A black background with blue and black border

AI-generated content may be incorrect.**ĐẠI HỌC KINH TẾ THÀNH PHỐ HỒ CHÍ MINH**

**KHOA KINH TẾ**

A blue and orange text on a black background

AI-generated content may be incorrect.

**BÀI TẬP GIỮA KỲ**

|  |  |  |
| --- | --- | --- |
| **Giảng viên** | **:** | TS. Đỗ Như Tài |
| **Mã lớp học phần** | **:** | 25C1ECO50118802 |
| **Nhóm** | **:** | Bộ tứ báo thủ |
| **Sinh viên thực hiện** | **:** | Nguyễn Văn Huy 31231026138  Võ Duy Khánh 31231027342  Phạm Bảo Ngân 31231022815  Lê Thị Kim Cương 31231023267 |

**TP Hồ Chí Minh, 22 tháng 10 năm 2025**

**Bảng đánh giá đóng góp**

|  |  |  |
| --- | --- | --- |
| **Sinh viên** | **Công việc nhiệm vụ** | **Phần trăm đóng góp** |
| Nguyễn Văn Huy | Chạy dữ liệu các bài thực hành lab1, lab2 | 100% |
| Võ Duy Khánh | Thực hiện nội dung lí thuyết lab1, lab2, tổng hợp kết quả các bài thực hành | 100% |
| Phạm Bảo Ngân | Thực hiện nội dung lí thuyết lab3, chạy dữ liệu, tổng hợp kết quả các bài thực hành | 100% |
| Lê Thị Kim Cương | Thực hiện nội dung lí thuyết lab3, chạy dữ liệu, tổng hợp kết quả các bài thực hành | 100% |

**MỤC LỤC**

[**Lời xin lỗi** 8](#_Toc212068156)

[**Lab\_01** 9](#_Toc212068157)

[**1.1: THỐNG KÊ MÔ TẢ** 9](#_Toc212068158)

[1.1.1. Ôn tập lý thuyết 9](#_Toc212068159)

[**+ Thống kê mô tả là gì? Nó khác gì với thống kê suy luận (inferential statistics)?** 9](#_Toc212068160)

[**+ Các thước đo thống kê mô tả chính (ví dụ: trung bình, trung vị, phương sai, độ lệch chuẩn) được sử dụng để làm gì? Trong trường hợp nào thì nên dùng trung vị thay vì trung bình?** 9](#_Toc212068161)

[**+ Làm thế nào để xác định phân bố của một tập dữ liệu? Các loại phân bố phổ biến là gì (ví dụ: phân bố chuẩn, lệch trái, lệch phải)?** 10](#_Toc212068162)

[**+ Độ lệch chuẩn và phạm vi (range) có ý nghĩa gì trong việc đánh giá sự phân tán của dữ liệu?** 11](#_Toc212068163)

[**+ Sự khác biệt giữa các thước đo như Q1, Q2, Q3 trong biểu đồ hộp (boxplot) là gì?** 12](#_Toc212068164)

[**+ Làm thế nào để xử lý giá trị thiếu (missing values) trước khi tính toán các chỉ số thống kê mô tả?** 13](#_Toc212068165)

[**+ Bạn có thể giải thích cách đọc và diễn giải một biểu đồ histogram hoặc boxplot từ dữ liệu thực tế không?** 14](#_Toc212068166)

[**+ Khi gặp một tập dữ liệu có giá trị ngoại lai (outliers), bạn sẽ xử lý chúng như thế nào trước khi thực hiện thống kê mô tả?** 15](#_Toc212068167)

[1.1.3. Bài tập thực hành 1: 16](#_Toc212068168)

[1.1.4. Bài tập thực hành 2: 18](#_Toc212068169)

[**1.2: XỬ LÝ VÀ TRỰC QUAN HÓA DỮ LIỆU** 21](#_Toc212068170)

[1.2.1. Ôn tập lý thuyết 21](#_Toc212068171)

[**+ Trực quan hóa dữ liệu có vai trò gì trong phân tích dữ liệu? Tại sao nó quan trọng trong khám phá dữ liệu (EDA)?** 21](#_Toc212068172)

[**+ Các loại biểu đồ phổ biến (như histogram, scatter plot, boxplot, bar chart) được sử dụng trong các trường hợp nào?** 22](#_Toc212068173)

[**+ Làm thế nào để chọn loại biểu đồ phù hợp với đặc điểm của dữ liệu (ví dụ: dữ liệu phân loại, dữ liệu số, dữ liệu thời gian)?** 23](#_Toc212068174)

[**+ Sự khác biệt giữa các thư viện trực quan hóa trong Python như Matplotlib, Seaborn và Plotly là gì?** 24](#_Toc212068175)

[**+ Những nguyên tắc thiết kế nào cần tuân thủ để tạo ra một biểu đồ trực quan hóa dễ hiểu và hiệu quả?** 26](#_Toc212068176)

[**+ Làm thế nào để tạo một biểu đồ đơn giản như histogram hoặc bar chart bằng Matplotlib? Bạn có thể chia sẻ đoạn code mẫu không?** 27](#_Toc212068177)

[**+ Làm thế nào để xuất biểu đồ từ Python ra các định dạng như PNG, PDF hoặc HTML để sử dụng trong báo cáo?** 28](#_Toc212068178)

[1.2.2. Bài tập thực hành 1 30](#_Toc212068179)

[1.2.3. Bài tập thực hành 2 33](#_Toc212068180)

[**1.3: PHÂN TÍCH ĐƠN BIẾN VÀ HAI BIẾN** 38](#_Toc212068181)

[1.3.1. Ôn tập lý thuyết 38](#_Toc212068182)

[**+ Phân tích đơn biến (univariate analysis) là gì? Nó khác gì với phân tích hai biến (bivariate analysis) trong khám phá dữ liệu?** 38](#_Toc212068183)

[**+ Các thước đo thống kê nào thường được sử dụng trong phân tích đơn biến (ví dụ: trung bình, trung vị, mode, độ lệch chuẩn)?** 38](#_Toc212068184)

[**+ Trong phân tích hai biến, làm thế nào để xác định mối quan hệ giữa hai biến (ví dụ: tương quan, nhân quả)?** 39](#_Toc212068185)

[**+ Sự khác biệt giữa tương quan (correlation) và hiệp biến (covariance) trong phân tích hai biến là gì?** 40](#_Toc212068186)

[**+ Khi nào nên sử dụng biểu đồ trực quan hóa trong phân tích đơn biến so với phân tích hai biến?** 41](#_Toc212068187)

[**+ Đoạn code mẫu để tạo biểu đồ scatter plot hoặc heatmap để phân tích mối quan hệ giữa hai biến** 42](#_Toc212068188)

[**+ Làm thế nào để trực quan hóa mối quan hệ giữa một biến số và một biến phân loại bằng biểu đồ boxplot hoặc violin plot trong Python?** 44](#_Toc212068189)

[1.3.3. Bài tập thực hành 1 46](#_Toc212068190)

[1.3.4. Bài tập thực hành 2 47](#_Toc212068191)

[**LAB\_02** 48](#_Toc212068192)

[**2.1: Giải thuật 1: CÂY QUYẾT ĐỊNH VÀ RỪNG CÂY** 48](#_Toc212068193)

[2.1.1. Ôn tập lý thuyết 48](#_Toc212068194)

[**+ Quy trình khai phá dữ liệu CRISP – DM (Cross Industry Standard Process for Data Mining) là gì ? Quy trình khai phá dữ liệu SEMMA (Sample, Explore, Modify, Model, Access) là gì?** 48](#_Toc212068195)

[**+ Cây quyết định hoạt động như thế nào? Hãy giải thích các thành phần chính (nút gốc, nút lá, nhánh) và cách cây đưa ra dự đoán.** 50](#_Toc212068196)

[**+ Các tiêu chí phân tách (splitting criteria) như Gini Index, Entropy, hay Information Gain được sử dụng trong cây quyết định là gì? Chúng khác nhau ra sao?** 51](#_Toc212068197)

[**+ Rừng cây (Random Forest) là gì? Nó khác gì so với một cây quyết định đơn lẻ? Tại sao Random Forest TRƯỜNG ĐẠI HỌC SÀI GÒN 2 thường có hiệu suất tốt hơn cây quyết định trong các bài toán phân loại?** 52](#_Toc212068198)

[**+ Những ưu điểm và hạn chế của cây quyết định và Random Forest là gì? Trong trường hợp nào thì cây quyết định có thể hoạt động kém hiệu quả?** 53](#_Toc212068199)

[**+ Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để xây dựng một mô hình cây quyết định không? Hãy mô tả các bước thực hiện.** 55](#_Toc212068200)

[**+ Làm thế nào để triển khai một mô hình Random Forest trong Python? Bạn thường thiết lập các tham số nào (ví dụ: n\_estimators, max\_depth)?** 56](#_Toc212068201)

[**+ Làm thế nào để đánh giá tầm quan trọng của các đặc trưng (feature importance) trong Random Forest bằng Python?** 57](#_Toc212068202)

[**+ Điều chỉnh siêu tham số (hyperparameter tuning) cho cây quyết định hoặc Random Forest chưa? Hãy mô tả cách bạn sử dụng GridSearchCV hoặc RandomizedSearchCV** 57](#_Toc212068203)

[2.1.3. Bài tập thực hành 1 58](#_Toc212068204)

[2.1.4. Bài tập thực hành 2: 63](#_Toc212068205)

[**2.2: Giải thuật 2: SUPPORT VECTOR MACHINE (SVM)** 68](#_Toc212068206)

[2.2.1: Lý thuyết 68](#_Toc212068207)

[**+ Giải thuật Support Vector Machine hoạt động như thế nào? Hãy giải thích khái niệm về ranh giới phân tách (hyperplane) và lề (margin)** 68](#_Toc212068208)

[**+ Các vector hỗ trợ (support vectors) có vai trò gì trong SVM? Tại sao chúng quan trọng?** 70](#_Toc212068209)

[**+ Sự khác biệt giữa SVM với lề cứng (hard margin) và lề mềm (soft margin) là gì? Khi nào nên sử dụng lề mềm?** 70](#_Toc212068210)

[**+ Hàm nhân (kernel) trong SVM là gì? Hãy giải thích các loại kernel phổ biến (linear, polynomial, RBF) và khi nào nên sử dụng chúng** 71](#_Toc212068211)

[**+ Tham số C trong SVM có ý nghĩa gì? Nó ảnh hưởng như thế nào đến hiệu suất của mô hình?** 73](#_Toc212068212)

[**+ Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để xây dựng một mô hình SVM cho bài toán phân loại không? Hãy mô tả các bước thực hiện** 73](#_Toc212068213)

[**+ Hàm nào trong Scikit-learn để chuẩn hóa dữ liệu (scaling) trước khi áp dụng SVM? Tại sao bước này quan trọng?** 75](#_Toc212068214)

[2.2.3. Bài tập thực hành 1 75](#_Toc212068215)

[2.2.4. Bài tập thực hành 2 77](#_Toc212068216)

[**2.3: Giải thuật 3: BAYES NGÂY THƠ (NAÏVE BAYES)** 81](#_Toc212068217)

[2.3.1. Ôn tập lý thuyết 81](#_Toc212068218)

[**+ Giải thuật Naive Bayes hoạt động như thế nào? Hãy giải thích định lý Bayes và giả định "ngây thơ" trong thuật toán này?** 81](#_Toc212068219)

[**+ Các loại mô hình Naive Bayes (Gaussian, Multinomial, Bernoulli) khác nhau ra sao? Khi nào nên sử dụng từng loại?** 82](#_Toc212068220)

[**+ Tại sao Naive Bayes được gọi là "ngây thơ"? Giả định về tính độc lập của các đặc trưng ảnh hưởng như thế nào đến hiệu suất của mô hình?** 83](#_Toc212068221)

[**+ Ưu điểm và hạn chế của Naive Bayes so với các thuật toán phân loại khác như SVM hoặc Random Forest là gì?** 83](#_Toc212068222)

[**+ Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để xây dựng một mô hình Naive Bayes (ví dụ: Gaussian Naive Bayes) không? Hãy mô tả các bước thực hiện** 84](#_Toc212068223)

[**+ Làm thế nào để xử lý dữ liệu phân loại (categorical data) trước khi áp dụng Multinomial Naive Bayes trong Python?** 86](#_Toc212068224)

[**+ Naive Bayes thường được sử dụng trong phân loại văn bản (text classification). Bạn có thể giải thích cách triển khai Naive Bayes cho bài toán này không?** 87](#_Toc212068225)

[2.3.1. Bài tập thực hành 1 89](#_Toc212068226)

[**LAB03** 92](#_Toc212068227)

[**3.1. Giải thuật K-Means** 92](#_Toc212068228)

[**3.1.1. Ôn tập lý thuyết** 92](#_Toc212068229)

[**Giải thuật K-Means hoạt động như thế nào?** 92](#_Toc212068230)

[**Tại sao cần chọn số lượng cụm K trước khi chạy? Làm sao xác định K tối ưu?** 93](#_Toc212068231)

[**Hàm mục tiêu (Objective Function) của K-Means là gì? Nó đo lường cái gì?** 93](#_Toc212068232)

[**Những hạn chế của K-Means là gì?** 93](#_Toc212068233)

[**Trong trường hợp nào K-Means có thể cho kết quả không tốt?** 94](#_Toc212068234)

[**Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để triển khai K-Means Clustering không? Hãy mô tả các bước thực hiện.** 94](#_Toc212068235)

[**Sử dụng phương pháp nào trong Python để chọn số cụm K tối ưu (ví dụ: Elbow Method, Silhouette Score)? Hãy chia sẻ một đoạn code mẫu** 95](#_Toc212068236)

[**K-Means nhạy cảm với giá trị khởi tạo (initial centroids), bạn sẽ làm gì để đảm bảo kết quả ổn định (ví dụ: K-Means++)?** 95](#_Toc212068237)

[**Làm thế nào để đánh giá chất lượng của các cụm được tạo bởi K-Means? Bạn sử dụng chỉ số nào (ví dụ: Silhouette Score, Within-Cluster Sum of Squares)?** 96](#_Toc212068238)

[**3.1.2. Bài tập thực hành** 96](#_Toc212068239)

[**3.2. Giải thuật phân cụm đa cấp** 104](#_Toc212068240)

[**3.2.1. Ôn tập lý thuyết** 104](#_Toc212068241)

[**Giải thuật phân cụm đa cấp hoạt động như thế nào?** 104](#_Toc212068242)

[**Các phương pháp liên kết (Linkage)** 104](#_Toc212068243)

[**Dendrogram trong phân cụm đa cấp:** 105](#_Toc212068244)

[**Ứng dụng với dữ liệu phi số (Non-numeric data):** 105](#_Toc212068245)

[**Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để triển khai phân cụm đa cấp hợp nhất (agglomerative clustering) không? Hãy mô tả các bước thực hiện.** 105](#_Toc212068246)

[**Làm thế nào để vẽ dendrogram trong Python sử dụng thư viện như scipy hoặc matplotlib? Hãy chia sẻ một đoạn code mẫu** 106](#_Toc212068247)

[**Các lớp trong gói Scipy hỗ trợ phân cụm đa cấp? và so sánh giữa cách tiếp cận scikit-learn và cách tiếp cận sử dụng Scipy** 106](#_Toc212068248)

[**3.2.2. Bài tập thực hành** 106](#_Toc212068249)

# **Lời xin lỗi**

Trước hết, nhóm chúng em xin gửi lời xin lỗi chân thành đến thầy vì đã nộp bài trễ hơn so với thời hạn mà thầy đã quy định, dù thầy đã nhiều lần gia hạn để tạo điều kiện cho chúng em. Nhóm đã cố gắng hoàn thành bài đúng hạn, tuy nhiên bài nộp lần đầu vẫn trễ 6 phút so với thời gian quy định.

Do quá vội vàng khi nộp bài sát giờ, chúng em đã không kiểm tra kỹ và vô tình bỏ sót một số yêu cầu mà thầy đã đề ra. Đến hôm nay, sau khi rà soát lại hướng dẫn, chúng em mới nhận ra rằng bài nộp chưa đúng theo định dạng quy định.

Vì vậy, nhóm chúng em xin phép được gửi lại bài giữa kỳ theo đúng format mà thầy yêu cầu. Chúng em thành thật xin lỗi về sự thiếu sót này và xin hoàn toàn chịu trách nhiệm cho sai lầm của mình. Rất mong thầy thông cảm và ghi nhận thiện chí khắc phục của nhóm.

Trân trọng,

Nhóm sinh viên

# **Lab\_01**

## **1.1: THỐNG KÊ MÔ TẢ**

### 1.1.1. Ôn tập lý thuyết

#### **+ Thống kê mô tả là gì? Nó khác gì với thống kê suy luận (inferential statistics)?**

* Thống kê mô tả (Descriptive Statistics) là một nhánh của thống kê dùng để tóm tắt, mô tả và trình bày dữ liệu sao cho dễ hiểu hơn. Nó không đưa ra kết luận vượt ra ngoài dữ liệu hiện có, mà chỉ phản ánh những gì đang có trong mẫu hoặc tập dữ liệu.
* Mục tiêu: mô tả và hiểu dữ liệu đã thu thập.
* Thống kê suy luận (Inferential Statistics) là bước tiếp theo, dùng mẫu dữ liệu để suy ra đặc điểm của tổng thể (population) mà mẫu đó đại diện. Nó dựa trên xác suất để đưa ra ước lượng hoặc kiểm định giả thuyết.
* Mục tiêu: rút ra kết luận hoặc dự đoán cho tổng thể dựa trên mẫu.

Phân biệt:

| Tiêu chí | Thống kê mô tả | Thống kê suy luận |
| --- | --- | --- |
| Mục tiêu | Mô tả dữ liệu có sẵn | Suy luận về tổng thể từ mẫu |
| Dữ liệu sử dụng | Toàn bộ tập dữ liệu hoặc mẫu | Mẫu đại diện cho tổng thể |
| Phương pháp | Trung bình, độ lệch chuẩn, biểu đồ... | Kiểm định giả thuyết, hồi quy, ước lượng... |
| Kết quả | Thông tin thực tế, không khái quát | Kết luận hoặc dự đoán cho tổng thể |

#### **+ Các thước đo thống kê mô tả chính (ví dụ: trung bình, trung vị, phương sai, độ lệch chuẩn) được sử dụng để làm gì? Trong trường hợp nào thì nên dùng trung vị thay vì trung bình?**

1. Các thước đo thống kê mô tả chính và ý nghĩa của chúng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Thước đo | Ý nghĩa | Dùng để làm gì |  |  |  |
| Trung bình (Mean) | Là giá trị bình quân của toàn bộ dữ liệu. | Dùng để biểu diễn mức độ điển hình của dữ liệu, thể hiện xu hướng trung tâm. |  |  |  |
| Trung vị (Median) | Là giá trị nằm ở giữa khi sắp xếp dữ liệu theo thứ tự tăng hoặc giảm. | Giúp xác định giá trị trung tâm khi dữ liệu có thể bị lệch (có ngoại lệ). |  |  |  |
| Phương sai (Variance) | Là bình phương độ lệch giữa các giá trị và trung bình. | Đo mức độ phân tán của dữ liệu quanh giá trị trung bình. |  |  |  |
| Độ lệch chuẩn (Standard Deviation) | Là căn bậc hai của phương sai. | Cho biết dữ liệu dao động nhiều hay ít quanh trung bình (càng lớn → phân tán càng mạnh). |  |  |  |
| Giá trị nhỏ nhất – lớn nhất (Min – Max) | Hai giá trị cực biên trong dữ liệu. | Giúp xác định khoảng biến thiên của dữ liệu. |  |  |  |
| Tứ phân vị (Quartiles) & IQR (Interquartile Range) | Chia dữ liệu thành 4 phần bằng nhau. IQR = Q3 – Q1. | Dùng để xác định mức độ phân tán và phát hiện giá trị ngoại lai. |  |  |  |

2. Dùng trung vị thay trung bình:

Dùng trung vị khi dữ liệu bị lệch hoặc có ngoại lệ (outliers).

Do đó:

Trung bình phù hợp khi dữ liệu phân bố đối xứng, không có ngoại lệ.

Trung vị phù hợp khi dữ liệu lệch (skewed) hoặc có giá trị cực đoan.

#### **+ Làm thế nào để xác định phân bố của một tập dữ liệu? Các loại phân bố phổ biến là gì (ví dụ: phân bố chuẩn, lệch trái, lệch phải)?**

1. Cách xác định phân bố của một tập dữ liệu

Để xác định dạng phân bố của dữ liệu, ta có thể sử dụng các phương pháp sau:

Biểu đồ tần suất (Histogram):  
Dùng để quan sát trực quan tần suất xuất hiện của các giá trị. Hình dạng biểu đồ cho biết dữ liệu có đối xứng, lệch trái hay lệch phải.

Biểu đồ hộp (Boxplot):  
Cho thấy vị trí trung vị, các tứ phân vị và giá trị ngoại lai. Hữu ích để phát hiện sự lệch trong dữ liệu.

Đường mật độ (Density Plot):  
Minh họa dạng liên tục của phân bố, giúp xác định xem dữ liệu có dạng chuông hay lệch về một phía.

Các thước đo mô tả:

Độ lệch (Skewness): đo mức độ lệch của phân bố.

Skewness ≈ 0: phân bố gần chuẩn.

Skewness > 0: phân bố lệch phải (đuôi dài bên phải).

Skewness < 0: phân bố lệch trái (đuôi dài bên trái).

Độ nhọn (Kurtosis): đo mức độ tập trung của dữ liệu quanh trung bình (đỉnh nhọn hay bẹt).

2. Các loại phân bố phổ biến

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Loại phân bố | Đặc điểm chính | Ví dụ điển hình |  |  |  |
| Phân bố chuẩn (Normal distribution) | Có dạng chuông đối xứng, trung bình = trung vị = mode. Phần lớn giá trị tập trung quanh trung tâm. | Chiều cao, cân nặng của con người. |  |  |  |
| Phân bố lệch phải (Right-skewed / Positively skewed) | Đuôi phân bố kéo dài về phía bên phải, trung bình lớn hơn trung vị. | Thu nhập, giá bất động sản. |  |  |  |
| Phân bố lệch trái (Left-skewed / Negatively skewed) | Đuôi phân bố kéo dài về phía bên trái, trung bình nhỏ hơn trung vị. | Điểm thi (đa số cao, ít người điểm thấp). |  |  |  |
| Phân bố đồng đều (Uniform distribution) | Các giá trị có xác suất xuất hiện gần như bằng nhau. | Kết quả khi tung xúc xắc công bằng. |  |  |  |
| Phân bố nhọn (Leptokurtic) | Phần lớn giá trị tập trung gần trung bình, đỉnh nhọn và đuôi dài. | Dữ liệu ít biến động nhưng có vài giá trị cực đoan. |  |  |  |
| Phân bố bẹt (Platykurtic) | Phân bố dàn trải, đỉnh thấp, đuôi ngắn. | Dữ liệu phân tán đồng đều quanh trung bình. |  |  |  |

#### **+ Độ lệch chuẩn và phạm vi (range) có ý nghĩa gì trong việc đánh giá sự phân tán của dữ liệu?**

Độ lệch chuẩn (Standard Deviation) và phạm vi (Range) đều là những thước đo cho biết mức độ phân tán hay biến động của dữ liệu quanh giá trị trung tâm, nhưng chúng phản ánh điều này ở mức độ khác nhau.

1. Phạm vi (Range)

Cách tính:

Range = Giá trị lớn nhất – Giá trị nhỏ nhất

Ý nghĩa:

Cho biết khoảng dao động tổng thể của dữ liệu, tức là dữ liệu trải dài từ đâu đến đâu.

Ưu điểm:

Dễ tính, dễ hiểu.

Phù hợp để có cái nhìn nhanh về độ rộng của dữ liệu.

Nhược điểm:

Chỉ dựa trên hai giá trị cực biên, nên rất nhạy cảm với giá trị ngoại lai (outliers).

Không phản ánh chính xác mức độ phân tán của phần lớn dữ liệu.

2. Độ lệch chuẩn (Standard Deviation)

Cách tính:  
Là căn bậc hai của phương sai, cho biết trung bình khoảng cách của các giá trị so với giá trị trung bình.

Ý nghĩa:  
Độ lệch chuẩn càng lớn, dữ liệu càng phân tán rộng quanh trung bình.  
Độ lệch chuẩn càng nhỏ, dữ liệu càng tập trung gần trung bình.

Ưu điểm:

Phản ánh tổng thể mức độ biến động của tất cả các giá trị.

Ít bị ảnh hưởng hơn bởi giá trị cực đoan so với phạm vi (dù vẫn có thể bị tác động).

Là thước đo chuẩn được sử dụng trong hầu hết các phân tích thống kê.

#### **+ Sự khác biệt giữa các thước đo như Q1, Q2, Q3 trong biểu đồ hộp (boxplot) là gì?**

1. Khái niệm chung về các thước đo Q1, Q2, Q3 trong biểu đồ hộp (Boxplot)

Trong thống kê mô tả, Q1, Q2, Q3 là các tứ phân vị (quartiles) - những giá trị chia tập dữ liệu đã được sắp xếp theo thứ tự tăng dần thành 4 phần bằng nhau.  
Mỗi phần chiếm 25% tổng số quan sát.

Cụ thể:

Q1 (Quartile 1 - Tứ phân vị thứ nhất):  
Là giá trị tại vị trí 25% của dữ liệu.  
🡪 25% dữ liệu nhỏ hơn Q1, 75% dữ liệu lớn hơn Q1.  
🡪 Còn gọi là phân vị thứ 25 (P25).

Q2 (Quartile 2 - Tứ phân vị thứ hai):  
Là giá trị tại vị trí 50% của dữ liệu.  
🡪 Cũng chính là trung vị (Median) của tập dữ liệu.  
🡪 50% dữ liệu nhỏ hơn Q2 và 50% dữ liệu lớn hơn Q2.

Q3 (Quartile 3 - Tứ phân vị thứ ba):  
Là giá trị tại vị trí 75% của dữ liệu.  
🡪 75% dữ liệu nhỏ hơn Q3, 25% dữ liệu lớn hơn Q3.  
🡪 Còn gọi là phân vị thứ 75 (P75).

2. Vai trò của Q1, Q2, Q3 trong biểu đồ hộp (Boxplot)

Biểu đồ hộp giúp minh họa sự phân bố và phát hiện giá trị ngoại lai thông qua các tứ phân vị này.

Cấu trúc của một boxplot cơ bản gồm:

Đường giữa hộp: là Q2 (trung vị).

Cạnh dưới của hộp: là Q1.

Cạnh trên của hộp: là Q3.

Độ dài của hộp (Q3 - Q1): gọi là khoảng tứ phân vị (IQR - Interquartile Range), thể hiện mức độ phân tán của 50% dữ liệu trung tâm.

Hai “râu” (whiskers): thường kéo dài đến giá trị nhỏ nhất và lớn nhất nằm trong phạm vi [Q1-1.5×IQR, Q3 + 1.5×IQR].

Các điểm nằm ngoài phạm vi đó: được xem là giá trị ngoại lai (outliers).

#### **+ Làm thế nào để xử lý giá trị thiếu (missing values) trước khi tính toán các chỉ số thống kê mô tả?**

1. Khái niệm giá trị thiếu (Missing Values)

Giá trị thiếu là những ô dữ liệu trống hoặc không có thông tin hợp lệ trong tập dữ liệu.  
Ví dụ: cột “tuổi” có vài dòng bị bỏ trống hoặc nhập sai định dạng (“-”, “N/A”, “null”).  
Nếu không xử lý, các phép tính thống kê như trung bình, phương sai, độ lệch chuẩn... sẽ bị sai lệch hoặc không thể thực hiện được.

2. Các bước xử lý giá trị thiếu trước khi tính toán thống kê mô tả

Bước 1. Xác định giá trị thiếu

Dùng công cụ hoặc ngôn ngữ lập trình (như Excel, SPSS, Python, R) để đếm số lượng giá trị bị thiếu ở từng biến.

Xem xét tỷ lệ thiếu (%) để quyết định cách xử lý phù hợp.

Bước 2. Phân tích nguyên nhân

Do lỗi nhập liệu (quên nhập, gõ sai)?

Do giá trị thực sự không tồn tại (ví dụ: người không có thu nhập)?

Do điều kiện không áp dụng được (ví dụ: câu hỏi chỉ dành cho một nhóm người)?

Cách xử lý sẽ khác nhau tùy nguyên nhân.

Bước 3. Chọn phương pháp xử lý phù hợp

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cách xử lý | Mô tả | Khi nên dùng |  |  |  |
| Loại bỏ (Deletion) | - Xóa các dòng (hoặc cột) có giá trị thiếu. - Có thể là xóa hoàn toàn (listwise) hoặc xóa từng phần (pairwise). | Khi tỷ lệ thiếu nhỏ (<5%) và không ảnh hưởng nhiều đến kết quả. |  |  |  |
| Thay thế bằng giá trị thống kê | - Điền giá trị thiếu bằng trung bình, trung vị hoặc mode của biến đó. | Khi dữ liệu thiếu ngẫu nhiên (random) và cần giữ nguyên số lượng quan sát. |  |  |  |
| Thay thế theo nhóm | - Điền giá trị thiếu bằng trung bình/trung vị trong từng nhóm con (ví dụ: theo giới tính, khu vực). | Khi có sự khác biệt rõ rệt giữa các nhóm và muốn duy trì tính đại diện. |  |  |  |
| Nội suy (Interpolation) | - Ước lượng giá trị dựa trên xu hướng của các điểm lân cận (thường dùng cho chuỗi thời gian). | Khi dữ liệu có tính tuần tự hoặc liên tục theo thời gian. |  |  |  |
| Dự đoán (Prediction / Imputation) | - Dùng mô hình hồi quy, cây quyết định hoặc kỹ thuật học máy để ước tính giá trị thiếu. | Khi dữ liệu phức tạp, thiếu nhiều nhưng vẫn cần phân tích sâu. |  |  |  |
| Giữ nguyên (Leave as Missing) | - Giữ nguyên các giá trị thiếu và loại biến đó khỏi phân tích. | Khi tỷ lệ thiếu quá cao hoặc giá trị đó không quan trọng cho mục tiêu nghiên cứu. |  |  |  |

#### **+ Bạn có thể giải thích cách đọc và diễn giải một biểu đồ histogram hoặc boxplot từ dữ liệu thực tế không?**

1. Cách đọc và diễn giải biểu đồ Histogram

a. Khái niệm:  
Biểu đồ Histogram (biểu đồ tần suất) thể hiện phân bố của dữ liệu liên tục thông qua các cột liền nhau, trong đó:

Trục hoành (X) biểu diễn các khoảng giá trị của dữ liệu.

Trục tung (Y) biểu diễn tần suất hoặc mật độ xuất hiện trong mỗi khoảng.

b. Cách đọc:

1. Quan sát hình dạng tổng thể:

Dạng chuông đối xứng 🡪 dữ liệu có phân bố chuẩn.

Dạng lệch phải (đuôi dài bên phải) 🡪 có nhiều giá trị nhỏ, ít giá trị lớn.

Dạng lệch trái (đuôi dài bên trái) 🡪 có nhiều giá trị lớn, ít giá trị nhỏ.

1. Độ cao của các cột:

Cột càng cao 🡪 khoảng giá trị đó xuất hiện thường xuyên hơn.

1. Độ rộng và vị trí của các cột:

Giúp xác định phạm vi biến thiên và độ tập trung của dữ liệu.

c. Ví dụ:  
Giả sử bạn có dữ liệu về điểm thi của 100 sinh viên.

Nếu histogram có dạng chuông đối xứng quanh điểm 7 🡪 điểm thi phân bố chuẩn, đa số sinh viên đạt điểm trung bình khá.

Nếu histogram lệch phải 🡪 nhiều sinh viên điểm thấp, ít sinh viên điểm cao.

Nếu histogram lệch trái 🡪 đa số sinh viên đạt điểm cao, ít người bị điểm thấp.

2. Cách đọc và diễn giải biểu đồ Boxplot (biểu đồ hộp)

a. Khái niệm:  
Boxplot dùng để mô tả phân bố dữ liệu, mức độ phân tán và giá trị ngoại lai thông qua 5 điểm chính:

Min: giá trị nhỏ nhất (không tính ngoại lai)

Q1: tứ phân vị thứ nhất (25%)

Q2 (Median): trung vị (50%)

Q3: tứ phân vị thứ ba (75%)

Max: giá trị lớn nhất (không tính ngoại lai)

b. Cách đọc:

1. Đường giữa hộp: trung vị - cho thấy vị trí trung tâm của dữ liệu.
2. Chiều dài hộp (Q3 – Q1): thể hiện độ phân tán của 50% dữ liệu trung tâm (IQR).

Hộp càng dài 🡪 dữ liệu càng phân tán.

Hộp càng ngắn 🡪 dữ liệu tập trung.

1. Vị trí của trung vị trong hộp:

Trung vị nằm giữa hộp 🡪 phân bố đối xứng.

Trung vị lệch về một bên 🡪 dữ liệu bị lệch trái hoặc lệch phải.

1. “Râu” (Whiskers): cho biết phạm vi giá trị bình thường.
2. Các điểm riêng lẻ bên ngoài râu: là giá trị ngoại lai (outliers).

#### **+ Khi gặp một tập dữ liệu có giá trị ngoại lai (outliers), bạn sẽ xử lý chúng như thế nào trước khi thực hiện thống kê mô tả?**

Khi gặp một tập dữ liệu có giá trị ngoại lai (outliers), việc xử lý chúng là bước quan trọng để đảm bảo kết quả thống kê mô tả không bị sai lệch. Dưới đây là các bước và cách tiếp cận phổ biến:

1. Xác định giá trị ngoại lai:  
Trước tiên, cần phát hiện các điểm dữ liệu bất thường. Một số cách phổ biến:

Dùng biểu đồ boxplot: Các điểm nằm ngoài khoảng [Q1 - 1.5\*IQR, Q3 + 1.5\*IQR] được xem là ngoại lai, trong đó IQR (Interquartile Range) = Q3 – Q1.

Dùng z-score: Một giá trị có z-score > 3 hoặc < -3 thường được coi là ngoại lai (nghĩa là nó cách trung bình hơn 3 độ lệch chuẩn).

Dùng biểu đồ phân bố (histogram): Quan sát trực quan các giá trị cực đoan ở hai đầu phân bố.

2. Phân tích nguyên nhân:  
Không nên loại bỏ outlier ngay lập tức. Cần kiểm tra:

Đó có phải là lỗi nhập liệu hay không.

Giá trị đó có thực sự hợp lý nhưng hiếm gặp.

3. Cách xử lý giá trị ngoại lai:

Loại bỏ (remove): Nếu đó là lỗi rõ ràng hoặc giá trị quá cực đoan, có thể loại bỏ khỏi tập dữ liệu.

Thay thế (impute): Thay bằng giá trị trung vị (median) hoặc giá trị gần nhất trong phạm vi hợp lý, đặc biệt khi dữ liệu không quá lớn.

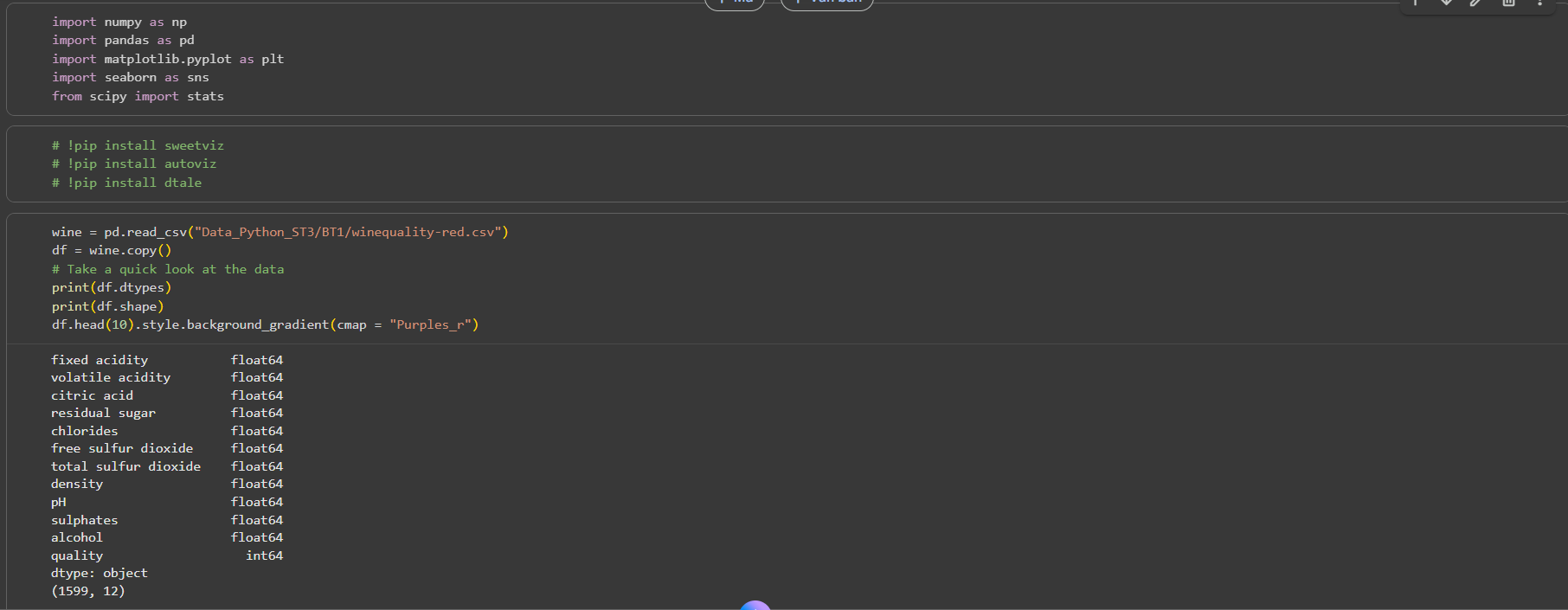
Biến đổi dữ liệu (transform): Áp dụng log, sqrt hoặc winsorization (giới hạn giá trị ở ngưỡng 5% và 95%) để giảm tác động của outlier.

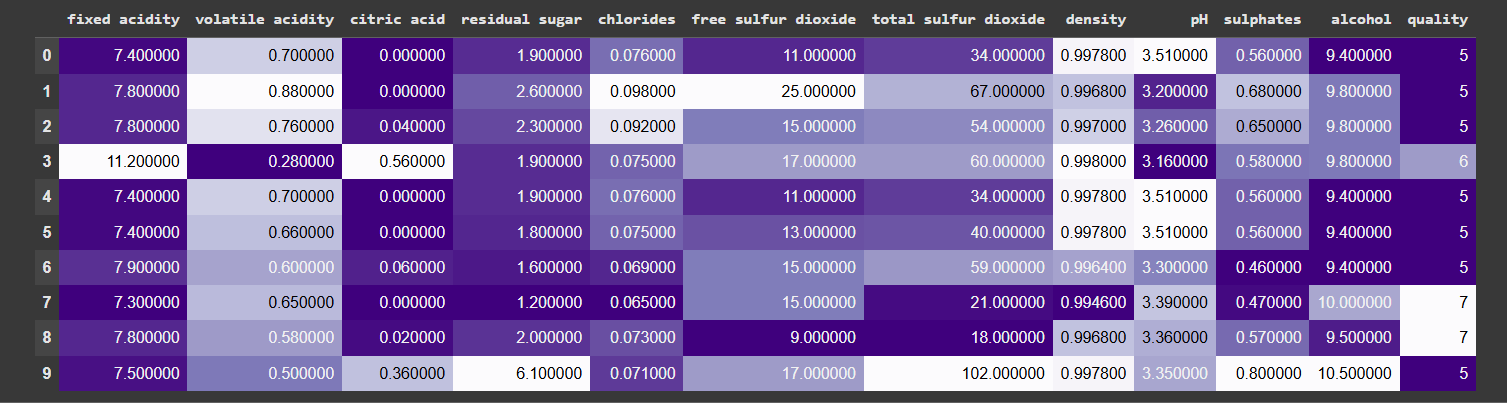
Giữ nguyên: Nếu giá trị ngoại lai phản ánh thực tế quan trọng (ví dụ trong nghiên cứu tài chính hay y học), có thể giữ lại nhưng cần phân tích riêng biệt.

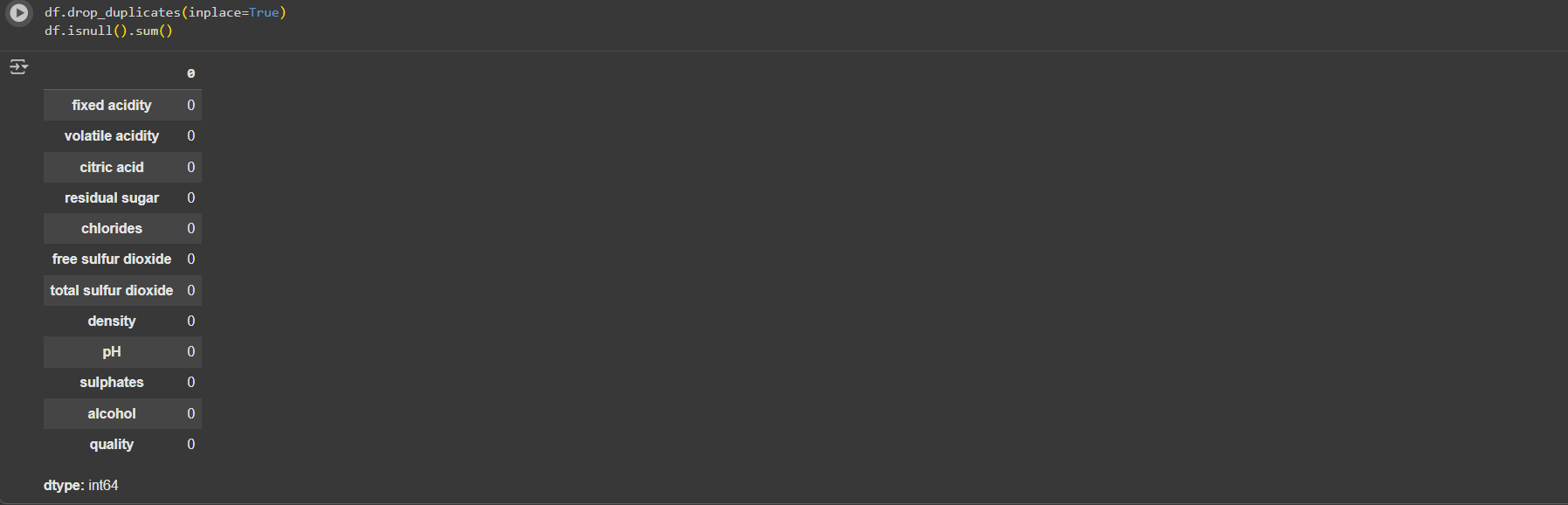
1. Ghi chú trong báo cáo:  
   Khi trình bày kết quả, nên nêu rõ phương pháp xử lý outlier để đảm bảo tính minh bạch và giúp người đọc hiểu cách bạn làm sạch dữ liệu.

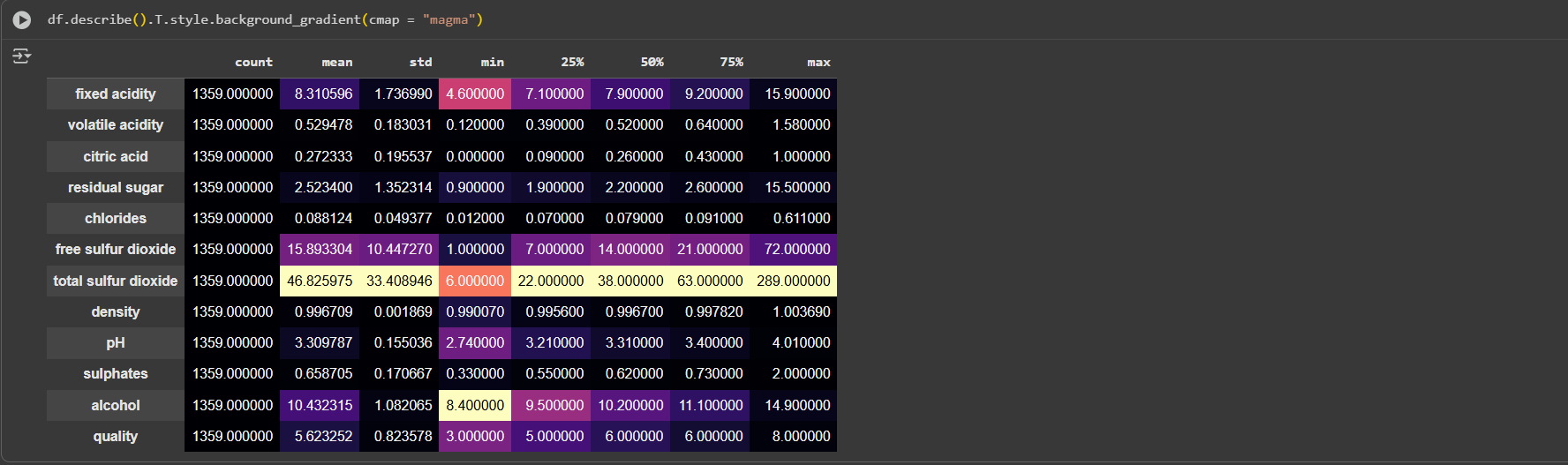
### 1.1.3. Bài tập thực hành 1:

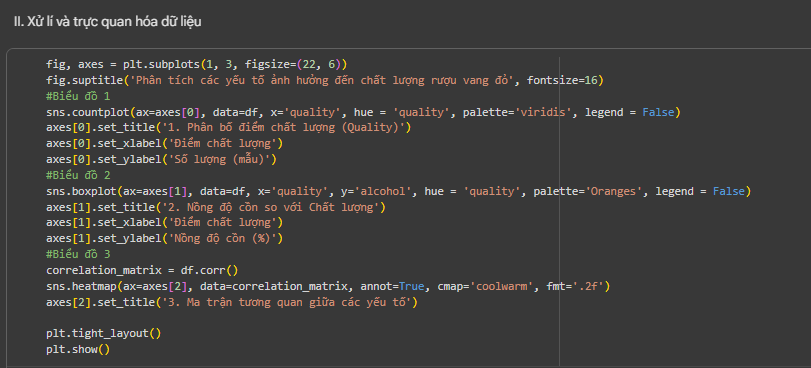
Thực hiện thống kê mô tả trên tập dữ liệu về phân loại chất lượng rượu đỏ. Dữ liệu lấy tại <https://www.kaggle.com/code/eisgandar/red-wine-quality-eda-classification>

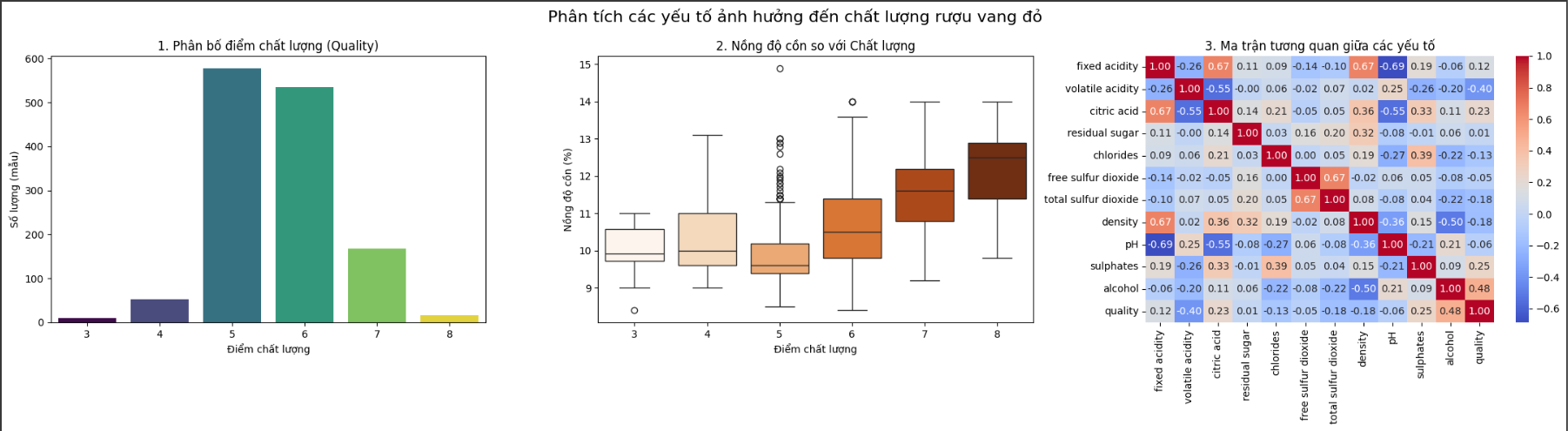






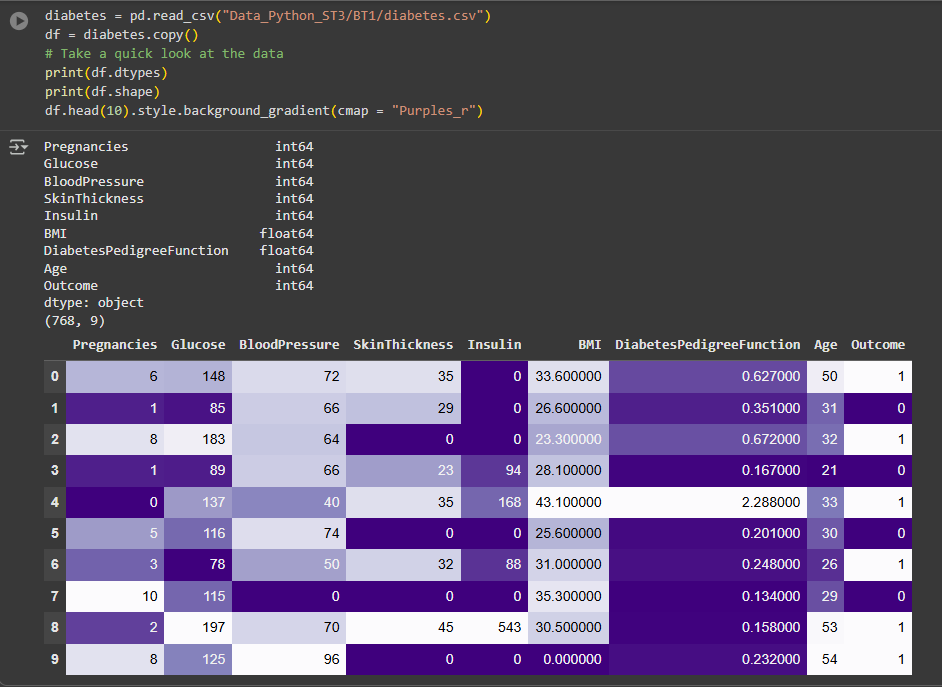
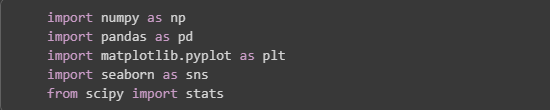


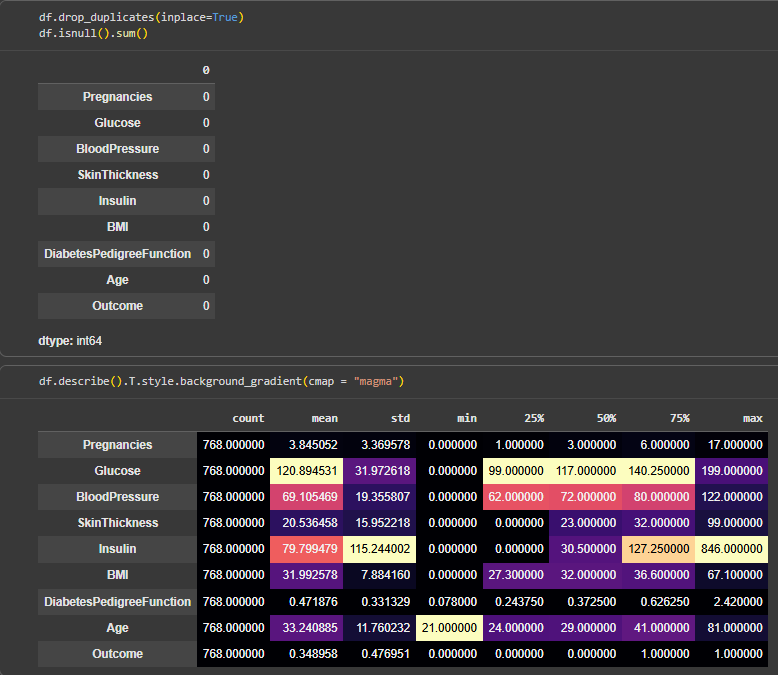


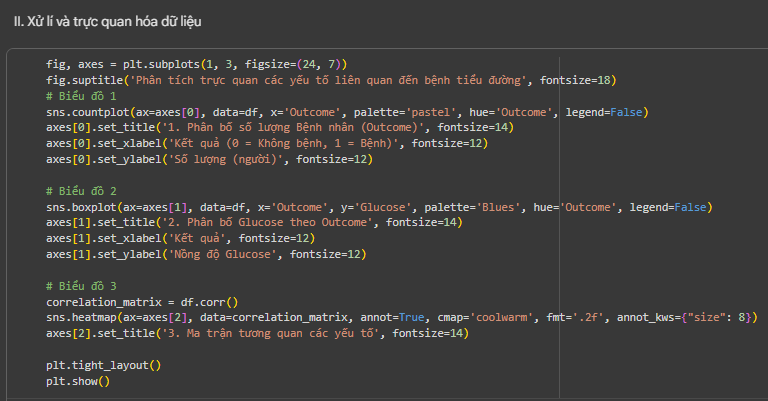


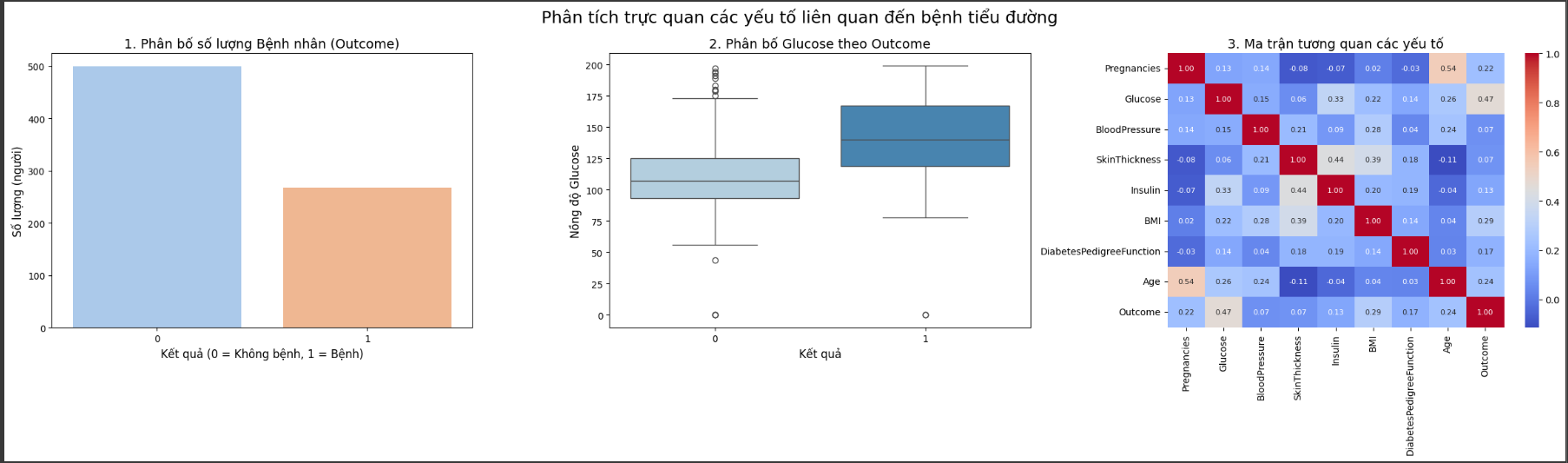
### 1.1.4. Bài tập thực hành 2:

Thực hiện thống kê mô tả trên tập dữ liệu về bệnh tiểu đường. Dữ liệu lấy tại <https://www.kaggle.com/code/vincentlugat/pima-indians-diabetes-eda-prediction-0-906>









## **1.2: XỬ LÝ VÀ TRỰC QUAN HÓA DỮ LIỆU**

### 1.2.1. Ôn tập lý thuyết

#### **+ Trực quan hóa dữ liệu có vai trò gì trong phân tích dữ liệu? Tại sao nó quan trọng trong khám phá dữ liệu (EDA)?**

Trực quan hóa dữ liệu (data visualization) là quá trình biểu diễn dữ liệu dưới dạng hình ảnh như biểu đồ, đồ thị hoặc bản đồ để giúp người phân tích dễ dàng nhận ra các xu hướng, mô hình và mối quan hệ trong dữ liệu.

1. Vai trò của trực quan hóa dữ liệu:

Giúp hiểu nhanh dữ liệu: Thay vì đọc hàng trăm hoặc hàng nghìn dòng dữ liệu, các biểu đồ cho phép nhận diện nhanh xu hướng, phân bố, hoặc bất thường.

Phát hiện mẫu và mối quan hệ: Các biểu đồ giúp thấy rõ mối liên hệ giữa các biến (ví dụ: tương quan giữa giá và doanh thu).

Phát hiện ngoại lệ (outliers): Những điểm dữ liệu bất thường thường được nhìn thấy dễ dàng trên biểu đồ như boxplot hoặc scatter plot.

Hỗ trợ ra quyết định: Giúp nhà phân tích hoặc nhà quản lý hiểu kết quả và đưa ra quyết định chính xác hơn.

Truyền đạt kết quả hiệu quả: Biểu đồ trực quan giúp trình bày kết quả dễ hiểu với người không chuyên về dữ liệu.

2. Tầm quan trọng trong khám phá dữ liệu (Exploratory Data Analysis – EDA):

Khám phá cấu trúc dữ liệu: Trực quan hóa giúp xác định kiểu dữ liệu, phạm vi, phân bố và sự thiếu hụt.

Phát hiện sai sót và bất thường: Biểu đồ giúp nhận ra lỗi nhập liệu, giá trị ngoại lai, hoặc xu hướng bất hợp lý.

Định hướng phân tích sâu hơn: Dựa vào các biểu đồ ban đầu, nhà phân tích có thể xác định hướng mô hình hóa hoặc lựa chọn biến quan trọng.

Tăng tính trực giác trong phân tích: Việc nhìn thấy dữ liệu một cách sinh động giúp hiểu sâu hơn về bản chất của vấn đề.

#### **+ Các loại biểu đồ phổ biến (như histogram, scatter plot, boxplot, bar chart) được sử dụng trong các trường hợp nào?**

1. Biểu đồ Histogram (biểu đồ tần suất)

Mục đích: Hiển thị phân bố của một biến liên tục (ví dụ: chiều cao, thu nhập, điểm thi).

Cách hiểu: Các cột thể hiện tần suất (hoặc mật độ) xuất hiện của dữ liệu trong từng khoảng giá trị.

Ứng dụng:

Xác định dạng phân bố (chuẩn, lệch trái, lệch phải).

Phát hiện giá trị ngoại lai hoặc dữ liệu bất thường.

So sánh phân bố giữa các nhóm (khi vẽ nhiều histogram).

2. Biểu đồ Scatter Plot (biểu đồ phân tán)

Mục đích: Thể hiện mối quan hệ giữa hai biến liên tục.

Cách hiểu: Mỗi điểm trên đồ thị biểu diễn một quan sát (x, y).

Ứng dụng:

Phát hiện xu hướng tuyến tính hoặc phi tuyến.

Xác định mức độ tương quan giữa hai biến.

Phát hiện nhóm dữ liệu hoặc outlier.

3. Biểu đồ Boxplot (biểu đồ hộp)

Mục đích: Tóm tắt phân bố dữ liệu bằng các thước đo như Q1, Q2 (median), Q3, và outlier.

Cách hiểu:

Đường giữa hộp là trung vị (median).

Mép hộp là Q1 và Q3 (25% và 75% dữ liệu).

Các điểm ngoài “râu” hộp là giá trị ngoại lai.

Ứng dụng:

So sánh phân bố của nhiều nhóm khác nhau.

Phát hiện outlier.

Đánh giá mức độ biến thiên của dữ liệu.

4. Biểu đồ Bar Chart (biểu đồ cột)

Mục đích: So sánh giá trị giữa các nhóm hoặc danh mục (categorical variables).

Cách hiểu: Mỗi cột thể hiện giá trị hoặc tần suất của một nhóm.

Ứng dụng:

So sánh quy mô giữa các nhóm (ví dụ: doanh thu theo khu vực).

Trình bày kết quả khảo sát, tỉ lệ phần trăm.

Theo dõi thay đổi qua thời gian (nếu các cột được sắp theo trình tự).

#### **+ Làm thế nào để chọn loại biểu đồ phù hợp với đặc điểm của dữ liệu (ví dụ: dữ liệu phân loại, dữ liệu số, dữ liệu thời gian)?**

1. Dữ liệu phân loại (Categorical Data)

Là dữ liệu chia theo nhóm, loại, hoặc danh mục (ví dụ: giới tính, nghề nghiệp, vùng miền).  
Mục tiêu: So sánh tần suất hoặc tỷ lệ giữa các nhóm.

Biểu đồ phù hợp:

Bar chart (biểu đồ cột): So sánh giá trị hoặc tần suất giữa các danh mục.

Pie chart (biểu đồ tròn): Hiển thị tỷ lệ phần trăm của từng nhóm (chỉ dùng khi số nhóm ít, dưới 6).

Stacked bar chart: So sánh cấu trúc thành phần giữa nhiều nhóm.

2. Dữ liệu số (Numerical Data)

Là dữ liệu định lượng có thể đo lường (ví dụ: thu nhập, điểm thi, tuổi).  
Mục tiêu: Xem phân bố, trung tâm, độ biến thiên, hoặc mối quan hệ.

Biểu đồ phù hợp:

Histogram: Thể hiện phân bố của một biến liên tục.

Boxplot: Tóm tắt phân bố và phát hiện giá trị ngoại lai.

Scatter plot: Phân tích mối quan hệ giữa hai biến liên tục.

Density plot: So sánh phân bố mượt hơn histogram, đặc biệt khi cần biểu diễn nhiều nhóm cùng lúc.

3. Dữ liệu theo thời gian (Time Series Data)

Là dữ liệu được thu thập theo thứ tự thời gian (ví dụ: doanh thu hàng tháng, giá cổ phiếu theo ngày).  
Mục tiêu: Theo dõi xu hướng, chu kỳ, hoặc biến động theo thời gian.

Biểu đồ phù hợp:

Line chart (biểu đồ đường): Hiển thị xu hướng thay đổi theo thời gian.

Area chart: Tương tự line chart nhưng nhấn mạnh tổng giá trị tích lũy theo thời gian.

Candlestick chart: Dùng cho dữ liệu tài chính (biến động giá mở, đóng, cao, thấp).

4. Dữ liệu có nhiều biến (Multivariate Data)

Khi phân tích hơn hai biến cùng lúc.

Biểu đồ phù hợp:

Heatmap: Biểu diễn mối quan hệ giữa nhiều biến dưới dạng màu sắc.

Bubble chart: Mở rộng scatter plot với thêm một biến thể hiện bằng kích thước điểm.

Pair plot: Trực quan hóa toàn bộ mối tương quan giữa nhiều biến số cùng lúc.

#### **+ Sự khác biệt giữa các thư viện trực quan hóa trong Python như Matplotlib, Seaborn và Plotly là gì?**

1. Matplotlib

Đặc điểm:

Là thư viện cơ bản và lâu đời nhất trong Python dùng để vẽ đồ thị 2D.

Cung cấp toàn quyền tùy chỉnh: màu sắc, kiểu đường, kích thước, nhãn, trục, chú thích, v.v.

Là nền tảng cho nhiều thư viện khác (như Seaborn, Pandas plotting).

Ưu điểm:

Linh hoạt, mạnh mẽ, có thể tạo mọi loại biểu đồ (từ đơn giản đến phức tạp).

Hỗ trợ tốt cho xuất file hình (PNG, PDF, SVG).

Phù hợp khi cần kiểm soát chi tiết từng phần của biểu đồ.

Nhược điểm:

Cú pháp dài và phức tạp.

Thiết kế mặc định không đẹp, cần nhiều tùy chỉnh để đạt tính thẩm mỹ cao.

2. Seaborn

Đặc điểm:

Xây dựng trên nền Matplotlib, được thiết kế để tạo biểu đồ thống kê đẹp và trực quan hơn.

Tích hợp tốt với Pandas DataFrame, thuận tiện cho phân tích dữ liệu.

Ưu điểm:

Tự động tối ưu hóa màu sắc, tỷ lệ, và phong cách đồ thị.

Có nhiều biểu đồ thống kê cao cấp: boxplot, violinplot, pairplot, heatmap,...

Hỗ trợ dễ dàng hiển thị mối quan hệ giữa các biến và phân bố dữ liệu.

Nhược điểm:

Tùy biến chi tiết thấp hơn Matplotlib.

Khó tạo các biểu đồ phi thống kê (ví dụ: biểu đồ tuỳ chỉnh không nằm trong các dạng hỗ trợ sẵn).

3. Plotly

Đặc điểm:

Là thư viện tương tác (interactive), cho phép người dùng phóng to, di chuyển, xem thông tin chi tiết trực tiếp trên biểu đồ.

Dựa trên JavaScript, có thể dùng trong trình duyệt hoặc kết hợp với Dash để xây dựng dashboard động.

Ưu điểm:

Biểu đồ đẹp, hiện đại và có tính tương tác cao.

Hỗ trợ nhiều loại biểu đồ phức tạp: 3D plot, map, bubble chart, radar chart, v.v.

Dễ dàng chia sẻ qua web, notebook hoặc dashboard.

Nhược điểm:

Nặng hơn, tốc độ chậm hơn khi xử lý dữ liệu lớn.

Tùy chỉnh chi tiết ít linh hoạt hơn Matplotlib trong một số trường hợp.

#### **+ Những nguyên tắc thiết kế nào cần tuân thủ để tạo ra một biểu đồ trực quan hóa dễ hiểu và hiệu quả?**

1. Xác định rõ mục tiêu và đối tượng người xem

Trước khi vẽ, cần hiểu rõ biểu đồ dùng để trả lời câu hỏi gì (ví dụ: xu hướng, so sánh, phân bố, mối quan hệ...).

Lựa chọn loại biểu đồ và mức độ chi tiết phù hợp với người xem (nhà phân tích, quản lý hay công chúng).

2. Chọn loại biểu đồ phù hợp với dữ liệu

Biểu đồ cột (bar chart): So sánh các nhóm hoặc danh mục.

Biểu đồ đường (line chart): Thể hiện xu hướng theo thời gian.

Histogram hoặc boxplot: Biểu diễn phân bố dữ liệu.

Scatter plot: Phân tích mối quan hệ giữa hai biến số.  
🡪 Sử dụng đúng loại biểu đồ giúp truyền đạt ý nghĩa chính xác và tránh hiểu nhầm.

3. Giữ thiết kế đơn giản và dễ đọc

Tránh rườm rà: Hạn chế màu sắc, hiệu ứng 3D hoặc chi tiết thừa.

Không lặp thông tin: Tránh gắn nhãn dữ liệu khi trục đã thể hiện rõ.

Tối ưu tỷ lệ: Trục tọa độ nên bắt đầu từ 0 (trừ khi có lý do thống kê cụ thể).

Sử dụng khoảng trắng hợp lý: Giúp biểu đồ thoáng và dễ nhìn hơn.

4. Sử dụng màu sắc hợp lý

Màu sắc nên có ý nghĩa: Ví dụ, màu đỏ cho giảm, xanh cho tăng.

Giới hạn số màu: Không dùng quá 5 màu khác nhau trong cùng một biểu đồ.

Chọn bảng màu có độ tương phản tốt, đặc biệt khi in trắng đen hoặc cho người bị mù màu.

Với dữ liệu phân loại 🡪 dùng màu rời rạc, với dữ liệu liên tục 🡪 dùng màu chuyển sắc (gradient).

5. Gắn nhãn và tiêu đề rõ ràng

Tiêu đề nên mô tả ngắn gọn nội dung biểu đồ.

Trục X, Y cần ghi rõ tên biến, đơn vị đo.

Chú thích (legend) phải dễ hiểu, đặt ở vị trí dễ nhìn.

Nếu cần, thêm ghi chú (annotation) để làm nổi bật điểm quan trọng.

6. Tránh gây hiểu nhầm

Không thay đổi tỷ lệ trục hoặc cắt trục tùy tiện, vì dễ làm sai lệch nhận thức.

Giữ tỷ lệ đúng giữa các giá trị: ví dụ, cột cao gấp đôi nghĩa là giá trị gấp đôi.

Không dùng biểu đồ tròn nếu các phần không cộng lại thành 100%.

7. Làm nổi bật thông tin quan trọng

Dùng màu sắc tương phản, nhãn dữ liệu hoặc chú thích để nhấn mạnh kết quả chính.

Giảm bớt các yếu tố phụ để người xem tập trung vào thông tin cốt lõi.

8. Đảm bảo tính nhất quán và dễ so sánh

Khi có nhiều biểu đồ trong cùng một báo cáo, hãy giữ cùng định dạng, màu sắc, tỷ lệ và phong cách.

Giúp người xem dễ dàng so sánh giữa các biểu đồ khác nhau.

#### **+ Làm thế nào để tạo một biểu đồ đơn giản như histogram hoặc bar chart bằng Matplotlib? Bạn có thể chia sẻ đoạn code mẫu không?**

1. Biểu đồ Histogram (biểu đồ tần suất)

Biểu đồ này dùng để thể hiện phân bố của dữ liệu liên tục, cho biết dữ liệu tập trung ở đâu và có dạng phân bố như thế nào.

Ví dụ:

import matplotlib.pyplot as plt

import numpy as np

# Tạo dữ liệu mẫu (ví dụ: điểm thi của 100 sinh viên)

data = np.random.normal(loc=70, scale=10, size=100)

# Vẽ biểu đồ histogram

plt.hist(data, bins=10, color='skyblue', edgecolor='black')

# Thêm tiêu đề và nhãn trục

plt.title("Phân bố điểm thi của sinh viên")

plt.xlabel("Điểm")

plt.ylabel("Tần suất")

# Hiển thị biểu đồ

plt.show()

Giải thích:

np.random.normal(loc, scale, size) tạo dữ liệu ngẫu nhiên theo phân bố chuẩn (ở đây trung bình 70, độ lệch chuẩn 10).

bins=10 chia dữ liệu thành 10 khoảng.

color và edgecolor giúp biểu đồ rõ ràng hơn.

2. Biểu đồ Bar Chart (biểu đồ cột)

Dùng để so sánh giá trị giữa các nhóm hoặc danh mục.

Ví dụ:

import matplotlib.pyplot as plt

# Dữ liệu mẫu: doanh thu theo khu vực

khu\_vuc = ['Bắc', 'Trung', 'Nam']

doanh\_thu = [120, 90, 150]

# Vẽ biểu đồ cột

plt.bar(khu\_vuc, doanh\_thu, color='orange')

# Thêm tiêu đề và nhãn trục

plt.title("Doanh thu theo khu vực")

plt.xlabel("Khu vực")

plt.ylabel("Doanh thu (triệu đồng)")

# Hiển thị giá trị trên đầu cột

for i, v in enumerate(doanh\_thu):

plt.text(i, v + 3, str(v), ha='center', fontweight='bold')

# Hiển thị biểu đồ

plt.show()

Giải thích:

plt.bar() tạo các cột đại diện cho từng nhóm.

Vòng lặp plt.text() hiển thị giá trị cụ thể trên đầu mỗi cột.

#### **+ Làm thế nào để xuất biểu đồ từ Python ra các định dạng như PNG, PDF hoặc HTML để sử dụng trong báo cáo?**

1. Xuất biểu đồ bằng Matplotlib

Cách 1: Lưu trực tiếp bằng savefig()

Sau khi tạo biểu đồ, bạn chỉ cần gọi lệnh:

import matplotlib.pyplot as plt

import numpy as np

# Tạo dữ liệu ví dụ

data = np.random.randn(100)

# Vẽ histogram

plt.hist(data, bins=15, color='skyblue', edgecolor='black')

plt.title("Phân bố dữ liệu")

plt.xlabel("Giá trị")

plt.ylabel("Tần suất")

# Lưu biểu đồ ra file PNG

plt.savefig("bieu\_do.png")

# Lưu dưới dạng PDF hoặc SVG

plt.savefig("bieu\_do.pdf")

plt.savefig("bieu\_do.svg")

# Hiển thị biểu đồ trên màn hình

plt.show()

Giải thích:

plt.savefig("ten\_file.png"): Lưu biểu đồ hiện tại ra tệp hình.

Bạn có thể đổi định dạng bằng cách thay phần mở rộng (.png, .pdf, .svg, .jpg, …).

Có thể thêm tham số để điều chỉnh chất lượng:

plt.savefig("bieu\_do\_cao.png", dpi=300, bbox\_inches='tight')

dpi=300: tăng độ phân giải (phù hợp khi chèn vào báo cáo hoặc in ấn).

bbox\_inches='tight': loại bỏ khoảng trắng dư quanh hình.

2. Xuất biểu đồ tương tác bằng Plotly

Plotly cho phép xuất biểu đồ tương tác ra HTML - rất hữu ích cho báo cáo web hoặc dashboard.

import plotly.express as px

import pandas as pd

# Tạo dữ liệu ví dụ

df = pd.DataFrame({

"Khu vực": ["Bắc", "Trung", "Nam"],

"Doanh thu": [120, 90, 150]

})

# Tạo biểu đồ cột

fig = px.bar(df, x="Khu vực", y="Doanh thu", title="Doanh thu theo khu vực")

# Lưu biểu đồ ra HTML

fig.write\_html("bieu\_do\_tuong\_tac.html")

# Ngoài ra có thể lưu ra PNG hoặc PDF (yêu cầu cài thêm kaleido)

fig.write\_image("bieu\_do\_plotly.png")

fig.write\_image("bieu\_do\_plotly.pdf")

Giải thích:

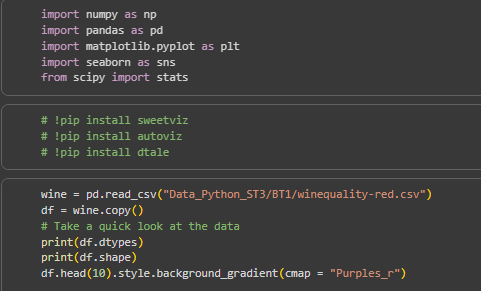
write\_html() → xuất file HTML tương tác (người xem có thể di chuột, zoom, phóng to).

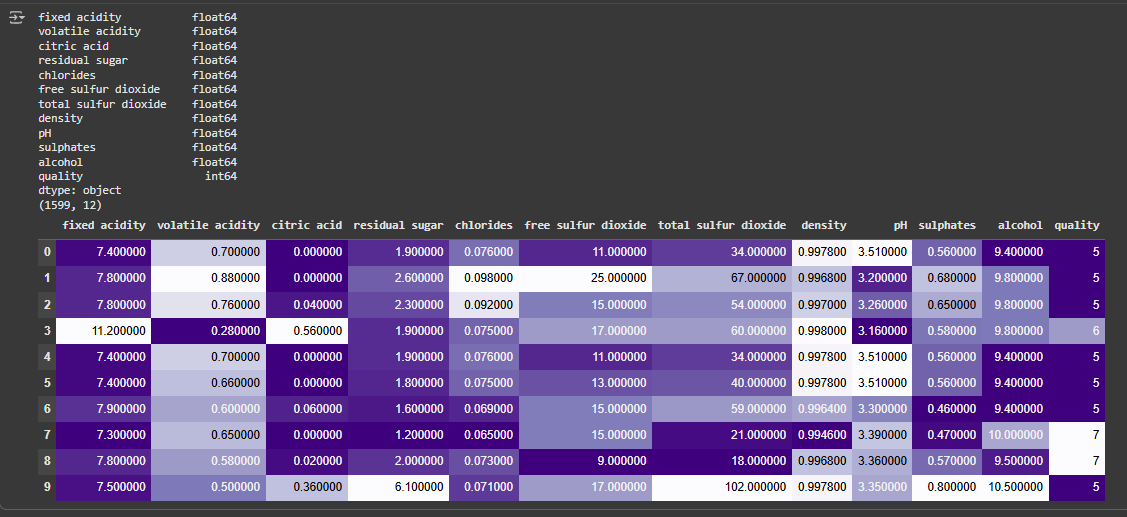
write\_image() → xuất ra ảnh tĩnh, cần cài thư viện phụ:

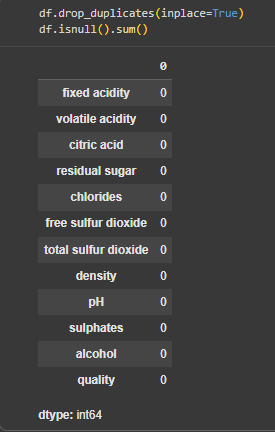
pip install kaleido

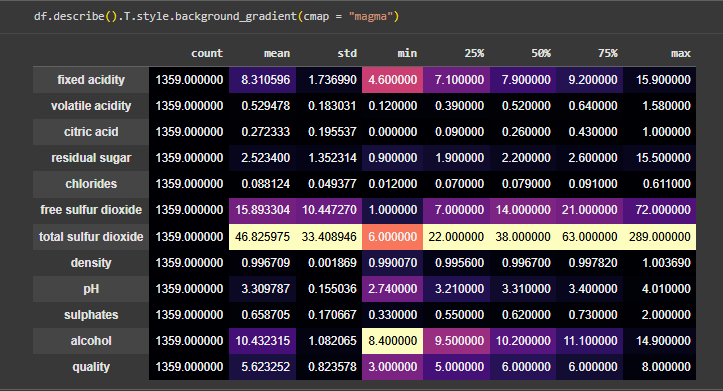
### 1.2.2. Bài tập thực hành 1

+ Thực hiện trực quan hóa dữ liệu trên tập dữ liệu về phân loại chất lượng rượu đỏ. Dữ liệu lấy tại <https://www.kaggle.com/code/eisgandar/red-wine-quality-eda-classification>

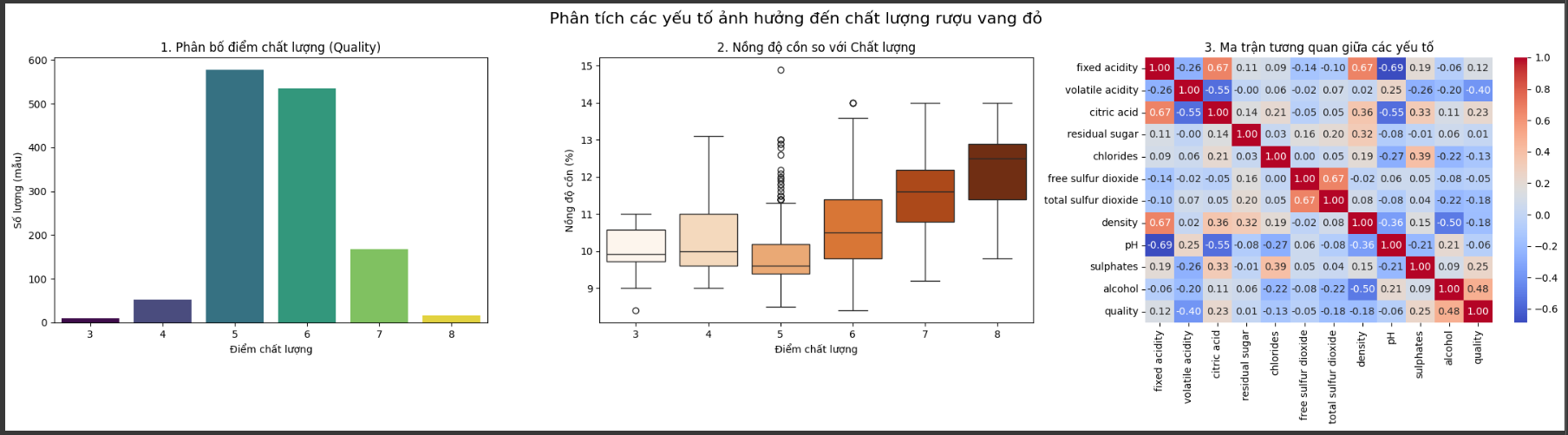






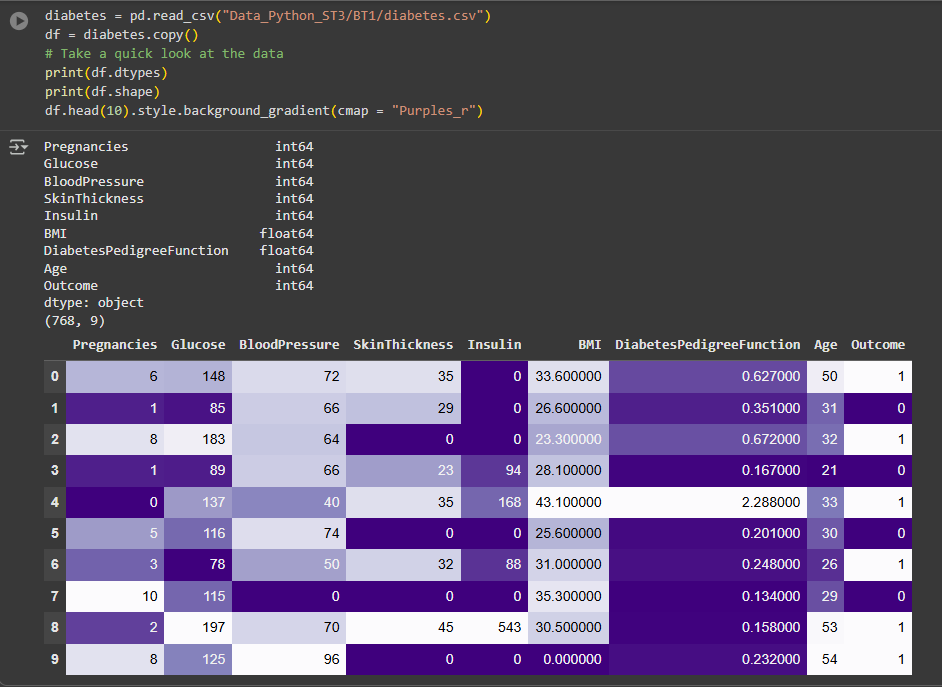
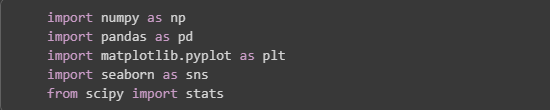


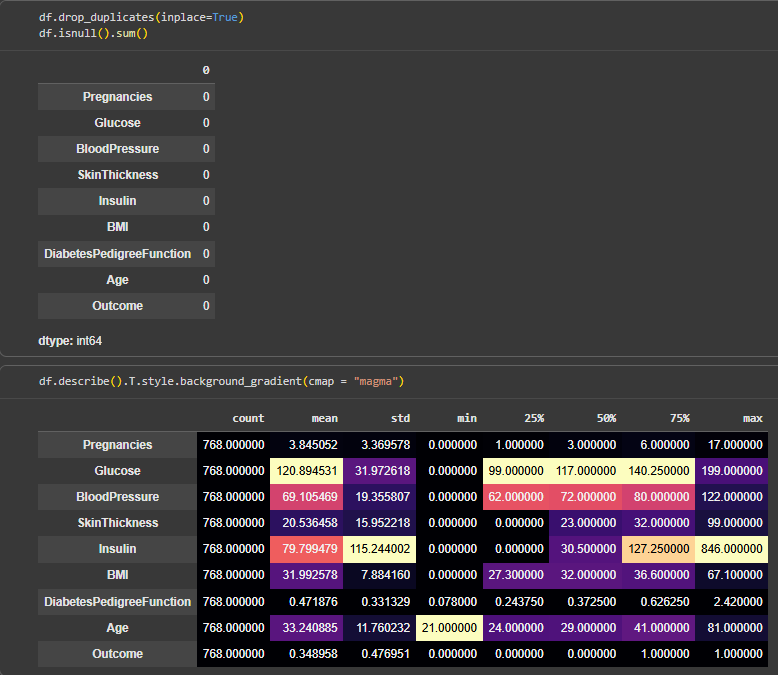


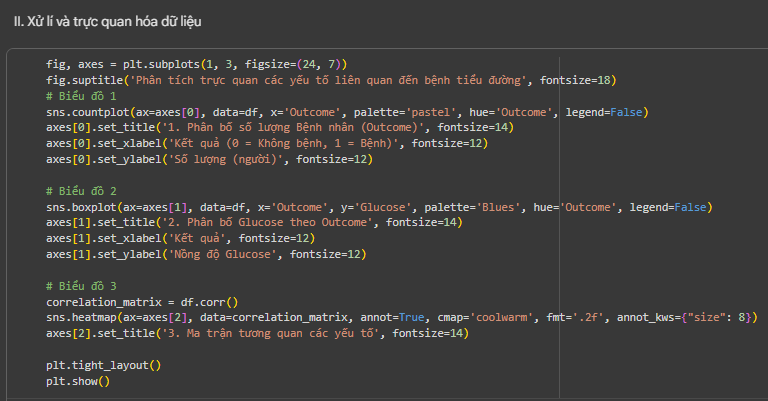


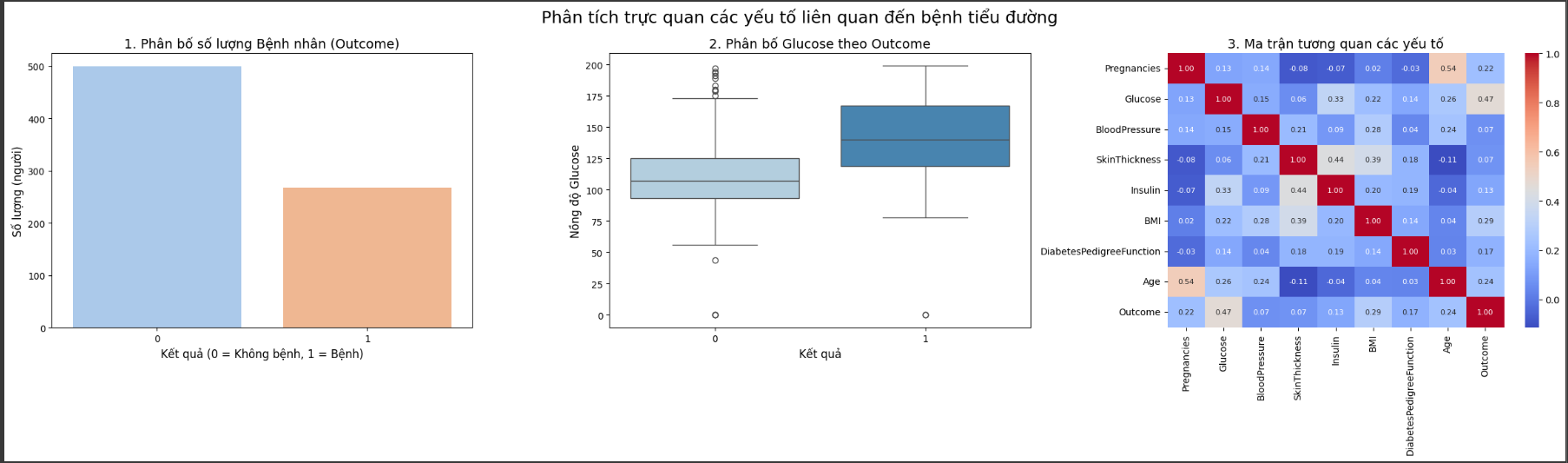
### 1.2.3. Bài tập thực hành 2

a) + Thực hiện trực quan hóa dữ liệu trên tập dữ liệu về bệnh tiểu đường. Dữ liệu lấy tại <https://www.kaggle.com/code/vincentlugat/pima-indians-diabetes-eda-prediction-0-906>

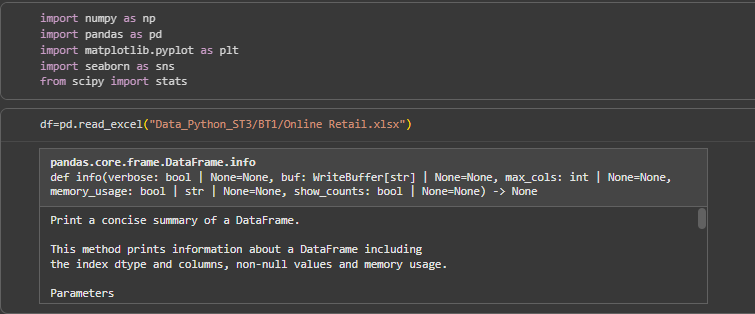


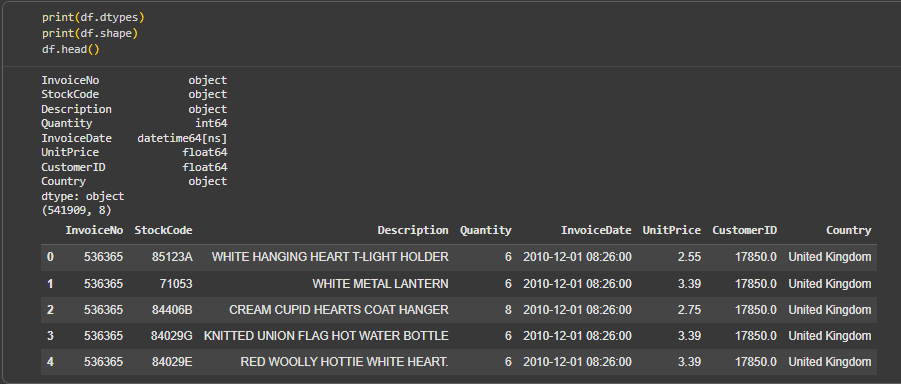


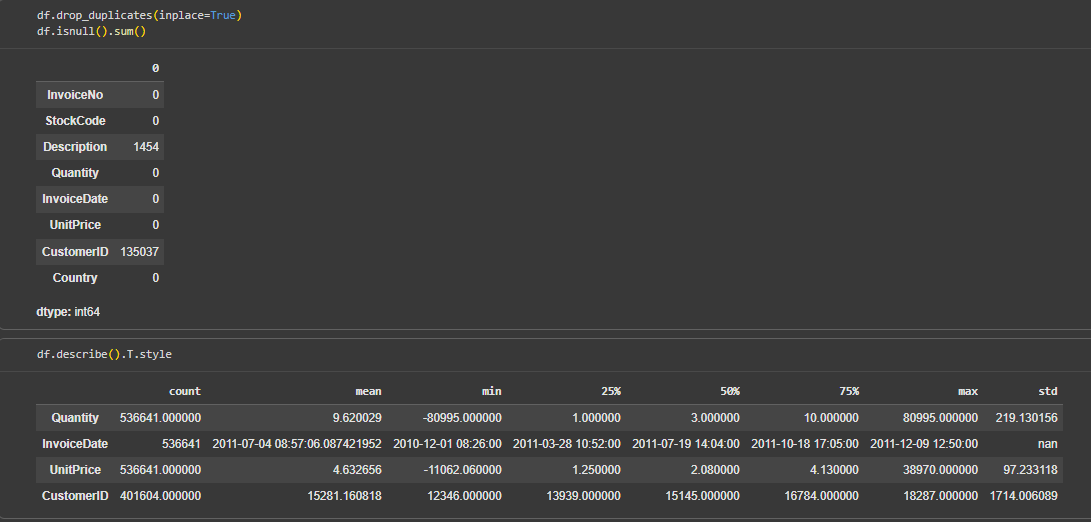


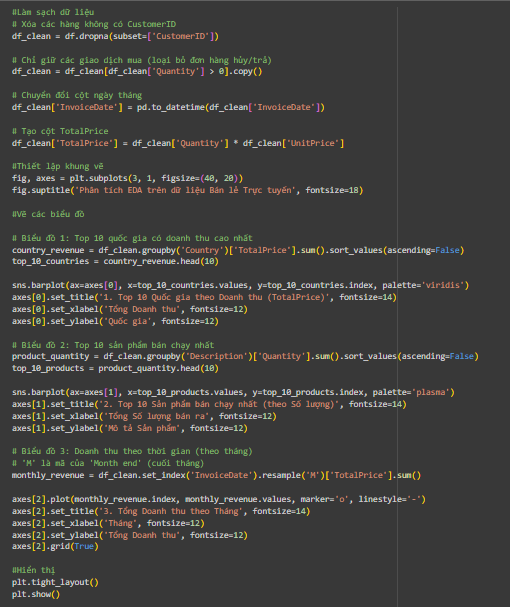


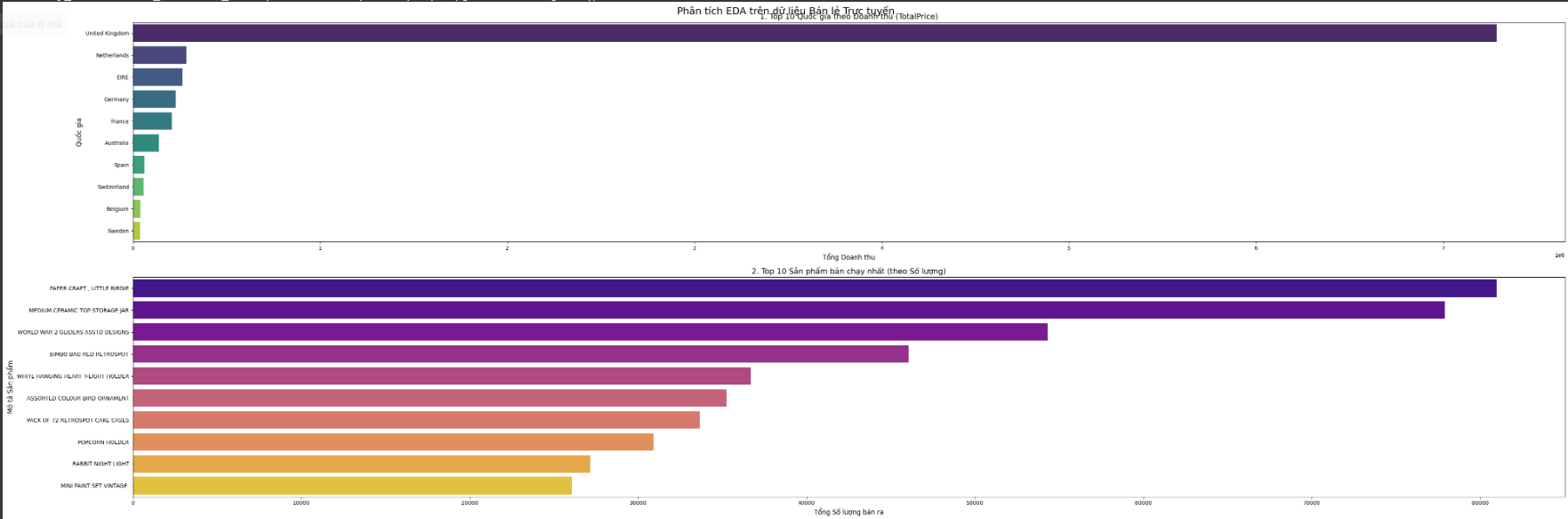
b) + Thực hiện EDA trên tập dữ liệu mua sắm tại siêu thị. Tập dữ liệu lấy từ <https://www.kaggle.com/code/rajatkumar30/eda-online-retail>

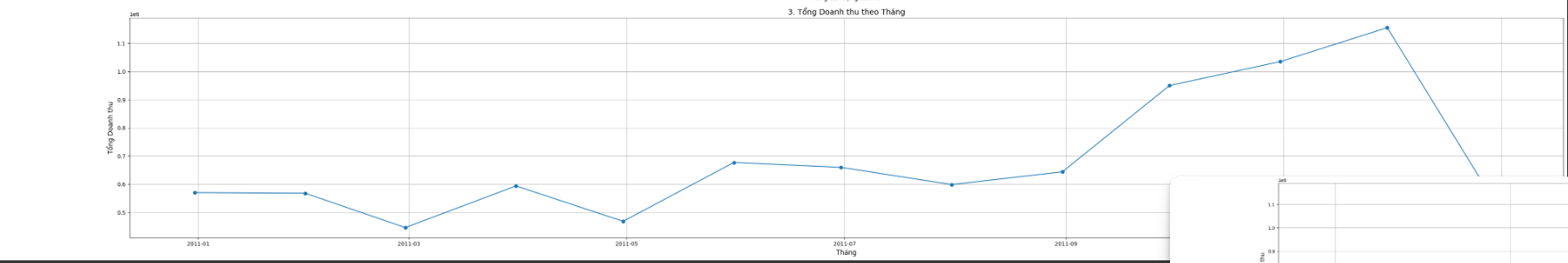












## **1.3: PHÂN TÍCH ĐƠN BIẾN VÀ HAI BIẾN**

### 1.3.1. Ôn tập lý thuyết

#### **+ Phân tích đơn biến (univariate analysis) là gì? Nó khác gì với phân tích hai biến (bivariate analysis) trong khám phá dữ liệu?**

Phân tích đơn biến (Univariate Analysis) là quá trình phân tích một biến duy nhất trong tập dữ liệu nhằm mô tả đặc điểm, phân bố và xu hướng của biến đó. Mục tiêu chính là hiểu rõ cấu trúc của biến thông qua các thước đo thống kê (trung bình, trung vị, độ lệch chuẩn, tần suất, v.v.) hoặc biểu đồ (histogram, boxplot, bar chart).

Phân tích hai biến (Bivariate Analysis) là quá trình phân tích mối quan hệ giữa hai biến trong tập dữ liệu. Mục tiêu là tìm hiểu xem hai biến có liên hệ, tương quan hay ảnh hưởng lẫn nhau hay không.  
Phân tích hai biến thường được thực hiện bằng:

Biểu đồ: scatter plot (đối với hai biến số), bar chart (đối với biến phân loại và biến số).

Chỉ số thống kê: hệ số tương quan Pearson, Spearman, hoặc kiểm định Chi-square.

Khác biệt chính:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tiêu chí | Phân tích đơn biến | Phân tích hai biến |  |  |  |
| Số lượng biến | 1 | 2 |  |  |  |
| Mục tiêu | Mô tả đặc điểm của từng biến | Xem mối quan hệ giữa hai biến |  |  |  |
| Kỹ thuật chính | Trung bình, trung vị, phương sai, histogram, boxplot | Scatter plot, cross-tabulation, hệ số tương quan, kiểm định giả thuyết |  |  |  |
| Ví dụ | Phân bố điểm thi của sinh viên | Mối quan hệ giữa điểm thi và số giờ học |  |  |  |

Tóm lại:  
Phân tích đơn biến giúp hiểu từng biến riêng lẻ, còn phân tích hai biến giúp hiểu cách các biến liên hệ với nhau, là bước quan trọng trước khi xây dựng mô hình thống kê hoặc máy học.

#### **+ Các thước đo thống kê nào thường được sử dụng trong phân tích đơn biến (ví dụ: trung bình, trung vị, mode, độ lệch chuẩn)?**

1. Các thước đo khuynh hướng trung tâm (Measures of Central Tendency)

Dùng để biểu diễn giá trị trung tâm hoặc điển hình của dữ liệu.

Trung bình (Mean):  
Là giá trị trung bình cộng của tất cả các quan sát.  
🡪 Phù hợp khi dữ liệu phân bố đều, không có ngoại lai.

Trung vị (Median):  
Là giá trị nằm giữa khi dữ liệu được sắp xếp theo thứ tự.  
🡪 Thích hợp khi dữ liệu có ngoại lai hoặc phân bố lệch.

Mode (Giá trị thường xuyên nhất):  
Là giá trị xuất hiện nhiều nhất trong tập dữ liệu.  
🡪 Hữu ích với dữ liệu phân loại hoặc rời rạc.

2. Các thước đo độ phân tán (Measures of Dispersion)

Cho biết mức độ biến động của dữ liệu quanh giá trị trung tâm.

Phạm vi (Range):  
Hiệu giữa giá trị lớn nhất và nhỏ nhất.  
🡪 Đơn giản nhưng dễ bị ảnh hưởng bởi ngoại lai.

Phương sai (Variance):  
Đo lường mức độ phân tán trung bình của dữ liệu so với trung bình.

Độ lệch chuẩn (Standard Deviation):  
Là căn bậc hai của phương sai, biểu thị mức độ dao động của dữ liệu.  
🡪 Giá trị càng lớn 🡪 dữ liệu càng phân tán.

IQR (Interquartile Range):  
Là khoảng giữa Q1 (phân vị 25%) và Q3 (phân vị 75%).  
🡪 Giúp loại bỏ ảnh hưởng của ngoại lai.

3. Các thước đo hình dạng phân bố (Shape of Distribution)

Dùng để mô tả hình dạng của dữ liệu.

Skewness (Độ lệch): cho biết dữ liệu lệch trái hay lệch phải.

Kurtosis (Độ nhọn): cho biết dữ liệu tập trung hay phân tán quanh trung tâm.

#### **+ Trong phân tích hai biến, làm thế nào để xác định mối quan hệ giữa hai biến (ví dụ: tương quan, nhân quả)?**

1. Xác định mối quan hệ tương quan (Correlation)

Tương quan thể hiện mức độ và chiều hướng liên hệ tuyến tính giữa hai biến định lượng.

Công cụ chính: Hệ số tương quan (correlation coefficient)

Hệ số tương quan Pearson (r): đo mối quan hệ tuyến tính giữa hai biến số.

r nằm trong khoảng [-1, 1]:

r > 0 🡪 tương quan thuận (biến này tăng thì biến kia tăng).

r < 0 🡪 tương quan nghịch (biến này tăng thì biến kia giảm).

r ≈ 0 🡪 không có tương quan tuyến tính rõ ràng.

Hệ số tương quan Spearman (ρ): dùng cho dữ liệu thứ bậc hoặc không tuyến tính.

Biểu đồ minh họa: Scatter plot giúp nhìn trực quan về mối quan hệ giữa hai biến.

2. Phân tích mối quan hệ giữa các loại biến khác

Tùy theo loại biến, ta chọn công cụ phù hợp:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Loại biến | Phương pháp phân tích | Ví dụ |  |  |  |
| Hai biến định lượng | Hệ số tương quan Pearson / Spearman, scatter plot | Chiều cao và cân nặng |  |  |  |
| Một biến định lượng + một biến phân loại | So sánh trung bình (t-test, ANOVA), boxplot | Thu nhập theo giới tính |  |  |  |
| Hai biến phân loại | Bảng chéo (cross-tabulation), kiểm định Chi-square | Giới tính và việc sử dụng mạng xã hội |  |  |  |

3. Xác định mối quan hệ nhân quả (Causation)

Tương quan không đồng nghĩa với nhân quả (“Correlation ≠ Causation”).  
Để xác định quan hệ nhân quả, cần có phân tích sâu hơn, thường qua:

Thực nghiệm có kiểm soát (Controlled experiments): thay đổi một biến độc lập để xem ảnh hưởng đến biến phụ thuộc.

Mô hình hồi quy (Regression analysis): kiểm soát các yếu tố khác để kiểm tra xem biến X có gây ra thay đổi trong biến Y không.

Phân tích chuỗi thời gian (Time series): dùng để xem biến nào thay đổi trước, có thể suy ra chiều hướng nhân quả.

#### **+ Sự khác biệt giữa tương quan (correlation) và hiệp biến (covariance) trong phân tích hai biến là gì?**

1. Hiệp biến (Covariance)

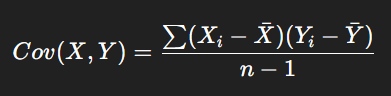
Khái niệm:  
Hiệp biến đo lường mức độ hai biến thay đổi cùng nhau.

Nếu hai biến tăng hoặc giảm cùng lúc, hiệp biến > 0.

Nếu một biến tăng khi biến kia giảm, hiệp biến < 0.

Nếu không có mối liên hệ rõ ràng, hiệp biến ≈ 0.

Công thức:



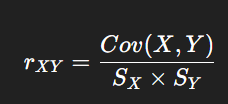
Trong đó:  
Xˉ,Yˉ là giá trị trung bình của hai biến X và Y.

Đơn vị:  
Phụ thuộc vào đơn vị đo lường của hai biến, nên khó so sánh giữa các cặp biến khác nhau.

2. Tương quan (Correlation)

Khái niệm:  
Tương quan là phiên bản chuẩn hóa của hiệp biến, giúp đo mức độ mạnh yếu và chiều hướng tuyến tính của mối quan hệ giữa hai biến.  
🡪 Có giá trị trong khoảng [-1, 1], dễ diễn giải hơn.

Công thức:



Trong đó SX, SY​ là độ lệch chuẩn của X và Y.

Giải thích:

r > 0 🡪 tương quan thuận.

r < 0 🡪 tương quan nghịch.

r = 0 🡪 không có mối tương quan tuyến tính.

|r| càng gần 1 🡪 mối quan hệ càng mạnh.

#### **+ Khi nào nên sử dụng biểu đồ trực quan hóa trong phân tích đơn biến so với phân tích hai biến?**

1. Phân tích đơn biến (Univariate Analysis)

Mục tiêu: Hiểu đặc điểm, phân bố và xu hướng của *một biến duy nhất*.

Khi nào dùng biểu đồ:

Khi muốn xem phân bố giá trị của biến.

Khi cần phát hiện ngoại lai (outliers) hoặc xu hướng lệch phân bố.

Khi muốn so sánh tần suất hoặc tỷ lệ trong dữ liệu.

Các biểu đồ thường dùng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Loại dữ liệu | Biểu đồ phù hợp | Mục đích |  |  |  |
| Dữ liệu định lượng (số) | Histogram, Boxplot, Density plot | Xem phân bố, độ lệch, ngoại lai |  |  |  |
| Dữ liệu định tính (phân loại) | Bar chart, Pie chart | So sánh tần suất, tỷ lệ các nhóm |  |  |  |

2. Phân tích hai biến (Bivariate Analysis)

Mục tiêu: Khám phá mối quan hệ giữa hai biến (có tương quan, ảnh hưởng hay không).

Khi nào dùng biểu đồ:

Khi muốn xem mức độ và chiều hướng mối quan hệ giữa hai biến số.

Khi cần so sánh sự khác biệt giữa các nhóm dữ liệu.

Các biểu đồ thường dùng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Loại biến | Biểu đồ phù hợp | Mục đích |  |  |  |
| Hai biến định lượng | Scatter plot, Line chart, Heatmap | Thể hiện tương quan hoặc xu hướng tuyến tính |  |  |  |
| Một biến định lượng + một biến phân loại | Boxplot, Violin plot, Bar chart | So sánh phân bố giá trị giữa các nhóm |  |  |  |
| Hai biến phân loại | Clustered bar chart, Mosaic plot, Stacked bar chart | So sánh tần suất hoặc tỷ lệ giữa hai nhóm phân loại |  |  |  |

#### **+ Đoạn code mẫu để tạo biểu đồ scatter plot hoặc heatmap để phân tích mối quan hệ giữa hai biến**

1. Biểu đồ Scatter Plot (Biểu đồ phân tán)

Dùng để xem mối quan hệ tuyến tính giữa hai biến định lượng.

# Import thư viện

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

# Tạo dữ liệu mẫu

data = pd.DataFrame({

'hours\_study': [2, 3, 4, 5, 6, 7, 8, 9],

'score': [50, 55, 65, 70, 75, 80, 85, 90]

})

# Tạo biểu đồ scatter plot

plt.figure(figsize=(6, 4))

sns.scatterplot(data=data, x='hours\_study', y='score', color='blue', s=80)

# Thêm tiêu đề và nhãn

plt.title('Mối quan hệ giữa số giờ học và điểm thi', fontsize=12)

plt.xlabel('Số giờ học')

plt.ylabel('Điểm thi')

# Hiển thị biểu đồ

plt.show()

Ý nghĩa:

Mỗi điểm thể hiện một quan sát (giá trị của hai biến).

Nếu các điểm có xu hướng nằm trên một đường chéo đi lên 🡪 tương quan thuận.

Nếu đi xuống 🡪 tương quan nghịch

2. Biểu đồ Heatmap (Bản đồ nhiệt)

Dùng để xem mức độ tương quan giữa nhiều biến số cùng lúc, thông qua ma trận tương quan.

# Import thư viện

import seaborn as sns

import matplotlib.pyplot as plt

import pandas as pd

# Tạo dữ liệu mẫu

data = pd.DataFrame({

'hours\_study': [2, 3, 4, 5, 6, 7, 8, 9],

'score': [50, 55, 65, 70, 75, 80, 85, 90],

'sleep\_hours': [8, 7.5, 7, 6.5, 6, 6, 5.5, 5]

})

# Tính ma trận tương quan

corr = data.corr()

# Vẽ heatmap

plt.figure(figsize=(5, 4))

sns.heatmap(corr, annot=True, cmap='coolwarm', linewidths=0.5)

# Thêm tiêu đề

plt.title('Ma trận tương quan giữa các biến', fontsize=12)

# Hiển thị biểu đồ

plt.show()

Ý nghĩa:

Mỗi ô trong heatmap thể hiện mức độ tương quan (r) giữa hai biến.

Giá trị gần 1 hoặc -1 🡪 tương quan mạnh (thuận hoặc nghịch).

Giá trị gần 0 🡪 ít hoặc không có tương quan tuyến tính.

#### **+ Làm thế nào để trực quan hóa mối quan hệ giữa một biến số và một biến phân loại bằng biểu đồ boxplot hoặc violin plot trong Python?**

1. Biểu đồ Boxplot (Biểu đồ hộp)

Boxplot dùng để so sánh phân bố của một biến số theo từng nhóm của biến phân loại.  
Nó giúp bạn thấy được trung vị, độ phân tán, và các giá trị ngoại lai (outliers) trong từng nhóm.

# Import thư viện

import seaborn as sns

import matplotlib.pyplot as plt

import pandas as pd

# Tạo dữ liệu mẫu

data = pd.DataFrame({

'gender': ['Nam', 'Nữ', 'Nam', 'Nữ', 'Nam', 'Nữ', 'Nam', 'Nữ'],

'income': [10, 12, 15, 18, 20, 22, 25, 27]

})

# Vẽ biểu đồ boxplot

plt.figure(figsize=(6, 4))

sns.boxplot(data=data, x='gender', y='income', palette='Set2')

# Thêm tiêu đề và nhãn

plt.title('So sánh thu nhập theo giới tính', fontsize=12)

plt.xlabel('Giới tính')

plt.ylabel('Thu nhập (triệu đồng)')

# Hiển thị biểu đồ

plt.show()

Giải thích:

Mỗi “hộp” thể hiện phân bố của thu nhập trong từng nhóm giới tính.

Đường giữa hộp là trung vị (median).

Các “râu” thể hiện phạm vi dữ liệu, còn dấu chấm ngoài râu là ngoại lai.

Giúp so sánh mức trung bình và độ biến động giữa các nhóm.

2. Biểu đồ Violin Plot

Violin plot thể hiện phân bố xác suất của dữ liệu trong từng nhóm, là sự kết hợp giữa boxplot và biểu đồ mật độ (density plot).  
Nó cho cái nhìn mượt mà và trực quan hơn về hình dạng phân bố của dữ liệu.

# Import thư viện

import seaborn as sns

import matplotlib.pyplot as plt

import pandas as pd

# Tạo dữ liệu mẫu

data = pd.DataFrame({

'gender': ['Nam', 'Nữ', 'Nam', 'Nữ', 'Nam', 'Nữ', 'Nam', 'Nữ'],

'income': [10, 12, 15, 18, 20, 22, 25, 27]

})

# Vẽ biểu đồ violin plot

plt.figure(figsize=(6, 4))

sns.violinplot(data=data, x='gender', y='income', palette='Pastel1', inner='box')

# Thêm tiêu đề và nhãn

plt.title('Phân bố thu nhập theo giới tính (Violin Plot)', fontsize=12)

plt.xlabel('Giới tính')

plt.ylabel('Thu nhập (triệu đồng)')

# Hiển thị biểu đồ

plt.show()

Giải thích:

Hình dạng “violin” biểu thị mật độ xuất hiện của giá trị thu nhập.

Phần rộng hơn 🡪 nhóm giá trị xuất hiện nhiều.

Có thể thấy rõ sự khác biệt trong hình dạng phân bố giữa các nhóm.

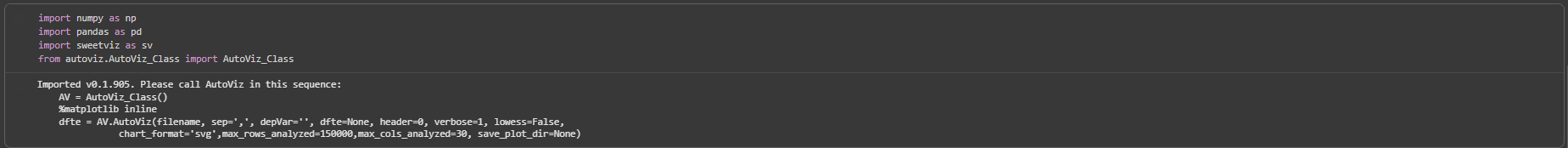
3. Khi nào dùng loại nào?

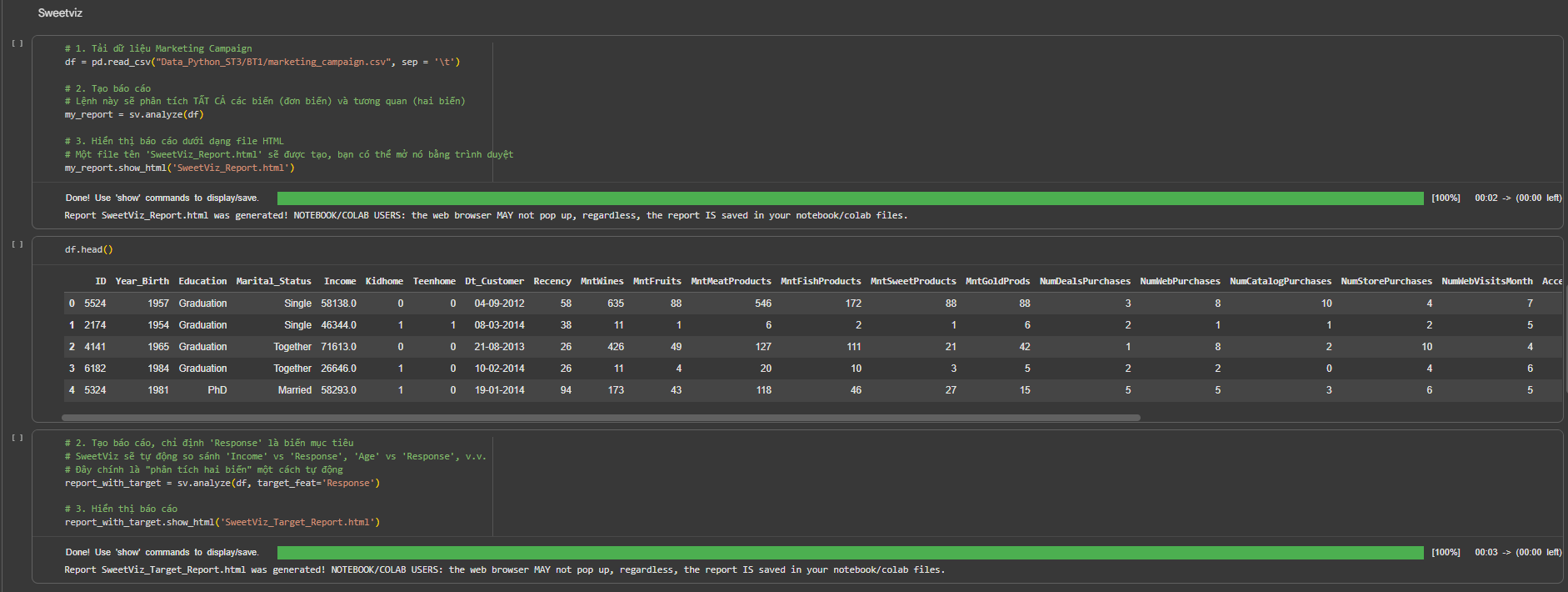
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Loại biểu đồ | Mục đích chính | Khi nên dùng |  |  |  |
| Boxplot | So sánh trung vị, độ phân tán và ngoại lai giữa các nhóm | Khi bạn muốn trình bày ngắn gọn, rõ ràng, dễ đọc |  |  |  |
| Violin plot | Hiển thị hình dạng phân bố xác suất của từng nhóm | Khi bạn muốn thấy rõ phân bố chi tiết và độ dày dữ liệu |  |  |  |

### 1.3.3. Bài tập thực hành 1

Tìm hiểu các tính năng và cách sử dụng sản phẩm SweetViz (<https://pypi.org/project/sweetviz/>) áp dụng trên tập dữ liệu Marketing Campaign







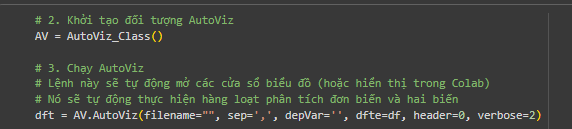
### 1.3.4. Bài tập thực hành 2

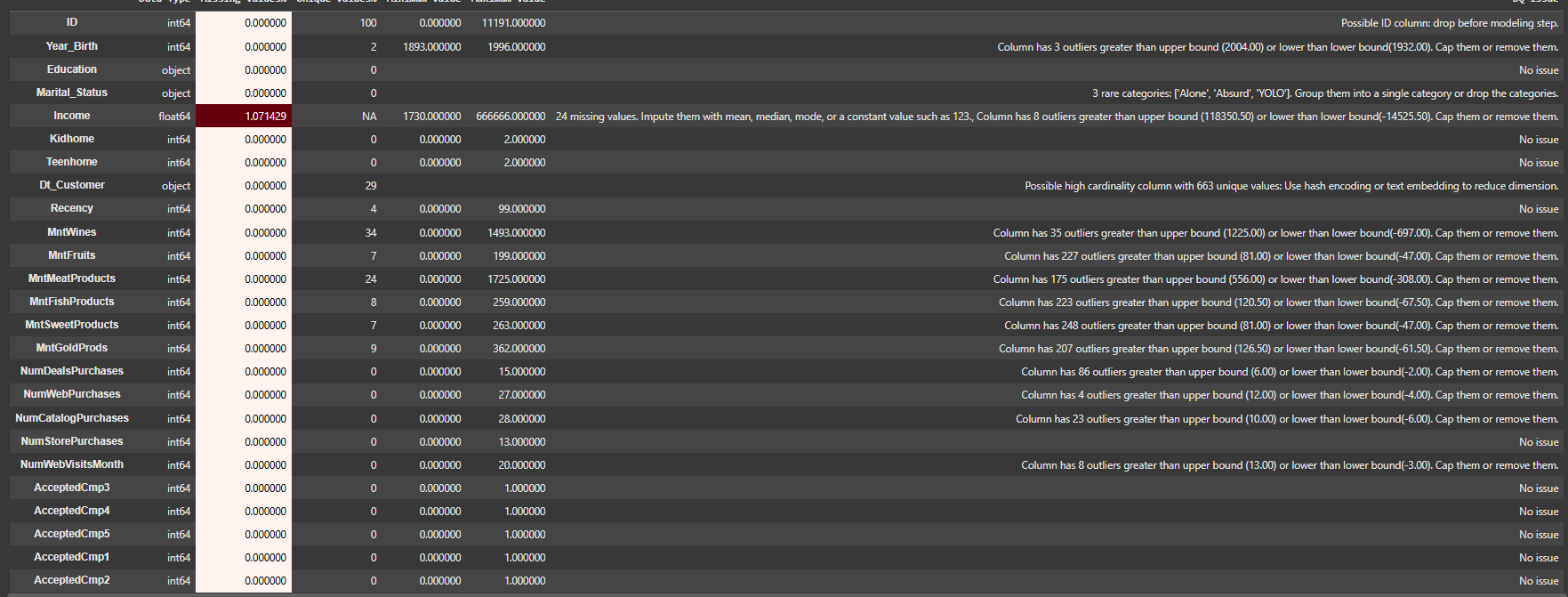
Tìm hiểu các tính năng và cách sử dụng sản phẩm AutoViz (<https://pypi.org/project/autoviz>) áp dụng trên tập dữ liệu Marketing Campaign











# **LAB\_02**

## **2.1: Giải thuật 1: CÂY QUYẾT ĐỊNH VÀ RỪNG CÂY**

### 2.1.1. Ôn tập lý thuyết

#### **+ Quy trình khai phá dữ liệu CRISP – DM (Cross Industry Standard Process for Data Mining) là gì ? Quy trình khai phá dữ liệu SEMMA (Sample, Explore, Modify, Model, Access) là gì?**

*a) Quy trình CRISP–DM (Cross-Industry Standard Process for Data Mining) là một mô hình quy chuẩn được sử dụng rộng rãi trong lĩnh vực khai phá dữ liệu (Data Mining).*Mục tiêu của CRISP–DM là cung cấp một khung làm việc có hệ thống, lặp lại và linh hoạt giúp các nhà phân tích triển khai dự án khai phá dữ liệu một cách hiệu quả, từ khâu hiểu bài toán kinh doanh đến khi ứng dụng kết quả vào thực tế.

CRISP–DM gồm 6 giai đoạn chính:

1. Hiểu biết kinh doanh (Business Understanding)  
Giai đoạn này tập trung vào việc xác định mục tiêu kinh doanh và yêu cầu của dự án. Người thực hiện cần hiểu rõ vấn đề thực tế mà tổ chức muốn giải quyết và chuyển đổi các mục tiêu kinh doanh đó thành mục tiêu khai phá dữ liệu cụ thể.

2. Hiểu biết dữ liệu (Data Understanding)  
Sau khi xác định được mục tiêu, bước tiếp theo là thu thập và khám phá dữ liệu. Mục tiêu là hiểu rõ đặc điểm, cấu trúc và chất lượng của dữ liệu hiện có. Việc này bao gồm việc kiểm tra dữ liệu bị thiếu, dữ liệu ngoại lệ, hoặc dữ liệu không hợp lệ.

3. Chuẩn bị dữ liệu (Data Preparation)  
Đây là giai đoạn chiếm nhiều thời gian nhất trong quy trình. Dữ liệu được làm sạch, chọn lọc, biến đổi và tích hợp để tạo ra tập dữ liệu cuối cùng dùng cho mô hình hóa.

4. Mô hình hóa (Modeling)  
Trong giai đoạn này, các kỹ thuật khai phá dữ liệu và thuật toán mô hình hóa được lựa chọn và áp dụng (như cây quyết định, hồi quy, mạng nơ-ron,…). Người thực hiện cần điều chỉnh các tham số của mô hình để đạt kết quả tốt nhất.

5. Đánh giá (Evaluation)  
Sau khi mô hình được xây dựng, cần đánh giá hiệu quả của mô hình thông qua các chỉ số như độ chính xác, độ nhạy, độ đặc hiệu, hoặc biểu đồ ROC. Ngoài ra, cần xem xét liệu mô hình có đáp ứng đúng mục tiêu kinh doanh đã đặt ra hay không.

6. Triển khai (Deployment)  
Giai đoạn cuối cùng là triển khai kết quả vào thực tế. Mô hình có thể được đưa vào sử dụng trong hệ thống, tạo báo cáo hoặc công cụ hỗ trợ ra quyết định. Sau khi triển khai, mô hình cần được theo dõi và cập nhật thường xuyên khi có sự thay đổi trong dữ liệu hoặc môi trường kinh doanh.

*b) Quy trình SEMMA (viết tắt của Sample, Explore, Modify, Model, Assess) là một phương pháp khai phá dữ liệu (data mining) được SAS Institute đề xuất.*Nó mô tả các bước tuần tự để phát hiện tri thức từ dữ liệu, tập trung vào việc xử lý và phân tích dữ liệu nhằm xây dựng mô hình dự đoán hiệu quả.

Cụ thể, SEMMA gồm 5 bước chính như sau:

1. Sample (Lấy mẫu)

Mục tiêu: Chọn ra một tập dữ liệu đại diện từ nguồn dữ liệu ban đầu.

Ý nghĩa: Giúp giảm khối lượng dữ liệu cần xử lý, tiết kiệm thời gian mà vẫn đảm bảo tính đại diện.

Công việc chính:

Lấy mẫu (sampling), chia tập dữ liệu thành tập huấn luyện và kiểm tra.

Kiểm tra phân bố và chất lượng dữ liệu.

2. Explore (Khám phá dữ liệu)

Mục tiêu: Hiểu đặc trưng và mối quan hệ trong dữ liệu.

Công việc chính:

Phân tích thống kê mô tả (mean, median, variance...).

Phát hiện dữ liệu thiếu, ngoại lai.

Trực quan hóa dữ liệu (biểu đồ, scatter plot…).

Kết quả: Có cái nhìn tổng quan và xác định hướng xử lý, biến cần quan tâm.

3. Modify (Biến đổi dữ liệu)

Mục tiêu: Chuẩn bị dữ liệu cho mô hình hóa.

Công việc chính:

Làm sạch dữ liệu (loại bỏ lỗi, điền giá trị thiếu).

Biến đổi, chuẩn hóa, hoặc tạo biến mới (feature engineering).

Chọn biến quan trọng cho mô hình.

4. Model (Mô hình hóa)

Mục tiêu: Xây dựng và huấn luyện mô hình dự đoán hoặc phân loại.

Công việc chính:

Chọn thuật toán (ví dụ: hồi quy, cây quyết định, mạng nơ-ron, SVM…).

Huấn luyện mô hình bằng dữ liệu đã chuẩn bị.

Tinh chỉnh tham số để đạt hiệu quả cao.

5. Assess (Đánh giá mô hình)

Mục tiêu: Kiểm tra hiệu quả mô hình và mức độ phù hợp với mục tiêu ban đầu.

Công việc chính:

So sánh mô hình dự đoán với dữ liệu thực tế.

Dùng chỉ số đánh giá (Accuracy, ROC, RMSE…).

Lựa chọn mô hình tốt nhất để triển khai.

#### **+ Cây quyết định hoạt động như thế nào? Hãy giải thích các thành phần chính (nút gốc, nút lá, nhánh) và cách cây đưa ra dự đoán.**

Cây quyết định (Decision Tree) là một thuật toán học có giám sát, được dùng cho bài toán phân loại hoặc hồi quy. Nó hoạt động bằng cách chia dữ liệu thành các nhánh dựa trên điều kiện của thuộc tính, cho đến khi đạt được kết quả cuối cùng tại các nút lá.

Cây quyết định chia dữ liệu thành các nhóm nhỏ hơn dựa vào các điều kiện. Quá trình này được thực hiện lặp lại cho đến khi dữ liệu trong nhóm đủ thuần nhất (thuộc cùng một lớp), hoặc không còn thuộc tính nào có thể chia tiếp.

Các thành phần chính của cây quyết định bao gồm:

a. Nút gốc (Root Node):

* Là nút đầu tiên của cây, đại diện cho toàn bộ dữ liệu ban đầu.
* Tại đây, thuật toán chọn thuộc tính tốt nhất để chia dữ liệu (dựa trên độ đo như *Information Gain*, *Gini Index*...).

b. Nút trung gian (Internal Node):

* Là các nút con của nút gốc.
* Mỗi nút thể hiện một điều kiện kiểm tra trên dữ liệu,

c. Nhánh (Branch):

* Là đường nối giữa hai nút, thể hiện kết quả của điều kiện tại nút cha.
* Mỗi nhánh tương ứng với một giá trị cụ thể.

d. Nút lá (Leaf Node):

* Là nút cuối cùng của cây, không còn chia nhỏ nữa.
* Đại diện cho kết quả hoặc nhãn dự đoán cuối cùng.

3. Cách cây đưa ra dự đoán

1. Bắt đầu từ nút gốc.
2. Kiểm tra điều kiện tại nút .
3. Dựa vào kết quả, đi theo nhánh tương ứng.
4. Tiếp tục cho đến khi gặp nút lá.
5. Nhãn hoặc giá trị tại nút lá chính là kết quả dự đoán.

#### **+ Các tiêu chí phân tách (splitting criteria) như Gini Index, Entropy, hay Information Gain được sử dụng trong cây quyết định là gì? Chúng khác nhau ra sao?**

Trong cây quyết định, tiêu chí phân tách (splitting criteria) được sử dụng để chọn thuộc tính tốt nhất tại mỗi nút, nhằm chia dữ liệu thành các nhóm thuần nhất nhất có thể (tức là các mẫu trong cùng một nhóm thuộc cùng một lớp).  
Ba tiêu chí phổ biến nhất là Gini Index, Entropy, và Information Gain.

1. Gini Index

- Được dùng trong thuật toán CART (Classification and Regression Tree).

- Mục tiêu: Đo mức độ không thuần khiết của một nút.

Công thức:  
[Gini(t) = 1 - \sum\_{i=1}^{n} p\_i^2]

2. Entropy

- Được dùng trong thuật toán ID3, C4.5, C5.0.

- Mục tiêu: Đo mức độ hỗn loạn (uncertainty) hoặc độ rối thông tin trong một nút.

Công thức:  
[Entropy(t) = - \sum\_{i=1}^{n} p\_i \log\_2(p\_i)]

3. Information Gain

- Được dùng trong: ID3, C4.5.

- Mục tiêu: Đo mức giảm độ hỗn loạn (Entropy) sau khi chia dữ liệu theo một thuộc tính.

Công thức:  
[Information\ Gain = Entropy(parent) - \sum\_{i} \frac{N\_i}{N} \times Entropy(child\_i)]  
Trong đó:

( N\_i ): số lượng mẫu ở nhánh con thứ i.

( N ): tổng số mẫu ở nút cha.

4. So sánh Gini Index, Entropy và Information Gain

| Tiêu chí | Được dùng trong | Đo cái gì | Giá trị tốt nhất | Ghi chú |
| --- | --- | --- | --- | --- |
| Gini Index | CART | Mức độ không thuần khiết | Nhỏ nhất | Tính toán nhanh hơn Entropy |
| Entropy | ID3, C4.5 | Độ hỗn loạn dữ liệu | Nhỏ nhất | Cần log₂, chậm hơn một chút |
| Information Gain | ID3, C4.5 | Mức giảm Entropy sau khi chia | Lớn nhất | Dựa trên Entropy |

#### **+ Rừng cây (Random Forest) là gì? Nó khác gì so với một cây quyết định đơn lẻ? Tại sao Random Forest TRƯỜNG ĐẠI HỌC SÀI GÒN 2 thường có hiệu suất tốt hơn cây quyết định trong các bài toán phân loại?**

Rừng cây (Random Forest) là một mô hình học tập tổ hợp được tạo thành từ nhiều cây quyết định hoạt động song song. Mục tiêu của nó là kết hợp sức mạnh của nhiều mô hình yếu để tạo ra một mô hình mạnh hơn, chính xác và ổn định hơn.

Random Forest gồm hàng chục, hàng trăm, thậm chí hàng nghìn cây quyết định độc lập.  
Mỗi cây được huấn luyện trên một tập dữ liệu mẫu ngẫu nhiên được chọn bằng kỹ thuật bootstrap sampling - lấy mẫu có hoàn lại và một tập con ngẫu nhiên của các thuộc tính.

Khi dự đoán:

+ Đối với bài toán phân loại: Mỗi cây bỏ phiếu cho một lớp 🡪 lớp có nhiều phiếu nhất là kết quả cuối cùng.

+ Đối với bài toán hồi quy: Kết quả dự đoán là trung bình giá trị của các cây.

*Sự khác biệt giữa Random Forest và cây quyết định đơn lẻ*

| Đặc điểm | Cây quyết định | Rừng cây (Random Forest) |
| --- | --- | --- |
| Cấu trúc | Một cây duy nhất | Tập hợp nhiều cây (ensemble) |
| Nguồn dữ liệu | Sử dụng toàn bộ dữ liệu huấn luyện | Mỗi cây dùng mẫu dữ liệu và tập thuộc tính ngẫu nhiên |
| Độ phức tạp | Đơn giản, dễ hiểu, dễ trực quan hóa | Phức tạp hơn, khó giải thích trực tiếp |
| Độ chính xác | Dễ bị ảnh hưởng bởi nhiễu và dữ liệu huấn luyện | Ổn định, chính xác cao hơn nhờ trung bình hóa kết quả |
| Nguy cơ overfitting | Dễ bị overfit (quá khớp với dữ liệu huấn luyện) | Giảm overfitting nhờ tính ngẫu nhiên và trung bình hóa |

*Tại sao Random Forest thường có hiệu suất tốt hơn cây quyết định*

1. Giảm overfitting:

+ Một cây quyết định có thể học “quá kỹ” dữ liệu huấn luyện và không tổng quát tốt cho dữ liệu mới.

+ Random Forest khắc phục bằng cách kết hợp nhiều cây độc lập, mỗi cây chỉ thấy một phần dữ liệu, giúp giảm sai lệch do từng cây riêng lẻ.

1. Tăng độ chính xác:

+ Khi nhiều cây “bỏ phiếu”, sai sót ngẫu nhiên của từng cây sẽ được triệt tiêu.

+ Kết quả trung bình (hoặc đa số) thường ổn định và chính xác hơn.

1. Tận dụng tính ngẫu nhiên:

+ Việc chọn ngẫu nhiên tập dữ liệu và thuộc tính giúp cây trong rừng khác biệt nhau → tăng khả năng mô hình học được nhiều khía cạnh của dữ liệu.

1. Khả năng chống nhiễu tốt:

+ Nếu có dữ liệu sai lệch hoặc nhiễu, ảnh hưởng của nó bị pha loãng trong toàn bộ rừng.

#### **+ Những ưu điểm và hạn chế của cây quyết định và Random Forest là gì? Trong trường hợp nào thì cây quyết định có thể hoạt động kém hiệu quả?**

1. Đối với cây quyết định:

*Ưu điểm:*

Dễ hiểu và trực quan: Cấu trúc dạng cây giúp người dùng dễ hình dung logic ra quyết định. Có thể vẽ, trình bày, hoặc giải thích cho người không chuyên.

Không cần chuẩn hóa dữ liệu: Cây quyết định không yêu cầu dữ liệu phải được chuẩn hóa hoặc biến đổi trước (không cần scale dữ liệu).

Xử lý được cả biến định tính và định lượng: Cây có thể hoạt động tốt với dữ liệu dạng số lẫn dạng danh mục (categorical).

Tự động chọn thuộc tính quan trọng: Trong quá trình huấn luyện, cây sẽ chọn ra các thuộc tính có giá trị phân tách cao nhất.

Hoạt động tốt với dữ liệu phi tuyến tính: Vì cây chia dữ liệu theo điều kiện, nó không giả định mối quan hệ tuyến tính giữa các biến.

*Hạn chế:*

Dễ bị overfitting: Cây càng sâu (nhiều nhánh), càng dễ khớp hoàn hảo với dữ liệu huấn luyện nhưng kém tổng quát trên dữ liệu mới.

Nhạy cảm với thay đổi nhỏ trong dữ liệu: Chỉ cần thay đổi vài mẫu trong tập huấn luyện cũng có thể tạo ra cây hoàn toàn khác (thiếu ổn định).

Thiếu khả năng khái quát khi dữ liệu phức tạp: Với dữ liệu có nhiều nhiễu hoặc mối quan hệ phức tạp, một cây duy nhất khó nắm bắt hết các mẫu ẩn.

Không phù hợp với dữ liệu có số chiều lớn: Khi số thuộc tính tăng, số cách chia cũng tăng mạnh, dễ dẫn đến cây quá phức tạp và hiệu suất giảm.

Trường hợp cây quyết định hoạt động kém hiệu quả:

Khi dữ liệu có nhiều nhiễu hoặc thuộc tính không rõ ràng, cây dễ học sai mô hình.

Khi số lượng đặc trưng quá lớn so với số lượng mẫu 🡪 mô hình bị phân tách quá nhiều, dẫn đến overfitting.

Khi ràng buộc giữa các thuộc tính mang tính liên tục, phức tạp 🡪 cây khó mô tả chính xác mối quan hệ.

1. Đối với Random Forest:

Ưu điểm:

Giảm overfitting: Kết hợp nhiều cây với dữ liệu và thuộc tính ngẫu nhiên giúp triệt tiêu sai số của từng cây riêng lẻ.

Hiệu suất dự đoán cao: Nhờ cơ chế trung bình hóa (hoặc bỏ phiếu), mô hình ổn định và chính xác hơn hầu hết cây đơn.

Khả năng kháng nhiễu tốt: Các mẫu hoặc đặc trưng nhiễu chỉ ảnh hưởng đến một phần nhỏ các cây, không chi phối toàn bộ mô hình.

Đánh giá được tầm quan trọng của thuộc tính: Random Forest có thể tính “feature importance” – mức đóng góp của từng biến vào dự đoán.

Hoạt động tốt với dữ liệu lớn, nhiều chiều.

Hạn chế:

Khó giải thích: Vì gồm hàng trăm cây, mô hình khó hiểu và khó biểu diễn trực quan.

Tốn tài nguyên: Cần nhiều thời gian huấn luyện và bộ nhớ, đặc biệt khi có nhiều cây hoặc dữ liệu lớn.

Khó tối ưu: Có nhiều siêu tham số (số cây, độ sâu, số đặc trưng chọn ngẫu nhiên…) cần điều chỉnh để đạt hiệu quả tối đa.

#### **+ Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để xây dựng một mô hình cây quyết định không? Hãy mô tả các bước thực hiện.**

Bước 1: Import các thư viện cần thiết

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

Bước 2: Tải dữ liệu

data = load\_iris()

X = data.data

Y = data.target

Bước 3: Chia dữ liệu thành tập huấn luyện và kiểm tra

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.3, random\_state=42, stratify=y)

Bước 4: Khởi tạo mô hình cây quyết định

Tham số kernel: 'linear', 'poly', 'rbf', 'sigmoid' Tham số C: điều chỉnh soft margin svm\_model = SVC(kernel='rbf', C=1.0, gamma='scale', random\_state=42)

Bước 5: Huấn luyện mô hình

model.fit(X\_train, y\_train)

Bước 6: Dự đoán trên tập kiểm tra

y\_pred = model.predict(X\_test)

Bước 7: Đánh giá mô hình

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Độ chính xác của mô hình: {accuracy:.2f}")

Bước 8: Trực quan hóa cây quyết định

plt.figure(figsize=(10,6))

plot\_tree(model, feature\_names=iris.feature\_names, class\_names=iris.target\_names, filled=True)

plt.show()

#### **+ Làm thế nào để triển khai một mô hình Random Forest trong Python? Bạn thường thiết lập các tham số nào (ví dụ: n\_estimators, max\_depth)?**

Bước 1: Import thư viện

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import load\_iris

from sklearn.metrics import accuracy\_score, classification\_report

Bước 2: Tải dữ liệu mẫu (VD: Iris dataset)

data = load\_iris()

X = data.data

y = data.targe

Bước 3: Chia dữ liệu train/test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

Bước 4: Khởi tạo mô hình Random Forest

rf = RandomForestClassifier(

n\_estimators=100, số cây trong rừng

max\_depth=None, độ sâu tối đa của mỗi cây (None = cho đến khi phân loại hoàn toàn)

min\_samples\_split=2, số mẫu tối thiểu để chia nhánh

min\_samples\_leaf=1, số mẫu tối thiểu ở một lá

max\_features='sqrt', số đặc trưng được xét tại mỗi phép chia

bootstrap=True, có dùng chọn mẫu ngẫu nhiên (bootstrap) hay không

random\_state=42

Bước 5: Huấn luyện mô hình

rf.fit(X\_train, y\_train)

Bước 6: Dự đoán và đánh giá

y\_pred = rf.predict(X\_test)

print("Độ chính xác:", accuracy\_score(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

#### **+ Làm thế nào để đánh giá tầm quan trọng của các đặc trưng (feature importance) trong Random Forest bằng Python?**

Bước 1: Import thư viện

import pandas as pd

from sklearn.datasets import load\_iris

from sklearn.ensemble import RandomForestClassifier

Bước 2: Tải dữ liệu ví dụ

data = load\_iris()

X = pd.DataFrame(data.data, columns=data.feature\_names)

y = data.target

Bước 3: Huấn luyện mô hình Random Forest

rf = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf.fit(X, y)

Bước 4: Lấy giá trị tầm quan trọng

importances = rf.feature\_importances\_

feature\_importance\_df = pd.DataFrame({'Feature': X.columns, 'Importance': importances }).sort\_values(by='Importance', ascending=False)

print(feature\_importance\_df)

#### **+ Điều chỉnh siêu tham số (hyperparameter tuning) cho cây quyết định hoặc Random Forest chưa? Hãy mô tả cách bạn sử dụng GridSearchCV hoặc RandomizedSearchCV**

GridSearchCV

from sklearn.ensemble import RandomForestClassifier

from sklearn.datasets import load\_iris

Bước 1: Chuẩn bị dữ liệu

data = load\_iris()

X, y = data.data, data.target

Bước 2: Khởi tạo mô hình

rf = RandomForestClassifier(random\_state=42)

Bước 3: Xác định lưới tham số để tìm kiếm

param\_grid = {

'n\_estimators': [100, 200, 300],

'max\_depth': [None, 5, 10, 15],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

'max\_features': ['sqrt', 'log2']}

Bước 4: Dùng GridSearchCV để tìm bộ tham số tối ưu

grid\_search = GridSearchCV(

estimator=rf,

param\_grid=param\_grid,

cv=5, # 5-fold cross-validation

scoring='accuracy',

n\_jobs=-1, # dùng tất cả CPU cores

verbose=2)

grid\_search.fit(X, y)

Bước 5: Xem kết quả

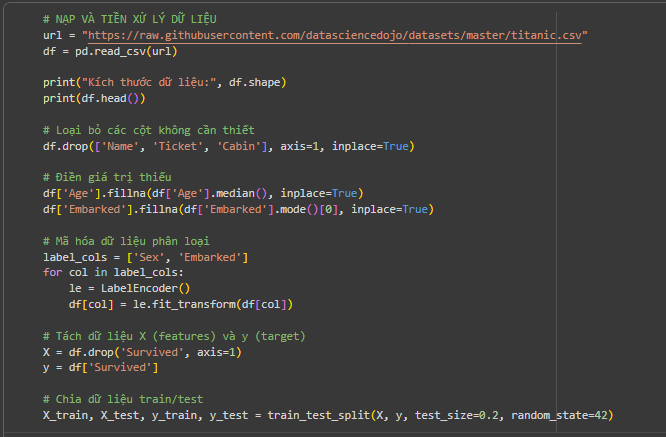
print("Best parameters found:", grid\_search.best\_params\_)

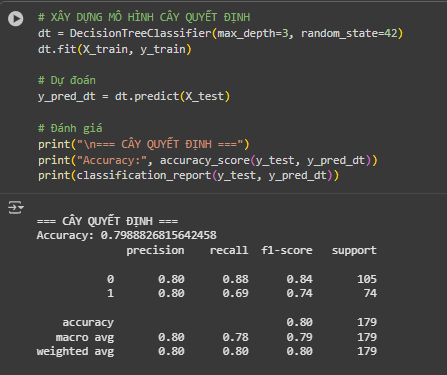
print("Best accuracy:", grid\_search.best\_score\_)

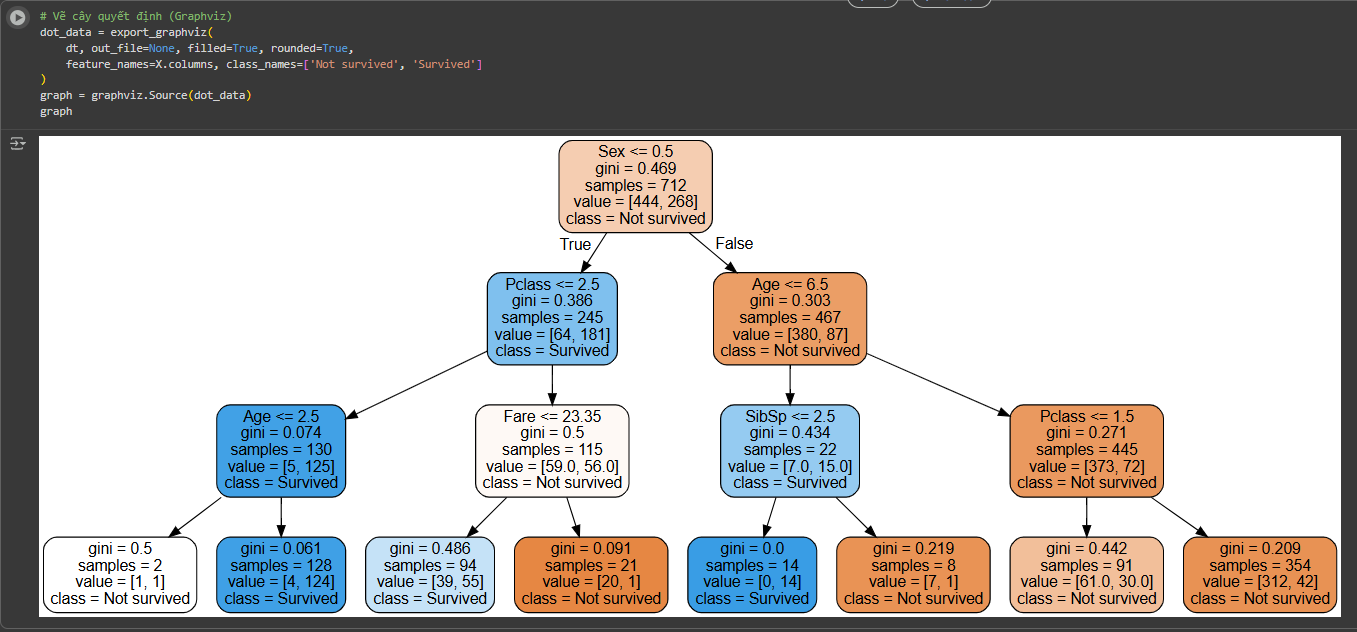
### 2.1.3. Bài tập thực hành 1

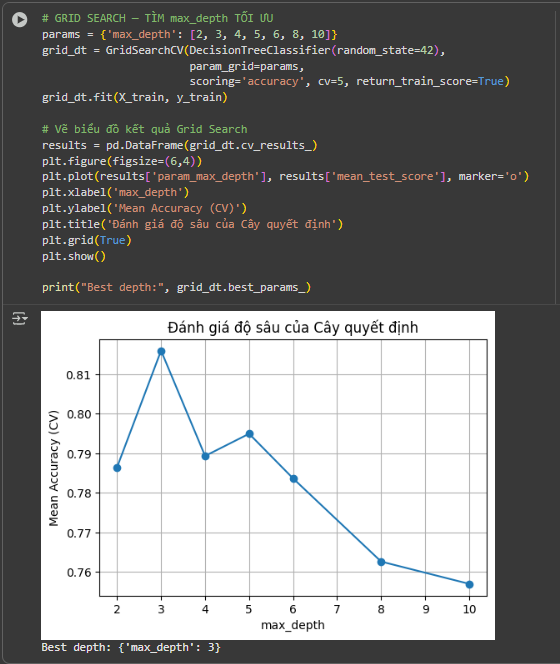
Xây dựng cây quyết định và rừng cây trên dữ liệu Titanic lấy từ <https://www.kaggle.com/code/dmilla/introduction-to-decision-trees-titanic-dataset>

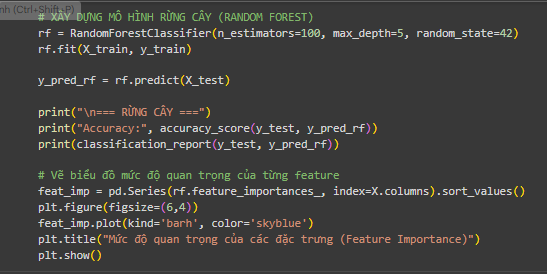


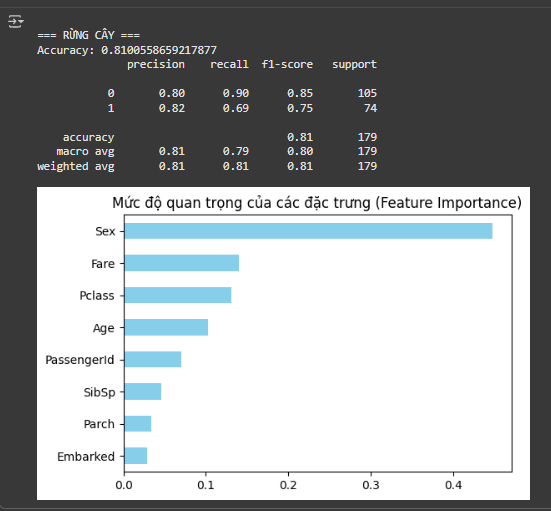


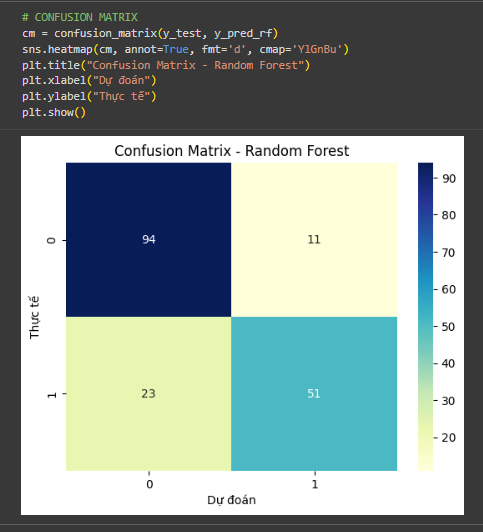






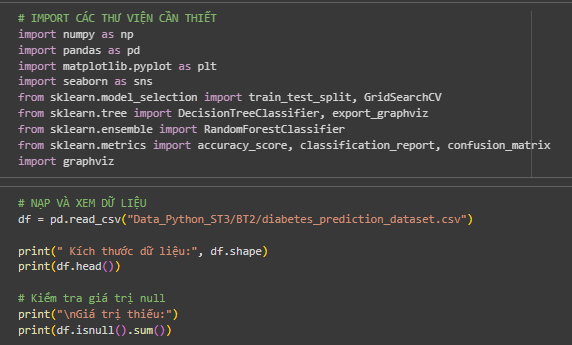


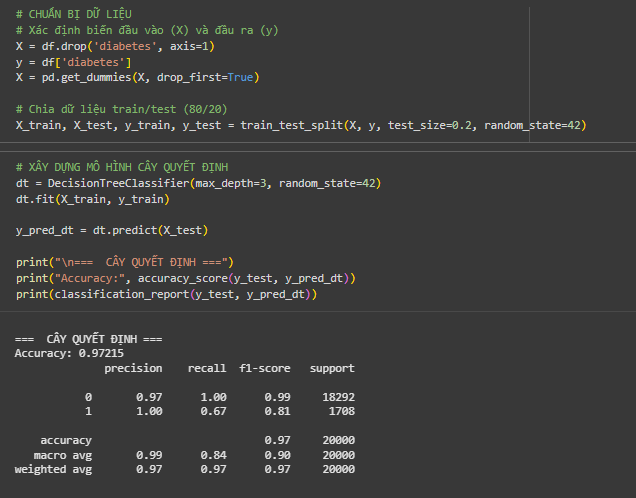


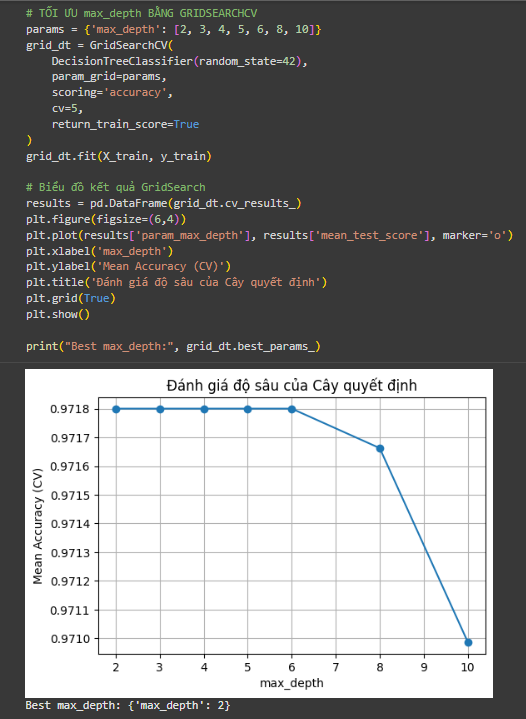
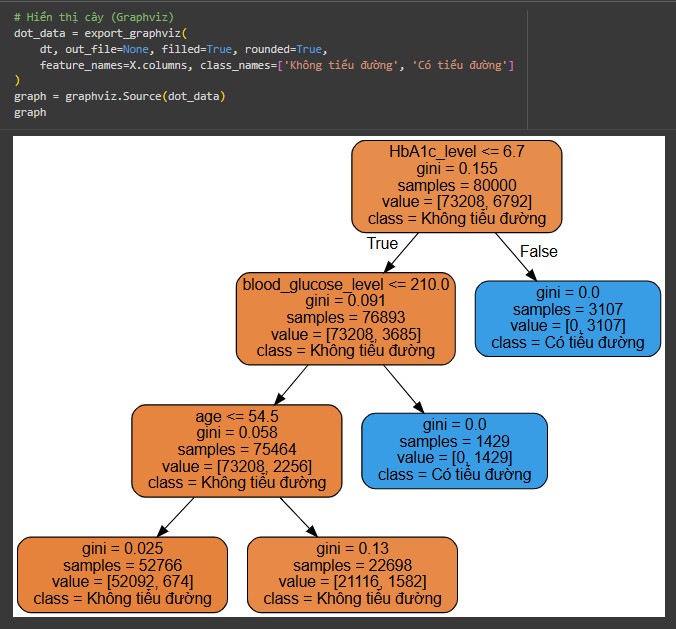


### 2.1.4. Bài tập thực hành 2:

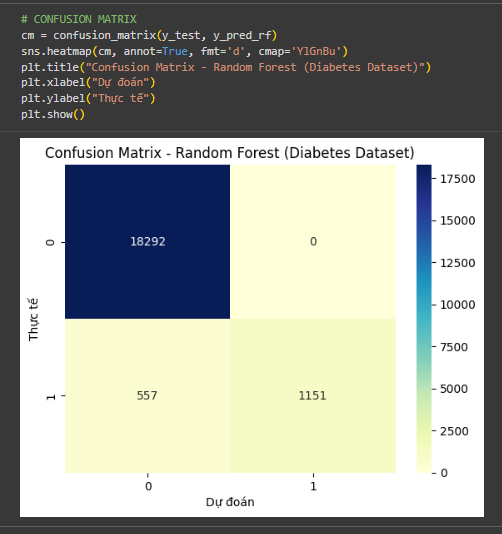
Xây dựng cây quyết định và rừng cây trên dữ liệu bệnh tiểu đường. Dữ liệu lấy từ <https://www.kaggle.com/code/tumpanjawat/diabetes-eda-random-forest-hp>











## **2.2: Giải thuật 2: SUPPORT VECTOR MACHINE (SVM)**

### 2.2.1: Lý thuyết

#### **+ Giải thuật Support Vector Machine hoạt động như thế nào? Hãy giải thích khái niệm về ranh giới phân tách (hyperplane) và lề (margin)**

1. Khái niệm chung về Support Vector Machine (SVM)

Support Vector Machine (SVM) là một thuật toán phân loại có giám sát (supervised learning). Mục tiêu của SVM là tìm ra một đường (hoặc mặt phẳng) phân tách các nhóm dữ liệu một cách tối ưu nhất.

* Với dữ liệu hai chiều, SVM tìm một đường thẳng.
* Với dữ liệu ba chiều, SVM tìm một mặt phẳng.
* Với dữ liệu n chiều, SVM tìm một siêu phẳng (hyperplane).

2. Nguyên lý hoạt động của SVM

Giả sử dữ liệu có hai lớp: lớp A và lớp B. SVM cố gắng tìm một đường phân tách sao cho:

1. Tất cả các điểm của lớp A nằm một bên của đường.
2. Tất cả các điểm của lớp B nằm bên còn lại.
3. Khoảng cách từ đường phân tách đến điểm gần nhất của mỗi lớp là lớn nhất có thể.

3. Ranh giới phân tách (Hyperplane)

Hyperplane là đường hoặc mặt phẳng dùng để phân chia các lớp dữ liệu. Trong không gian n chiều, hyperplane có phương trình:

w · x + b = 0

Trong đó:

* w: vector trọng số xác định hướng của hyperplane
* b: hằng số điều chỉnh vị trí (bias)
* x: vector đặc trưng (feature vector)

Một điểm x\_i được phân loại theo quy tắc:

* Nếu w · x\_i + b > 0 → lớp 1
* Nếu w · x\_i + b < 0 → lớp 2

4. Lề (Margin)

Margin là khoảng cách từ hyperplane đến các điểm gần nhất của hai lớp. SVM cố gắng tối đa hóa margin, tức là chọn đường phân tách sao cho nó cách xa các điểm biên nhất của hai lớp.

Các điểm nằm gần nhất với hyperplane gọi là Support Vectors. Đây là các điểm quan trọng nhất, quyết định vị trí và hướng của hyperplane

5. Mô hình toán học

SVM giải bài toán tối ưu hóa:

minimize (1/2) ||w||²

với điều kiện:

y\_i (w · x\_i + b) ≥ 1, với mọi i

* Mục tiêu: giảm độ lớn của w, tức là tăng margin
* Ràng buộc đảm bảo các điểm được phân loại đúng

6. Dữ liệu không tách tuyến tính

Khi dữ liệu không thể phân tách tuyến tính, SVM sử dụng:

1. Soft Margin SVM – Cho phép một số điểm phân loại sai nhẹ, điều chỉnh bằng tham số C.
2. Kernel Trick – Biến đổi dữ liệu sang không gian chiều cao hơn để tìm hyperplane tách được dữ liệu.

Các kernel phổ biến: linear, polynomial, RBF (Gaussian), sigmoid.

7. Các khái niệm chính

* Hyperplane: đường hoặc mặt phẳng phân tách dữ liệu
* Margin: khoảng cách giữa hyperplane và các điểm gần nhất
* Support Vectors: các điểm biên quan trọng quyết định hyperplane
* Mục tiêu của SVM: tìm hyperplane có margin lớn nhất để phân loại chính xác

8. Kết luận

Support Vector Machine là mô hình mạnh, đặc biệt khi dữ liệu phân tách rõ ràng hoặc có thể biến đổi qua kernel. SVM hoạt động dựa trên việc tối đa hóa khoảng cách (margin) giữa các lớp dữ liệu. Hyperplane là ranh giới phân tách, margin là vùng đệm xung quanh hyperplane, và các support vectors là điểm quan trọng quyết định mô hình.

#### **+ Các vector hỗ trợ (support vectors) có vai trò gì trong SVM? Tại sao chúng quan trọng?**

Support Vectors là những điểm dữ liệu nằm gần nhất với siêu phẳng phân tách (hyperplane) trong SVM. Chúng nằm trên hoặc rất gần ranh giới lề (margin), do đó quyết định vị trí và hướng của hyperplane.

Các support vectors là những điểm quan trọng nhất trong tập dữ liệu, vì hyperplane được xác định hoàn toàn dựa trên chúng.

Nếu các điểm dữ liệu khác bị loại bỏ, hyperplane vẫn không thay đổi miễn là các support vectors vẫn giữ nguyên.

Chúng giúp tối đa hóa margin giữa hai lớp, làm cho mô hình phân loại ổn định và khả năng tổng quát hóa tốt hơn.

SVM chỉ phụ thuộc vào support vectors để xác định siêu phẳng phân tách. Các điểm dữ liệu nằm xa margin không ảnh hưởng đến kết quả.

Nhờ vậy, SVM có khả năng chống nhiễu tốt, vì những điểm không quan trọng (không phải support vectors) sẽ không ảnh hưởng đến mô hình.

Số lượng support vectors cũng phản ánh độ phức tạp của mô hình: nhiều support vectors 🡪 mô hình phức tạp hơn, ít support vectors 🡪 mô hình đơn giản hơn.

#### **+ Sự khác biệt giữa SVM với lề cứng (hard margin) và lề mềm (soft margin) là gì? Khi nào nên sử dụng lề mềm?**

1. SVM lề cứng (Hard Margin SVM)

Khái niệm: Lề cứng yêu cầu rằng tất cả các điểm dữ liệu phải được phân loại đúng, không được phép có bất kỳ điểm nào nằm trong hoặc vi phạm margin.

Đặc điểm:

Hyperplane phân tách dữ liệu hoàn hảo.

Không cho phép lỗi phân loại.

Phù hợp với dữ liệu tách tuyến tính hoàn hảo, không có nhiễu.

Hạn chế:

Nếu dữ liệu có nhiễu hoặc các điểm ngoại lai (outliers), hard margin có thể không tìm được hyperplane hoặc dễ bị overfitting.

1. SVM lề mềm (Soft Margin SVM)

Khái niệm: Lề mềm cho phép một số điểm vi phạm margin hoặc bị phân loại sai với một mức phạt.

Đặc điểm:

Giới thiệu tham số C để điều chỉnh mức độ chấp nhận lỗi:

C lớn 🡪 phạt nặng, ít lỗi, gần giống hard margin.

C nhỏ 🡪 cho phép nhiều lỗi hơn, margin rộng hơn.

Tăng khả năng tổng quát hóa, giảm overfitting.

Khi nào nên sử dụng:

Dữ liệu không tách tuyến tính hoàn hảo.

Dữ liệu có nhiễu hoặc outliers.

Muốn SVM cân bằng giữa độ chính xác trên tập huấn luyện và khả năng tổng quát hóa.

#### **+ Hàm nhân (kernel) trong SVM là gì? Hãy giải thích các loại kernel phổ biến (linear, polynomial, RBF) và khi nào nên sử dụng chúng**

Hàm nhân (Kernel) trong SVM

1. Khái niệm chung

Hàm nhân (kernel) trong SVM là một kỹ thuật giúp biến đổi dữ liệu từ không gian gốc sang một không gian chiều cao hơn để tìm được hyperplane phân tách dữ liệu, ngay cả khi dữ liệu không thể phân tách tuyến tính trong không gian ban đầu.  
Nói cách khác, kernel cho phép SVM xây dựng các ranh giới phi tuyến (non-linear decision boundary) mà không cần tính trực tiếp tọa độ trong không gian chiều cao hơn.

Khi sử dụng kernel, phương trình hyperplane trong không gian gốc được thay bằng một hàm nhân K(x\_i, x\_j), đo độ tương đồng giữa hai điểm x\_i và x\_j.

1. Các loại kernel phổ biến

Kernel tuyến tính (Linear Kernel)

Công thức: K(x\_i, x\_j) = x\_i · x\_j

Dùng khi dữ liệu có thể phân tách tuyến tính.

Ưu điểm: nhanh, ít tốn tài nguyên, dễ giải thích.

Kernel đa thức (Polynomial Kernel)

Công thức: K(x\_i, x\_j) = (γ x\_i · x\_j + r)^d, trong đó γ, r, d là tham số.

Dùng khi dữ liệu có quan hệ phi tuyến, đặc biệt khi mô hình cần tạo đường cong phức tạp.

Thích hợp với dữ liệu có tương tác giữa các đặc trưng.

Kernel RBF (Radial Basis Function / Gaussian Kernel)

Công thức: K(x\_i, x\_j) = exp(-γ ||x\_i - x\_j||²), γ > 0

Dùng khi dữ liệu không tuyến tính phức tạp và không biết dạng đường phân tách trước.

RBF có khả năng tạo biên phân loại mềm, linh hoạt, rất phổ biến trong thực tế.

Kernel sigmoid (ít phổ biến hơn)

Công thức: K(x\_i, x\_j) = tanh(γ x\_i · x\_j + r)

Tương tự như một mạng neuron đơn giản.

1. Khi nào nên sử dụng từng Kernel

| Kernel | Khi sử dụng | Ưu điểm |
| --- | --- | --- |
| Linear | Dữ liệu tách tuyến tính hoặc số lượng đặc trưng rất lớn | Nhanh, dễ giải thích, ít nguy cơ overfitting |
| Polynomial | Dữ liệu có quan hệ phi tuyến, cần tạo đường cong phức tạp | Tạo quyết định phi tuyến, linh hoạt hơn linear |
| RBF | Dữ liệu phi tuyến, quan hệ phức tạp, không biết dạng đường phân tách | Linh hoạt, phổ biến, hiệu quả với đa số bài toán thực tế |

#### **+ Tham số C trong SVM có ý nghĩa gì? Nó ảnh hưởng như thế nào đến hiệu suất của mô hình?**

Tham số C trong SVM

1. Khái niệm

Tham số C trong SVM là một tham số điều chỉnh lề mềm (Soft Margin). Nó xác định mức độ chấp nhận lỗi phân loại trong quá trình tìm hyperplane tối ưu.

Giá trị C càng lớn 🡪 mô hình phạt nặng các điểm phân loại sai.

Giá trị C nhỏ 🡪 mô hình cho phép một số điểm phân loại sai, ưu tiên margin rộng.

1. Ảnh hưởng đến hiệu suất mô hình

C lớn (C 🡪 ∞)

Hyperplane cố gắng phân loại chính xác tất cả các điểm.

Margin thường hẹp hơn, có nguy cơ overfitting, đặc biệt khi dữ liệu có nhiễu hoặc outliers.

C nhỏ

Cho phép một số điểm bị phân loại sai, hyperplane có thể cách đều các lớp hơn.

Margin rộng hơn, mô hình tăng khả năng tổng quát hóa, ít bị overfitting.

Tuy nhiên, nếu quá nhỏ, mô hình có thể underfitting, bỏ qua nhiều điểm quan trọng.

#### **+ Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để xây dựng một mô hình SVM cho bài toán phân loại không? Hãy mô tả các bước thực hiện**

Bước 1: Import thư viện

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

Bước 2: Chuẩn bị dữ liệu

data = load\_iris()

X = data.data Các đặc trưng

y = data.target Nhãn lớp

Bước 3: Chia dữ liệu thành tập huấn luyện và tập kiểm tra

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.3, random\_state=42, stratify=y)

Bước 4: Khởi tạo mô hình SVM

Tham số kernel: 'linear', 'poly', 'rbf', 'sigmoid'

Tham số C: điều chỉnh soft margin

svm\_model = SVC(kernel='rbf', C=1.0, gamma='scale', random\_state=42)

Bước 5: Huấn luyện mô hình

svm\_model.fit(X\_train, y\_train)

Bước 6: Dự đoán và đánh giá

y\_pred = svm\_model.predict(X\_test)

Độ chính xác tổng thể

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

Báo cáo chi tiết từng lớp

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

Ma trận nhầm lẫn

print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

*Mô tả các bước*

1. Import thư viện: Sử dụng scikit-learn cho mô hình SVM, chia dữ liệu, và đánh giá.
2. Chuẩn bị dữ liệu: Ở đây dùng dataset Iris làm ví dụ, gồm các đặc trưng và nhãn lớp.
3. Chia dữ liệu: train\_test\_split tách 70% cho huấn luyện và 30% cho kiểm tra, giữ tỷ lệ lớp bằng stratify.
4. Khởi tạo SVM:

kernel='rbf': sử dụng kernel Gaussian, phù hợp với dữ liệu phi tuyến.

C=1.0: điều chỉnh soft margin.

gamma='scale': tham số kernel RBF, mặc định là phù hợp với hầu hết dữ liệu.

1. Huấn luyện: Dùng .fit() với tập huấn luyện.
2. Dự đoán và đánh giá: Dùng .predict() trên tập kiểm tra, sau đó tính accuracy, classification report, confusion matrix.

#### **+ Hàm nào trong Scikit-learn để chuẩn hóa dữ liệu (scaling) trước khi áp dụng SVM? Tại sao bước này quan trọng?**

Trong Scikit-learn, có một số cách phổ biến để chuẩn hóa dữ liệu:

**StandardScaler**

Công thức: z=(x−μ)/z

Biến đổi dữ liệu sao cho trung bình bằng 0 và độ lệch chuẩn bằng 1.

Dùng cho SVM rất phổ biến.

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

**MinMaxScaler**

Công thức: x′=(x−xmin)/(xmax−xminx')

Biến đổi dữ liệu về khoảng [0, 1].

Thường dùng khi dữ liệu có giới hạn cụ thể.

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X\_scaled = scaler.fit\_transform(X)

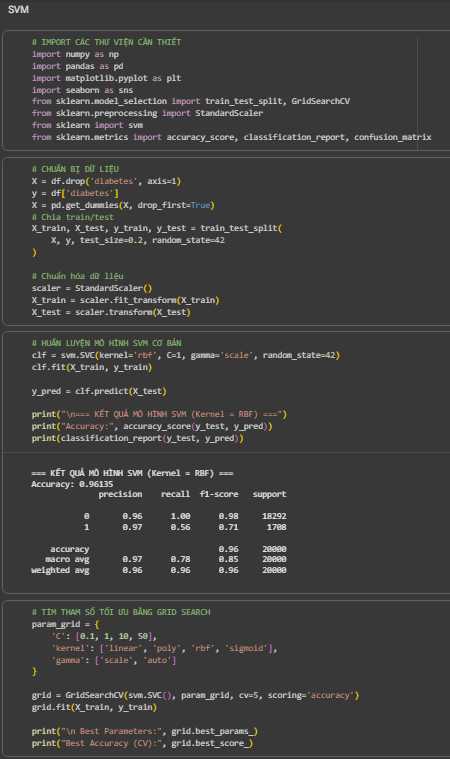
**MaxAbsScaler**

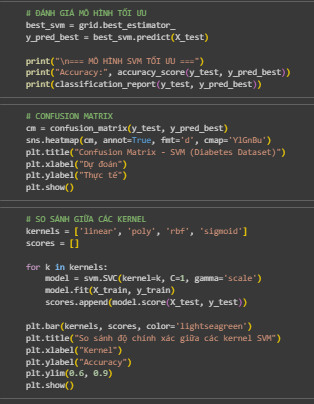
Biến đổi dữ liệu sao cho giá trị tuyệt đối lớn nhất bằng 1.

Thường dùng cho dữ liệu sparse (thưa).

### 2.2.3. Bài tập thực hành 1

Xây dựng mô hình từ giải thuật SVM trên dữ liệu bệnh tiểu đường. Dữ liệu lấy từ <https://www.kaggle.com/code/tumpanjawat/diabetes-eda-random-forest-hp>



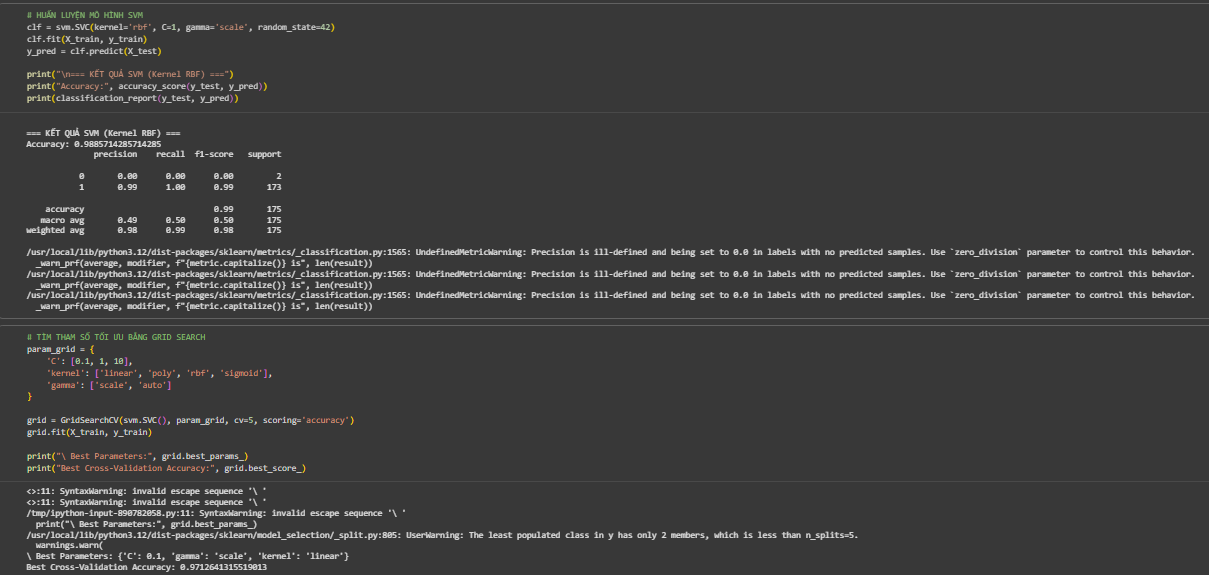


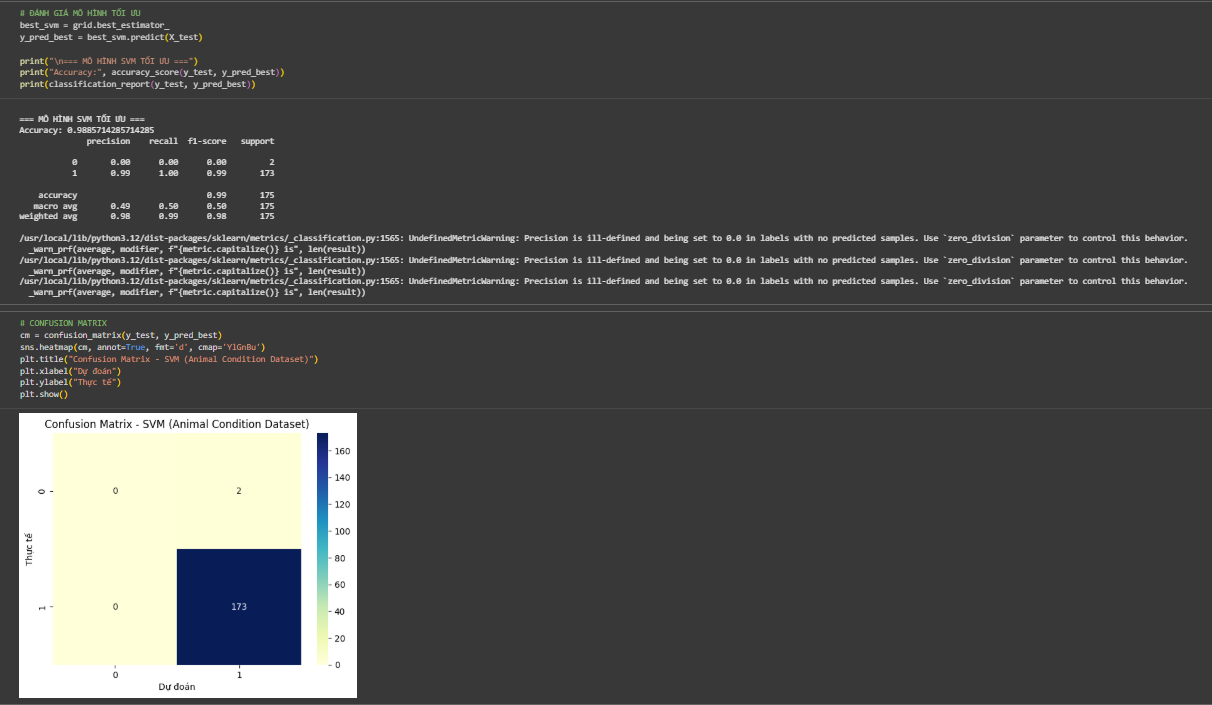
### 2.2.4. Bài tập thực hành 2

Xây dựng mô hình từ giải thuật SVM trên dữ liệu các con thú trong rừng. Dữ liệu lấy từ <https://www.kaggle.com/code/kareemellithy/animal-condition-predict-svm-knn>











## **2.3: Giải thuật 3: BAYES NGÂY THƠ (NAÏVE BAYES)**

### 2.3.1. Ôn tập lý thuyết

#### **+ Giải thuật Naive Bayes hoạt động như thế nào? Hãy giải thích định lý Bayes và giả định "ngây thơ" trong thuật toán này?**

1. Khái niệm chung

Naive Bayes là một thuật toán phân loại có giám sát dựa trên định lý Bayes, thường dùng cho dữ liệu phân loại văn bản, spam email, phân loại bệnh lý, v.v. Thuật toán này gọi là “Naive” (ngây thơ) vì nó giả định các đặc trưng là độc lập có điều kiện với nhau, mặc dù trong thực tế chúng có thể liên quan.

1. Định lý Bayes  
   Định lý Bayes mô tả mối quan hệ giữa xác suất có điều kiện của hai sự kiện. Công thức:

P(C∣X)=[P(X∣C)⋅P(C)]/P(X)

Trong đó:

P(C∣X): xác suất của lớp C khi biết đặc trưng X (posterior probability).

P(X∣C): xác suất quan sát X thuộc lớp C (likelihood).

P(C): xác suất tiên nghiệm của lớp C (prior probability).

P(X): xác suất quan sát X xảy ra (evidence).

Ý tưởng:  
Naive Bayes chọn lớp C sao cho P(C∣X) lớn nhất.

1. Giả định "ngây thơ" (Naive assumption)  
   Naive Bayes giả định rằng tất cả các đặc trưng x1,x2,...,xnx\_1, x\_2, ..., x\_nx1​,x2​,...,xn​ là độc lập có điều kiện với nhau khi biết lớp C.

Giả định này giúp tính toán đơn giản và nhanh chóng, vì không cần ước lượng xác suất phối hợp của tất cả các đặc trưng.

Mặc dù giả định này hiếm khi đúng hoàn toàn trong thực tế, Naive Bayes vẫn hoạt động hiệu quả trong nhiều bài toán phân loại.

1. Nguyên lý hoạt động của Naive Bayes

Bước cơ bản:

1. Tính xác suất tiên nghiệm P(C)P(C)P(C) cho mỗi lớp dựa trên dữ liệu huấn luyện.
2. Tính xác suất có điều kiện P(xi∣C)P(x\_i | C)P(xi​∣C) cho từng đặc trưng và từng lớp.
3. Khi dự đoán, tính posterior probability cho mỗi lớp:
4. Chọn lớp có xác suất cao nhất.

#### **+ Các loại mô hình Naive Bayes (Gaussian, Multinomial, Bernoulli) khác nhau ra sao? Khi nào nên sử dụng từng loại?**

1. Gaussian Naive Bayes

Đặc điểm:

Giả định rằng các đặc trưng liên tục tuân theo phân phối Gaussian (phân phối chuẩn).

Tính xác suất P(xi∣C)P(x\_i|C)P(xi​∣C) dựa trên mean và variance của đặc trưng trong lớp C.

Khi sử dụng:

Dữ liệu đặc trưng liên tục, ví dụ: chiều cao, cân nặng, điểm số.

Dataset Iris là ví dụ điển hình.

1. Multinomial Naive Bayes

Đặc điểm:

Dùng cho dữ liệu đếm hoặc tần suất, ví dụ: số lần từ xuất hiện trong văn bản.

Mỗi đặc trưng thể hiện số lần hoặc tần suất xuất hiện của một sự kiện.

Khi sử dụng:

Phân loại văn bản (spam email, sentiment analysis).

Dữ liệu dạng rời rạc, đếm được, không liên tục.

1. Bernoulli Naive Bayes

Đặc điểm:

Dữ liệu đặc trưng là nhị phân (0/1), chỉ quan tâm sự xuất hiện hay không xuất hiện của một đặc trưng.

Khi sử dụng:

Văn bản nhị phân (presence/absence) thay vì tần suất.

Dữ liệu dạng boolean.

#### **+ Tại sao Naive Bayes được gọi là "ngây thơ"? Giả định về tính độc lập của các đặc trưng ảnh hưởng như thế nào đến hiệu suất của mô hình?**

1. Tại sao gọi là “ngây thơ”

Naive Bayes được gọi là ngây thơ (naive) vì thuật toán giả định rằng tất cả các đặc trưng (features) là độc lập có điều kiện với nhau khi biết lớp (class).

Nói cách khác, mô hình coi mỗi đặc trưng không bị ảnh hưởng bởi các đặc trưng khác trong cùng một lớp, mặc dù trong thực tế điều này hiếm khi đúng.

Giả định này giúp tính toán xác suất đơn giản và nhanh chóng, nhưng bỏ qua mối quan hệ giữa các đặc trưng.

1. Ảnh hưởng của giả định độc lập đến hiệu suất mô hình

Ưu điểm:

Giúp mô hình tính toán nhanh, ít tốn tài nguyên, đặc biệt với dữ liệu có nhiều đặc trưng.

Trong nhiều bài toán, ngay cả khi giả định không hoàn toàn đúng, Naive Bayes vẫn cho kết quả dự đoán chính xác và khả năng tổng quát hóa tốt.

Hạn chế:

Nếu các đặc trưng có mối quan hệ mạnh, giả định độc lập có thể làm sai lệch xác suất ước lượng, dẫn đến hiệu suất kém hơn so với các mô hình không giả định độc lập.

#### **+ Ưu điểm và hạn chế của Naive Bayes so với các thuật toán phân loại khác như SVM hoặc Random Forest là gì?**

Ưu điểm và hạn chế của Naive Bayes so với các thuật toán phân loại khác

1. Ưu điểm của Naive Bayes

Đơn giản và nhanh chóng: Dễ triển khai, tính toán nhanh, đặc biệt với dữ liệu lớn và nhiều đặc trưng.

Hiệu quả với dữ liệu nhỏ: Không cần nhiều dữ liệu để ước lượng xác suất, vẫn cho kết quả tốt.

Hoạt động tốt với dữ liệu văn bản: Rất phù hợp với các bài toán phân loại văn bản, spam detection, sentiment analysis.

Ít bị overfitting: Vì mô hình đơn giản, ít phụ thuộc vào các tham số phức tạp.

1. Hạn chế của Naive Bayes

Giả định độc lập “ngây thơ”: Khi các đặc trưng có mối quan hệ mạnh, xác suất ước lượng có thể không chính xác, làm giảm hiệu suất.

Không học được tương tác phức tạp: Không thể nắm bắt mối quan hệ phi tuyến giữa các đặc trưng như SVM hoặc Random Forest.

Hiệu suất thấp với dữ liệu không phù hợp.

1. So sánh với SVM và Random Forest

| Tiêu chí | Naive Bayes | SVM | Random Forest |
| --- | --- | --- | --- |
| Độ phức tạp | Thấp, nhanh | Trung bình – cao, tùy kernel | Cao, nhiều cây |
| Khả năng xử lý dữ liệu lớn | Rất tốt | Tốt, nhưng kernel phức tạp có thể chậm | Tốt, nhưng training nhiều cây tốn thời gian |
| Khả năng học phi tuyến | Giới hạn | Tốt với kernel phi tuyến | Rất tốt, tự động học tương tác phức tạp |
| Dữ liệu nhiễu/outliers | Nhạy với giả định độc lập | Nhạy, nhưng kernel RBF linh hoạt | Khá bền, robust nhờ trung bình nhiều cây |
| Ứng dụng phổ biến | Văn bản, spam, sentiment | Hầu hết bài toán phân loại, đặc biệt phi tuyến | Bài toán tổng quát, dữ liệu phức tạp, phi tuyến |

#### **+ Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để xây dựng một mô hình Naive Bayes (ví dụ: Gaussian Naive Bayes) không? Hãy mô tả các bước thực hiện**

Bước 1: Import thư viện

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

Bước 2: Chuẩn bị dữ liệu

data = load\_iris()

X = data.data Các đặc trưng

y = data.target Nhãn lớp

Bước 3: Chia dữ liệu thành tập huấn luyện và tập kiểm tra

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.3, random\_state=42, stratify=y)

Bước 4: Khởi tạo mô hình Gaussian Naive Bayes

gnb = GaussianNB()

Bước 5: Huấn luyện mô hình

gnb.fit(X\_train, y\_train)

Bước 6: Dự đoán và đánh giá

y\_pred = gnb.predict(X\_test)

Độ chính xác tổng thể

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

Báo cáo chi tiết từng lớp

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

Ma trận nhầm lẫn

print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

Mô tả các bước

1. Import thư viện: Sử dụng scikit-learn cho Gaussian Naive Bayes, chia dữ liệu, và đánh giá hiệu suất.
2. Chuẩn bị dữ liệu: Ở đây dùng dataset Iris làm ví dụ, gồm các đặc trưng và nhãn lớp.
3. Chia dữ liệu: train\_test\_split tách 70% cho huấn luyện và 30% cho kiểm tra, giữ tỷ lệ lớp bằng stratify.
4. Khởi tạo GaussianNB: Gaussian Naive Bayes phù hợp với dữ liệu liên tục và giả định các đặc trưng tuân theo phân phối chuẩn.
5. Huấn luyện: Dùng .fit() với tập huấn luyện.
6. Dự đoán và đánh giá: Dùng .predict() trên tập kiểm tra, sau đó tính accuracy, classification report, và confusion matrix.

#### **+ Làm thế nào để xử lý dữ liệu phân loại (categorical data) trước khi áp dụng Multinomial Naive Bayes trong Python?**

1. Vấn đề

Multinomial Naive Bayes chỉ chấp nhận dữ liệu dạng số (numeric), đại diện cho tần suất hoặc số lượng của các đặc trưng.

Các đặc trưng phân loại (categorical) như “red”, “blue”, “small”, “medium”, “large” không thể sử dụng trực tiếp.

1. Các bước xử lý dữ liệu phân loại

Bước 1: Chuyển categorical sang numeric

Sử dụng One-Hot Encoding: mỗi giá trị của đặc trưng phân loại được biến thành một cột nhị phân (0/1).

Sử dụng pandas.get\_dummies() hoặc sklearn.preprocessing.OneHotEncoder.

import pandas as pd

from sklearn.preprocessing import OneHotEncoder

Giả sử df là DataFrame có cột phân loại 'color'

df = pd.DataFrame({'color': ['red', 'blue', 'green', 'red', 'blue']})

One-Hot Encoding với pandas

df\_encoded = pd.get\_dummies(df, columns=['color'])

print(df\_encoded)

Bước 2: Chuẩn bị dữ liệu đầu vào cho MultinomialNB

MultinomialNB yêu cầu dữ liệu là integer ≥ 0, đại diện tần suất hoặc presence.

Nếu đặc trưng biểu thị tần suất, có thể giữ nguyên giá trị.

Nếu đặc trưng chỉ là sự xuất hiện (presence), One-Hot Encoding tạo ra giá trị 0/1 là phù hợp.

from sklearn.naive\_bayes import MultinomialNB

from sklearn.model\_selection import train\_test\_split

Giả sử X là df\_encoded và y là nhãn lớp

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df\_encoded, y, test\_size=0.3, random\_state=42)

model = MultinomialNB()

model.fit(X\_train, y\_train)

#### **+ Naive Bayes thường được sử dụng trong phân loại văn bản (text classification). Bạn có thể giải thích cách triển khai Naive Bayes cho bài toán này không?**

Triển khai Naive Bayes cho bài toán phân loại văn bản

1. Khái niệm chung

Bài toán phân loại văn bản thường là dự đoán nhãn lớp của văn bản dựa trên nội dung, ví dụ: spam/ham email, sentiment analysis (tích cực/tiêu cực).

Naive Bayes rất phù hợp vì dữ liệu văn bản có rất nhiều đặc trưng (từ ngữ) và mô hình có thể xử lý tốt dữ liệu sparse.

1. Các bước triển khai

Bước 1: Chuẩn bị dữ liệu

Tập dữ liệu gồm văn bản và nhãn lớp.

Ví dụ: DataFrame với 2 cột: text (nội dung) và label (nhãn).

Bước 2: Tiền xử lý văn bản (text preprocessing)

Loại bỏ ký tự đặc biệt, số, dấu câu.

Chuyển về chữ thường.

Tách từ (tokenization).

(Tuỳ chọn) Loại stopwords, stemming hoặc lemmatization.

Bước 3: Biểu diễn văn bản dưới dạng numeric

Sử dụng Bag-of-Words (CountVectorizer) hoặc TF-IDF (TfidfVectorizer) để chuyển văn bản thành ma trận tần suất từ ngữ.

Kết quả là dữ liệu dạng sparse matrix, phù hợp với Multinomial Naive Bayes.

Bước 4: Chia dữ liệu thành tập huấn luyện và tập kiểm tra

Sử dụng train\_test\_split để đánh giá mô hình trên dữ liệu chưa thấy trước đó.

Bước 5: Khởi tạo và huấn luyện mô hình Naive Bayes

Dùng MultinomialNB() cho văn bản, vì đặc trưng là tần suất từ.

Huấn luyện bằng .fit(X\_train, y\_train).

Bước 6: Dự đoán và đánh giá

Dự đoán nhãn văn bản mới bằng .predict(X\_test).

Đánh giá mô hình bằng accuracy, classification report, confusion matrix.

1. Ví dụ code Python minh họa

Import thư viện

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score, classification\_report

Giả sử df có 2 cột: 'text' và 'label'

df = pd.DataFrame({

'text': ["I love this product", "This is spam message", "Happy with the service", "Buy now spam offer"],

'label': ["positive", "spam", "positive", "spam"]

})

Chia dữ liệu

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df['text'], df['label'], test\_size=0.3, random\_state=42)

Biểu diễn văn bản thành tần suất từ

vectorizer = CountVectorizer()

X\_train\_vec = vectorizer.fit\_transform(X\_train)

X\_test\_vec = vectorizer.transform(X\_test)

Khởi tạo và huấn luyện mô hình Multinomial Naive Bayes

model = MultinomialNB()

model.fit(X\_train\_vec, y\_train)

Dự đoán và đánh giá

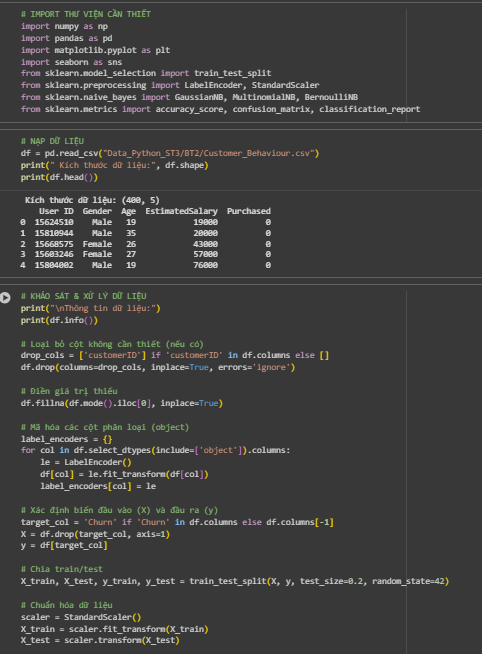
y\_pred = model.predict(X\_test\_vec)

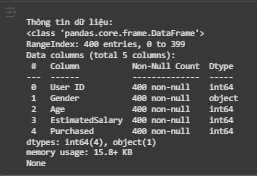
print("Accuracy:", accuracy\_score(y\_test, y\_pred))

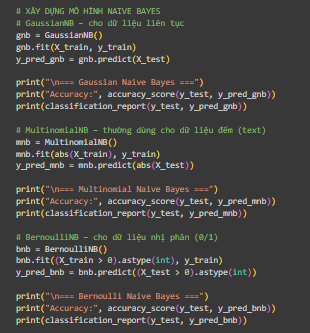
print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

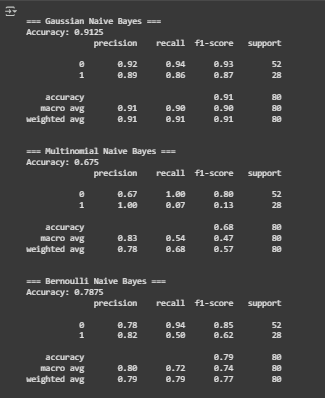
### 2.3.1. Bài tập thực hành 1

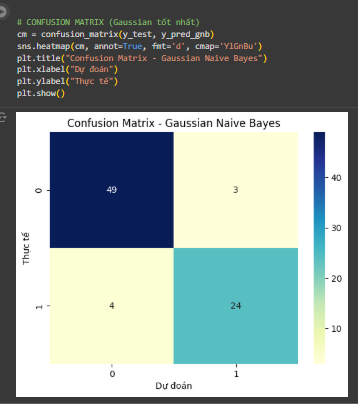
Xây dựng mô hình Naïve ngây thơ trên tập dữ liệu hành vi của khách hàng lấy tại <https://www.kaggle.com/code/arezalo/customer-behaviour-prediction-naive-bayes>













# **LAB03**

## **3.1. Giải thuật K-Means**

### **3.1.1. Ôn tập lý thuyết**

#### **Giải thuật K-Means hoạt động như thế nào?**

K-Means là một thuật toán phân cụm không giám sát (unsupervised learning), dùng để chia dữ liệu thành k nhóm (clusters) sao cho các điểm trong cùng cụm thì giống nhau, còn khác cụm thì khác nhau.

Các bước chính:

* Bước 1: Chọn số cụm K (do người dùng xác định).
* Bước 2: Khởi tạo K tâm cụm ban đầu (centroids): thường được chọn ngẫu nhiên hoặc bằng K-Means++.
* Bước 3: Gán nhãn cho từng điểm dữ liệu vào cụm có tâm gần nhất (dựa trên khoảng cách Euclidean).
* Bước 4: Cập nhật lại tâm cụm mới = trung bình các điểm trong cụm.
* Bước 5: Lặp lại bước 3 và 4 cho đến khi các tâm cụm không còn thay đổi đáng kể (hội tụ).

#### **Tại sao cần chọn số lượng cụm K trước khi chạy? Làm sao xác định K tối ưu?**

Thuật toán K-Means cần biết bao nhiêu nhóm để bắt đầu gom cụm. Nếu không có K, nó không biết phải chia dữ liệu ra làm mấy phần.

Cách xác định k tối ưu:

* Phương pháp Elbow:Tính tổng bình phương khoảng cách trong cụm (SSE – Sum of Squared Errors) cho nhiều giá trị K khác nhau.  
  Khi vẽ đồ thị SSE theo K, điểm gấp khúc (elbow) là K tối ưu.
* Phương pháp Silhouette Score:Đo mức độ chặt chẽ trong từng cụm và cách biệt giữa các cụm.  
  Giá trị Silhouette càng cao → phân cụm càng tốt.

#### **Hàm mục tiêu (Objective Function) của K-Means là gì? Nó đo lường cái gì?**

Hàm mục tiêu của K-Means có dạng như sau:

Trong đó:

* : tập các điểm thuộc cụm k
* : tâm cụm (centroid)
* : khoảng cách Euclidean bình phương
* : một điểm dữ liệu

Như vậy, là tổng bình phương khoảng cách trong cụm (Within-Cluster Sum of Squares), đo lường mức độ phân tán của các điểm trong cụm. Nếu các điểm trong cụm nằm càng gần nhau, càng nhỏ.

#### **Những hạn chế của K-Means là gì?**

Thuật toán K-Means tồn tại nhiều hạn chế như: phải xác định trước số cụm K, nhạy cảm với điểm khởi tạo, chỉ phù hợp với cụm có dạng hình cầu và mật độ tương đương, dễ bị ảnh hưởng bởi nhiễu và ngoại lệ, không áp dụng tốt cho dữ liệu phân loại và sẽ bị sai lệch nếu dữ liệu chưa được chuẩn hóa.

Trước hết, K-Means yêu cầu phải xác định trước số cụm K. Việc này khiến người dùng phải dự đoán trước số nhóm trong dữ liệu, dẫn đến khả năng chọn sai, làm kết quả phân cụm không phản ánh đúng cấu trúc thực tế. Nếu chọn quá ít, nhiều nhóm dữ liệu sẽ bị gộp chung; còn nếu chọn quá nhiều, một nhóm có thể bị chia nhỏ thành nhiều cụm không cần thiết.

Thứ hai, K-Means rất nhạy cảm với điểm khởi tạo ban đầu. Nếu các tâm cụm ban đầu được chọn khác nhau, thuật toán có thể hội tụ đến các kết quả phân cụm khác nhau, thậm chí không tối ưu.

Thứ ba, K-Means chỉ hoạt động tốt khi các cụm có hình dạng gần như hình cầu, có kích thước và mật độ tương đương. Khi dữ liệu có cụm kéo dài, chồng lấn hoặc mật độ không đồng đều, ranh giới phân cụm do K-Means tạo ra sẽ không chính xác.

Thứ tư, do sử dụng giá trị trung bình để tính tâm cụm, K-Means rất dễ bị ảnh hưởng bởi nhiễu hoặc điểm ngoại lệ. Chỉ vài điểm lệch cũng có thể khiến tâm cụm bị kéo ra khỏi vùng trung tâm thật, làm giảm độ chính xác của mô hình.

Thứ năm, thuật toán này chỉ phù hợp với dữ liệu số, vì nó dựa vào khoảng cách Euclidean để tính độ tương đồng. Với dữ liệu dạng phân loại, khoảng cách giữa các giá trị không có ý nghĩa, khiến K-Means không thể áp dụng hiệu quả.

Cuối cùng, nếu dữ liệu chưa được chuẩn hóa về cùng một thang đo, các biến có giá trị lớn sẽ chi phối toàn bộ quá trình tính toán khoảng cách, làm sai lệch kết quả phân cụm. Vì vậy, việc chuẩn hóa dữ liệu là bước bắt buộc khi sử dụng K-Means.

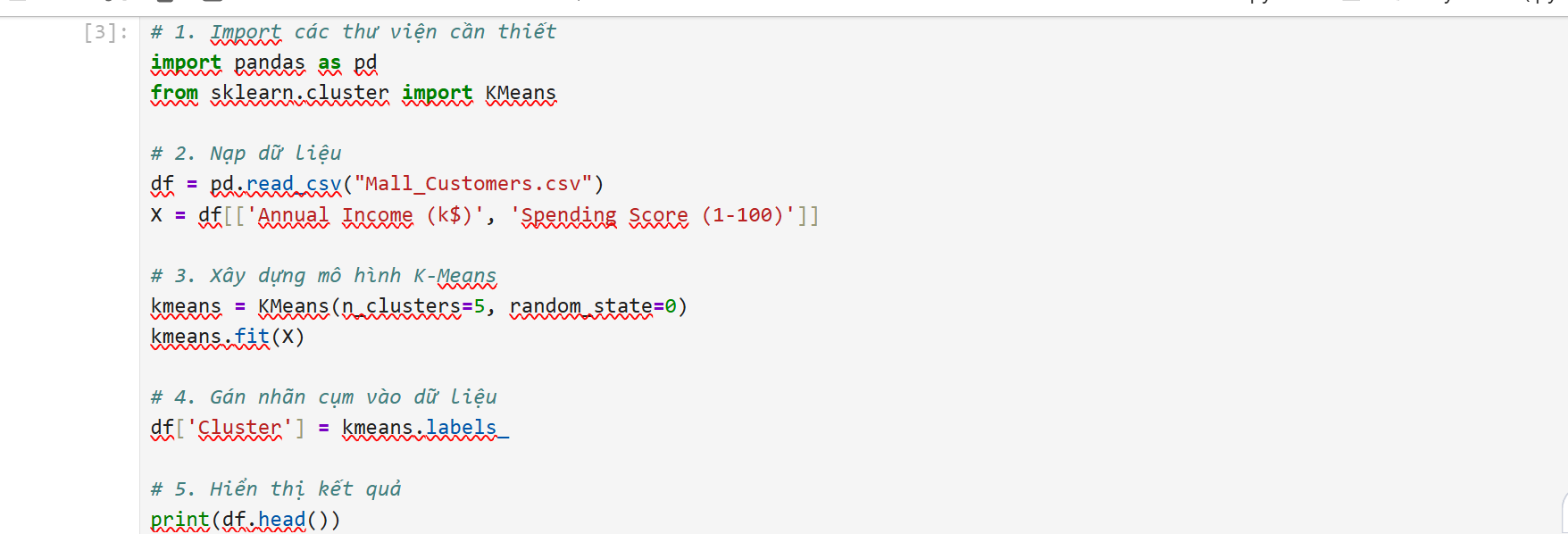
#### **Trong trường hợp nào K-Means có thể cho kết quả không tốt?**

K-Means có thể cho kết quả không tốt trong nhiều tình huống khác nhau. Trước hết, khi dữ liệu có các cụm hình dạng phức tạp như hình vòng tròn hay lưỡi liềm, thuật toán này sẽ phân chia sai vì nó chỉ tạo ranh giới thẳng. Ví dụ, nếu có hai cụm dữ liệu dạng vòng tròn đồng tâm, K-Means có xu hướng cắt đôi bằng một đường thẳng thay vì nhận ra một cụm nằm bên trong, một cụm nằm bên ngoài.

Tiếp theo, với dữ liệu có mật độ không đồng đều, K-Means thường chia sai cụm. Chẳng hạn, nếu có một cụm rất dày và một cụm rất thưa, thuật toán sẽ cố làm đều khoảng cách giữa các điểm, khiến cụm thưa bị tách nhỏ hoặc bị gộp chung với cụm dày.

Ngoài ra, khi dữ liệu có điểm ngoại lệ hoặc nhiễu, K-Means cũng dễ cho kết quả sai lệch. Ví dụ, chỉ một vài điểm nằm xa trung tâm cũng có thể kéo tâm cụm ra xa, khiến các điểm khác bị gán nhầm vào cụm không đúng.

#### **Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để triển khai K-Means Clustering không? Hãy mô tả các bước thực hiện.**

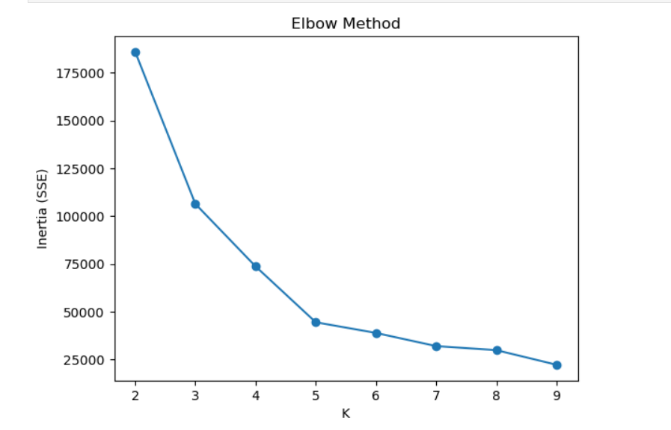
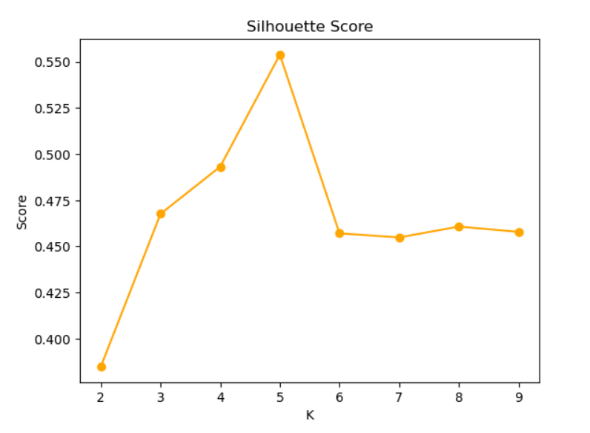
****

#### **Sử dụng phương pháp nào trong Python để chọn số cụm K tối ưu (ví dụ: Elbow Method, Silhouette Score)? Hãy chia sẻ một đoạn code mẫu**

Ta chạy đoạn code sau:

****

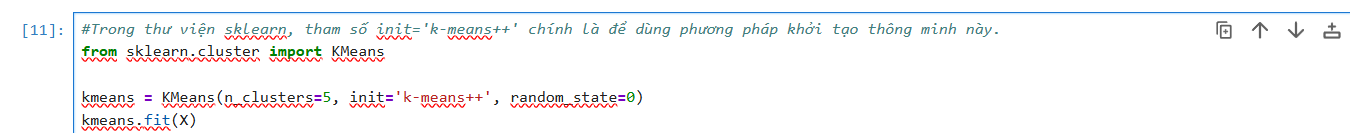
Kết quả nhận được:

** **

#### **K-Means nhạy cảm với giá trị khởi tạo (initial centroids), bạn sẽ làm gì để đảm bảo kết quả ổn định (ví dụ: K-Means++)?**

Sử dụng phương pháp khởi tạo K-Means++, giúp chọn các tâm phân bố xa nhau một cách có kiểm soát, tránh hội tụ xấu:

* Chọn tâm đầu tiên ngẫu nhiên trong dữ liệu.
* Với mỗi điểm còn lại, tính khoảng cách đến tâm gần nhất đã chọn.
* Chọn tâm mới với xác suất tỉ lệ thuận với bình phương khoảng cách đó, tức là những điểm xa các tâm hiện có sẽ có cơ hội cao hơn được chọn làm tâm mới.
* Lặp lại cho đến khi có đủ k tâm.
* Các tâm ban đầu phân bố đều khắp dữ liệu, không bị dồn một chỗ.

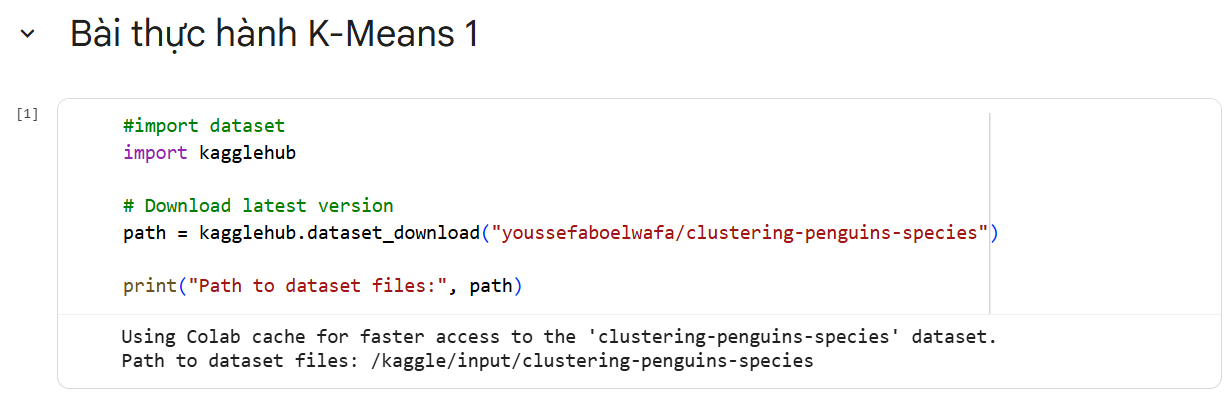


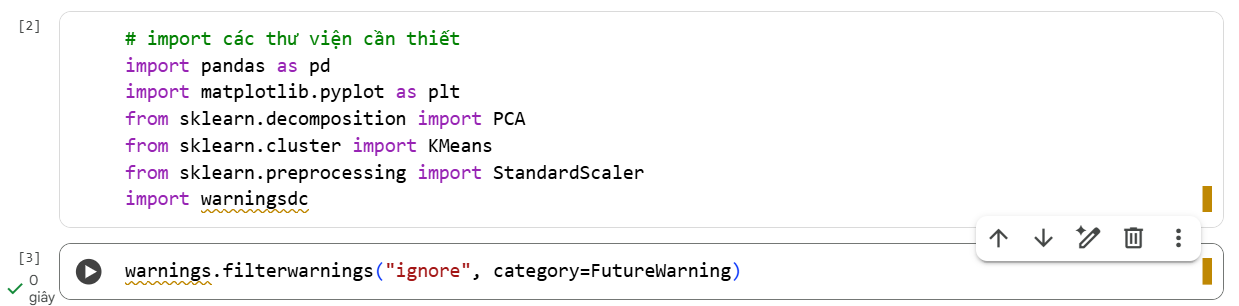
#### **Làm thế nào để đánh giá chất lượng của các cụm được tạo bởi K-Means? Bạn sử dụng chỉ số nào (ví dụ: Silhouette Score, Within-Cluster Sum of Squares)?**

* Within-Cluster Sum of Squares (WCSS / Inertia): Tổng bình phương khoảng cách trong cụm 🡪 càng nhỏ càng tốt.
* Silhouette Score: Đo sự tách biệt giữa các cụm (giá trị từ -1 đến 1, càng gần 1 càng tốt).
* Davies-Bouldin Index (DBI): Càng thấp càng thể hiện phân cụm tốt.

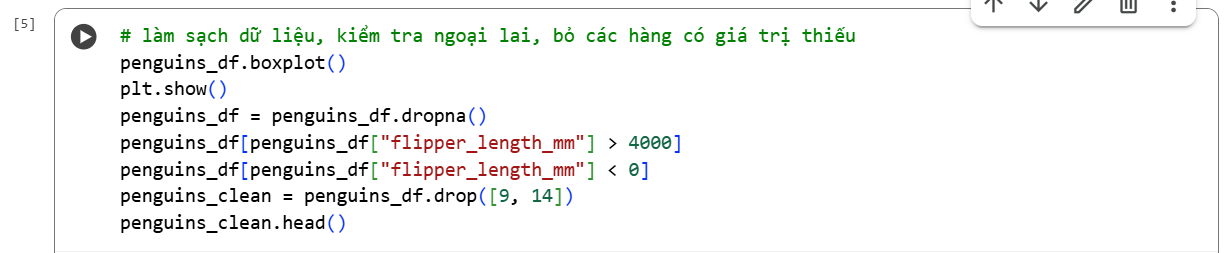
### **3.1.2. Bài tập thực hành**

1. **Xây dựng mô hình phân cụm K-means trên tập dữ liệu chim cánh cụt. Dữ liệu lấy tại** [**https://www.kaggle.com/code/youssefaboelwafa/clustering-penguins-species-k-means-clustering**](https://www.kaggle.com/code/youssefaboelwafa/clustering-penguins-species-k-means-clustering)

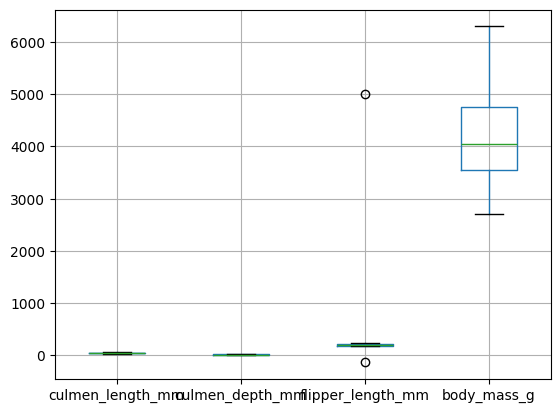
****

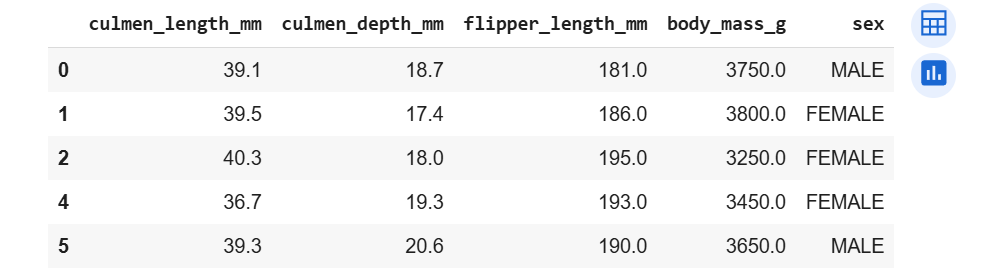
****

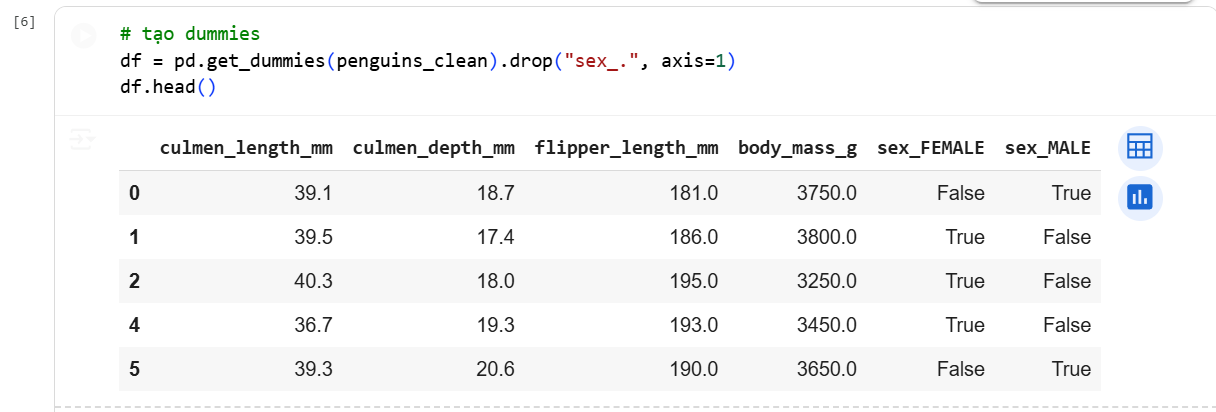
****

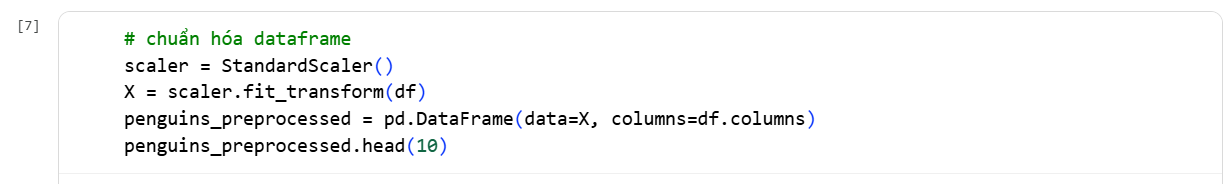
****

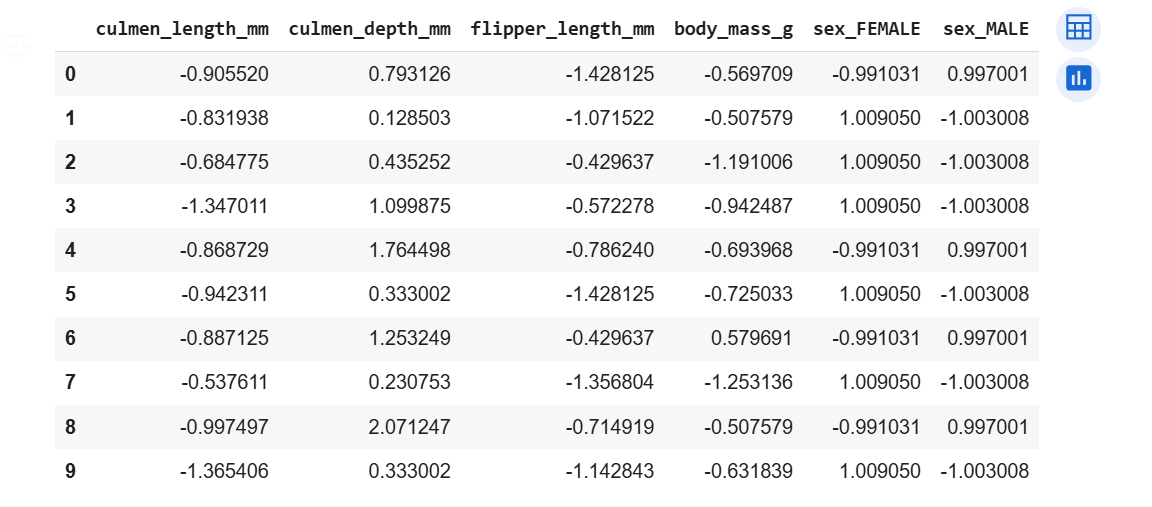
Kết quả đạt được:

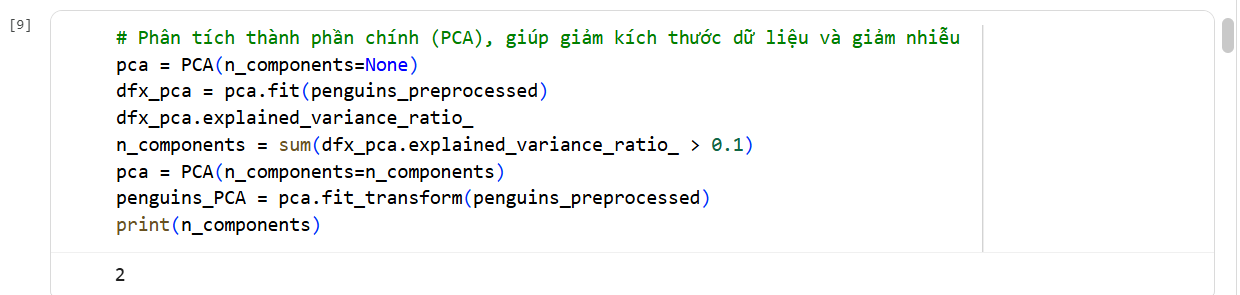
****

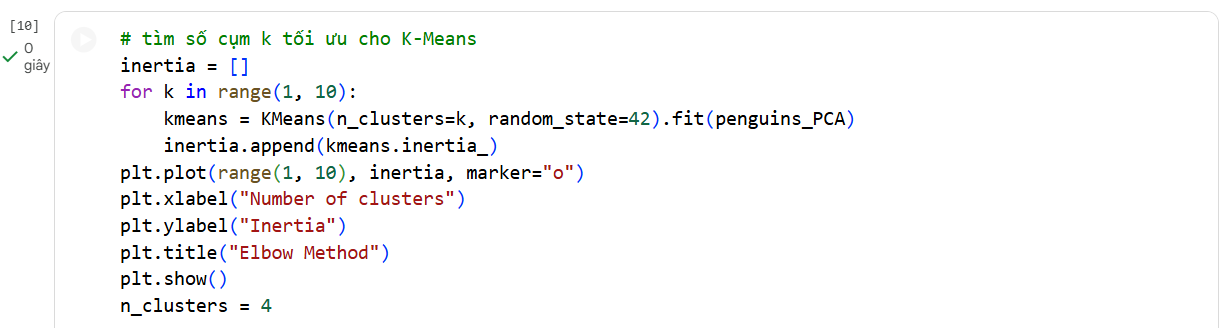
****

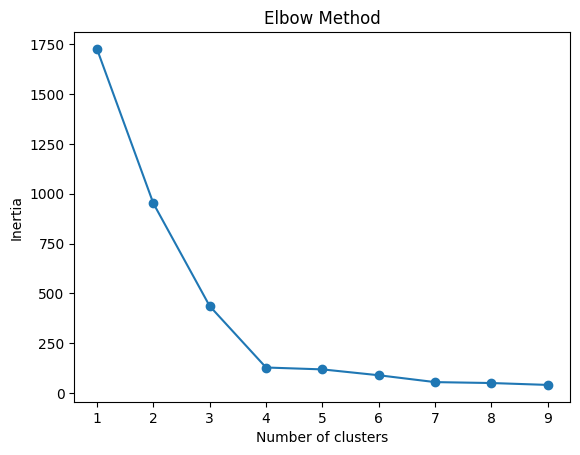
****

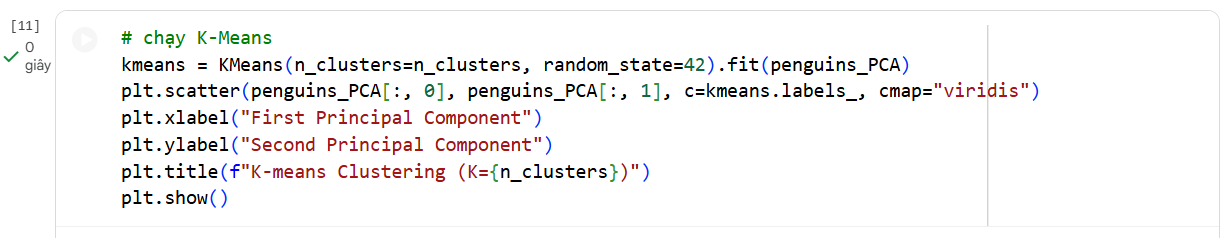
****

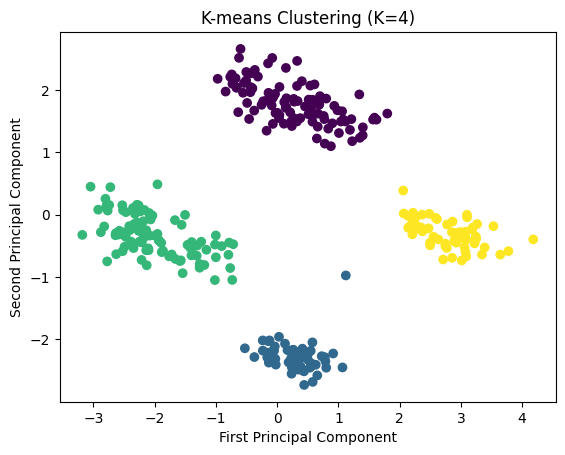
****

****

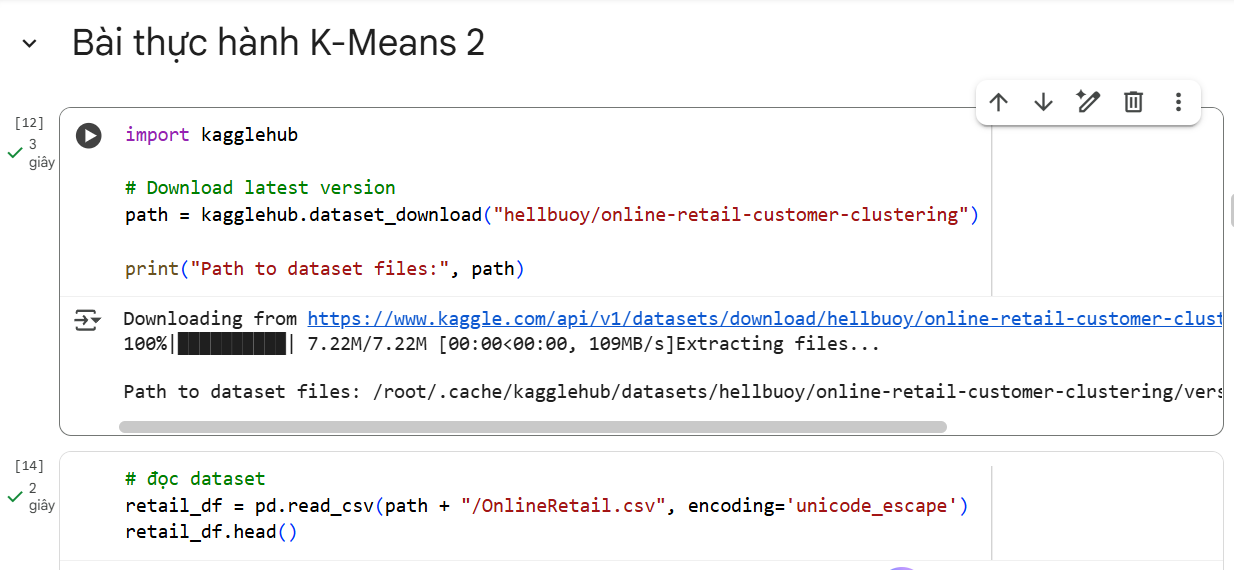
****

****

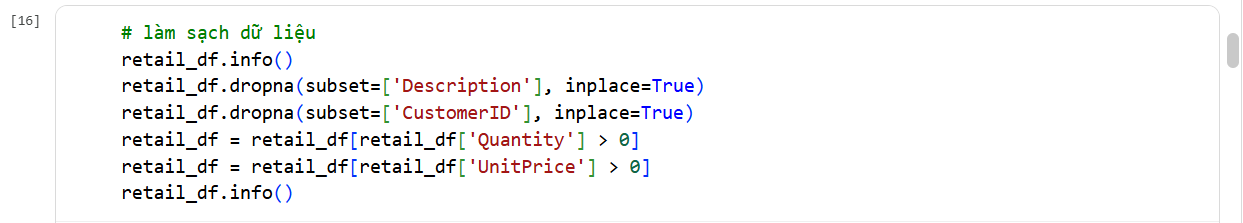
****

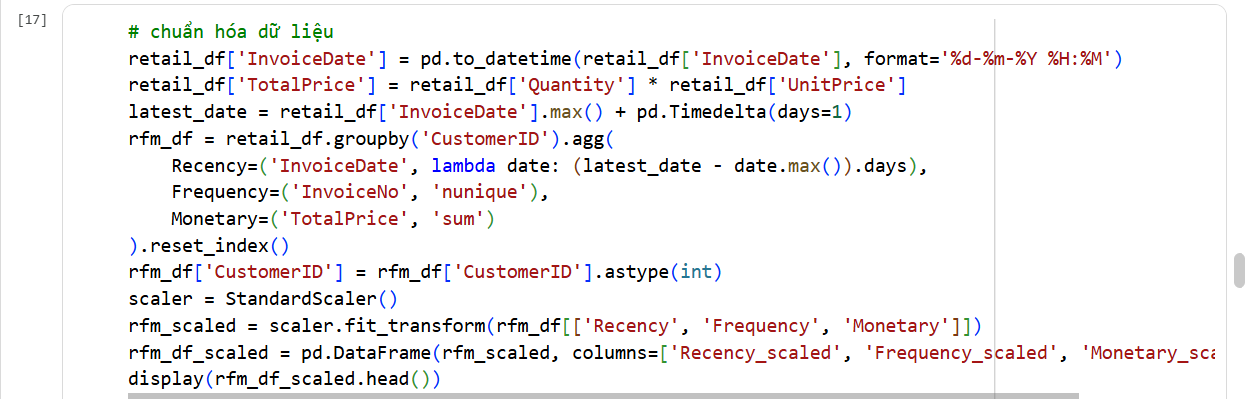
****

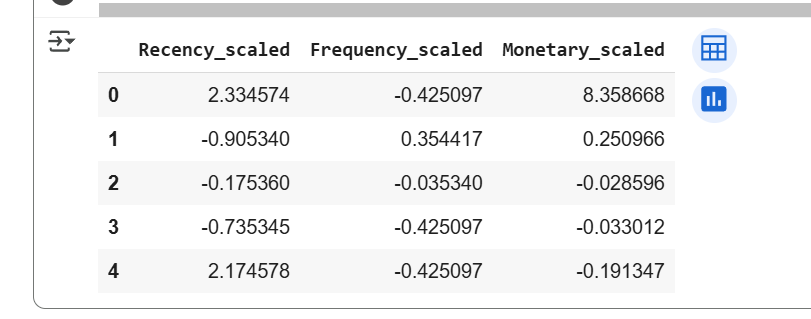
1. **Xây dựng mô hình phân cụm K-means trên tập dữ liệu mua sắm tại siêu thị. Dữ liệu lấy tại** [**https://www.kaggle.com/datasets/hellbuoy/online-retail-customer-clustering**](https://www.kaggle.com/datasets/hellbuoy/online-retail-customer-clustering)

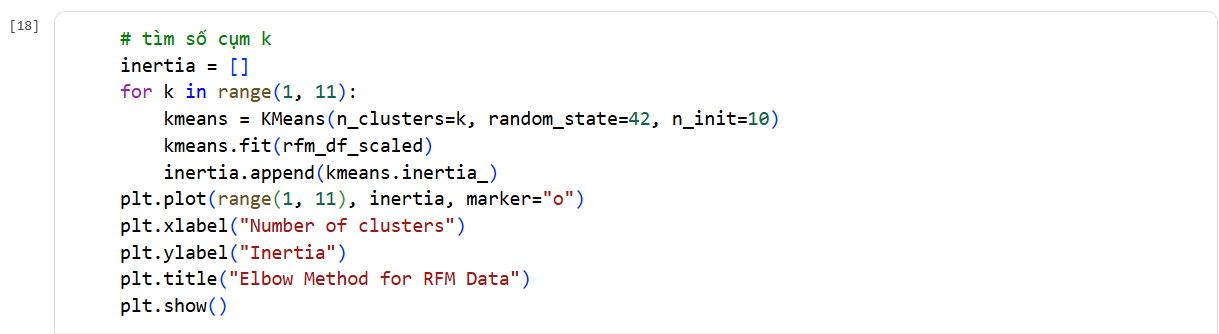
****

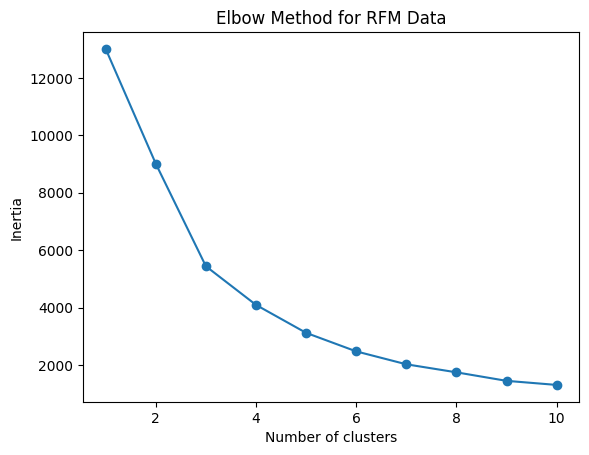
****

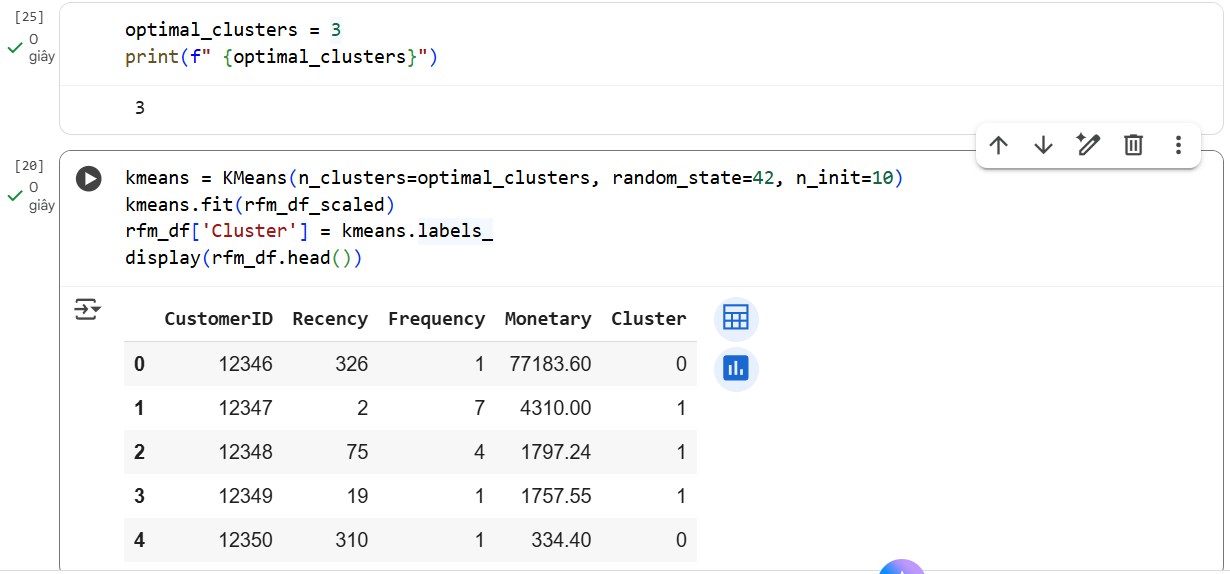
****

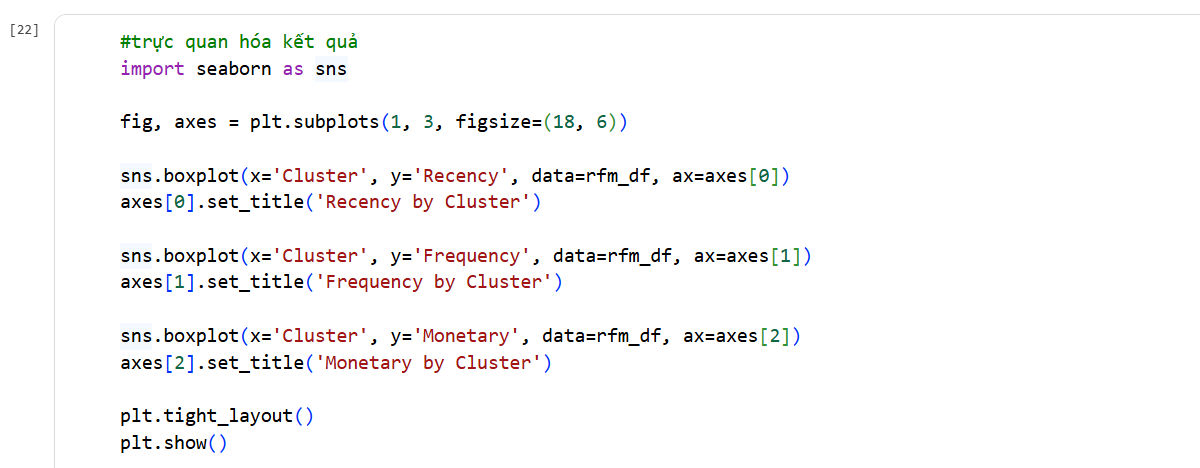
****

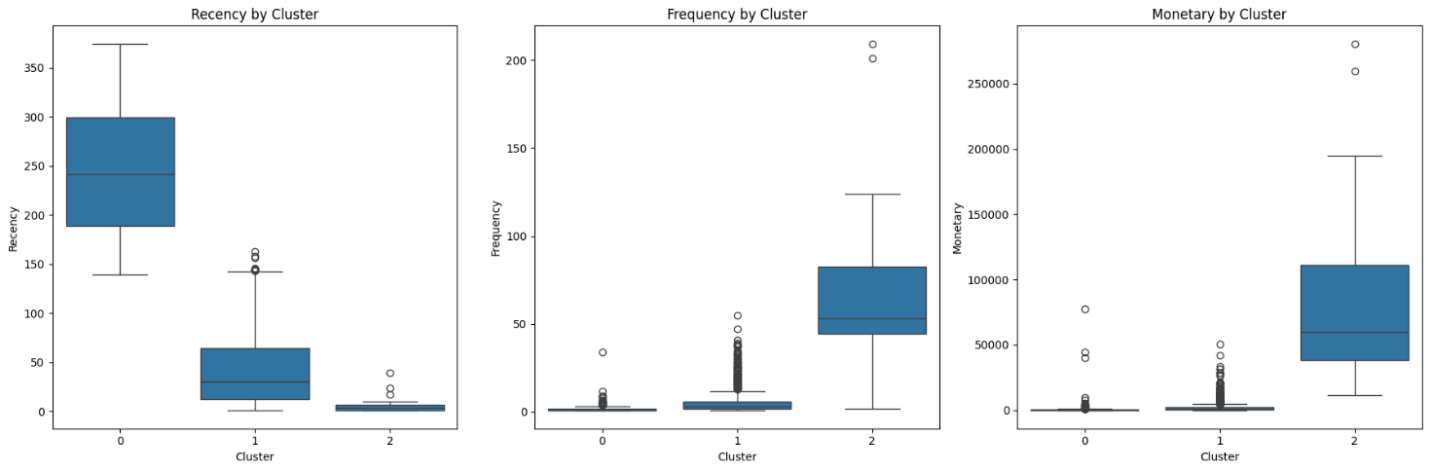
****

****

****

****

****

****

## **3.2. Giải thuật phân cụm đa cấp**

### **3.2.1. Ôn tập lý thuyết**

#### **Giải thuật phân cụm đa cấp hoạt động như thế nào?**

Phân cụm đa cấp (Hierarchical Clustering) là một phương pháp phân cụm không giám sát, tạo cấu trúc cây phân cấp (dendrogram) thể hiện mối quan hệ giữa các điểm dữ liệu.  
Có hai hướng tiếp cận chính:

* **Agglomerative (Hợp nhất):** Mỗi điểm dữ liệu ban đầu là một cụm riêng, sau đó các cụm gần nhau nhất sẽ được hợp lại dần cho đến khi còn một cụm duy nhất.
* **Divisive (Phân tách):** Toàn bộ dữ liệu ban đầu là một cụm duy nhất, sau đó được tách dần thành nhiều cụm nhỏ hơn.

#### **Các phương pháp liên kết (Linkage)**

* **Single linkage:** Khoảng cách giữa hai cụm là khoảng cách nhỏ nhất giữa hai điểm.
* **Complete linkage:** Khoảng cách giữa hai cụm là khoảng cách lớn nhất giữa hai điểm.
* **Average linkage:** Dựa trên khoảng cách trung bình giữa các điểm.
* **Ward’s linkage:** Tối thiểu hóa phương sai trong mỗi cụm, tạo ra các cụm có kích thước cân bằng hơn.

→ Trong thực tế, **Ward’s linkage** thường được sử dụng cho dữ liệu số (numeric data) vì cho kết quả ổn định và dễ diễn giải.

#### **Dendrogram trong phân cụm đa cấp:**

* Là biểu đồ dạng cây biểu diễn quá trình gộp hoặc tách các cụm dữ liệu.
* Chiều cao tại vị trí gộp hai cụm thể hiện khoảng cách giữa chúng.
* Dendrogram được sử dụng để xác định **số lượng cụm tối ưu** bằng cách quan sát các nhánh tách biệt rõ ràng nhất.

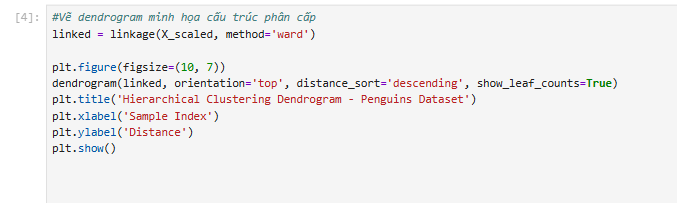
#### **Ứng dụng với dữ liệu phi số (Non-numeric data):**

Có thể áp dụng được nếu dữ liệu phi số được mã hóa thành dạng số (ví dụ: one-hot encoding).  
Trong trường hợp dữ liệu dạng chuỗi hoặc phân loại, có thể dùng các thước đo khoảng cách đặc biệt như **Hamming**, **Jaccard**, hoặc **Cosine distance**.

#### **Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để triển khai phân cụm đa cấp hợp nhất (agglomerative clustering) không? Hãy mô tả các bước thực hiện.**

****

#### **Làm thế nào để vẽ dendrogram trong Python sử dụng thư viện như scipy hoặc matplotlib? Hãy chia sẻ một đoạn code mẫu**

****

#### **Các lớp trong gói Scipy hỗ trợ phân cụm đa cấp? và so sánh giữa cách tiếp cận scikit-learn và cách tiếp cận sử dụng Scipy**

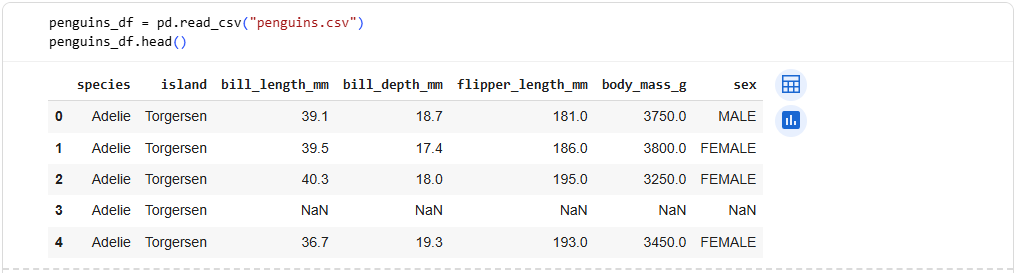
* **linkage()**: Tính toán ma trận liên kết giữa các điểm dữ liệu để xây dựng cấu trúc phân cấp.
* **dendrogram()**: Vẽ biểu đồ cây phân cấp (dendrogram) minh họa quá trình gộp hoặc tách cụm.
* **fcluster()**: Cắt cây dendrogram tại một ngưỡng khoảng cách nhất định để tạo các cụm cụ thể.
* **cophenet()**: Đánh giá độ tương đồng giữa ma trận khoảng cách ban đầu và cấu trúc phân cấp.

### **3.2.2. Bài tập thực hành**

1. **Xây dựng mô hình phân cụm đa cấp trên tập dữ liệu chim cánh cụt. Dữ liệu lấy tại** <https://www.kaggle.com/code/youssefaboelwafa/clustering-penguins-species-k-means-clustering>

****

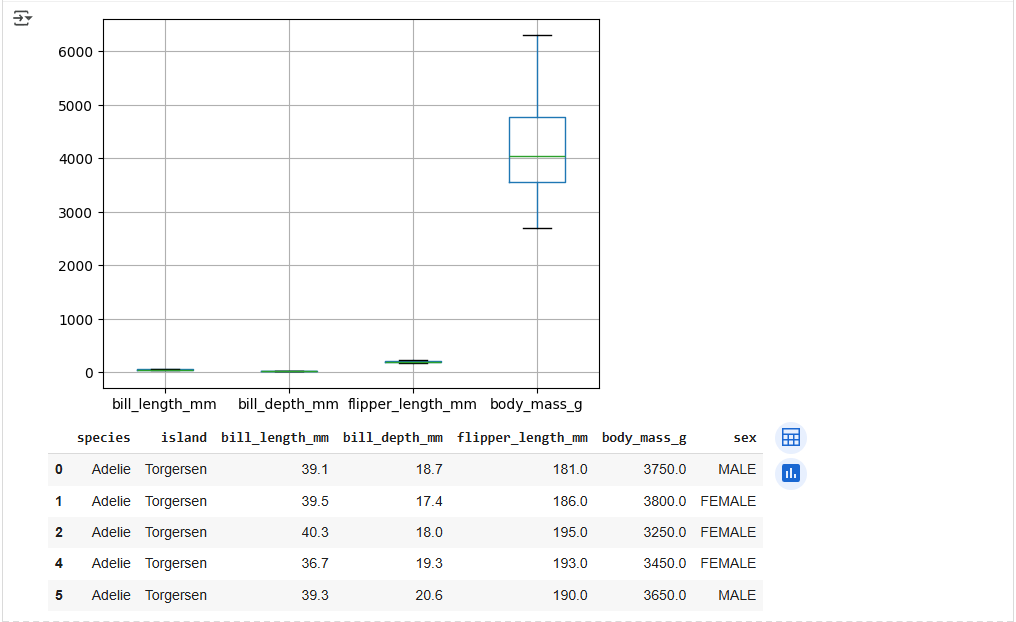
****

****

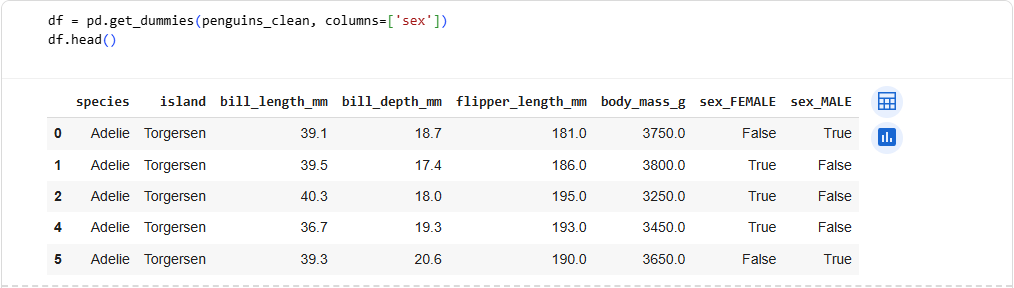
**Xử lý null và giá trị ngoại lệ**

****

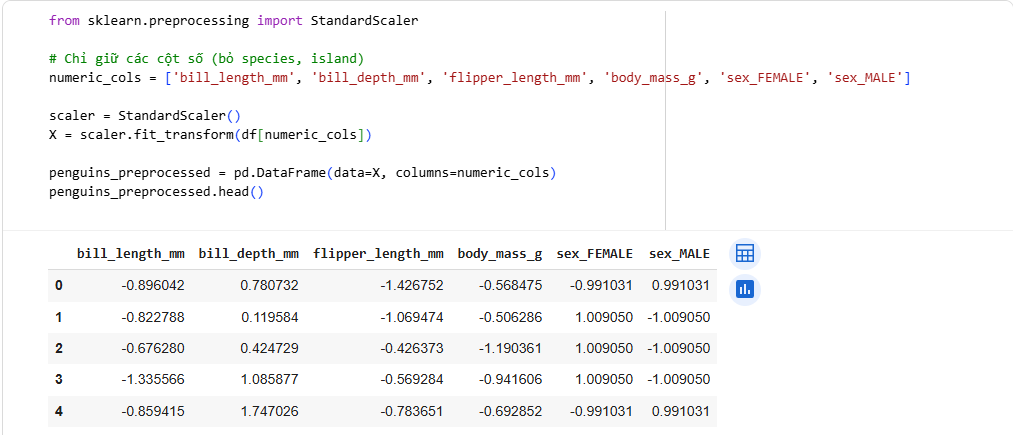
Kết quả đạt được:

****

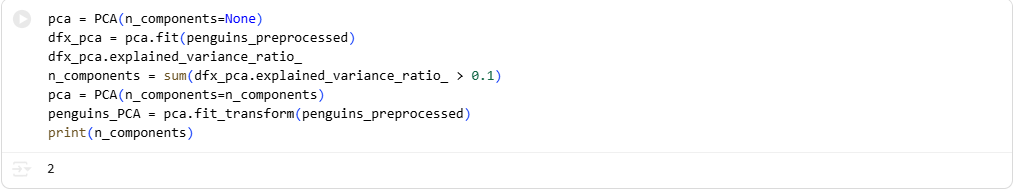
**Tiền xử lý dữ liệu**

****

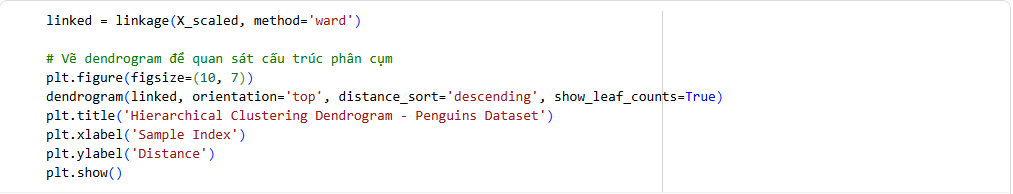
**Scaling**

****

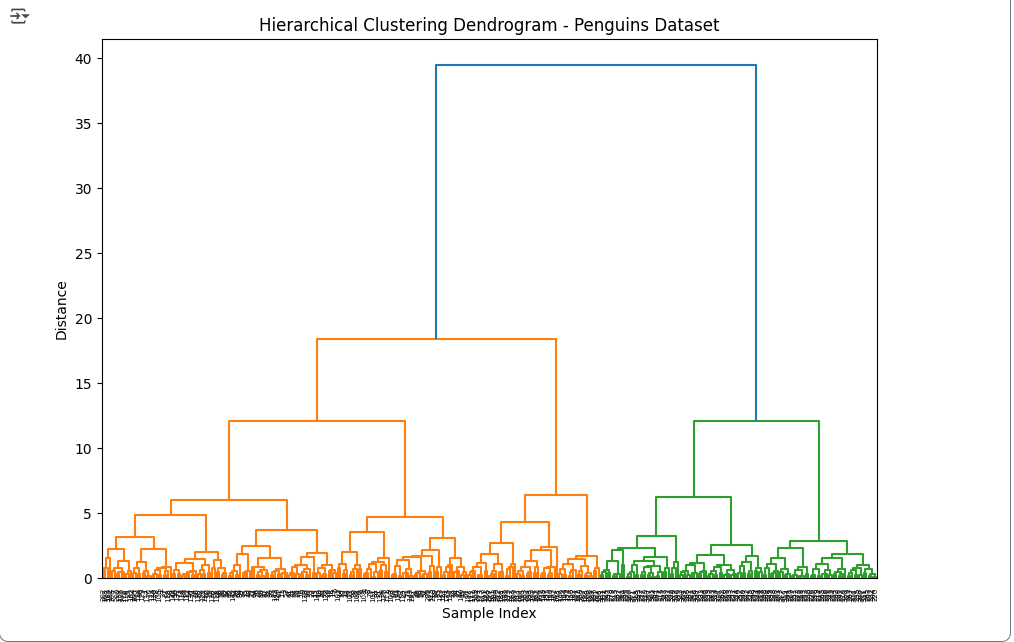
**Perform PCA**

****

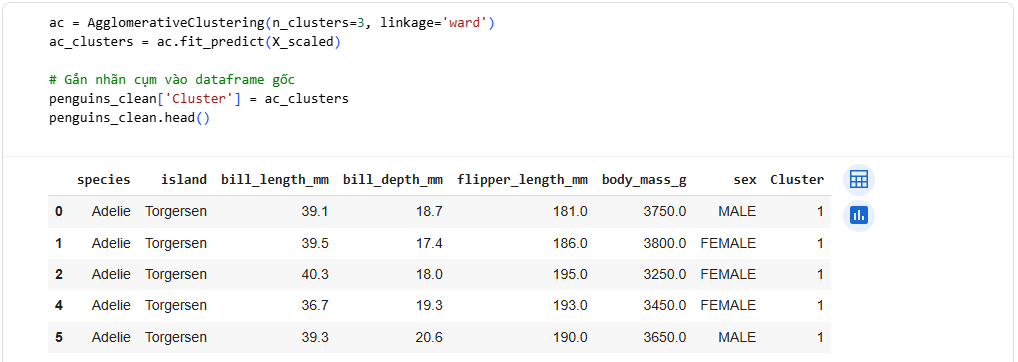
**Dendrogram**

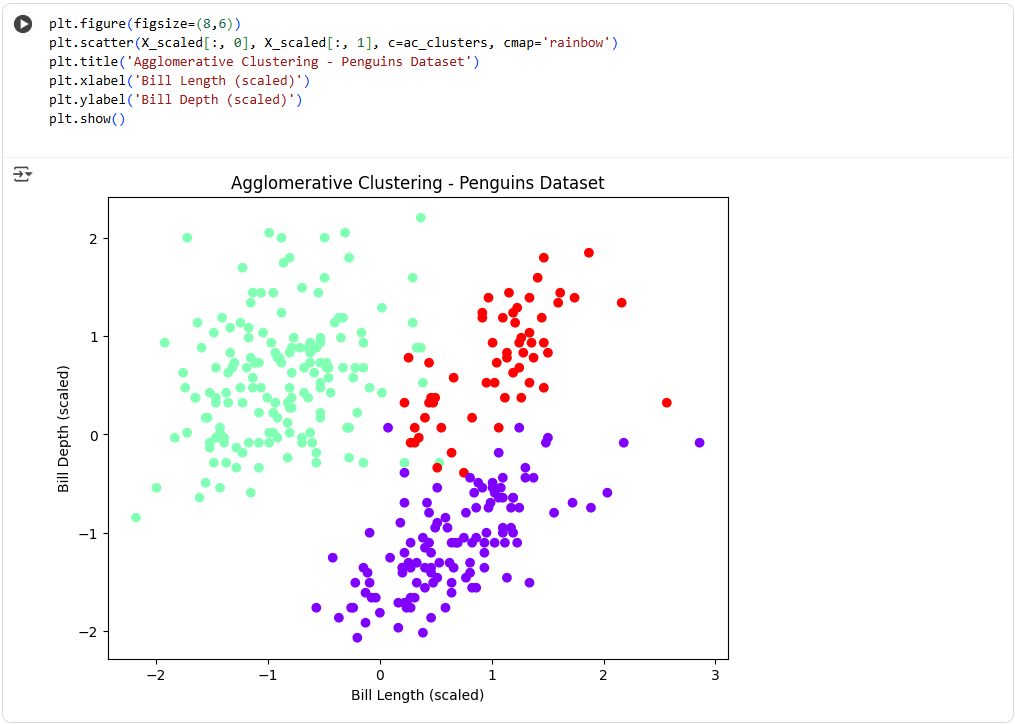
**** ****

Kết quả đạt được:

****

**Agglomerative Clustering**

****

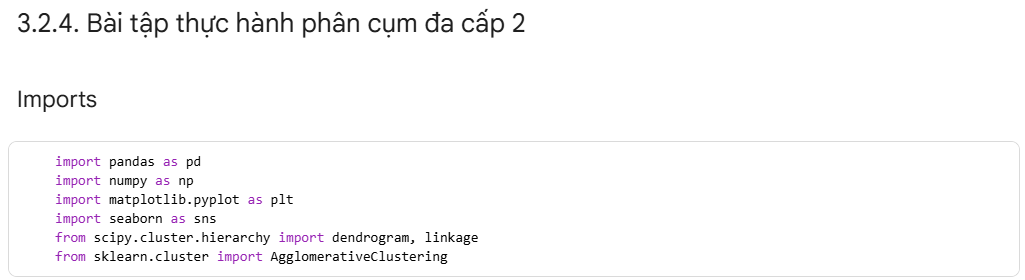
****

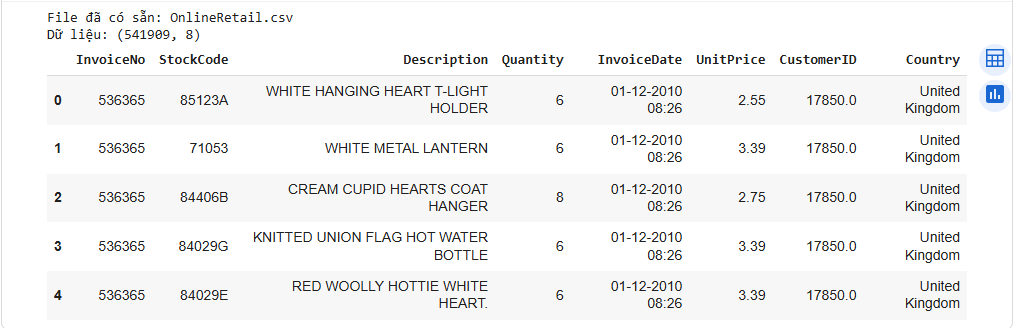
**Phân cụm đa cấp (Hierarchical Clustering):**

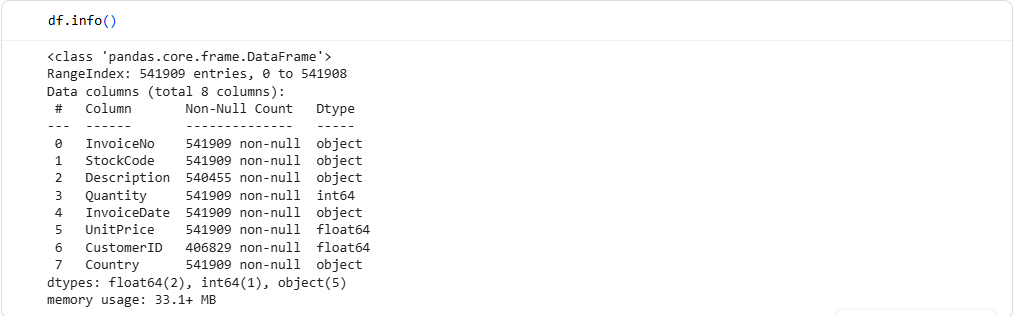
* **Cụm 0:** Những cá thể chim có **mỏ dài** và **cơ thể nặng hơn**, đại diện cho **loài Gentoo**.
* **Cụm 1:** Những cá thể có **mỏ ngắn** và **thân hình nhỏ nhẹ**, chủ yếu tương ứng với **loài Adelie**.
* **Cụm 2:** Những cá thể có **kích thước mỏ trung bình** và **khối lượng cơ thể vừa phải**, tương tự **loài Chinstrap**.

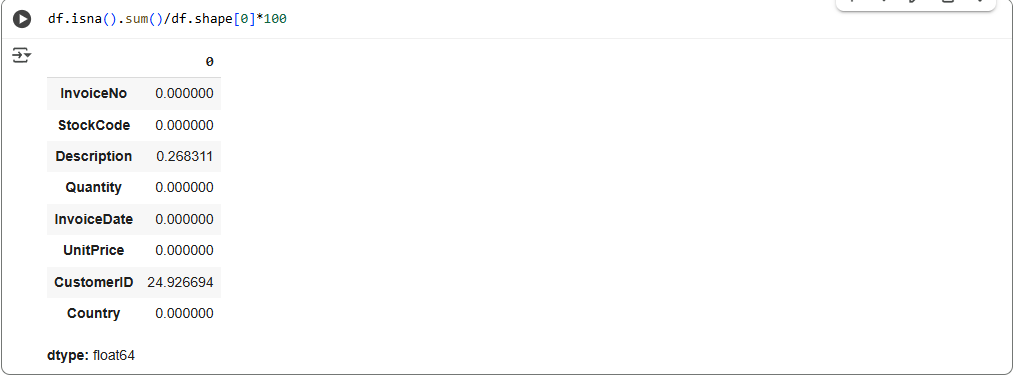
Nhìn chung, mô hình phân cụm đa cấp đã **phân nhóm chim cánh cụt thành ba cụm riêng biệt**, phản ánh **rõ ràng sự khác biệt về đặc điểm hình thái và loài**.

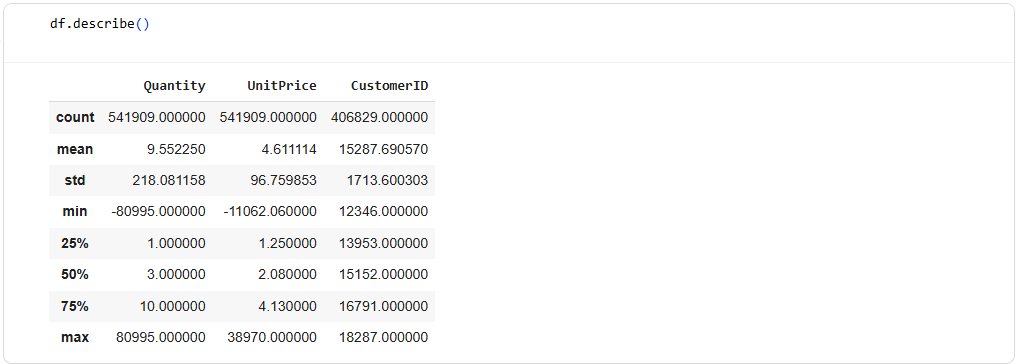
1. **Xây dựng mô hình phân cụm đa cấp trên tập dữ liệu mua sắm tại siêu thị. Dữ liệu lấy tại** <https://www.kaggle.com/datasets/hellbuoy/online-retail-customer-clustering>

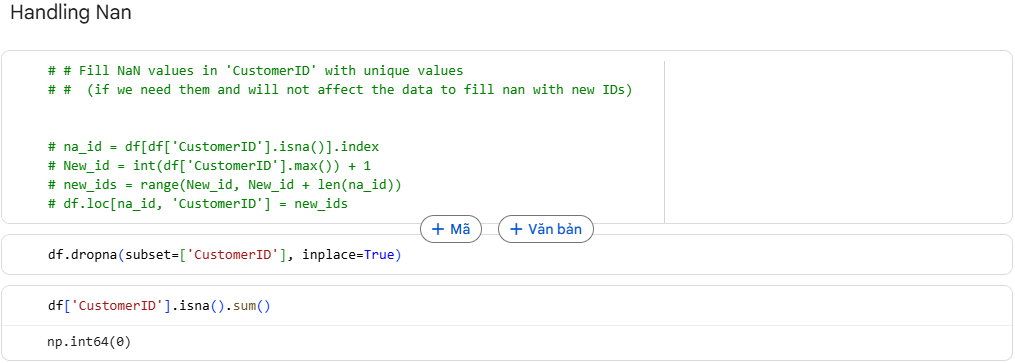
****

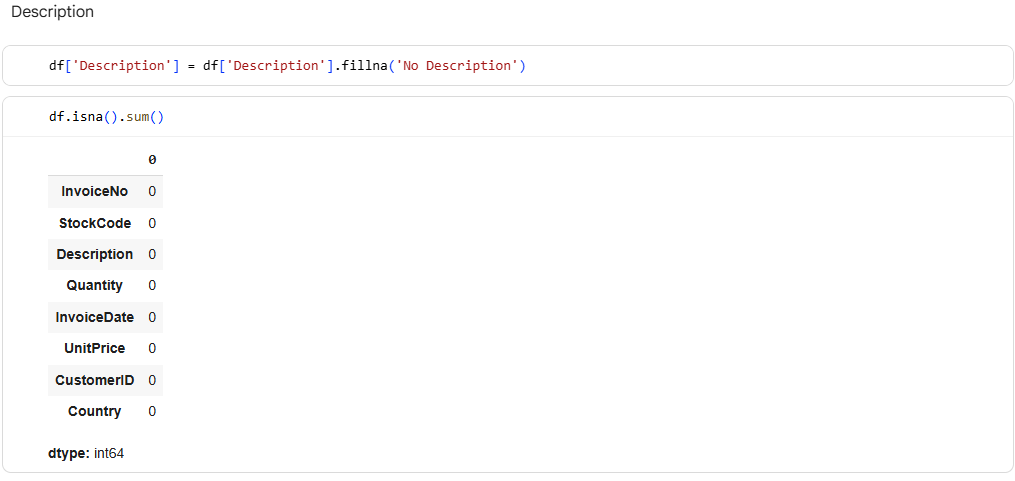
****

****

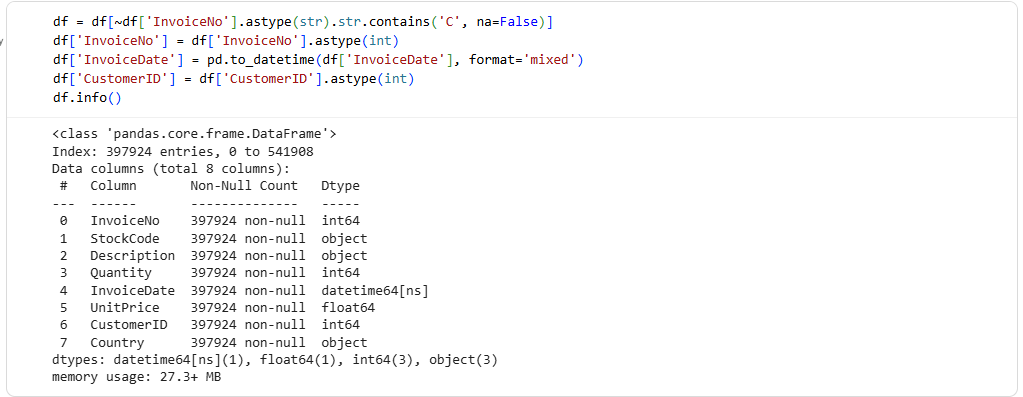
****

****

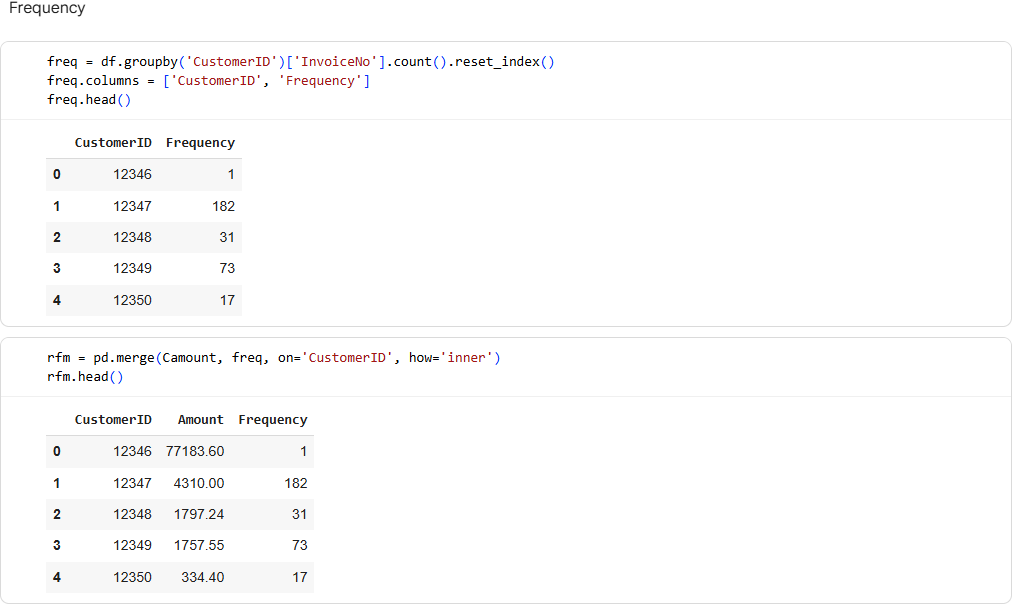
****

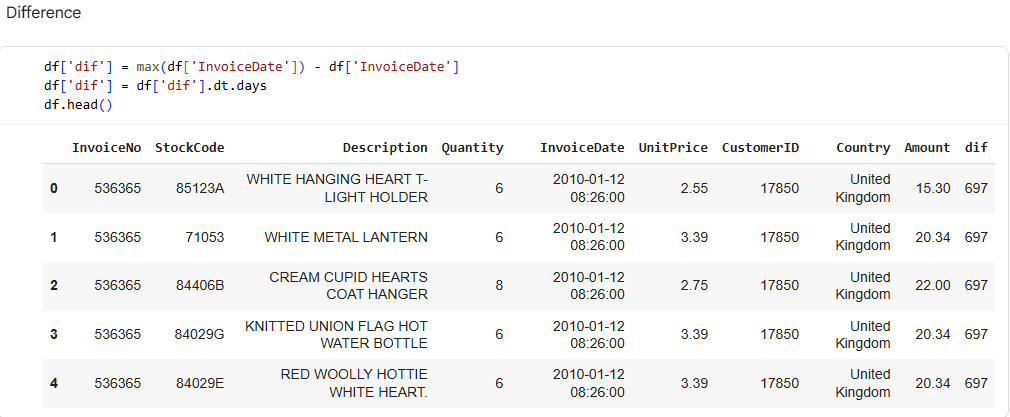
****

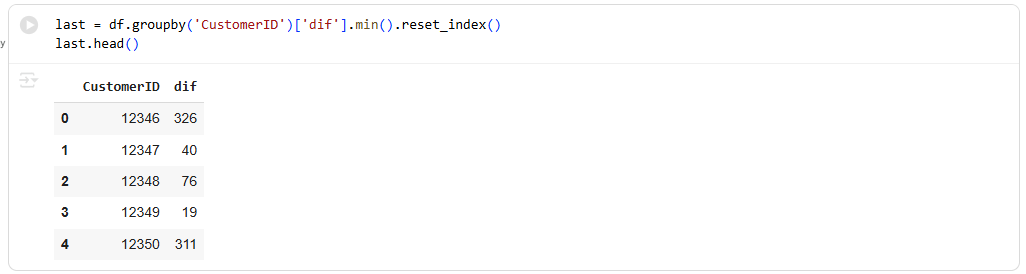
****

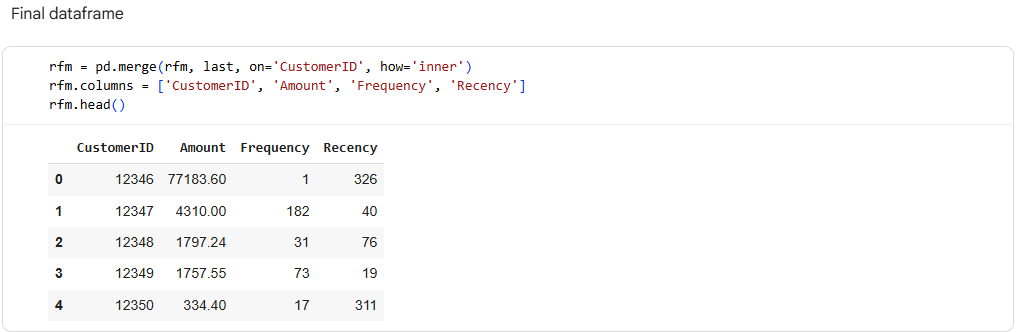
****

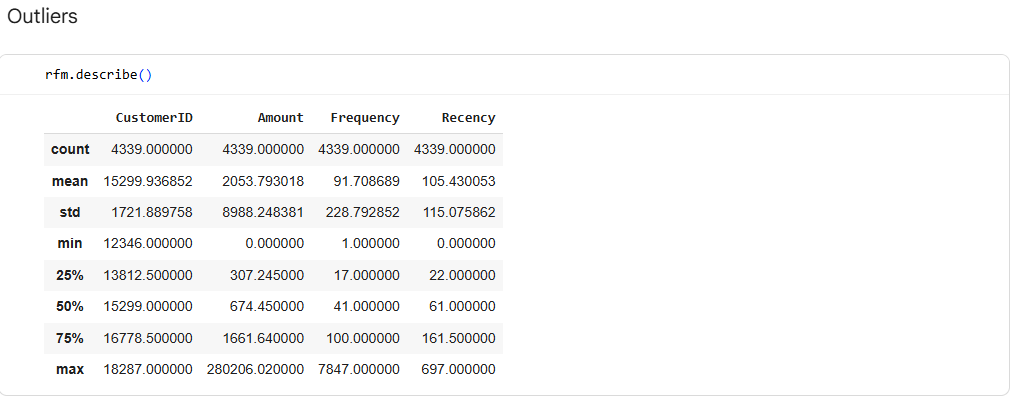
****

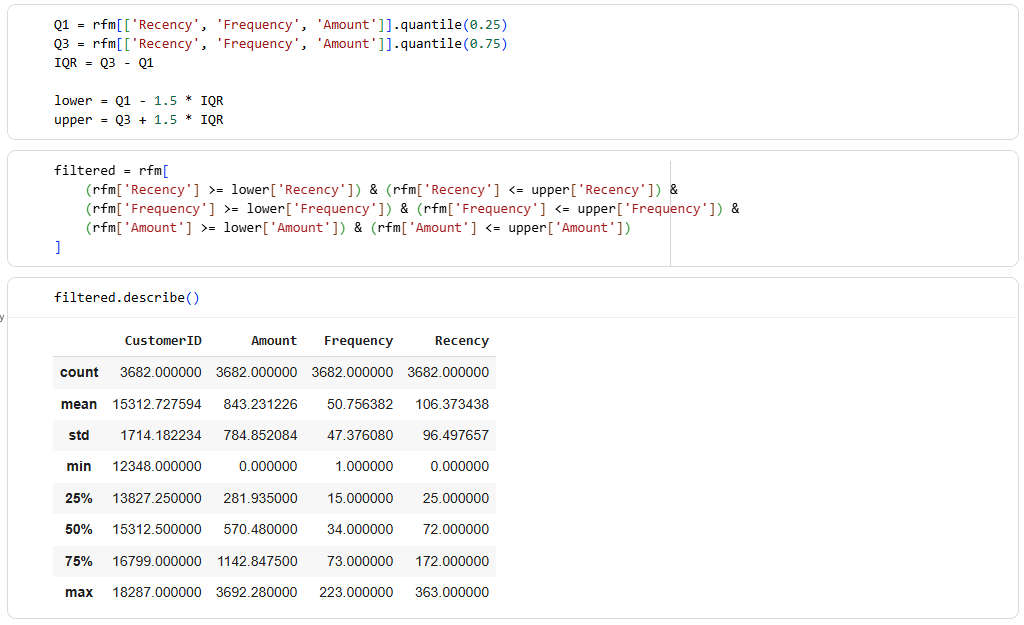
****

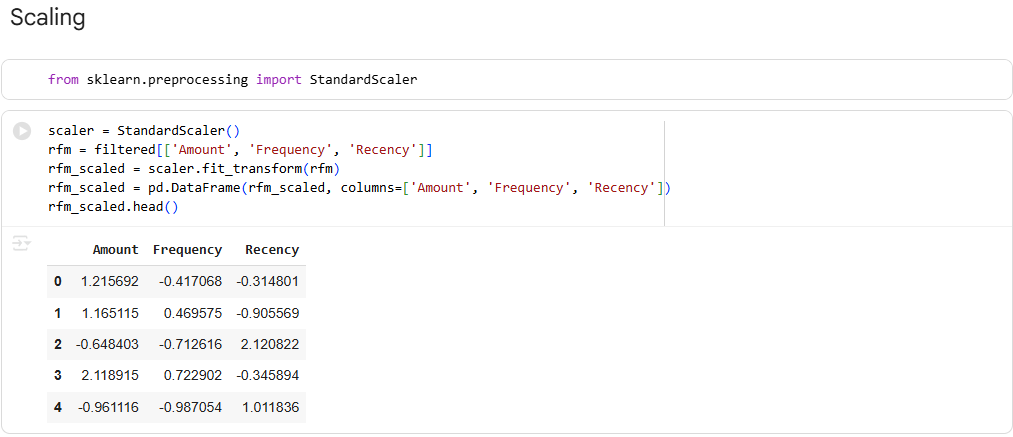
****

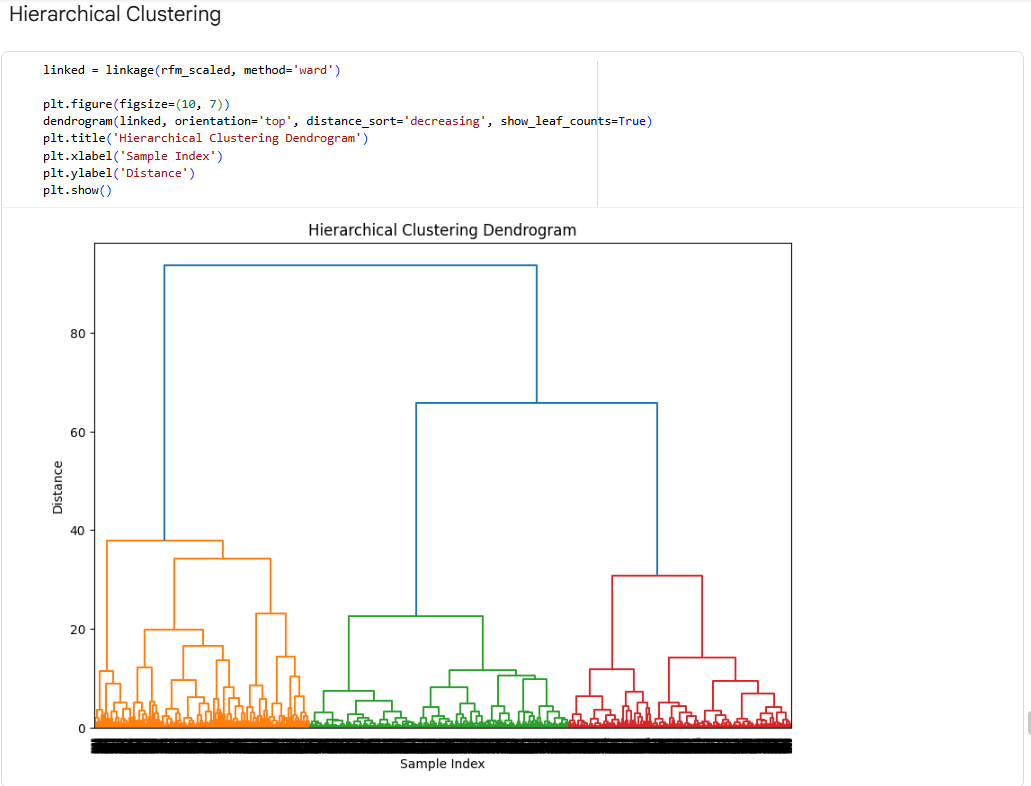
****

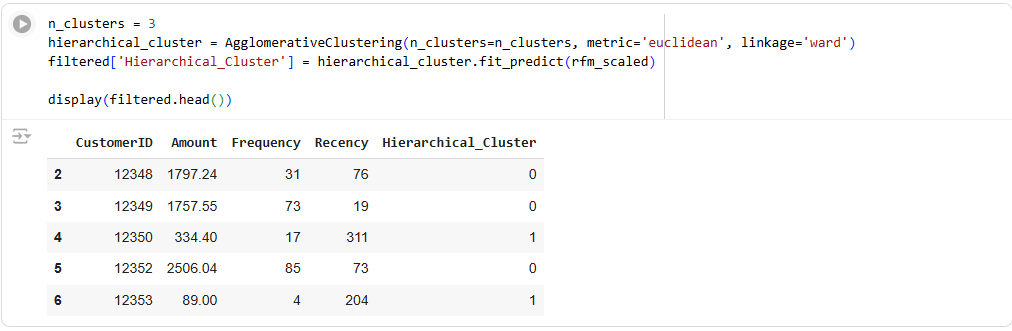
****

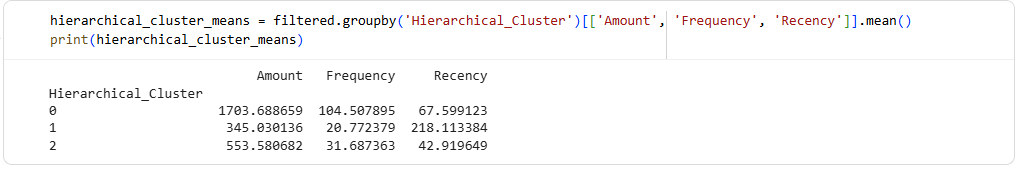
****

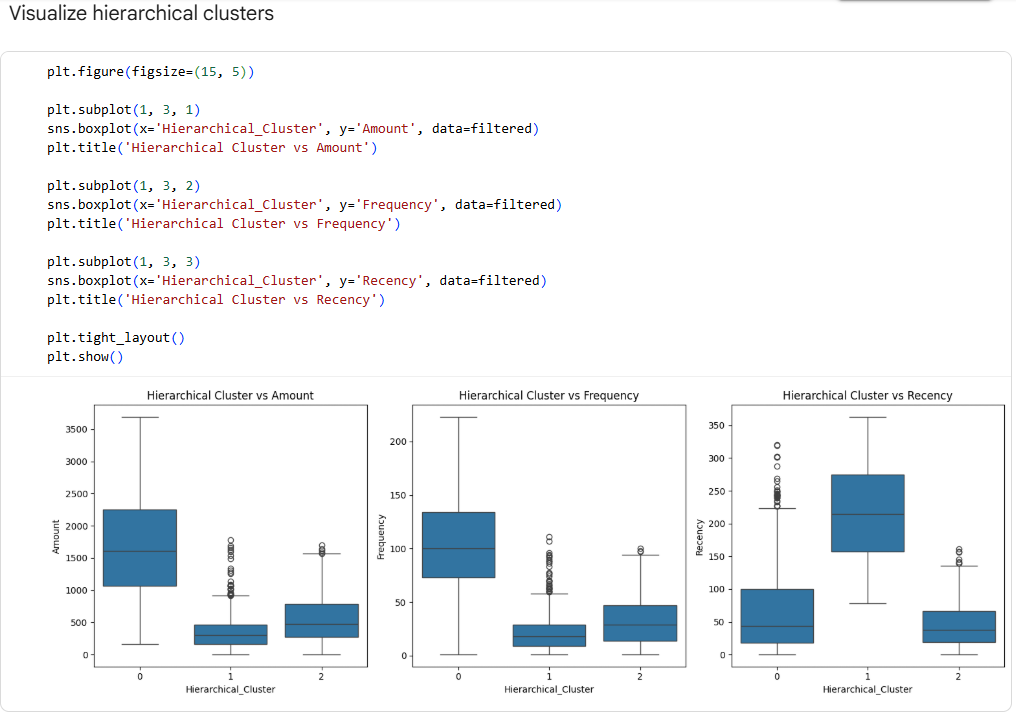
****

****

****

****

****

****

**Phân cụm đa cấp (Hierarchical Clustering):**

* **Cụm 0:** Có **giá trị chi tiêu trung bình cao** ($1703.69) và **tần suất mua hàng cao** (104.51), cùng với **thời gian gần mua gần đây thấp** (Recency = 67.60). → Nhóm **khách hàng trung thành và chi tiêu lớn**.
* **Cụm 1:** Có **mức chi tiêu trung bình thấp** ($345.03) và **tần suất mua hàng thấp** (20.77), nhưng **Recency cao** (218.11). → Nhóm **khách hàng ít mua và đã lâu không quay lại**.
* **Cụm 2:** Có **mức chi tiêu và tần suất ở mức trung bình** ($553.58 và 31.69), cùng với **Recency thấp** (42.92). → Nhóm **khách hàng mới hoặc mua hàng đều đặn ở mức vừa phải**.