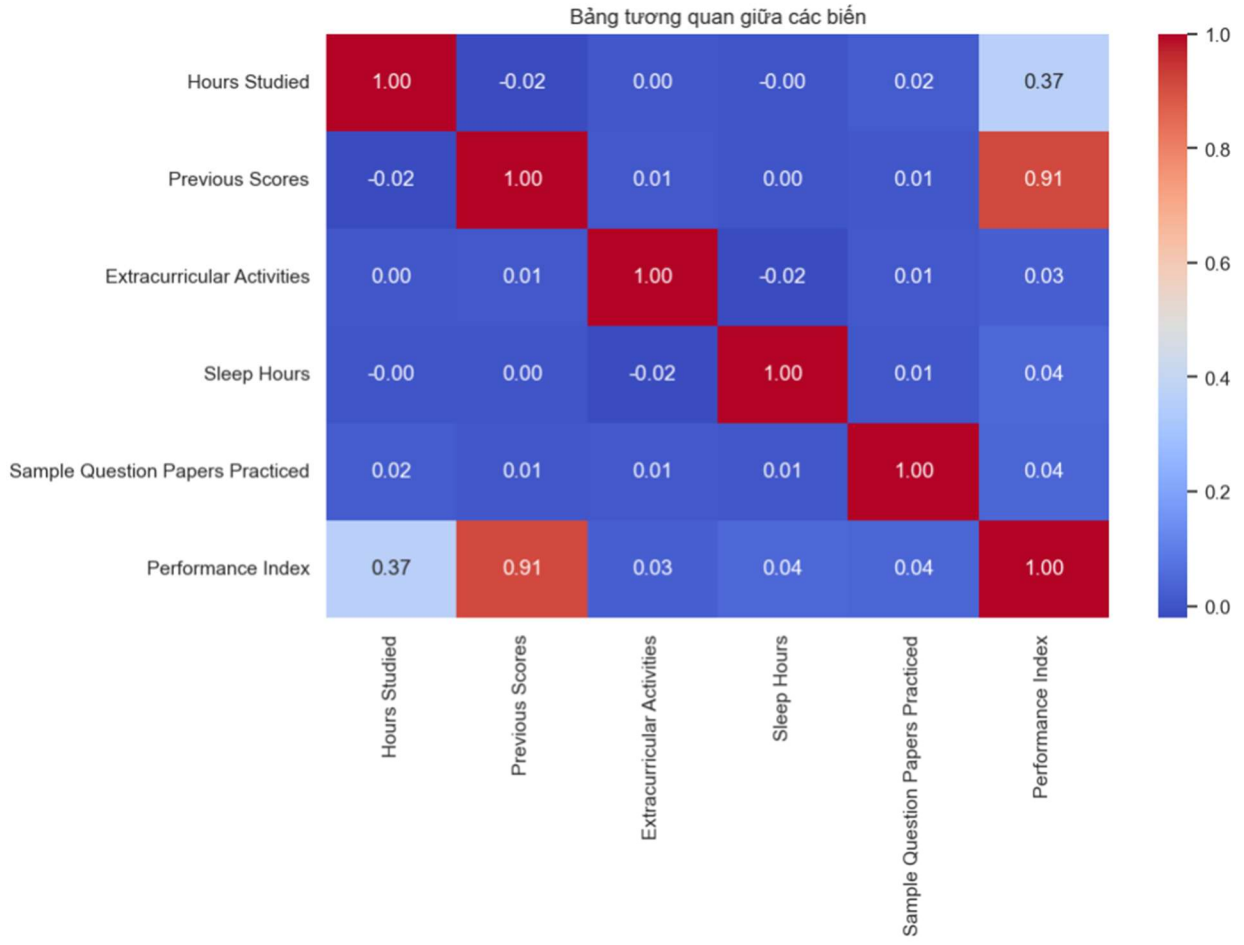
➢ "Sample Question Papers Practiced" có sự phân tán cao, cho thấy các học sinh có chiến lược ôn tập rất khác nhau.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9000 entries, 0 to 8999
Data columns (total 6 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Hours Studied                   9000 non-null   int64
 1   Previous Scores                 9000 non-null   int64
 2   Extracurricular Activities      9000 non-null   int64
 3   Sleep Hours                     9000 non-null   int64
 4   Sample Question Papers Practiced  9000 non-null   int64
 5   Performance Index               9000 non-null   float64
dtypes: float64(1), int64(5)
memory usage: 422.0 KB
```

Phân tích:

- Tất cả các cột đều có 9000 entries, và không có giá trị null trong bất kỳ cột nào. Điều này xác nhận rằng không có dữ liệu bị thiếu trong tập dữ liệu.
- Các cột Hours Studied, Previous Scores, Extracurricular Activities, Sleep Hours, và Sample Question Papers Practiced đều là kiểu int64, nghĩa là các giá trị trong các cột này đều là số nguyên. Đây là các dữ liệu rời rạc.
- Cột Performance Index là kiểu float64, nghĩa là giá trị trong cột này có thể là số thập phân. Điều này hợp lý vì chỉ số hiệu suất thường có thể là một giá trị trung bình hoặc được tính toán dựa trên một công thức.
- Bộ dữ liệu có tổng cộng 9000 hàng và 6 cột, sử dụng 422.0 KB bộ nhớ. Điều này cho thấy tập dữ liệu của bạn không quá lớn, dễ dàng để quản lý và phân tích trên các hệ thống thông thường mà không yêu cầu tối ưu hóa bộ nhớ đặc biệt.

Nhận xét: Bộ dữ liệu này được tổ chức tốt với các kiểu dữ liệu phù hợp và không có dữ liệu bị thiếu



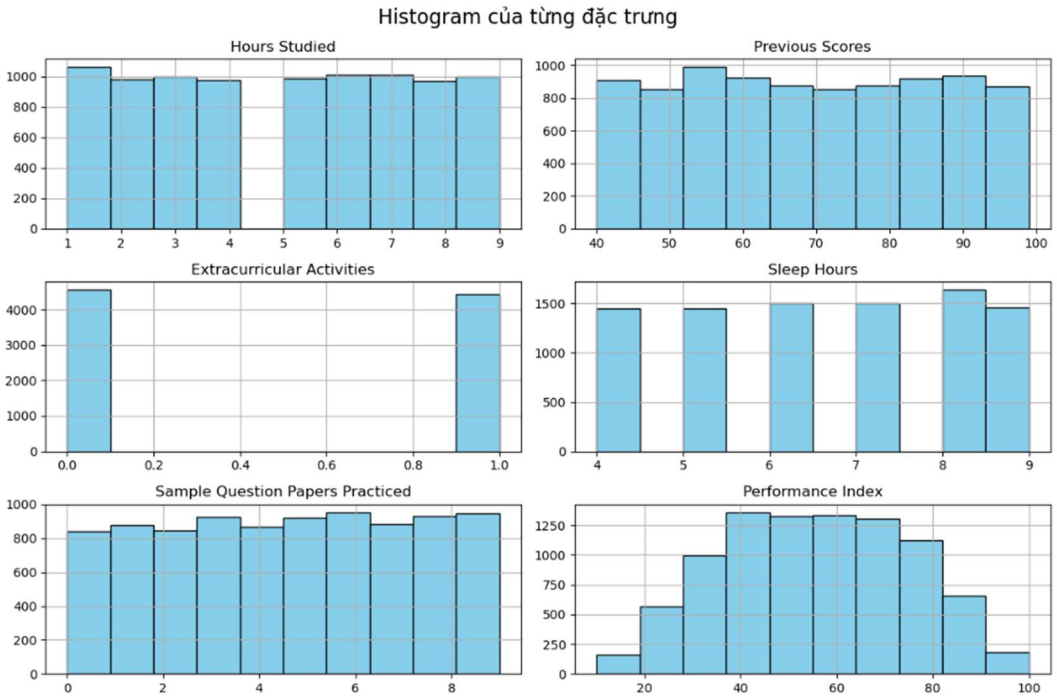
Phân tích:

- Có một mối tương quan rất mạnh giữa **Previous Scores** và **Performance Index** với hệ số tương quan là **0.91**. Điều này cho thấy rằng những học sinh có điểm số cao trong các kỳ trước thường có chỉ số hiệu suất cao. Điều này có thể là do các yếu tố như khả năng duy trì hiệu suất học tập hoặc tiếp tục áp dụng các phương pháp học tập hiệu quả.
- Mối tương quan giữa **Hours Studied** và **Performance Index** có giá trị **0.37**, cho thấy có một mối tương quan dương nhưng không quá mạnh giữa số giờ học và chỉ số hiệu suất. Điều này có nghĩa là việc học nhiều giờ hơn có thể góp phần tăng cường hiệu suất học tập, nhưng nó không phải là yếu tố quyết định duy nhất.
- Các biến khác như **Extracurricular Activities**, **Sleep Hours**, và **Sample Question Papers Practiced** đều có mối tương quan rất thấp với **Performance Index** (khoảng từ 0.03 đến 0.04). Điều này có thể cho thấy rằng các yếu tố này không có nhiều ảnh hưởng trực tiếp đến chỉ số hiệu suất của học sinh.
- Nhìn chung, các cặp biến khác cũng không có mối tương quan đáng kể với nhau (đều có hệ số tương quan gần 0), cho thấy rằng chúng hoạt động tương đối độc lập trong việc ảnh hưởng đến chỉ số hiệu suất. Do đó phần vẽ biểu đồ sau chỉ thực hiện giữa các đặc trưng với **Performance Index**.

Nhận xét:

- **Previous Scores** là yếu tố quan trọng nhất ảnh hưởng đến **Performance Index**, với mối tương quan mạnh mẽ.
- **Hours Studied** có tác động đến **Performance Index**, nhưng không mạnh bằng **Previous Scores**.
- Các yếu tố khác như **Extracurricular Activities**, **Sleep Hours**, và **Sample Question Papers Practiced** có tác động không đáng kể đến hiệu suất học tập.

<Figure size 1200x800 with 0 Axes>



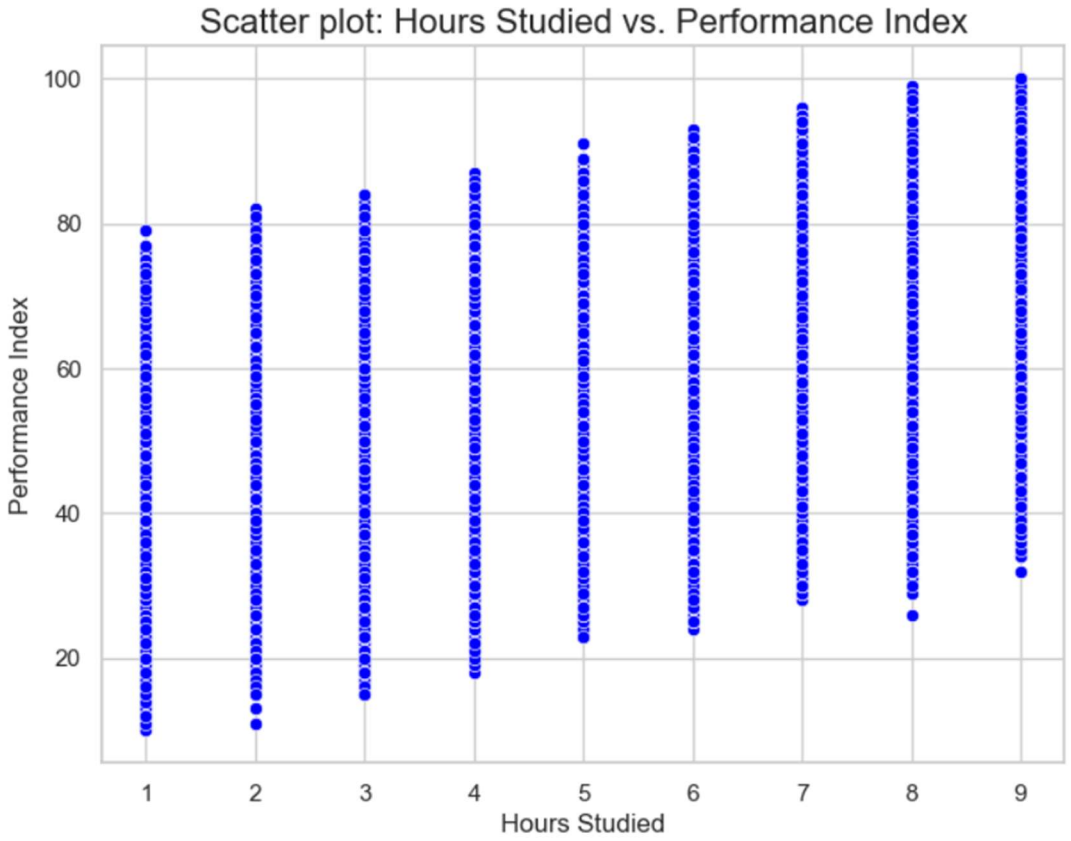
Phân tích:

- Biểu đồ phân phối của số giờ học khá đồng đều, với mỗi khoảng thời gian từ 1 đến 9 giờ đều có số lượng học sinh tương đối tương đương (khoảng từ 800 đến 1000 học sinh). Điều này cho thấy rằng học sinh trong tập dữ liệu có thời gian học khá đa dạng, không có nhóm nào quá chênh lệch về số giờ học.
- Phân phối điểm số trước đây cũng khá đồng đều từ 40 đến 100, với một số đỉnh cao tại các khoảng 50-60 và 80-90. Điều này chỉ ra rằng học sinh có mức điểm khá phân tán, không tập trung vào một mức cụ thể, với sự phân phối điểm số trải đều trên toàn bộ phổ điểm.
- Biểu đồ cho thấy rõ ràng rằng có hai nhóm học sinh, một nhóm tham gia hoạt động ngoại khóa (1) và một nhóm không tham gia (0), với số lượng gần như bằng nhau (khoảng 4500 học sinh mỗi nhóm). Điều này phù hợp với kết luận trước đó rằng số lượng học sinh tham gia và không tham gia hoạt động ngoại khóa gần như bằng nhau.
- Phân phối số giờ ngủ cho thấy sự tập trung vào khoảng 6-8 giờ ngủ mỗi đêm, với đỉnh tại 8 giờ. Điều này chỉ ra rằng phần lớn học sinh có giấc ngủ khoa học trong khoảng này, cho thấy một sự nhất quán tương đối về thời gian ngủ giữa các học sinh.

- Biểu đồ phân phối số bài thi mẫu được làm tương đối đồng đều từ 0 đến 9 bài, cho thấy rằng học sinh có sự phân bổ khác nhau trong việc ôn tập, từ không làm bài thi mẫu nào cho đến làm đến 9 bài.
- Biểu đồ chỉ số hiệu suất có hình dạng gần như phân phối chuẩn (bell curve), với đỉnh tại khoảng 50-60 điểm. Điều này cho thấy rằng phần lớn học sinh có hiệu suất trung bình, với một số ít học sinh có hiệu suất rất cao hoặc rất thấp.

Nhận xét:

- Nhiều biến số như số giờ học, số bài thi mẫu được làm, và điểm trước đây cho thấy sự phân phối đồng đều, chỉ ra rằng các yếu tố này có sự đa dạng trong tập dữ liệu.
- Chỉ số hiệu suất có dạng phân phối chuẩn, điều này phù hợp với kỳ vọng về một biến số đo lường hiệu suất.
- Có sự phân tách rõ ràng giữa những học sinh tham gia và không tham gia hoạt động ngoại khóa.
- Phần lớn học sinh có giấc ngủ từ 6 đến 8 giờ, cho thấy sự tương đồng trong thói quen ngủ.

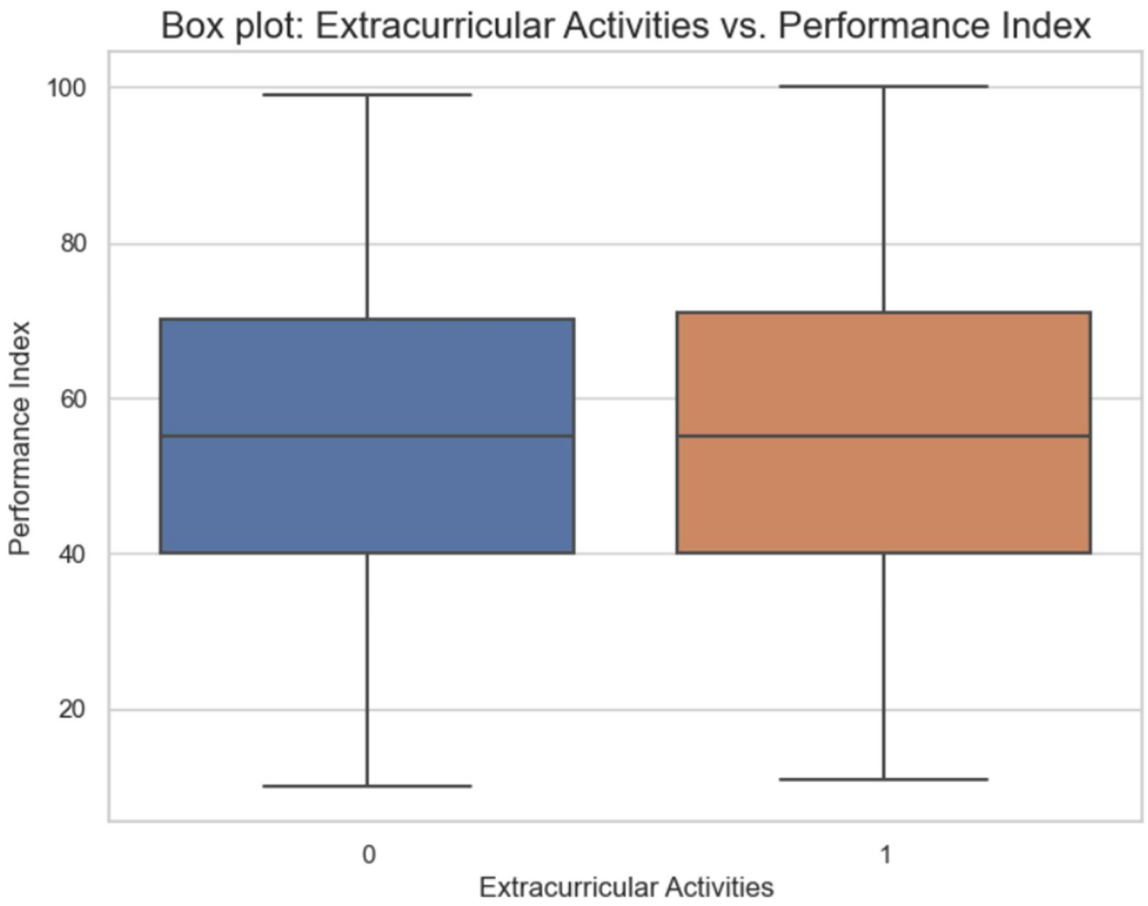


Phân tích:

- Biểu đồ cho thấy một mối quan hệ dương giữa **Hours Studied** và **Performance Index**. Khi số giờ học tăng lên, chỉ số hiệu suất có xu hướng tăng. Điều này cho thấy rằng học sinh học nhiều giờ hơn thường có chỉ số hiệu suất cao hơn.
- Các điểm phân tán trên biểu đồ cho thấy rằng học sinh với cùng một số giờ học có thể có chỉ số hiệu suất rất khác nhau. Chẳng hạn, với 6 giờ học, chỉ số hiệu suất có thể nằm trong khoảng từ 20 đến 100. Điều này cho thấy rằng ngoài số giờ học, còn nhiều yếu tố khác ảnh hưởng đến chỉ số hiệu suất.
- Dường như có hiệu ứng giảm dần khi số giờ học tăng lên. Cụ thể, với số giờ học từ 1 đến 5, mức độ tăng của chỉ số hiệu suất khá rõ ràng. Tuy nhiên, từ 6 đến 9 giờ học, chỉ số hiệu suất không tăng mạnh nữa, và có xu hướng phân tán rộng hơn. Điều này có thể chỉ ra rằng sau một mức độ nhất định, việc tăng thêm giờ học không còn làm tăng hiệu suất một cách tương ứng nữa.
- Mặc dù số giờ học cao hơn có liên quan đến chỉ số hiệu suất cao hơn, nhưng biểu đồ cũng cho thấy sự đa dạng về hiệu suất trong mỗi nhóm số giờ học. Ví dụ, ngay cả học sinh học 9 giờ cũng có thể có chỉ số hiệu suất từ khoảng 20 đến 100.

Nhận xét:

- Biểu đồ này chỉ ra rằng có một mối quan hệ dương giữa số giờ học và chỉ số hiệu suất, nhưng không hoàn toàn tuyến tính.
- Hiệu ứng tăng thêm giờ học không làm tăng chỉ số hiệu suất theo một tỷ lệ nhất định, đặc biệt là khi số giờ học đã vượt qua mức 5-6 giờ.
- Có nhiều yếu tố khác ngoài số giờ học có thể ảnh hưởng đến hiệu suất của học sinh, điều này được thể hiện qua sự phân tán rộng của chỉ số hiệu suất trong mỗi nhóm giờ học.

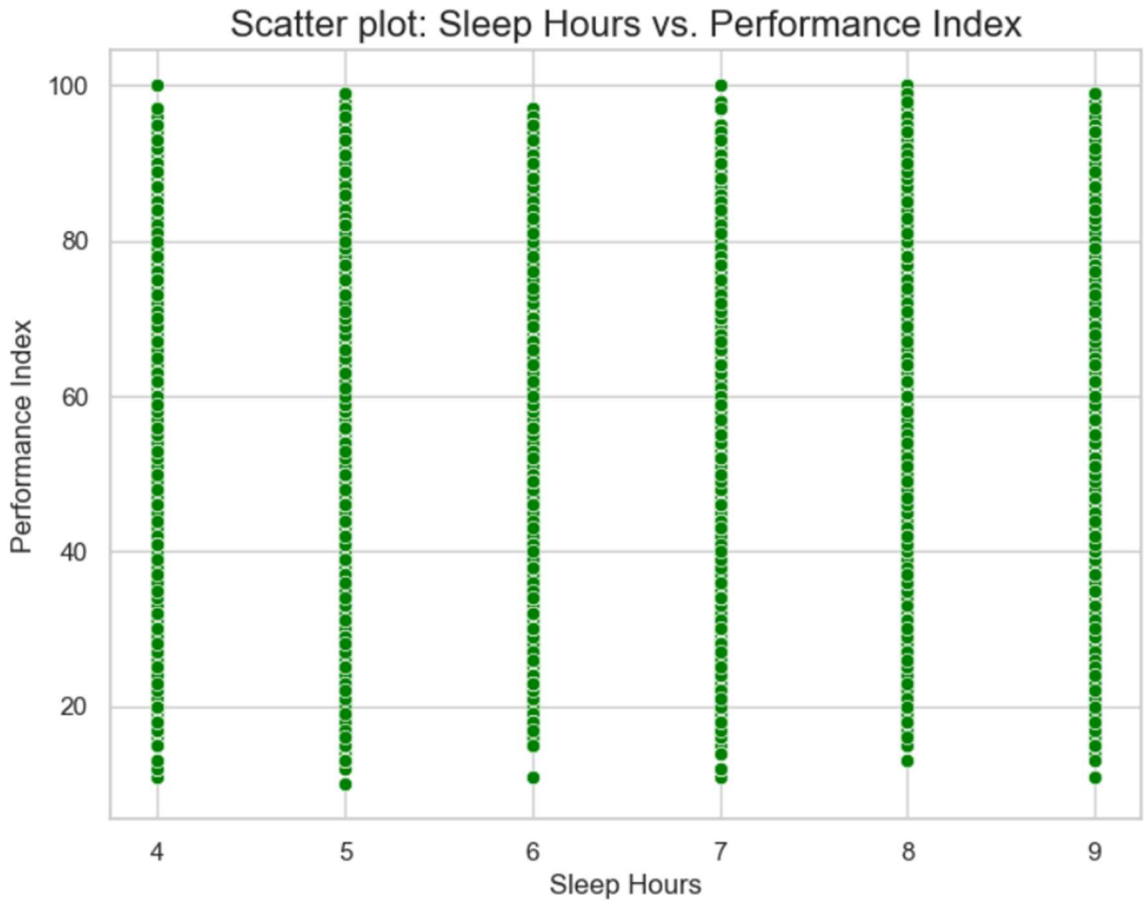


Phân tích:

- Trung vị của chỉ số hiệu suất ở cả hai nhóm (có tham gia và không tham gia hoạt động ngoại khóa) là khá tương đương nhau, đều khoảng 60. Điều này cho thấy rằng việc tham gia hay không tham gia hoạt động ngoại khóa không có ảnh hưởng lớn đến trung bình hiệu suất học tập của học sinh.
- Phạm vi của các giá trị giữa phân vị thứ 25 (Q1) và phân vị thứ 75 (Q3) cũng tương đương giữa hai nhóm, cho thấy sự phân tán của hiệu suất học tập giữa những học sinh tham gia và không tham gia hoạt động ngoại khóa là tương tự nhau.
- Độ dài của các đuôi phân phối (whiskers) cũng tương đối tương đồng, kéo dài từ khoảng 20 đến 100. Điều này cho thấy cả hai nhóm đều có phạm vi chỉ số hiệu suất khá rộng.
- Biểu đồ không có dấu hiệu của các dữ liệu ngoại lệ (outliers), điều này chỉ ra rằng hầu hết các giá trị nằm trong phạm vi phân bố thông thường.

Nhận xét:

- Việc tham gia hay không tham gia hoạt động ngoại khóa dường như không có ảnh hưởng rõ ràng đến chỉ số hiệu suất học tập của học sinh. Cả hai nhóm đều có phân bố chỉ số hiệu suất khá giống nhau về trung vị, độ phân tán, và phạm vi dữ liệu.
- Điều này có thể gợi ý rằng các yếu tố khác (như số giờ học hoặc điểm số trước đây) có thể đóng vai trò quan trọng hơn trong việc ảnh hưởng đến hiệu suất học tập so với việc tham gia hoạt động ngoại khóa. Tuy nhiên, hoạt động ngoại khóa có thể có các lợi ích khác mà chỉ số hiệu suất không thể đo lường trực tiếp.



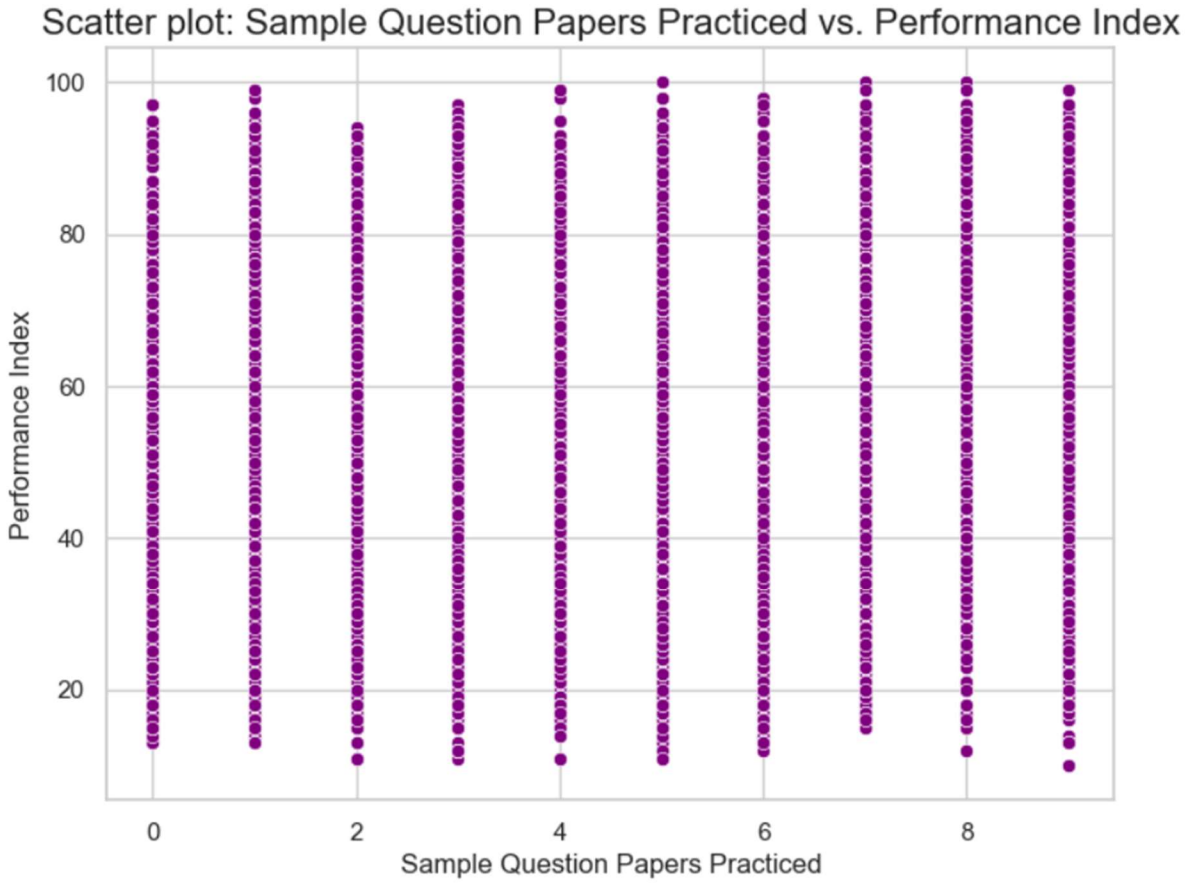
Phân tích:

- Biểu đồ cho thấy các điểm phân tán khá đồng đều và không có xu hướng rõ ràng giữa số giờ ngủ và chỉ số hiệu suất. Điều này chỉ ra rằng không có mối quan hệ mạnh mẽ giữa số giờ ngủ và hiệu suất học tập. Việc ngủ nhiều hay ít dường như không ảnh hưởng trực tiếp đến chỉ số hiệu suất của học sinh.

- Các điểm phân tán đều trải dài từ giá trị thấp đến giá trị cao của chỉ số hiệu suất, cho thấy rằng học sinh có hiệu suất từ thấp đến cao đều có số giờ ngủ khác nhau, từ 4 đến 9 giờ. Không có bất kỳ sự tập trung nào ở một mức hiệu suất cụ thể liên quan đến số giờ ngủ.
- Ở mỗi mức giờ ngủ, chỉ số hiệu suất dao động từ khoảng 20 đến 100. Điều này chỉ ra rằng số giờ ngủ không phải là một yếu tố quyết định lớn đối với hiệu suất học tập, vì học sinh có thể đạt hiệu suất cao hoặc thấp dù họ ngủ bao nhiêu giờ đi chăng nữa.

Nhận xét:

- Biểu đồ này cho thấy rằng số giờ ngủ không có mối liên hệ rõ ràng với chỉ số hiệu suất học tập. Các học sinh có thể có hiệu suất học tập cao hoặc thấp dù số giờ ngủ là bao nhiêu.
- Việc thiếu mối quan hệ rõ ràng có thể gợi ý rằng các yếu tố khác, chẳng hạn như chất lượng giấc ngủ, phương pháp học tập, hoặc các yếu tố cá nhân khác, có thể quan trọng hơn trong việc ảnh hưởng đến hiệu suất học tập.

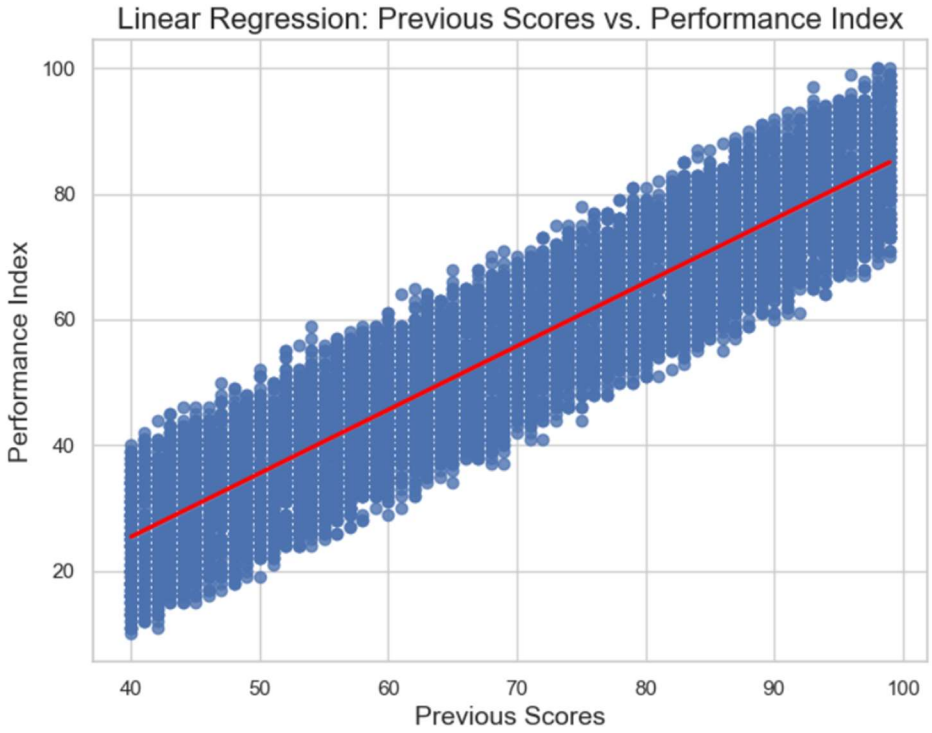


Phân tích:

- Giống như biểu đồ giữa số giờ ngủ và chỉ số hiệu suất, biểu đồ này cũng không cho thấy một mối quan hệ mạnh mẽ giữa số bài thi mẫu được làm và chỉ số hiệu suất. Các điểm phân tán rất đều, cho thấy rằng học sinh có thể đạt các mức hiệu suất khác nhau dù họ có làm nhiều hay ít bài thi mẫu.
- Dữ liệu cho thấy các học sinh có hiệu suất học tập đa dạng (từ thấp đến cao) ở mỗi mức số lượng bài thi mẫu được làm. Tức là dù làm 0 bài, 4 bài, hay 8 bài, học sinh đều có thể có chỉ số hiệu suất nằm trong khoảng từ 20 đến 100.
- Tại mỗi mức số lượng bài thi mẫu được làm, chỉ số hiệu suất có một dải giá trị rộng, từ rất thấp đến rất cao. Điều này cho thấy rằng việc làm nhiều bài thi mẫu không đảm bảo rằng học sinh sẽ có hiệu suất cao, và ngược lại.

Nhận xét:

- **Không có mối liên hệ rõ ràng** giữa số bài thi mẫu được làm và chỉ số hiệu suất, chỉ ra rằng số lượng bài thi mẫu được làm không phải là yếu tố quyết định quan trọng đối với hiệu suất học tập.
- **Sự phân tán dữ liệu rộng** cho thấy rằng hiệu suất học tập của học sinh phụ thuộc vào nhiều yếu tố khác ngoài số lượng bài thi mẫu, chẳng hạn như kỹ năng học tập, sự chuẩn bị, và các yếu tố cá nhân khác.



Phân tích:

- Đường hồi quy tuyến tính màu đỏ cho thấy một mối quan hệ tuyến tính dương giữa điểm số trước đây và chỉ số hiệu suất. Điều này có nghĩa là khi điểm số trước đây của học sinh tăng lên, chỉ số hiệu suất cũng có xu hướng tăng theo. Đây là một mối quan hệ tương quan mạnh mẽ, như đã thấy ở các phân tích trước đó với hệ số tương quan rất cao (0.91).
- Mặc dù có mối quan hệ tuyến tính rõ ràng, nhưng có một số sự phân tán trong dữ liệu. Ví dụ, tại một mức điểm số cụ thể (chẳng hạn như 70 hoặc 80), chỉ số hiệu suất có thể dao động từ khoảng 40 đến 100. Điều này cho thấy rằng mặc dù điểm số trước đây là một yếu tố quan trọng, nhưng không phải là yếu tố duy nhất quyết định đến hiệu suất học tập.
- Khi điểm số trước đây tăng từ 40 đến 100, chỉ số hiệu suất cũng tăng từ khoảng 20 đến 100, cho thấy rằng những học sinh có điểm số cao trước đây thường có chỉ số hiệu suất cao hơn trong học tập.
- Đường hồi quy phù hợp tốt với dữ liệu, cho thấy rằng mô hình hồi quy tuyến tính là một cách tiếp cận hợp lý để mô hình hóa mối quan hệ giữa điểm số trước đây và chỉ số hiệu suất.

Nhận xét:

- Biểu đồ này khẳng định rằng có một mối quan hệ rất mạnh giữa điểm số trước đây và chỉ số hiệu suất, với điểm số trước đây cao hơn thường dẫn đến chỉ số hiệu suất cao hơn.
- Mặc dù mối quan hệ là rõ ràng, sự phân tán dữ liệu cho thấy có các yếu tố khác ngoài điểm số trước đây cũng đóng vai trò quan trọng trong việc quyết định hiệu suất học tập.
- Đường hồi quy tuyến tính là một mô hình thích hợp để dự đoán chỉ số hiệu suất dựa trên điểm số trước đây, nhưng nên xem xét thêm các yếu tố khác nếu muốn có mô hình dự đoán toàn diện hơn.

BÁO CÁO VÀ NHẬN XÉT KẾT QUẢ

Yêu cầu 2a: Xây dựng mô hình sử dụng toàn bộ 5 đặc trưng:

- Huấn luyện mô hình hồi quy tuyến tính với tập dữ liệu huấn luyện X_train và y_train sau khi đã được xử lý trước bằng hàm preprocess() như trình bày trên miêu tả hàm. Hàm preprocess() đảm bảo rằng dữ liệu đầu vào không bị lệch (bias) và phù hợp để đưa vào mô hình.
- Thực hiện việc huấn luyện mô hình sử dụng phương pháp hồi quy tuyến tính bằng cách gọi phương thức fit trên mô hình OLSLinearRegression(). Quá trình này sẽ tìm ra bộ trọng số tối ưu cho các biến độc lập, từ đó giúp giảm thiểu sai số giữa giá trị dự đoán và giá trị thực tế trên tập huấn luyện.
- Sau khi mô hình đã được huấn luyện, các trọng số được lấy ra bằng cách sử dụng phương thức get_params(). Những trọng số này phản ánh mức độ ảnh hưởng của từng biến độc lập lên biến mục tiêu trong mô hình.

$$w = [-33.969 \;\; 2.852 \;\; 1.018 \;\; 0.604 \;\; 0.474 \;\; 0.192]$$

- Tiến hành dự đoán giá trị y_pred trên tập kiểm tra X_test sau khi đã xử lý trước tương tự như tập huấn luyện. Quá trình dự đoán được thực hiện bằng phương thức predict() của mô hình.
- Để đánh giá mô hình, tính toán sai số trung bình tuyệt đối (MAE) giữa các giá trị dự đoán y_pred và giá trị thực tế y_test trên tập kiểm tra. MAE là chỉ số đánh giá độ chính xác của mô hình, phản ánh mức độ chênh lệch trung bình giữa giá trị thực tế và giá trị dự đoán.

$$MAE = 1.596$$

⇨ Với 5 đặc trưng thì xây dựng được mô hình có công thức hồi quy tuyến tính:

$$Student\ Performance = -33.969 + (2.852 * Hours\ Studied) + (1.018 * Previous\ Scores) + (0.604 * Extracurricular\ Activities) + (0.474 * Sleep\ Hours) + (0.192 * Sample\ Question\ Papers\ Practiced)$$

Nhận xét: Đây là mô hình đơn giản khi xem 5 đặc trưng đều ảnh hưởng đến kết quả học tập tương lai và cũng có MAE thấp nhất tránh được sự phức tạp không cần thiết và các vấn đề như overfitting hay đa cộng tuyến và cho thấy khá ít dữ liệu nhiễu trong test.csv. [1[1]][2]

[1] Chương 7 giải thích về bias-variance tradeoff và cách mô hình đơn giản thường hoạt động tốt hơn

Yêu cầu 2b: Xây dựng mô hình sử dụng 1 đặc trưng:

- Khởi tạo biến label bằng hàm X_train.columns.values() đã trình bày ở trên để lấy tên các đặc trưng, đồng thời tạo list_mae trống để lưu kết quả MAE các đặc trưng và n_splits để chia số lượng phần mà tập dữ liệu được chia ở đây là 5.
- Xáo trộn dữ liệu thành 5 bộ đề chung cho mỗi đặc trưng nhằm đáp ứng tính chính xác của việc đánh giá mô hình bằng các hàm:
  - indices = np.arange(X_train.shape[0]): Tạo một mảng các chỉ số từ 0 đến số lượng hàng trong X_train.
  - np.random.shuffle(indices): Xáo trộn ngẫu nhiên các chỉ số trong indices.
  - X_train_shuffled = X_train.iloc[indices].reset_index(drop=True): Sử dụng các chỉ số đã xáo trộn để sắp xếp lại các hàng của X_train và đặt lại chỉ số (index).
  - y_train_shuffled = y_train.iloc[indices].reset_index(drop=True): Tương tự, sắp xếp lại các hàng của y_train và đặt lại chỉ số.
  - X_splits = np.array_split(X_train_shuffled, n_splits): Chia X_train_shuffled thành 5 phần bằng nhau hoặc gần bằng nhau và lưu vào X_splits.
  - y_splits = np.array_split(y_train_shuffled, n_splits): Tương tự, chia y_train_shuffled thành 5 phần và lưu vào y_splits.
- Lặp qua từng mô hình 1 đặc trưng trong label và tạo mae_list trống để lưu giá trị MAE cho từng lần lặp bộ đề nhằm chọn ra MAE nhỏ nhất để xét với các đặc trưng khác. Lặp qua từng bộ đề, với quy tắc là 1 đề test test_X = X_splits[i][[name]]: Lấy cột đặc trưng name từ phần thứ i trong X_splits làm tập kiểm tra, tương tự cho test_y và 4 đề để train train_X = pd.concat([X_splits[j][[name]] for j in range(n_splits) if j != i]): Kết hợp các phần còn lại của X_splits, ngoại trừ phần thứ i, để làm tập huấn luyện, tương tự cho train_y. (kfold cross-validation) [5]
- Tiền xử lý dữ liệu bằng hàm preprocess() thêm cột đầu tiên toàn giá trị 1 trước khi huấn luyện mô hình và dự đoán. Từ đó, tính toán mae bằng hàm mae_cal() rồi lưu vào mae_list.
- Tìm giá trị nhỏ nhất trong mae_list để thêm vào list_mae một cặp tên đặc trưng và giá trị mae tương ứng nhỏ nhất.
- Tìm mô hình 1 đặc trưng tốt nhất và tạo bảng kết quả.

STT	Mô hình 1 đặc trưng	MAE
1	Hours Studied	15.342
2	Previous Scores	6.515
3	Extracurricular Activities	16.050
4	Sleep Hours	16.023
5	Sample Question Papers Practiced	16.044

⇨ Mô hình 1 đặc trưng tốt nhất là "Previous Scores" cho thấy "Previous Scores" có ảnh hưởng lớn đến với kết quả "Performance Index"

- Huấn luyện mô hình 1 đặc trưng tốt nhất trên tập test.csv, lấy trọng số bằng phương thức get_params()

$$w = [-14.989 \quad 1.011]$$

- Tiến hành dự đoán giá trị y_pred trên tập kiểm tra X_test sau khi đã xử lý trước tương tự như tập huấn luyện. Quá trình dự đoán được thực hiện bằng phương thức predict() của mô hình.
- Để đánh giá mô hình, tính toán sai số trung bình tuyệt đối (MAE) giữa các giá trị dự đoán y_pred và giá trị thực tế y_test trên tập kiểm tra. MAE là chỉ số đánh giá độ chính xác của mô hình, phản ánh mức độ chênh lệch trung bình giữa giá trị thực tế và giá trị dự đoán.

$$MAE = 6.544$$

⇨ Với 1 đặc trưng "Previous Scores" thì xây dựng được mô hình có công thức hồi quy tuyến tính:

$$Student\ Performance = -14.989 + 1.011 * Previos\ Scores$$

Giả thuyết:

Previous Scores (Điểm số trước đây) là yếu tố có mối tương quan mạnh mẽ nhất với **Performance Index** (Chỉ số hiệu suất), như đã thấy từ biểu đồ hồi quy tuyến tính và phân tích tương quan. Điều này ngụ ý rằng **Previous Scores** có khả năng là một chỉ báo đáng tin cậy cho hiệu suất học tập tương lai của học sinh.

- Biểu đồ hồi quy tuyến tính đã chỉ ra rằng có một mối quan hệ tuyến tính rõ ràng giữa **Previous Scores** và **Performance Index**. Điều này có nghĩa là **Previous Scores** không chỉ là một biến dự đoán quan trọng mà còn là một

biến dự đoán mạnh mẽ, vì nó có thể giải thích một phần lớn biến động trong **Performance Index**.

- So với các đặc trưng khác, như **Hours Studied** hoặc **Sample Question Papers Practiced**, **Previous Scores** đã chứng tỏ khả năng cao hơn trong việc dự đoán hiệu suất học tập. Điều này là bởi vì điểm số trước đây đã phản ánh một cách toàn diện kết quả học tập trước đó của học sinh, bao gồm kiến thức, kỹ năng và các yếu tố khác như khả năng quản lý thời gian và sự cam kết học tập.

Kết luận: Do vậy, khi thực hiện cross-validation, việc chọn **Previous Scores** làm đặc trưng duy nhất trong mô hình có thể mang lại giá trị MAE (Mean Absolute Error) thấp nhất, bởi vì đây là đặc trưng có tính dự đoán tốt nhất. Các đặc trưng khác mặc dù có thể có mối tương quan với **Performance Index**, nhưng không đủ mạnh hoặc không đủ trực tiếp để làm giảm MAE hiệu quả như **Previous Scores**.

Yêu cầu 2c: Xây dựng/ thiết kế mô hình cho kết quả tốt nhất:

- Khởi tạo biến label bằng hàm X_train.columns.values() đã trình bày ở trên để lấy tên các đặc trưng, đồng thời tạo list_mae trống để lưu kết quả MAE các đặc trưng và n_splits để chia số lượng phần mà tập dữ liệu được chia ở đây là 5.
- Xáo trộn dữ liệu thành 5 bộ đề chung cho mỗi đặc trưng nhằm đáp ứng tính chính xác của việc đánh giá mô hình bằng các hàm:
  - indices = np.arange(X_train.shape[0]): Tạo một mảng các chỉ số từ 0 đến số lượng hàng trong X_train.
  - np.random.shuffle(indices): Xáo trộn ngẫu nhiên các chỉ số trong indices.
  - X_train_shuffled = X_train.iloc[indices].reset_index(drop=True): Sử dụng các chỉ số đã xáo trộn để sắp xếp lại các hàng của X_train và đặt lại chỉ số (index).
  - y_train_shuffled = y_train.iloc[indices].reset_index(drop=True): Tương tự, sắp xếp lại các hàng của y_train và đặt lại chỉ số.
  - X_splits = np.array_split(X_train_shuffled, n_splits): Chia X_train_shuffled thành 5 phần bằng nhau hoặc gần bằng nhau và lưu vào X_splits.
  - y_splits = np.array_split(y_train_shuffled, n_splits): Tương tự, chia y_train_shuffled thành 5 phần và lưu vào y_splits.

- Lặp qua từng mô hình trong models dictionary, trong đó model_name là tên mô hình và model_func là hàm tương ứng để biến đổi dữ liệu theo các hàm model1(), model2(), model3(). Song song tạo mae_list cho mỗi lần lặp.
- Lặp qua từng bộ đề, test_X = X_splits[i].copy(): Lấy phần thứ i trong X_splits làm tập kiểm tra và tạo một bản sao (copy) để tránh ảnh hưởng đến dữ liệu gốc, tương tự với test_y và train_X = pd.concat([X_splits[j] for j in range(n_splits) if j != i]): Kết hợp các phần còn lại của X_splits, ngoại trừ phần thứ i, để làm tập huấn luyện, tương tự train_y. (kfold cross-validation) [5]
- Áp dụng hàm tương ứng để xây dựng mô hình và huấn luyện hồi quy tuyến tính. Tính ra MAE thấp nhất rồi lưu vào list_mae theo cặp tên model và MAE thấp nhất tương ứng.
- Tìm mô hình tốt nhất và in ra kết quả

STT	Mô hình	MAE
1	Sử dụng 4 đặc trưng trừ Extracurricular Activities	1.641
2	Sử dụng 2 đặc trưng tương quan cao nhất + biến tương tác giữa chúng	1.816
3	Sử dụng 2 đặc trưng tương quan cao nhất + biến tương tác giữa chúng + sqrt của 3 đặc trưng còn lại	1.629

⇨ Mô hình tốt nhất trong 3 mô hình là model 3

- Huấn luyện mô hình tốt nhất trên tập test.csv, lấy trọng số bằng phương thức get_params()

$$w = [-3.721 \;\; 1.019 \;\; 2.862 \;\; -1.547 \;\; 5.993 \;\; 2.367 \;\; 6.022]$$

- Tiến hành dự đoán giá trị y_pred trên tập kiểm tra X_test sau khi đã xử lý trước tương tự như tập huấn luyện. Quá trình dự đoán được thực hiện bằng phương thức predict() của mô hình.
- Để đánh giá mô hình, tính toán sai số trung bình tuyệt đối (MAE) giữa các giá trị dự đoán y_pred và giá trị thực tế y_test trên tập kiểm tra. MAE là chỉ số đánh giá độ chính xác của mô hình, phản ánh mức độ chênh lệch trung bình giữa giá trị thực tế và giá trị dự đoán.

$$MAE = 1.608$$

⇨ Công thức hồi quy tuyến tính:

$$Student\ Performance = -3.721 + 1.019 * Previous\ Scores + 2.862 * Hours\ Studied - 1.547 * (Previous\ Scores * Hours\ Studied) + 5.993 * \sqrt{Extracurricular\ Activities} + 2.367 * \sqrt{Sleep\ Hours} + 6.022 * \sqrt{Sample\ Question\ Papers\ Practiced}$$

Lý do chọn mô hình:

- Với model 1, Trong quá trình phân tích tương quan và biểu đồ hộp (box plot), chúng ta đã thấy rằng Extracurricular Activities có mối quan hệ rất yếu với Performance Index. Cụ thể, trung vị của Performance Index cho cả hai nhóm học sinh tham gia và không tham gia hoạt động ngoại khóa gần như tương đương nhau, và khoảng phân bố của dữ liệu cũng tương tự. Điều này chỉ ra rằng Extracurricular Activities không phải là yếu tố có tác động lớn đến hiệu suất học tập. Đồng thời, hệ số tương quan giữa Extracurricular Activities và Performance Index là rất thấp, gần như bằng 0, cho thấy rằng việc tham gia hay không tham gia hoạt động ngoại khóa không liên quan đáng kể đến chỉ số hiệu suất. Vì vậy, đặc trưng này có thể không đóng góp nhiều vào việc giải thích sự biến động trong dữ liệu. Khi thêm một đặc trưng không có ý nghĩa hoặc có mối quan hệ yếu vào mô hình, điều này có thể dẫn đến việc thêm "nhiễu" vào mô hình. Nhiễu này không chỉ làm tăng độ phức tạp mà còn có thể làm giảm khả năng tổng quát hóa của mô hình, đặc biệt khi áp dụng trên dữ liệu mới. Do đó, việc loại bỏ Extracurricular Activities giúp mô hình trở nên đơn giản hơn và tập trung vào những đặc trưng có ý nghĩa hơn.
- Với model 2, Previous Scores có hệ số tương quan rất cao với Performance Index, cụ thể là r = 0.91. Điều này có nghĩa là điểm số trước đây của học sinh là một trong những yếu tố dự đoán mạnh mẽ nhất về hiệu suất học tập tương lai. Một mối quan hệ tương quan mạnh mẽ như vậy cho thấy rằng học sinh có điểm số cao trong quá khứ có khả năng cao cũng sẽ có hiệu suất học tập tốt hơn trong tương lai. Và, Hours Studied có hệ số tương quan r = 0.37 với Performance Index, cho thấy rằng có một mối quan hệ dương giữa số giờ học và hiệu suất học tập. Mặc dù mối quan hệ này không mạnh như Previous Scores, nhưng nó vẫn đủ đáng kể để cho thấy rằng học sinh học nhiều giờ hơn thường có hiệu suất học tập tốt hơn. Nên biến tương tác được thêm vào để kiểm tra xem liệu 2 đặc trưng có tương

quan cao khi kết hợp lại có tạo ra tác động cộng gộp lên hiệu suất hay không? [1[2]][3]

- Với model 3, như ở model 2, Previous Scores và Hours Studied được giữ lại vì chúng là hai đặc trưng quan trọng nhất có mối quan hệ trực tiếp và mạnh mẽ với Performance Index và biến tương tác cũng tương tự. Việc sử dụng căn bậc hai của các đặc trưng này là để kiểm tra liệu có tồn tại mối quan hệ phi tuyến giữa các biến này và Performance Index hay không [1][4[3]]. Sleep Hours giả thuyết rằng giấc ngủ không chỉ đơn thuần có tác động tuyến tính đến hiệu suất học tập. Tương tự cho Sample Question Papers Practiced và Extracurricular Activities.

[2] Chương 3,9 bàn luận chi tiết về hồi quy tuyến tính và phi tuyến, bao gồm việc sử dụng biến tương tác

[3] Chương 5 đề cập đến các phương pháp để mô hình hóa các mối quan hệ phi tuyến

TÀI LIỆU THAM KHẢO

[1]	Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer
[2]	Burnham, K. P., & Anderson, D. R. (2002). *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer
[3]	Aiken, L. S., & West, S. G. (1991). *Multiple Regression: Testing and Interpreting Interactions*. SAGE Publications.
[4]	Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). *Introduction to Linear Regression Analysis*. Wiley.
[5]	Antoniadis, P. (2024, March 18). *Cross-Validation: K-Fold vs. Leave-One-Out*. Baeldung. Retrieved August 17, 2024, from https://www.baeldung.com/cs/cross-validation-k-fold-loo
[6]	Jain, S. (2024, August 7). *Linear Regression in Machine learning*. GeeksforGeeks. Retrieved August 17, 2024, from https://www.geeksforgeeks.org/ml-linear-regression/
[7]	Lab 04: OLS Linear Regression (Cô Uyên)
[8]	*API Reference — Matplotlib 3.9.2 documentation*. (n.d.). Matplotlib. Retrieved August 17, 2024, from https://matplotlib.org/stable/api/index.html
[9]	*NumPy reference — NumPy v2.0 Manual*. (2024, June 16). NumPy -. Retrieved August 17, 2024, from https://numpy.org/doc/stable/reference/index.html
[10]	*API reference — pandas 2.2.2 documentation*. (n.d.). Pandas. Retrieved August 17, 2024, from https://pandas.pydata.org/docs/reference/index.html
[11]	*API reference — seaborn 0.13.2 documentation*. (n.d.). Seaborn. Retrieved August 17, 2024, from https://seaborn.pydata.org/api.html