

**ĐẠI HỌC KINH TẾ THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG KINH TẾ, LUẬT VÀ QUẢN LÝ NHÀ NƯỚC**

**KHOA KINH TẾ**

**----- 🙠🞟🙢 -----**

**BÁO CÁO GIỮA KỲ**

**LAB 1+ 2+ 3 MIDTERM**

**Môn:** Kỹ thuật lập trình với Stata và Python

**Giảng viên**   
**Mã lớp học phần  
Thành viên nhóm**

TP. Hồ Chí Minh, ngày 19 tháng 10 năm 2025

**:** TS. Đỗ Như Tài  
**:** 25C1ECO50118801  
**:** Lê Trần Khánh Linh – 31231023353  
 Trần Hải Minh – 31231024829  
 Nguyễn Thanh Trúc – 31231022693  
 Trịnh Nhất Mạnh – 31231027146

**MỤC LỤC**

[ĐÁNH GIÁ ĐÓNG GÓP 3](#_Toc211812546)

[BẢNG PHÂN CÔNG CHI TIẾT 3](#_Toc211812547)

[LAB01 – PHÂN TÍCH KHÁM PHÁ DỮ LIỆU 5](#_Toc211812548)

[1. Thống kê mô tả 5](#_Toc211812549)

[1.1. Ôn tập lý thuyết 6](#_Toc211812550)

[1.2. Bài tập thực hành 1 - Thống kê mô tả trên tập dữ liệu *Wine Quality (red)* 8](#_Toc211812551)

[1.3. Bài tập thực hành 2 - Tập dữ liệu *Pima Indians Diabetes* 19](#_Toc211812552)

[2. Xử lý và trực quan hoá dữ liệu 32](#_Toc211812553)

[2.1. Ôn tập lý thuyết 32](#_Toc211812554)

[2.2. Bài tập thực hành 1 - Tập dữ liệu Red Wine Quality 35](#_Toc211812555)

[2.3. Bài tập thực hành 2 - Tập dữ liệu *Pima Indians Diabetes (1)* 39](#_Toc211812556)

[2.4. Bài tập thực hành 2 - Tập dữ liệu *EDA Online Retail (2)* 46](#_Toc211812557)

[3. Phân tích đơn biến và hai biến 51](#_Toc211812558)

[3.1. Ôn tập lý thuyết 51](#_Toc211812559)

[3.2. Bài tập thực hành 1 - Ứng dụng SweetViz trên tập dữ liệu *Marketing Campaign* 53](#_Toc211812560)

[3.3. Bài tập thực hành 2 - Ứng dụng AutoViz trên tập dữ liệu Marketing Campaign 56](#_Toc211812561)

[LAB02 – THUẬT TOÁN PHÂN LỚP 61](#_Toc211812562)

[2.1. GIẢI THUẬT 1: CÂY QUYẾT ĐỊNH VÀ RỪNG CÂY 61](#_Toc211812563)

[2.1.1. Ôn tập lý thuyết 61](#_Toc211812564)

[2.1.2. Bài tập thực hành 1 78](#_Toc211812565)

[2.1.3. Bài tập thực hành 2 81](#_Toc211812566)

[2.2. GIẢI THUẬT 2: SUPPORT VECTOR MACHINE (SVM) 83](#_Toc211812567)

[2.2.1 Ôn tập lý thuyết 83](#_Toc211812568)

[2.2.2. Bài tập thực hành 1 88](#_Toc211812569)

[2.2.3. Bài tập thực hành 2 90](#_Toc211812570)

[2.3. GIẢI THUẬT 3: BAYES NGÂY THƠ (NAÏVE BAYES) 93](#_Toc211812571)

[2.3.1 Ôn tập lý thuyết 93](#_Toc211812572)

[2.3.2. Bài tập thực hành 1 98](#_Toc211812573)

[2.3.3. Bài tập thực hành 2 101](#_Toc211812574)

[LAB03: CÁC THUẬT TOÁN PHÂN CỤM CƠ BẢN 104](#_Toc211812575)

[3.1. GIẢI THUẬT K-MEANS 104](#_Toc211812576)

[3.1.1 Ôn tập lý thuyết 104](#_Toc211812577)

[3.1.2. Bài tập thực hành 1 – Phân cụm K-means trên tập dữ liệu chim cánh cụt 107](#_Toc211812578)

[3.1.3. Bài tập thực hành 2 Xây dựng mô hình phân cụm K-means trên tập dữ liệu mua sắm tại siêu thị. 112](#_Toc211812579)

[3.2. GIẢI THUẬT PHÂN CỤM ĐA CẤP 115](#_Toc211812580)

[3.2.1. Ôn tập lý thuyết 115](#_Toc211812581)

[3.2.2. Bài tập thực hành 1 126](#_Toc211812582)

[3.2.3. Bài tập thực hành 2 133](#_Toc211812583)

# ĐÁNH GIÁ ĐÓNG GÓP

|  |  |  |
| --- | --- | --- |
| HỌ & TÊN | MSSV | ĐÁNH GIÁ ĐÓNG GÓP |
| Lê Trần Khánh Linh | 31231023353 | 100% |
| Trần Hải Minh | 31231024829 | 100% |
| Nguyễn Thanh Trúc | 31231022693 | 100% |
| Trịnh Nhất Mạnh | 31231027146 | 100% |

# BẢNG PHÂN CÔNG CHI TIẾT

| **STT** | **Nội dung / Hạng mục** | **Người phụ trách chính** | **Ghi chú** |
| --- | --- | --- | --- |
| **I. LAB01 – PHÂN TÍCH KHÁM PHÁ DỮ LIỆU** | | | |
| 1 | 1.1 Ôn tập lý thuyết – Thống kê mô tả | **Trần Hải Minh** | Tóm tắt lý thuyết, ví dụ minh họa. |
| 2 | 1.2 Bài tập thực hành 1 – Wine Quality (red) | **Nguyễn Thanh Trúc** | Làm EDA + thống kê mô tả cơ bản. |
| 3 | 1.3 Bài tập thực hành 2 – Pima Indians Diabetes | **Trịnh Nhất Mạnh** | Phân tích, mô tả biến & kết quả thống kê. |
| 4 | 2.1 Ôn tập lý thuyết – Xử lý & trực quan hóa dữ liệu | **Lê Trần Khánh Linh** | Lý thuyết xử lý missing, outlier, visualization. |
| 5 | 2.2 Bài tập thực hành 1 – Red Wine Quality | **Trần Hải Minh** | Thực hành xử lý dữ liệu, biểu đồ trực quan. |
| 6 | 2.3 Bài tập thực hành 2 – Pima Indians Diabetes | **Trịnh Nhất Mạnh** | Thực hành trực quan hóa dữ liệu. |
| 7 | 2.4 Bài tập thực hành 3 – EDA Online Retail | **Lê Trần Khánh Linh** | Tổng hợp kết quả EDA + nhận xét chuyên sâu. |
| 8 | 3.1 Ôn tập lý thuyết – Phân tích đơn biến & hai biến | **Nguyễn Thanh Trúc** | Chuẩn bị nội dung lý thuyết. |
| 9 | 3.2 Bài tập thực hành 1 – SweetViz | **Trần Hải Minh** | Chạy công cụ SweetViz, phân tích kết quả. |
| 10 | 3.3 Bài tập thực hành 2 – AutoViz | **Lê Trần Khánh Linh** | So sánh kết quả AutoViz & SweetViz. |
| **II. LAB02 – THUẬT TOÁN PHÂN LỚP** | | | |
| 11 | 2.1.1 Ôn tập lý thuyết – Cây quyết định & Rừng cây | **Trần Hải Minh** | Tóm tắt, sơ đồ Decision Tree. |
| 12 | 2.1.2–2.1.3 Bài tập thực hành 1–2 | **Trịnh Nhất Mạnh** | Xây dựng mô hình, đánh giá độ chính xác. |
| 13 | 2.2.1 Ôn tập lý thuyết – SVM | **Nguyễn Thanh Trúc** | Lý thuyết & ý nghĩa tham số. |
| 14 | 2.2.2–2.2.3 Bài tập thực hành 1–2 | **Lê Trần Khánh Linh** | Thực hành mô hình SVM, so sánh kết quả. |
| 15 | 2.3.1 Ôn tập lý thuyết – Naïve Bayes | **Trần Hải Minh** | So sánh các loại Bayes, ưu nhược điểm. |
| 16 | 2.3.2–2.3.3 Bài tập thực hành 1–2 | **Lê Trần Khánh Linh** | Áp dụng mô hình trên tập dữ liệu. |
| **III. LAB03 – PHÂN CỤM CƠ BẢN** | | | |
| 17 | 3.1.1 Ôn tập lý thuyết – K-means | **Trịnh Nhất Mạnh** | Mô tả quy trình & công thức. |
| 18 | 3.1.2 Bài tập thực hành 1 – Chim cánh cụt | **Nguyễn Thanh Trúc** | Thực hành & vẽ biểu đồ phân cụm. |
| 19 | 3.1.3 Bài tập thực hành 2 – Dữ liệu siêu thị | **Trần Hải Minh** | Phân tích kết quả, nhận xét nhóm khách hàng. |
| 20 | 3.2.1 Ôn tập lý thuyết – Phân cụm đa cấp | **Trịnh Nhất Mạnh** | Lý thuyết Hierarchical Clustering. |
| 21 | 3.2.2–3.2.3 Bài tập thực hành 1–2 | **Lê Trần Khánh Linh** | Thực hành mô hình & so sánh với K-means. |
| **IV. NHIỆM VỤ CHUNG VÀ TỔNG HỢP** | | | |
| 22 | Tổng hợp nội dung, định dạng file Word | **Lê Trần Khánh Linh** | Nhóm trưởng – chịu trách nhiệm tổng hợp & trình bày. |
| 23 | Tổng hợp, làm sạch và sắp xếp code (Python notebook) | **Lê Trần Khánh Linh** | Gom code các lab, kiểm tra lỗi, format. |
| 24 | Kiểm tra, chỉnh sửa cuối cùng | **Cả nhóm** | Rà soát trước khi nộp. |

# LAB01 – PHÂN TÍCH KHÁM PHÁ DỮ LIỆU

## 1. Thống kê mô tả

### 1.1. Ôn tập lý thuyết

#### 1.1.1. Thống kê mô tả và thống kê suy luận

Thống kê mô tả (descriptive statistics) nhằm tóm tắt và mô tả đặc tính của tập dữ liệu hiện có bằng bảng, đồ thị và các số đo thống kê (ví dụ: trung bình, trung vị, độ lệch chuẩn). Thống kê suy luận (inferential statistics) sử dụng thông tin từ mẫu để suy ra hoặc kiểm định giả thuyết về tổng thể thông qua các phương pháp như ước lượng khoảng tin cậy và kiểm định giả thuyết. Về bản chất, thống kê mô tả trả lời “dữ liệu như thế nào?”, trong khi thống kê suy luận trả lời “ta có thể kết luận gì về tổng thể?”.

#### 1.1.2. Các thước đo trung tâm và phân tán - ý nghĩa và ứng dụng

* Trung bình (mean): Phản ánh giá trị trung bình của dữ liệu, phù hợp khi phân bố gần chuẩn và không có nhiều ngoại lai.
* Trung vị (median, Q2): giá trị chia dữ liệu làm đôi (50%). Trung vị ổn định trước ảnh hưởng của ngoại lai và phân bố lệch.
* Mode (giá trị xuất hiện nhiều nhất): hữu ích cho dữ liệu phân loại.
* Phương sai & Độ lệch chuẩn (variance, SD): đo mức độ phân tán quanh trung bình; độ lệch chuẩn biểu diễn đơn vị giống dữ liệu.
* Phạm vi (range): Đơn giản nhưng nhạy cảm với ngoại lai.

Khi nào dùng trung vị thay cho trung bình? - Chọn trung vị khi dữ liệu có ngoại lai rõ rệt hoặc phân bố lệch (skewed), vì trung vị ít bị ảnh hưởng bởi các giá trị cực trị.

#### 1.1.3. Xác định dạng phân bố của dữ liệu

* Quan sát đồ họa: histogram (biểu đồ tần suất), kernel density plot, Q-Q plot (so sánh với phân bố chuẩn).
* Kiểm định thống kê: ví dụ kiểm định Shapiro-Wilk để kiểm tra tính chuẩn; lưu ý kiểm định nhạy với kích thước mẫu (mẫu lớn dễ phát hiện sai lệch nhỏ).
* Các dạng phân bố thường gặp: phân bố chuẩn (symmetrical), lệch phải (positive skew), lệch trái (negative skew), phân bố nhiều đỉnh (multimodal).

#### 1.1.4. Phân vị, IQR và boxplot

* Q1, Q2, Q3 lần lượt là phân vị 25%, 50% (trung vị), 75%.
* IQR (Interquartile Range) = Q3 − Q1; đo độ phân tán của phần giữa 50% dữ liệu.
* Boxplot biểu diễn Q1, Q2, Q3, “whiskers” (thường kéo tới Q1 − 1.5·IQR và Q3 + 1.5·IQR) và các điểm được coi là ngoại lai.

#### 1.1.5. Ý nghĩa độ lệch chuẩn và phạm vi

* Phạm vi cho biết biên độ toàn bộ dữ liệu nhưng bị chi phối bởi ngoại lai.
* Độ lệch chuẩn phản ánh sự dao động trung bình quanh trung bình mẫu; dùng tốt cho dữ liệu có phân bố tương đối đối xứng.

#### 1.1.6. Xử lý giá trị thiếu (missing values)

* Nhận dạng cơ chế thiếu: MCAR (Missing Completely at Random), MAR (Missing at Random), MNAR (Missing Not at Random) - phân biệt cơ chế này giúp chọn phương pháp xử lý phù hợp.
* Các phương pháp xử lý:

Loại bỏ quan sát thiếu (listwise deletion) - đơn giản nhưng có thể làm mất công suất nếu tỉ lệ thiếu lớn.

Điền thay thế (imputation): giá trị tĩnh (mean/median/mode), điền theo mô hình (regression imputation), hoặc phương pháp đa giá trị (multiple imputation) để bảo toàn biến động.

* Nguyên tắc báo cáo: luôn ghi rõ tỉ lệ missing theo biến và phương pháp xử lý đã áp dụng.

#### 1.1.7. Đọc và diễn giải histogram và boxplot

* Histogram: mô tả hình dạng phân bố (độ lệch, độ dốc, đa đỉnh), giúp phát hiện skewness và multimodality.
* Boxplot: tập trung vào phân vị, IQR và ngoại lai; thuận tiện để so sánh phân bố giữa các nhóm.
* Kết hợp: dùng cả hai để có cái nhìn đầy đủ: histogram cho hình dạng tổng thể, boxplot cho tóm tắt phân vị và ngoại lai.

#### 1.1.8. Phát hiện và xử lý ngoại lai (outliers)

* Phát hiện: quy tắc IQR (ngoại lai nếu < Q1 − 1.5·IQR hoặc > Q3 + 1.5·IQR), z-score (|z|>3), hoặc phương pháp đồ họa.
* Xử lý:
  1. Kiểm tra nguyên nhân (lỗi ghi nhận, lỗi nhập liệu, hay giá trị hợp lệ nhưng bất thường).
  2. Nếu do lỗi: chỉnh sửa hoặc loại bỏ.
  3. Nếu giá trị hợp lý: cân nhắc giữ lại nhưng báo cáo chi tiết; hoặc áp dụng kỹ thuật làm suy giảm ảnh hưởng (log-transform, winsorizing) hoặc dùng phương pháp thống kê “robust” (ví dụ dùng trung vị và IQR thay cho trung bình và SD).
* Nguyên tắc báo cáo: mọi quyết định về ngoại lai phải được mô tả rõ ràng trong phần phương pháp (số lượng, tiêu chí phát hiện, xử lý).

### 1.2. Bài tập thực hành 1 - Thống kê mô tả trên tập dữ liệu *Wine Quality (red)*

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>

#### 1.2.1. Mô tả dữ liệu

* Nguồn & kích thước: Tập dữ liệu wine quality (red) gồm 1599 mẫu; không phát hiện giá trị bị thiếu trong dữ liệu gốc.
* Biến mục tiêu: *quality* (điểm rời rạc từ 3 đến 8; tập trung mạnh ở 5–6; mean = 5.64, median = 6).
* Các biến đầu vào (11 biến hóa–lý) được nhóm như sau:

Yếu tố độ axit: *fixed acidity, volatile acidity, citric acid, pH*.

Phụ gia & thành phần: *residual sugar, chlorides, sulphates, free sulfur dioxide, total sulfur dioxide*.

Tính chất vật lý: *density, alcohol*.

***Bảng 1. Thống kê mô tả tóm tắt***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | count | mean | std | min | 25% | 50% | 75% | max |
| fixed acidity | 1599.0 | 8.319637 | 1.741096 | 4.60000 | 7.1000 | 7.90000 | 9.200000 | 15.90000 |
| volatile acidity | 1599.0 | 0.527821 | 0.179060 | 0.12000 | 0.3900 | 0.52000 | 0.640000 | 1.58000 |
| citric acid | 1599.0 | 0.270976 | 0.194801 | 0.00000 | 0.0900 | 0.26000 | 0.420000 | 1.00000 |
| residual sugar | 1599.0 | 2.538806 | 1.409928 | 0.90000 | 1.9000 | 2.20000 | 2.600000 | 15.50000 |
| chlorides | 1599.0 | 0.087467 | 0.047065 | 0.01200 | 0.0700 | 0.07900 | 0.090000 | 0.61100 |
| free sulfur dioxide | 1599.0 | 15.874922 | 10.460157 | 1.00000 | 7.0000 | 14.00000 | 21.000000 | 72.00000 |
| total sulfur dioxide | 1599.0 | 46.467792 | 32.895324 | 6.00000 | 22.0000 | 38.00000 | 62.000000 | 289.00000 |
| density | 1599.0 | 0.996747 | 0.001887 | 0.99007 | 0.9956 | 0.99675 | 0.997835 | 1.00369 |
| pH | 1599.0 | 3.311113 | 0.154386 | 2.74000 | 3.2100 | 3.31000 | 3.400000 | 4.01000 |
| sulphates | 1599.0 | 0.658149 | 0.169507 | 0.33000 | 0.5500 | 0.62000 | 0.730000 | 2.00000 |
| alcohol | 1599.0 | 10.422983 | 1.065668 | 8.40000 | 9.5000 | 10.20000 | 11.100000 | 14.90000 |
| quality | 1599.0 | 5.636023 | 0.807569 | 3.00000 | 5.0000 | 6.00000 | 6.000000 | 8.00000 |

#### 1.2.2. Diễn giải theo nhóm biến

A. Yếu tố độ axit

* *volatile acidity* (mean ≈ 0.528, median ≈ 0.52): phân bố tương đối đối xứng nhưng tồn tại quan sát lớn (max = 1.58) – có khả năng ảnh hưởng tiêu cực đến cảm quan (vị giấm), cần kiểm tra kỹ.
* *fixed acidity* (mean 8.32 > median 7.9): lệch phải nhẹ → một số mẫu có độ axit cố định cao.
* *pH* ổn định (mean ≈ 3.31, sd ≈ 0.154) → ít biến động giữa mẫu.

B. Phụ gia & thành phần khác

* *free/total sulfur dioxide*: phân bố lệch phải rõ (total SO₂ median = 38, max = 289) → tồn tại ngoại lai lớn.
* *chlorides* và *residual sugar* cũng cho thấy đuôi phải dài (max tương ứng 0.611 và 15.5) → một số mẫu khác biệt mạnh so với phần lớn dữ liệu.  
  Hệ quả: nhóm này có skewness dương; các biến này thường cần biến đổi (log/winsorize) khi đưa vào mô hình.

C. Tính chất vật lý

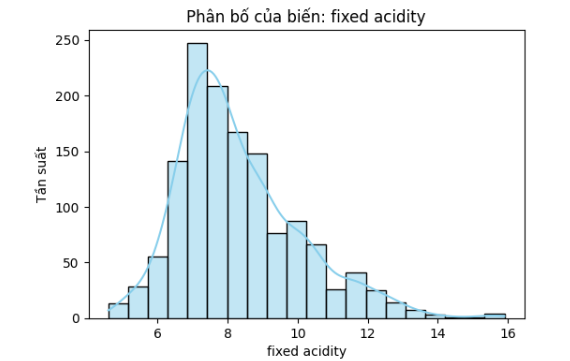
* *density* có phương sai rất nhỏ → khả năng đóng góp thông tin phân biệt thấp.
* *alcohol* có phạm vi rộng và phân bố tương đối đối xứng (mean ≈ 10.42, median = 10.2) → thường liên quan tích cực tới quality và là biến dự đoán tiềm năng.

D. Biến mục tiêu *quality*

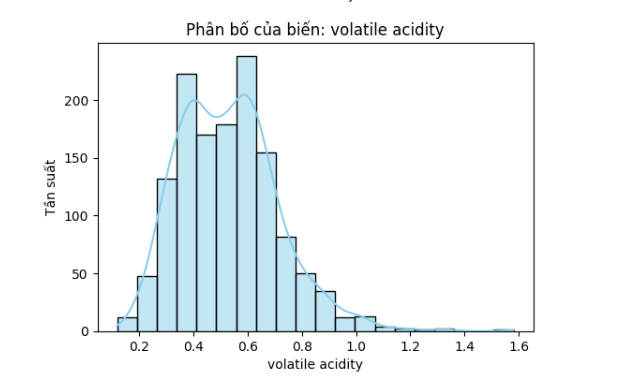
Phân bố rời rạc, tập trung ở 5–6; 75% phân vị = 6 → dataset bị mất cân bằng lớp (class imbalance), cần chú ý trong lựa chọn thuật toán và đánh giá.

#### 1.2.3. Kiểm tra phân bố của từng nhóm biến

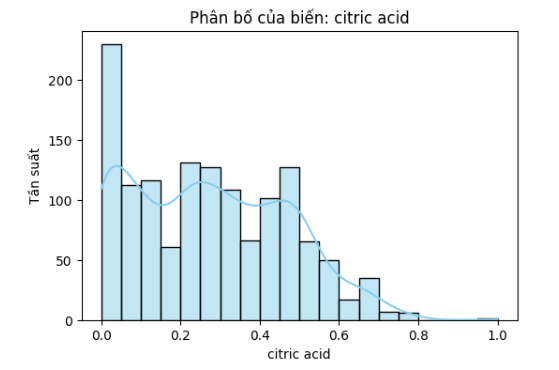
A. Yếu tố độ axit



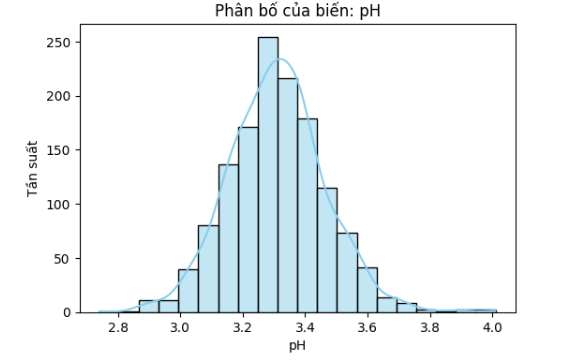
Fixed acidity: Phân bố gần chuẩn với dạng hình chuông rõ, đỉnh tập trung quanh 7.0–8.0. Tuy nhiên, đuôi phải kéo dài đến khoảng 15.9 cho thấy có một số quan sát giá trị cao bất thường. Các mẫu có độ axit >14 được xem là ngoại lai tiềm năng.



Volatile acidity: Phân bố đa đỉnh với hai đỉnh chính quanh 0.4 và 0.6, thể hiện khả năng tồn tại hai nhóm rượu khác biệt về mức độ axit bay hơi. Đuôi phải nhỏ kéo dài đến >1.0 xác nhận sự hiện diện của một số mẫu có vị chua gắt, thường liên quan đến chất lượng thấp.

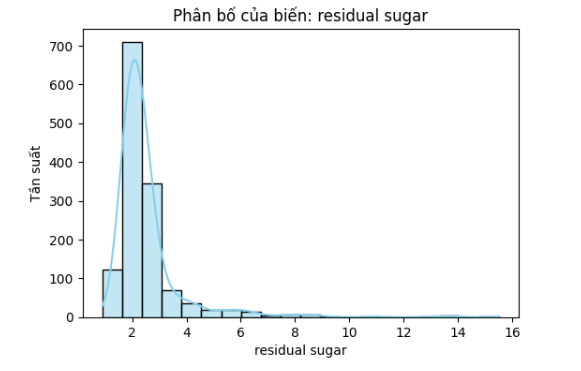


Citric acid: Phân bố không đối xứng, đỉnh lớn nhất tại 0.0. Nhiều mẫu có giá trị bằng 0 cho thấy axit citric được bổ sung không đồng nhất giữa các nhà sản xuất. Phân bố phân tán và nhiều đỉnh phụ, phản ánh tính biến thiên cao của thành phần này.

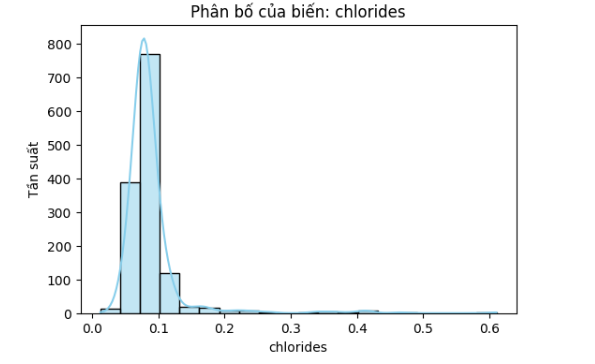


pH: Phân bố đối xứng, gần chuẩn, tập trung mạnh tại khoảng 3.3–3.4. Điều này thể hiện độ ổn định cao của độ axit tổng thể giữa các mẫu rượu.

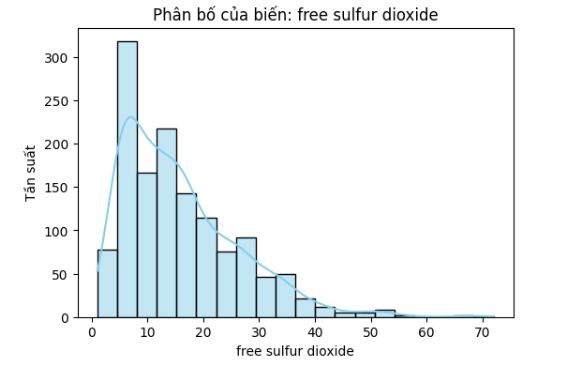
B. Phụ gia & thành phần khác



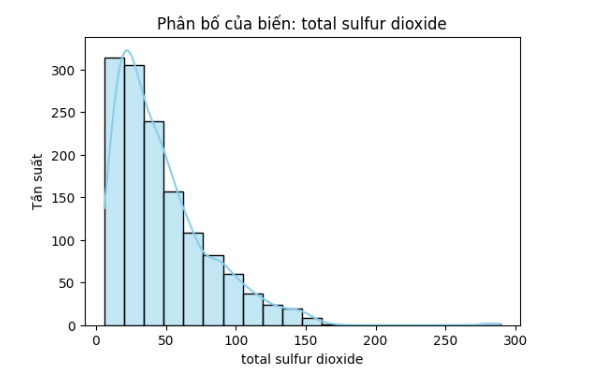
Residual sugar: Lệch phải rõ rệt, phần lớn mẫu tập trung trong khoảng 1.0–2.0. Đuôi kéo dài đến 15.5 phản ánh một vài mẫu rượu ngọt hoặc giá trị cực trị. Dữ liệu này nên được biến đổi log để giảm độ lệch.



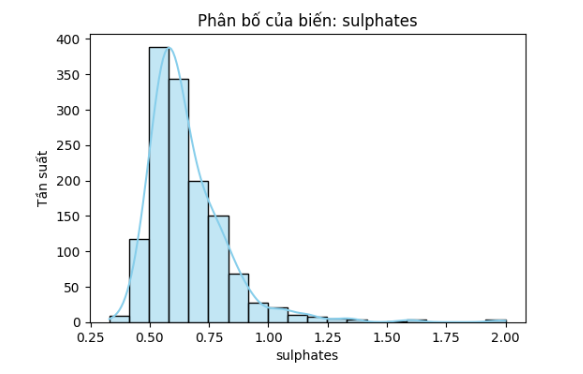
Chlorides: Phân bố lệch phải mạnh, tập trung ở 0.07–0.10. Các giá trị vượt 0.2 là ngoại lai tiềm năng, có thể do sự khác biệt trong quá trình lên men hoặc nguồn nguyên liệu.



Free sulfur dioxide: Lệch phải, đỉnh tần suất cao ở khoảng 5–10 và đuôi kéo dài đến 72. Sự biến động lớn này cho thấy mức sử dụng SO₂ tự do không đồng nhất, cần được chuẩn hóa hoặc biến đổi trước khi mô hình hóa.

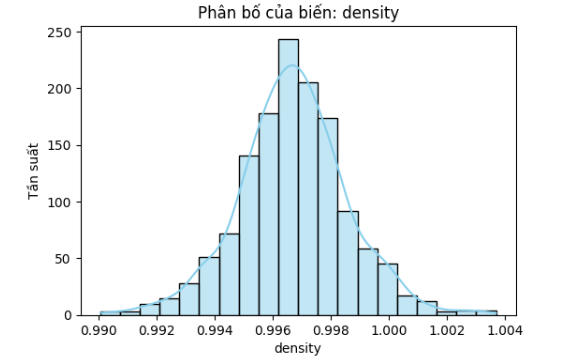


Total sulfur dioxide: Lệch phải rất mạnh, tần suất cao nhất ở 20–40 nhưng kéo dài đến 289. Mức độ biến thiên lớn và sự xuất hiện của ngoại lai cực trị cho thấy cần biến đổi log hoặc áp dụng phương pháp winsorizing để giảm ảnh hưởng cực trị.

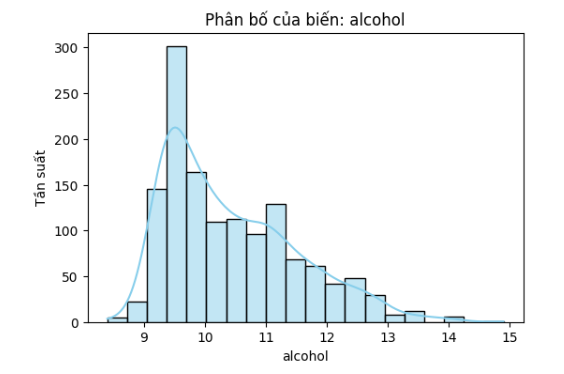


Sulphates: Phân bố lệch phải nhẹ, đỉnh chính ở khoảng 0.5–0.6 và đuôi kéo dài đến >1.0. Một số mẫu có giá trị cao bất thường có thể ảnh hưởng đến các giả định chuẩn trong mô hình thống kê.

C. Tính chất vật lý

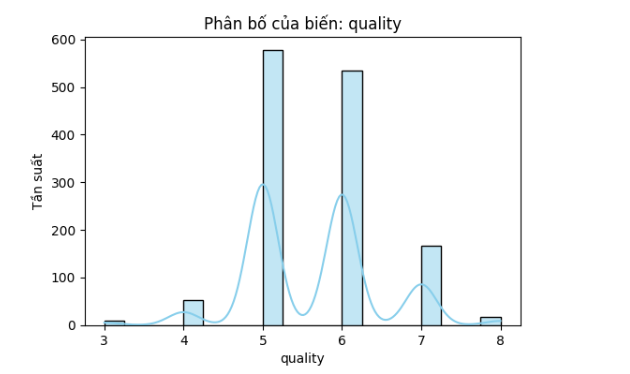


Density: Phân bố đối xứng gần chuẩn, tập trung mạnh quanh 0.996–0.997. Biến này có độ biến thiên rất nhỏ, do đó khả năng đóng góp phân biệt trong mô hình thấp.



Alcohol: Phân bố đa đỉnh nhẹ với hai đỉnh tại khoảng 9.0–10.0% và 11.0–12.0%, thể hiện sự tồn tại của hai nhóm rượu vang theo phong cách hoặc nồng độ cồn khác nhau. Nồng độ cồn có xu hướng tương quan dương với chất lượng cảm quan.

D. Biến mục tiêu *quality*



quality là biến rời rạc, phân bố đa đỉnh với tần suất cao nhất ở mức 5 và 6. Các mức 3–4 (chất lượng thấp) và 7–8 (chất lượng cao) chiếm tỷ lệ nhỏ, thể hiện hiện tượng mất cân bằng lớp (class imbalance) rõ rệt. Sự chênh lệch này cần được lưu ý trong giai đoạn mô hình hóa.

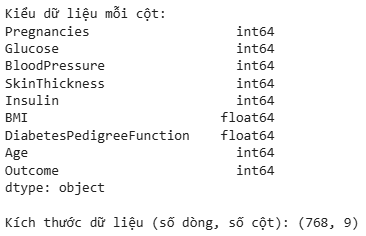
**⇒** Kết quả kiểm tra phân bố cho thấy hầu hết các biến hóa–lý có độ lệch phải (positive skewness), đặc biệt là nhóm SO₂, chlorides và residual sugar. Các biến này nên được biến đổi log hoặc chuẩn hóa để đáp ứng giả định của các mô hình tuyến tính. Ngược lại, pH và density có phân bố gần chuẩn và độ biến thiên thấp, trong khi alcohol thể hiện tiềm năng cao trong việc dự đoán chất lượng rượu.

### 1.3. Bài tập thực hành 2 - Tập dữ liệu *Pima Indians Diabetes*

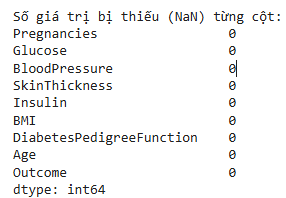
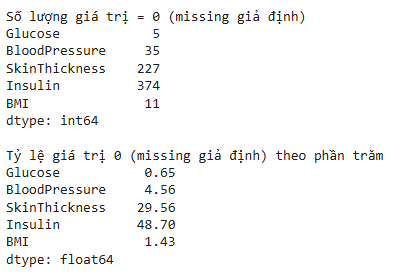
Thực hiện thống kê mô tả trên tập dữ liệu về bệnh tiểu đường. Dữ liệu lấy tại <https://www.kaggle.com/code/vincentlugat/pima-indians-diabetes-eda-prediction-0-906>

#### 1.3.1. Mô tả và kiểm tra dữ liệu

Tập dữ liệu Pima Indians Diabetes gồm 768 quan sát và 9 biến, trong đó có 8 biến định lượng và 1 biến phân loại *Outcome*.

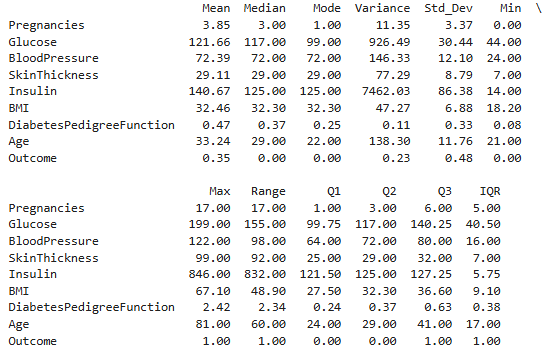
Các biến bao gồm:

* Pregnancies: Số lần mang thai của bệnh nhân.
* Glucose: Nồng độ glucose trong máu (mg/dL).
* BloodPressure: Huyết áp tâm trương (mmHg).
* SkinThickness: Độ dày lớp da (mm).
* Insulin: Nồng độ insulin trong máu (μU/ml).
* BMI: Chỉ số khối cơ thể (kg/m²).
* DiabetesPedigreeFunction: Chỉ số phản ánh yếu tố di truyền của bệnh tiểu đường.
* Age: Tuổi của bệnh nhân (năm).
* Outcome: Kết quả chẩn đoán — 0 = không mắc tiểu đường, 1 = mắc tiểu đường.

Kết quả kiểm tra dữ liệu:

* Không có bản ghi trùng lặp và không có giá trị bị thiếu trực tiếp trong dữ liệu.
* Tuy nhiên, 5 biến gồm Glucose, BloodPressure, SkinThickness, Insulin và BMI xuất hiện giá trị bằng 0, điều này không hợp lý về mặt y học (vì các chỉ số này không thể thực sự bằng 0).
* Do đó, các giá trị này được xem là giá trị thiếu giả định (missing value) và được thay thế bằng trung vị (median) của từng biến tương ứng.
* Việc sử dụng trung vị giúp giảm ảnh hưởng của các giá trị ngoại lai, đồng thời giữ nguyên đặc trưng trung tâm của phân bố dữ liệu, phù hợp với các biến có phân phối lệch.

#### 1.3.2. Thống kê mô tả

Sau khi tiền xử lý, dữ liệu được thống kê mô tả bằng các thước đo trung tâm và phân tán, bao gồm trung bình (mean), trung vị (median), phương sai, độ lệch chuẩn (std), giá trị cực trị (min–max), các phần tử phân vị (Q1, Q2, Q3) và độ lệch tứ phân vị (IQR).

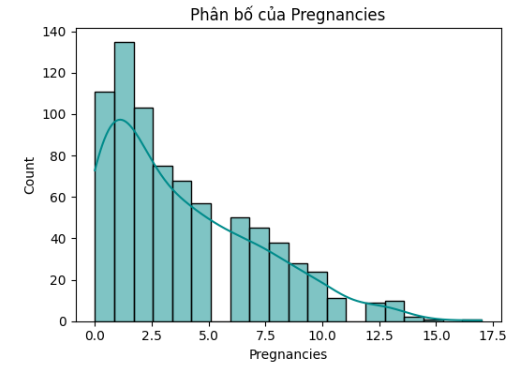
Kết quả chính:

* Glucose: mean = 121.66, median = 117, std = 30.44 → phân bố lệch phải nhẹ, cho thấy có nhiều bệnh nhân có chỉ số đường huyết cao.
* Insulin: mean = 149.68, std = 86.34, range = 15–846 → biến động rất lớn, xuất hiện nhiều giá trị ngoại lai.
* BMI: mean ≈ median ≈ 32.3 → phân bố tương đối đối xứng, phản ánh sự đồng đều về chỉ số khối cơ thể.
* Age: mean = 33.24 → phần lớn người trong mẫu thuộc nhóm trưởng thành, ít người cao tuổi.
* Outcome: mean = 0.35 → khoảng 35% đối tượng trong mẫu mắc bệnh tiểu đường.

Nhìn tổng thể, Glucose, Insulin và BMI thể hiện mức biến động cao nhất — đây cũng là những yếu tố sinh học có khả năng tương quan mạnh với kết quả chẩn đoán (Outcome).

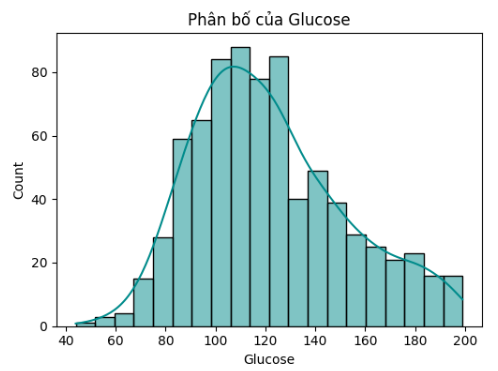
#### 1.3.3. Trực quan hoá và diễn giải

Kết quả trực quan hóa cho thấy phần lớn các biến trong tập dữ liệu có phân bố lệch phải (positively skewed) – đặc trưng phổ biến của dữ liệu y học, khi hầu hết giá trị tập trung ở mức trung bình hoặc thấp, trong khi một số ít quan sát có giá trị rất cao. Cụ thể:



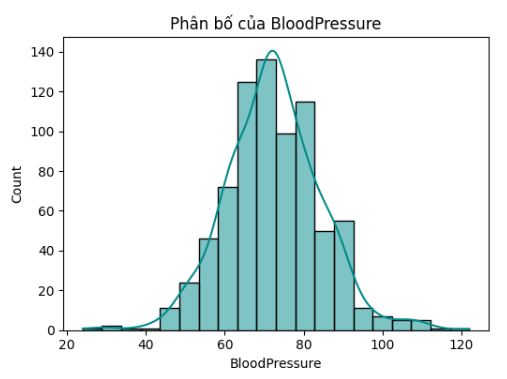
Pregnancies (Số lần mang thai):

Phân bố lệch phải rõ rệt, với phần lớn giá trị nằm trong khoảng 0–3. Số cá nhân có hơn 10 lần mang thai chiếm tỷ lệ rất nhỏ. Điều này phản ánh đặc điểm nhân khẩu học thực tế của mẫu — đa số phụ nữ trong nghiên cứu chỉ mang thai ít lần.



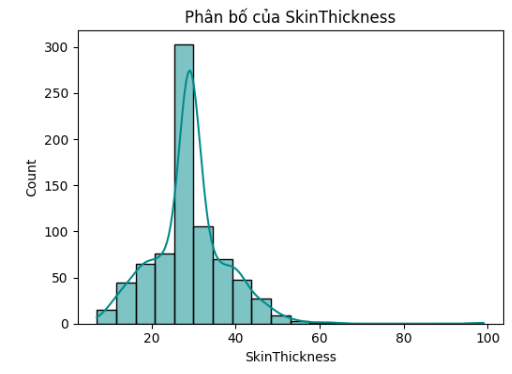
Glucose (Đường huyết):

Phân bố gần chuẩn nhưng hơi lệch phải, tập trung chủ yếu trong khoảng 90–140 mg/dL. Các giá trị lớn hơn 180 đại diện cho nhóm có đường huyết cao bất thường, nhiều khả năng thuộc nhóm mắc tiểu đường hoặc tiền tiểu đường. Đây là biến có độ phân tán cao và ý nghĩa lâm sàng mạnh.



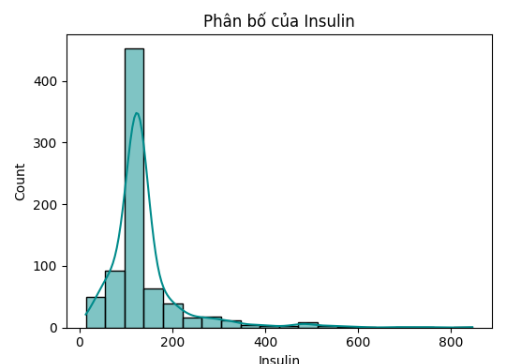
BloodPressure (Huyết áp):

Phân bố gần đối xứng, tập trung trong khoảng 60–80 mmHg — mức huyết áp bình thường ở người khỏe mạnh. Một số giá trị thấp (<50) hoặc cao (>100) có thể phản ánh sự khác biệt sinh lý hoặc sai lệch khi đo.



SkinThickness (Độ dày lớp da):

Phân bố lệch phải đáng kể, phần lớn giá trị trong khoảng 20–30 mm. Một số quan sát vượt 60 mm được xem là ngoại lai. Vì biến này phản ánh lượng mỡ dưới da, sự lệch phải là hợp lý, do chỉ một số cá nhân có độ dày da cao bất thường.



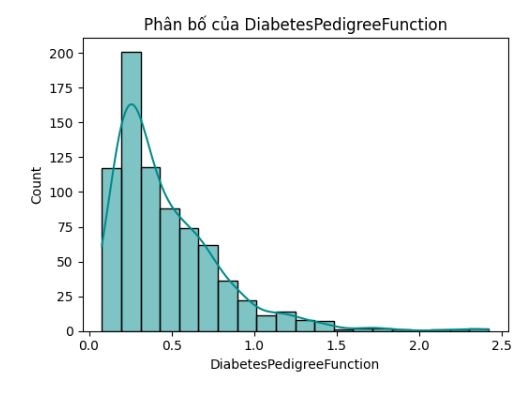
Insulin (Nồng độ insulin):

Lệch phải rất mạnh, phần lớn tập trung ở mức 50–200 μU/ml, trong khi một số giá trị vượt 600–800 cho thấy sự tồn tại của ngoại lai cực trị. Điều này khẳng định sự cần thiết của bước tiền xử lý bằng cách thay thế giá trị 0 và chuẩn hóa dữ liệu.



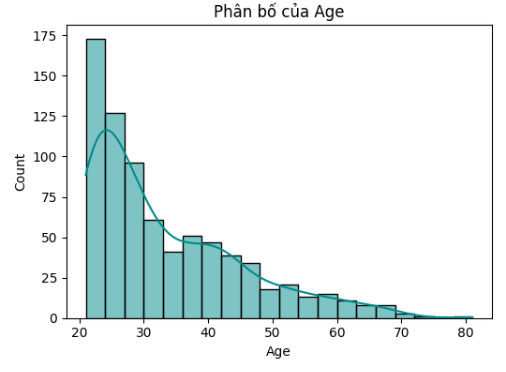
BMI (Chỉ số khối cơ thể):

Phân bố tương đối đối xứng, chủ yếu trong khoảng 25–35 kg/m². Đa số cá nhân trong mẫu nằm trong nhóm thừa cân hoặc béo phì — yếu tố nguy cơ đã được chứng minh có liên hệ mật thiết với bệnh tiểu đường type 2.



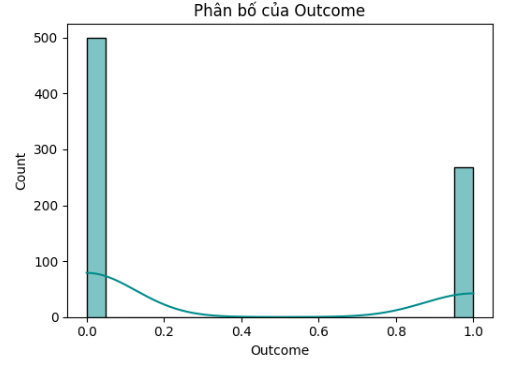
DiabetesPedigreeFunction (Chỉ số di truyền):

Lệch phải mạnh, đa số giá trị < 0.6, chỉ một số ít trường hợp vượt 1.5. Điều này cho thấy phần lớn người tham gia có nguy cơ di truyền thấp, tuy nhiên vẫn tồn tại vài cá nhân có nền tảng di truyền đáng kể.



Age (Tuổi):

Phân bố lệch phải, tập trung nhiều ở nhóm 20–40 tuổi và giảm dần ở nhóm cao tuổi (>60). Mẫu dữ liệu chủ yếu bao gồm người trưởng thành và trung niên, phản ánh cấu trúc dân số nghiên cứu phổ biến.



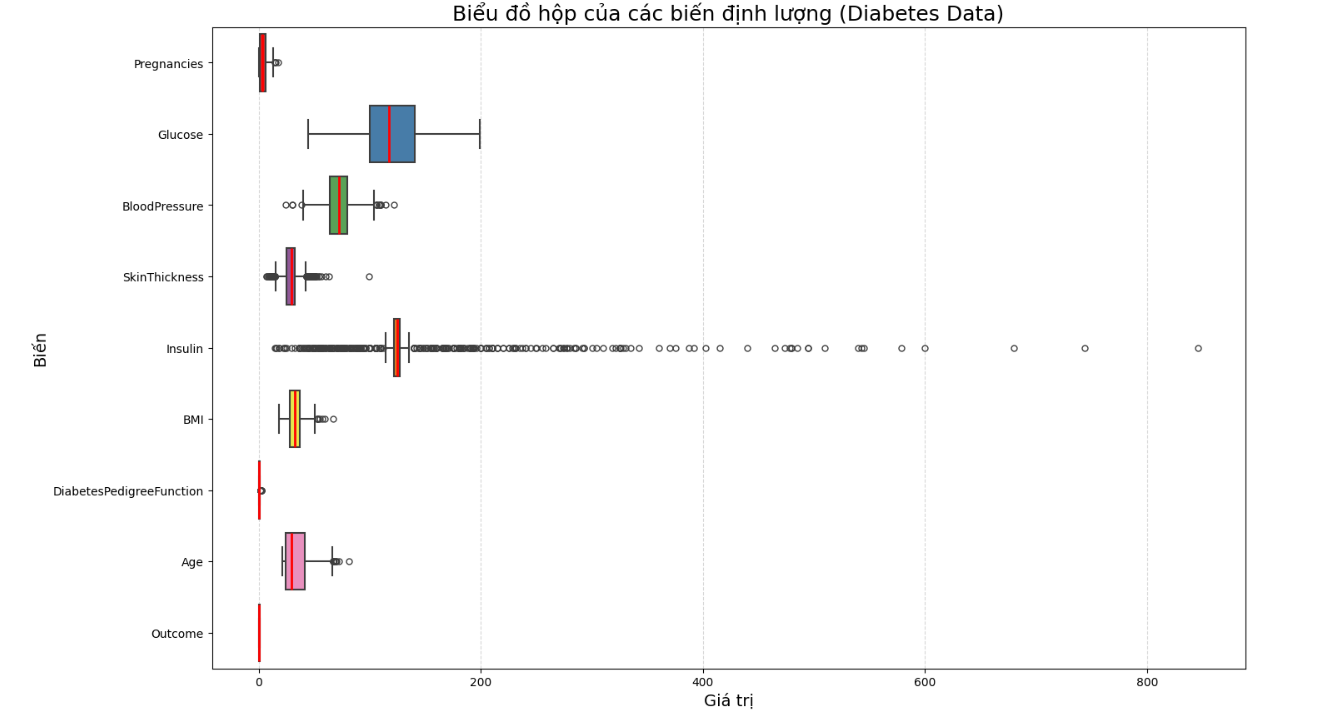
Outcome (Kết quả chẩn đoán):

Là biến nhị phân với hai giá trị 0 (không tiểu đường) và 1 (tiểu đường). Khoảng 65% người tham gia thuộc nhóm không mắc bệnh, 35% thuộc nhóm có bệnh. Phân bố này thể hiện sự mất cân bằng lớp (class imbalance), cần được lưu ý khi xây dựng mô hình dự đoán.

Từ kết quả biểu đồ cho thấy:

* Các biến Glucose, Insulin, và SkinThickness có phân bố lệch phải và xuất hiện nhiều ngoại lai, phản ánh đúng đặc trưng sinh học của dữ liệu y học.
* Các biến BMI và BloodPressure có phân bố gần chuẩn, thể hiện tính ổn định cao hơn giữa các cá nhân.
* Khi đối chiếu với Outcome, có thể nhận thấy Glucose và Insulin là hai yếu tố phân biệt rõ rệt nhất giữa nhóm mắc và không mắc bệnh.

Phân tích biểu đồ hộp (boxplot) củng cố kết luận này:



* Hầu hết các biến thể hiện phân bố lệch phải với nhiều điểm ngoại lai.
* Glucose và Insulin có số lượng ngoại lai cao nhất — phản ánh nhóm bệnh nhân có chỉ số đường huyết và nồng độ insulin vượt chuẩn.
* SkinThickness và Age cũng có vài ngoại lai đáng chú ý, trong khi BloodPressure, BMI và DiabetesPedigreeFunction tương đối ổn định.
* Tổng thể, dữ liệu cho thấy mức độ phân tán trung bình – lệch phải rõ rệt – phù hợp với đặc điểm lâm sàng của bệnh tiểu đường, đồng thời khẳng định vai trò trọng yếu của Glucose và Insulin trong việc phân biệt hai nhóm đối tượng.

## 2. Xử lý và trực quan hoá dữ liệu

### 2.1. Ôn tập lý thuyết

#### 2.1.1. Vai trò và tầm quan trọng của trực quan hóa dữ liệu trong EDA

Trực quan hóa dữ liệu (Data Visualization) là công cụ cốt lõi trong quá trình phân tích khám phá dữ liệu (Exploratory Data Analysis – EDA).  
Nó cho phép người phân tích hiểu nhanh cấu trúc, xu hướng và mối quan hệ tiềm ẩn trong dữ liệu thông qua biểu đồ, đồ thị, hoặc bản đồ trực quan.  
Cụ thể, trực quan hóa giúp:

* Phát hiện quy luật, mô hình, và xu hướng tiềm ẩn trong dữ liệu.
* Nhận diện bất thường, ngoại lai hoặc dữ liệu bị thiếu.
* Hỗ trợ kiểm tra giả định trước khi áp dụng mô hình thống kê hoặc máy học.
* Tăng tính hiệu quả truyền tải thông tin khi trình bày kết quả phân tích.

#### 2.2.2. Các loại biểu đồ phổ biến và ứng dụng

* Histogram: Thể hiện phân bố của biến số liên tục, giúp quan sát dạng phân bố (chuẩn, lệch, đa đỉnh...).
* Scatter Plot: Mô tả mối tương quan giữa hai biến định lượng.
* Boxplot: So sánh phân bố, phát hiện ngoại lai và sự chênh lệch giữa các nhóm.
* Bar Chart: Biểu diễn giá trị trung bình hoặc tần suất giữa các nhóm hoặc danh mục.
* Heatmap: Thể hiện mức độ tương quan giữa nhiều biến, thường dùng trong EDA tổng quát.

#### 2.2.3. Cách lựa chọn biểu đồ phù hợp với loại dữ liệu

* Dữ liệu định lượng (số): Dùng Histogram, Boxplot, Scatter Plot để thể hiện phân bố và mối quan hệ.
* Dữ liệu phân loại: Dùng Bar Chart hoặc Pie Chart để so sánh nhóm.
* Dữ liệu chuỗi thời gian: Dùng Line Chart hoặc Area Chart để thể hiện xu hướng theo thời gian.

#### 2.2.4. So sánh giữa các thư viện trực quan hóa trong Python

|  |  |  |  |
| --- | --- | --- | --- |
| **Thư viện** | **Đặc điểm nổi bật** | **Ưu điểm** | **Hạn chế** |
| **Matplotlib** | Thư viện nền tảng, cho phép tạo hầu hết các loại biểu đồ tĩnh. | Linh hoạt, tùy chỉnh cao. | Cú pháp dài, ít thân thiện. |
| **Seaborn** | Xây dựng trên Matplotlib, hỗ trợ biểu đồ thống kê đẹp và dễ dùng. | Thẩm mỹ cao, phù hợp cho EDA. | Ít tùy biến hơn Matplotlib. |
| **Plotly** | Tạo biểu đồ tương tác (interactive). | Phù hợp cho báo cáo và dashboard web. | Cấu hình phức tạp hơn. |

#### 2.2.5. Nguyên tắc thiết kế biểu đồ hiệu quả

* Giữ biểu đồ đơn giản, dễ đọc, tập trung vào thông tin chính.
* Đặt tiêu đề, nhãn trục và đơn vị đo rõ ràng.
* Màu sắc hài hòa, tránh gây nhiễu thị giác.
* Chọn loại biểu đồ phù hợp với mục tiêu phân tích.
* Hạn chế hiệu ứng đồ họa phức tạp hoặc thừa chi tiết.

#### 2.2.6. Ví dụ mã Python cơ bản bằng Matplotlib

*import matplotlib.pyplot as plt*

*# Histogram*

*plt.hist(data, bins=20, color='skyblue')*

*plt.title('Histogram')*

*plt.xlabel('Giá trị')*

*plt.ylabel('Tần suất')*

*plt.show()*

*# Bar chart*

*plt.bar(categories, values, color='orange')*

*plt.title('Biểu đồ cột')*

*plt.xlabel('Danh mục')*

*plt.ylabel('Giá trị')*

*plt.show()*

#### 2.2.7. Xuất biểu đồ sang các định dạng phổ biến

*plt.savefig("bieu\_do.png", dpi=300, bbox\_inches='tight') # Xuất PNG*

*plt.savefig("bieu\_do.pdf") # Xuất PDF*

*plt.savefig("bieu\_do.svg") # Định dạng vector*

Với Plotly, có thể xuất biểu đồ tương tác HTML bằng lệnh:

*fig.write\_html("bieu\_do.html")*

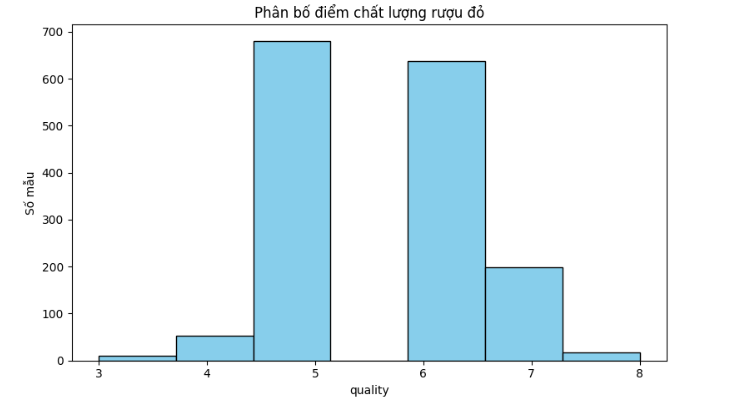
### 2.2. Bài tập thực hành 1 - Tập dữ liệu Red Wine Quality

#### 2.2.1. Chuẩn bị dữ liệu

* Tập dữ liệu: Wine Quality – Red Wine (Kaggle, Eisgandar 2023).
* Kích thước: 1599 mẫu quan sát, gồm 11 biến đầu vào và 1 biến mục tiêu (quality).
* Biến mục tiêu: quality – thang điểm 0–10, đánh giá cảm quan tổng thể.
* Kiểm tra dữ liệu: Không có giá trị bị thiếu hoặc trùng lặp.
* Các biến đặc trưng chính: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol.

#### 2.2.2. Trực quan hóa và diễn giải

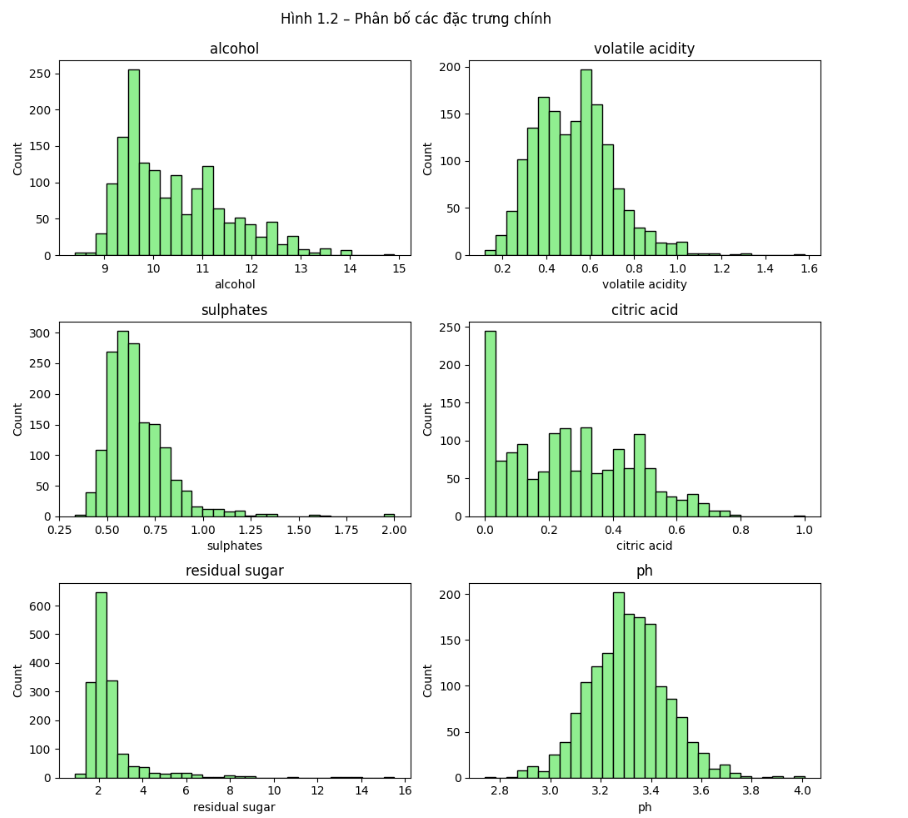
1. Phân bố điểm chất lượng (Quality Distribution)



Loại biểu đồ: Histogram – phù hợp cho biến định lượng rời rạc (quality).

* Đa số mẫu có điểm 5–6, phản ánh rượu vang chất lượng trung bình chiếm ưu thế.
* Rượu có quality ≥ 7 chiếm tỷ lệ nhỏ → nhóm rượu cao cấp hiếm.
* Phân bố lệch phải nhẹ, thể hiện sự tồn tại của ít mẫu đạt chất lượng vượt trội.

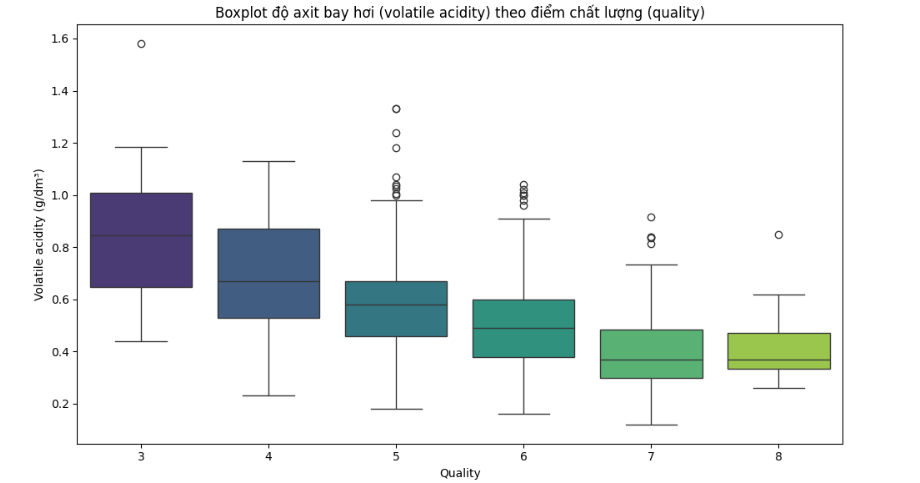
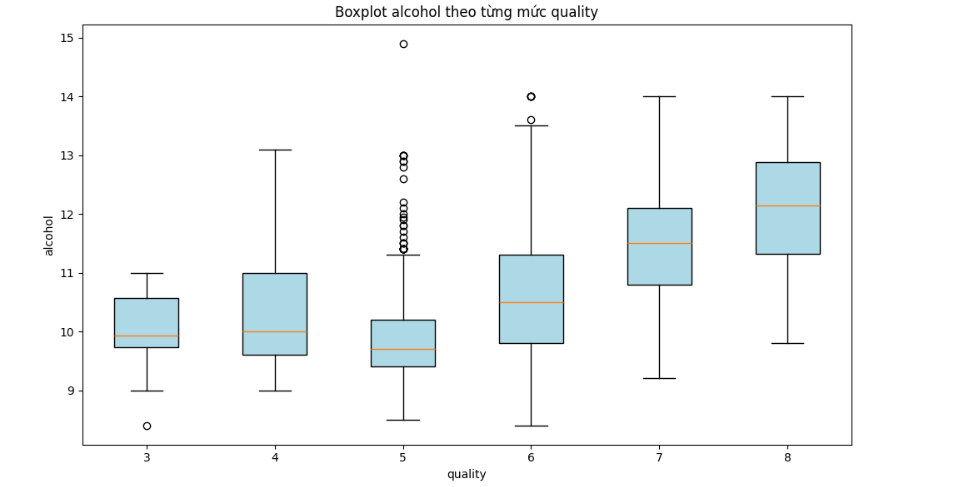
2. Phân bố các đặc trưng hóa học (Histogram – Univariate Analysis)



Loại biểu đồ: Histogram cho các biến định lượng.

* Alcohol: lệch phải nhẹ, tập trung khoảng 9–11% → phù hợp mức nồng độ tiêu chuẩn.
* Volatile acidity: lệch phải rõ, phần lớn dưới 0.6 g/dm³, thể hiện nhóm rượu đạt vị chua cân bằng.
* Sulphates & Citric acid: tập trung ở giá trị thấp → phần lớn rượu có phụ gia và axit hữu cơ ở mức nhẹ.

3. Boxplot theo nhóm chất lượng (Bivariate Analysis)



Loại biểu đồ: Boxplot – phù hợp so sánh phân bố giữa các nhóm (quality).

(a) Alcohol vs. Quality:

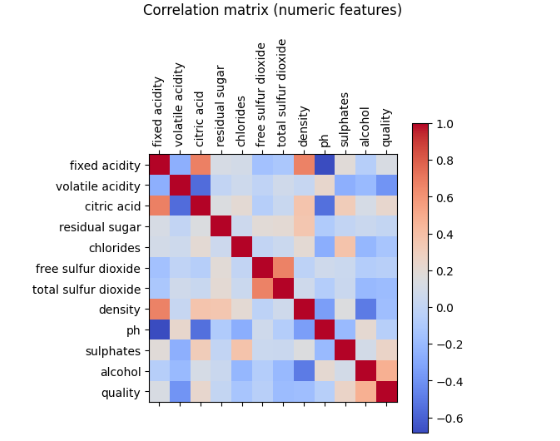
* Trung vị của alcohol tăng dần theo quality.
* Nhóm quality ≥ 7 có nồng độ cồn cao hơn đáng kể → cho thấy tương quan dương mạnh giữa alcohol và cảm quan.

(b) Volatile Acidity vs. Quality:

* Mức volatile acidity giảm khi quality tăng.
* Nhóm chất lượng thấp có độ phân tán lớn hơn → phản ánh quy trình sản xuất kém ổn định.

4. Ma trận tương quan (Correlation Heatmap)

Loại biểu đồ: Heatmap – sử dụng để đánh giá mức độ tương quan tuyến tính giữa các biến định lượng.



* Alcohol có tương quan dương mạnh với quality.
* Volatile acidity tương quan âm rõ rệt với quality.
* Density và alcohol tương quan âm cao, phản ánh quy luật vật lý pha loãng.
* free sulfur dioxide và total sulfur dioxide có tương quan cao → cảnh báo hiện tượng đa cộng tuyến.

***Kết luận***

* Các biểu đồ trực quan đã xác nhận vai trò của các biến hóa–lý chính trong việc ảnh hưởng đến chất lượng rượu vang đỏ.
* Những yếu tố có tác động mạnh gồm:

Tác động tích cực: Alcohol, Sulphates, Citric Acid.

Tác động tiêu cực: Volatile Acidity, Density.

* Việc sử dụng histogram, boxplot, và heatmap giúp:

Khái quát được cấu trúc dữ liệu và xu hướng phân bố.

Phát hiện mối quan hệ quan trọng giữa các biến đầu vào và chất lượng cảm quan (quality).

Chuẩn bị cơ sở cho bước mô hình hóa dự đoán (predictive modeling) sau này.

### 2.3. Bài tập thực hành 2 - Tập dữ liệu *Pima Indians Diabetes (1)*

#### 2.3.1. Mô tả và tiền xử lý dữ liệu

Bộ dữ liệu được sử dụng là Pima Indians Diabetes Dataset, chứa thông tin y tế của 768 phụ nữ gốc Pima (Mỹ) nhằm dự đoán khả năng mắc bệnh tiểu đường.  
Tập dữ liệu gồm 9 biến, trong đó có 8 biến định lượng và 1 biến phân loại (Outcome).

Kiểm tra dữ liệu ban đầu cho thấy không có bản ghi trùng lặp. Tuy nhiên, một số biến có chứa giá trị bằng 0 không hợp lý về mặt sinh lý, được xem là giá trị thiếu giả định. Cụ thể như sau:

|  |  |
| --- | --- |
| Cột | Số lượng 0 |
| Pregnancies | 0 |
| Glucose | 5 |
| BloodPressure | 35 |
| SkinThickness | 227 |
| Insulin | 374 |
| BMI | 11 |
| Diabetes Pedigree Function | 0 |
| Age | 0 |
| Outcome | 0 |

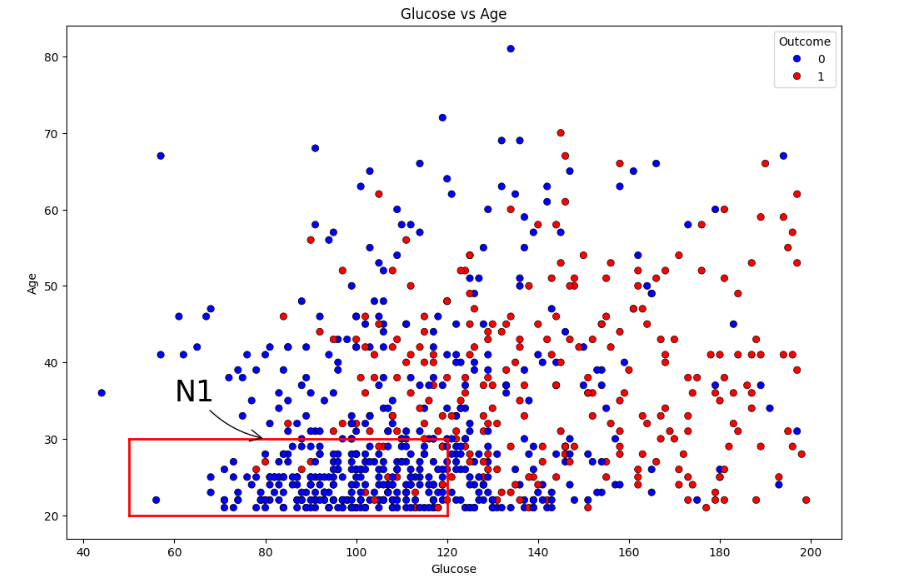
Để đảm bảo tính chính xác trong phân tích, các giá trị 0 này được thay thế bằng trung vị (median) của từng biến tương ứng. Phương pháp sử dụng trung vị giúp:

* Giảm ảnh hưởng của ngoại lai (outliers).
* Bảo toàn đặc trưng trung tâm của phân bố dữ liệu.
* Phù hợp với bản chất dữ liệu y học vốn thường có phân bố lệch phải.

Sau khi xử lý, toàn bộ dữ liệu được chuẩn hóa bằng StandardScaler nhằm đưa các biến về cùng thang đo.

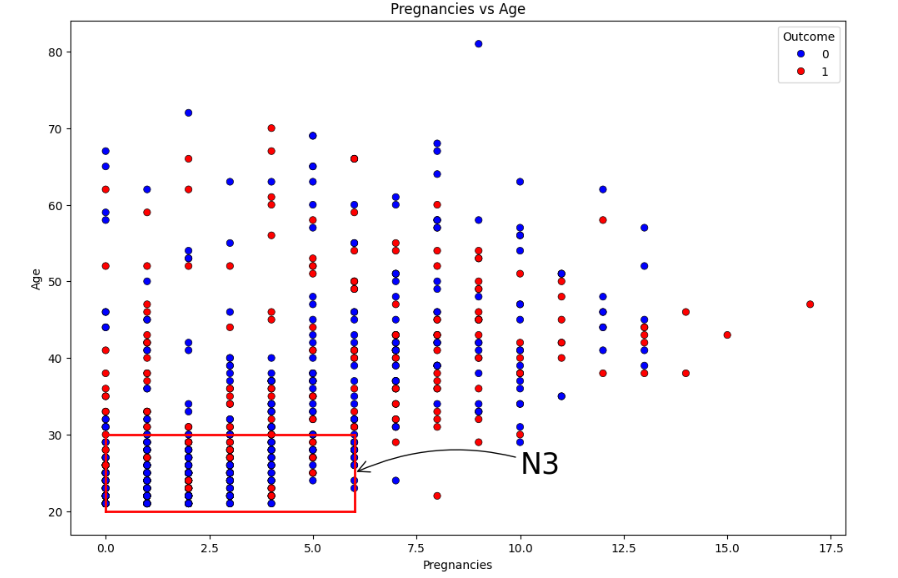
#### 2.3.2. Trực quan hóa và diễn giải

1. Biểu đồ tán xạ “Glucose vs Age”



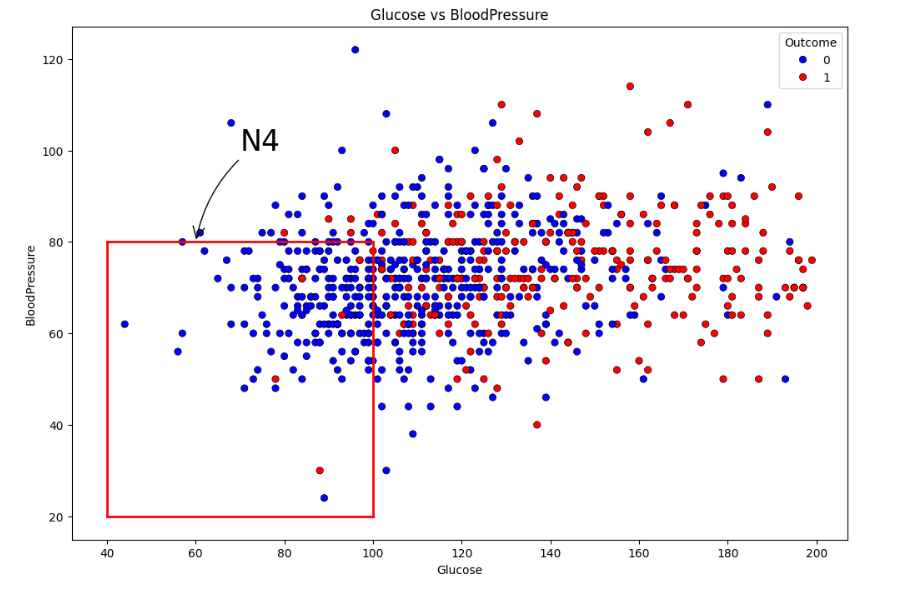
Biểu đồ thể hiện mối quan hệ giữa mức đường huyết (Glucose) và độ tuổi (Age).  
Quan sát cho thấy người cao tuổi có xu hướng có mức glucose cao hơn, và các trường hợp mắc tiểu đường (màu vàng) tập trung nhiều ở vùng glucose > 140 mg/dL.  
Điều này phù hợp với lý thuyết y học rằng tuổi tác là yếu tố nguy cơ làm giảm hiệu quả chuyển hóa insulin, dẫn đến gia tăng khả năng mắc bệnh tiểu đường tuýp 2.

2. Biểu đồ tán xạ “Pregnancies vs Age”



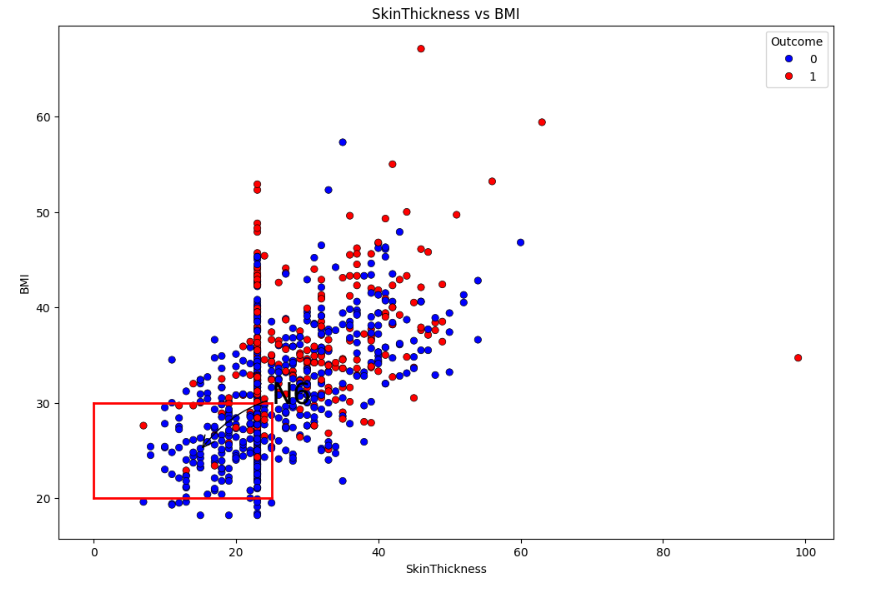
Biểu đồ cho thấy mối quan hệ giữa số lần mang thai (Pregnancies) và độ tuổi (Age).  
Khi tuổi và số lần mang thai tăng, tỷ lệ mắc tiểu đường (thể hiện qua mật độ điểm vàng) cũng tăng rõ rệt.  
Điều này phản ánh nguy cơ tiểu đường thai kỳ và ảnh hưởng nội tiết tố lâu dài có thể làm tăng khả năng phát triển bệnh ở phụ nữ có nhiều lần sinh nở.

3. Biểu đồ tán xạ “Glucose vs BloodPressure”



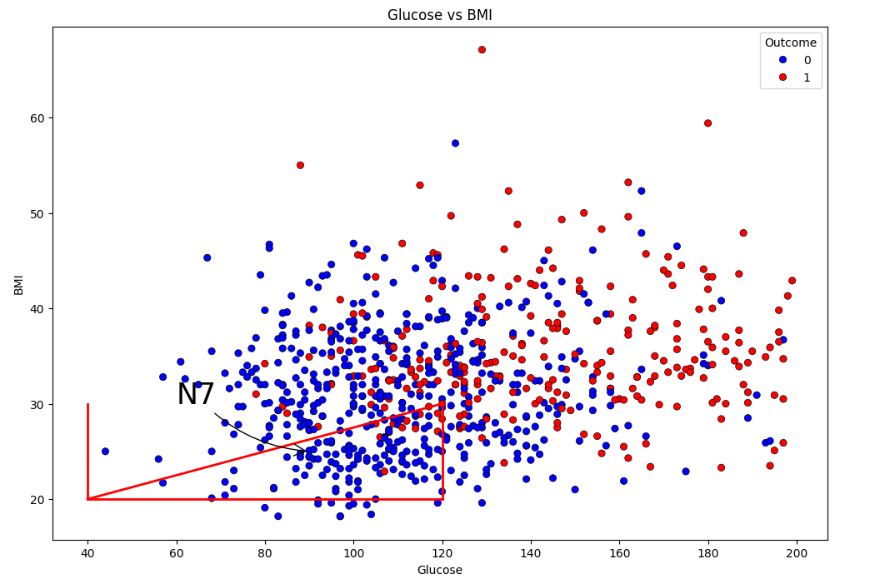
Biểu đồ mô tả mối quan hệ giữa mức đường huyết (Glucose) và huyết áp tâm trương (BloodPressure).  
Quan sát cho thấy đường huyết cao là yếu tố phân tách rõ ràng nhất giữa hai nhóm mắc và không mắc tiểu đường, trong khi huyết áp không thể hiện sự khác biệt lớn.  
Điều này gợi ý rằng BloodPressure chỉ đóng vai trò hỗ trợ, không phải là biến dự đoán chính trong bộ dữ liệu này.

4. Biểu đồ tán xạ “SkinThickness vs BMI”



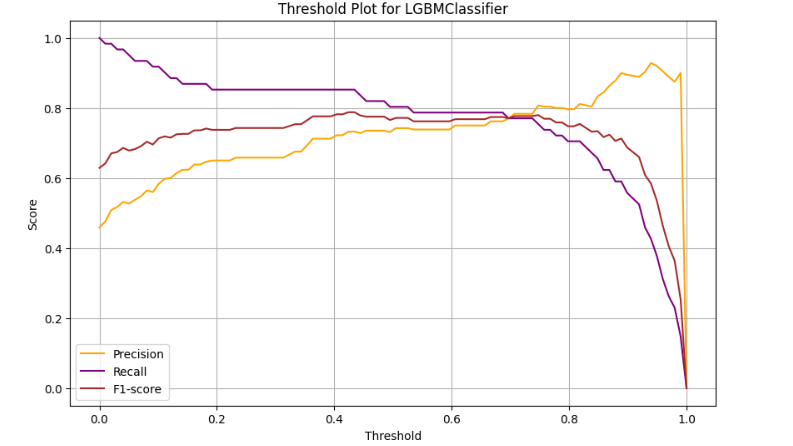
Biểu đồ cho thấy mối liên hệ giữa độ dày lớp da (SkinThickness) và chỉ số khối cơ thể (BMI).  
Kết quả cho thấy những người có lớp da mỏng và BMI trung bình (20–30) phần lớn không mắc tiểu đường, trong khi nhóm có BMI > 30 (béo phì) và SkinThickness cao hơn có khả năng mắc bệnh lớn hơn.  
Điều này củng cố nhận định rằng béo phì là yếu tố nguy cơ hàng đầu trong cơ chế phát triển tiểu đường.

5. Biểu đồ tán xạ “Glucose vs BMI”



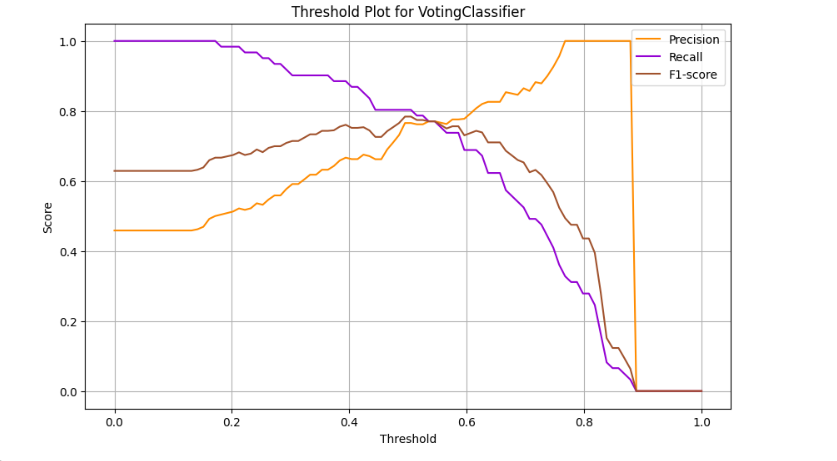
Biểu đồ thể hiện tương quan giữa đường huyết (Glucose) và BMI.  
Các trường hợp có Glucose > 140 mg/dL và BMI > 30 xuất hiện mật độ điểm vàng cao, cho thấy đường huyết cao và béo phì kết hợp làm tăng đáng kể nguy cơ mắc bệnh.  
Mối quan hệ này thể hiện tác động tương hỗ giữa hai yếu tố chuyển hóa quan trọng trong dự đoán tiểu đường.

6. Biểu đồ “Threshold Plot – LGBMClassifier”



Biểu đồ minh họa mối quan hệ giữa ngưỡng phân loại (threshold) với các chỉ số đánh giá mô hình: Precision, Recall và F1-score.  
Khi threshold thấp, Recall tăng nhưng Precision giảm; ngược lại, khi threshold cao, Precision tăng nhưng Recall giảm.  
Điểm cực đại của F1-score thể hiện ngưỡng tối ưu, giúp mô hình đạt được sự cân bằng tốt nhất giữa độ chính xác và khả năng bao phủ mẫu dương tính.

7. Biểu đồ “Threshold Plot – VotingClassifier”



Kết quả tương tự được quan sát với VotingClassifier. Khi ngưỡng phân loại giảm, Recall tăng nhưng Precision giảm; khi ngưỡng tăng, điều ngược lại xảy ra.  
Giá trị F1-score đạt cao nhất tại điểm cân bằng, là ngưỡng tối ưu giúp mô hình duy trì hiệu năng dự đoán ổn định.  
Điều này thể hiện sự đánh đổi giữa Precision và Recall – một nguyên lý quan trọng trong thiết kế mô hình học máy.

***Kết luận***  
Các biểu đồ trực quan trên giúp minh chứng cho mối liên hệ giữa các yếu tố sinh lý (Glucose, BMI, Age, SkinThickness) với khả năng mắc bệnh tiểu đường. Đồng thời, việc trực quan hóa các ngưỡng dự đoán giúp người phân tích đánh giá và tối ưu hóa mô hình một cách trực quan, đảm bảo cân bằng giữa độ chính xác và khả năng phát hiện bệnh trong thực tế.

### 2.4. Bài tập thực hành 2 - Tập dữ liệu *EDA Online Retail (2)*

#### 2.4.1. Mô tả và tiền xử lý dữ liệu

Nguồn dữ liệu:  
Tập dữ liệu “Online Retail Dataset” từ Kaggle, ghi nhận các giao dịch bán hàng trực tuyến của một công ty bán lẻ tại Anh giai đoạn 2010–2011.

Kích thước dữ liệu: 541.909 dòng × 8 cột

Các biến chính:

* InvoiceNo: Mã hóa đơn
* StockCode: Mã sản phẩm
* Description: Mô tả sản phẩm
* Quantity: Số lượng sản phẩm bán
* InvoiceDate: Ngày giao dịch
* UnitPrice: Đơn giá (GBP)
* CustomerID: Mã khách hàng
* Country: Quốc gia

Tình trạng dữ liệu:

* Thiếu dữ liệu:

Description: 1.454 dòng bị thiếu

CustomerID: 135.080 dòng bị thiếu

* Giá trị trùng lặp: Có xuất hiện các giao dịch trùng (do lỗi ghi nhận hoặc trả hàng).
* Giá trị âm: Một số Quantity âm thể hiện giao dịch trả lại hàng (refund).

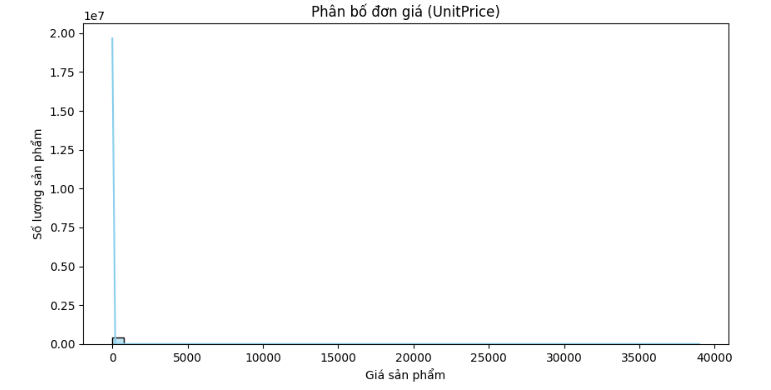
Xử lý dữ liệu:

* Loại bỏ dòng thiếu CustomerID để đảm bảo tính chính xác khi phân tích hành vi khách hàng.
* Xóa các bản ghi trùng lặp.
* Tạo biến mới TotalPrice = Quantity × UnitPrice để biểu thị doanh thu của từng giao dịch.
* Lọc bỏ các giao dịch có Quantity ≤ 0 hoặc UnitPrice ≤ 0.

→ Dữ liệu sau xử lý sạch và nhất quán hơn, sẵn sàng cho các bước phân tích mô tả và trực quan hóa.

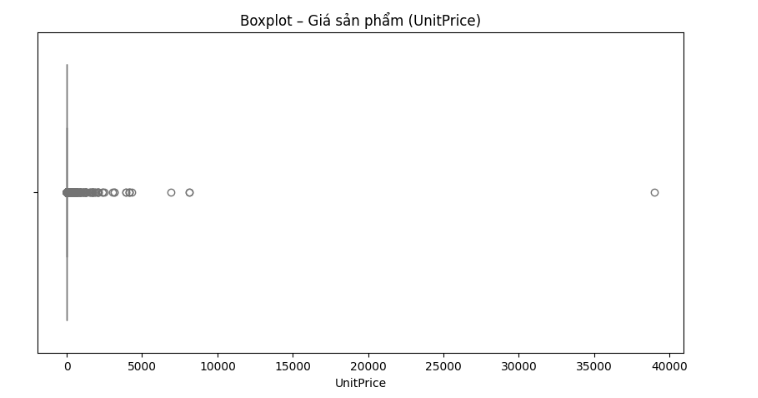
#### 2.4.2. Trực quan hóa và diễn giải

1. Phân bố đơn giá (UnitPrice)



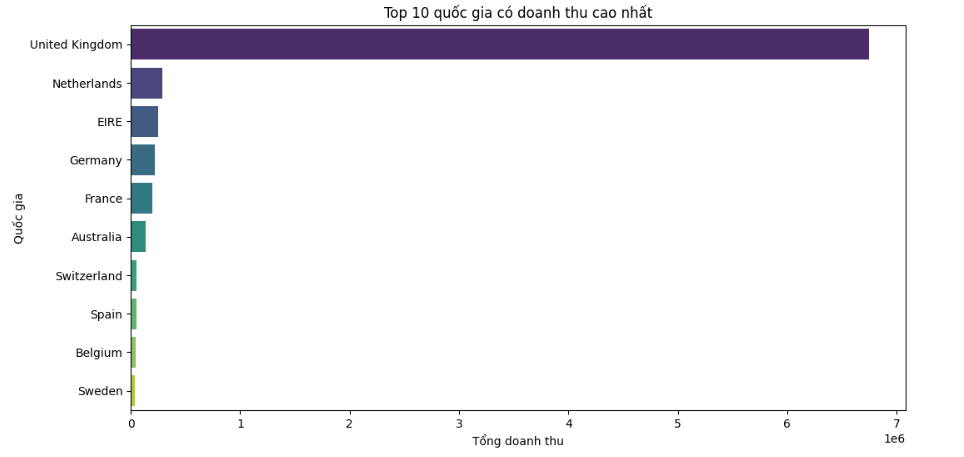
Biểu đồ histogram cho thấy phần lớn sản phẩm có đơn giá dưới 5 GBP, thể hiện đặc điểm hàng hóa phổ thông, giá thấp.  
Một số giá trị cao bất thường (>100 GBP) được xác định là ngoại lai (outliers), có thể do lỗi nhập liệu hoặc sản phẩm đặc biệt.  
→ Phân bố lệch phải mạnh, phù hợp với quy luật giá trong bán lẻ (phần lớn hàng hóa giá thấp, ít hàng cao cấp).

2. Boxplot – Đơn giá sản phẩm



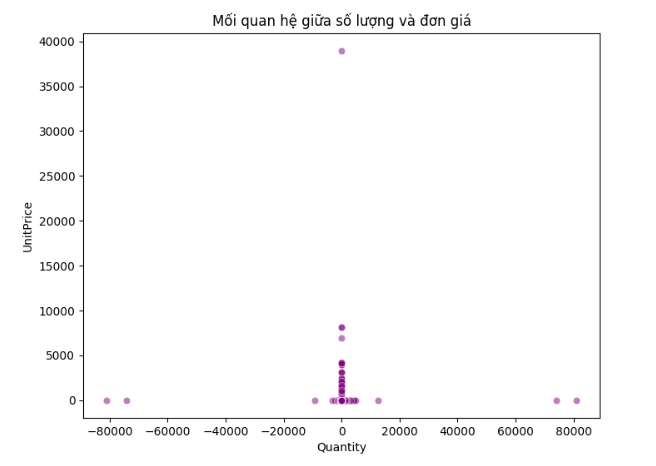
Boxplot cho thấy nhiều điểm ngoại lệ ở vùng giá cao, trong khi phần lớn sản phẩm tập trung ở vùng giá thấp (<10 GBP).  
→ Các outlier này nên được kiểm tra trước khi mô hình hóa để tránh ảnh hưởng đến các phép đo trung bình hoặc hồi quy.

3. Top 10 quốc gia có doanh thu cao nhất



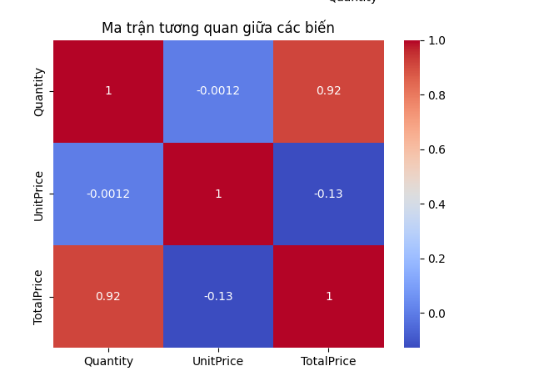
Biểu đồ cột thể hiện United Kingdom chiếm ưu thế tuyệt đối về doanh thu, vượt xa các quốc gia khác như Hà Lan, Đức, Pháp.  
→ Điều này hợp lý vì Anh là thị trường nội địa chính của doanh nghiệp, chiếm hơn 85–90% tổng giao dịch.

4. Mối quan hệ giữa số lượng và đơn giá (Quantity vs UnitPrice)



Biểu đồ tán xạ cho thấy không có tương quan tuyến tính rõ ràng giữa số lượng bán và đơn giá.  
Một số giá trị Quantity âm phản ánh các đơn hàng bị hoàn trả.  
→ Kết quả gợi ý rằng giá không ảnh hưởng trực tiếp đến lượng tiêu thụ trong dataset này, có thể do chính sách giá đồng nhất cho từng loại hàng.

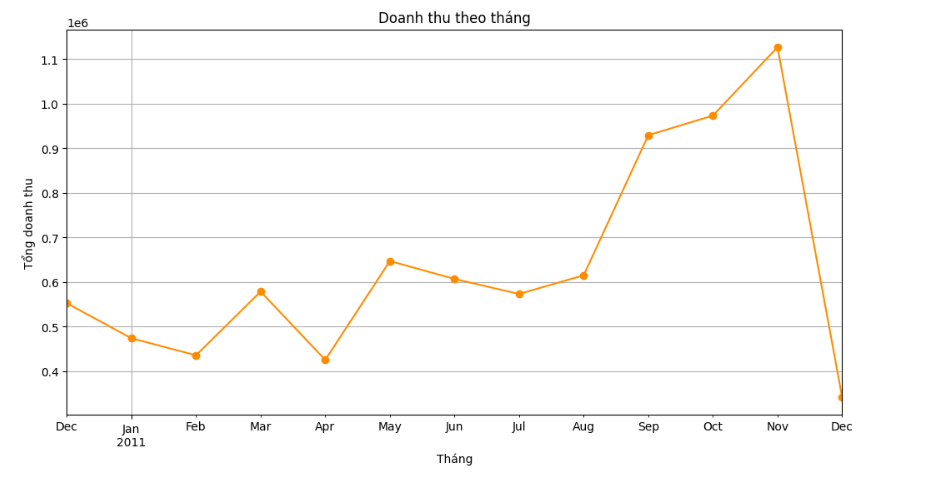
5. Ma trận tương quan giữa các biến



Biểu đồ heatmap cho thấy:

* Quantity tương quan mạnh với TotalPrice (r ≈ 0.92) → giá trị doanh thu chủ yếu bị chi phối bởi số lượng bán ra.
* UnitPrice hầu như không tương quan với các biến khác.  
  → Đây là mô hình doanh thu điển hình: “số lượng quyết định doanh thu nhiều hơn đơn giá”, phản ánh tính chất kinh doanh hàng tiêu dùng.

6. Doanh thu theo tháng (Monthly Revenue)



Biểu đồ line chart cho thấy doanh thu tăng dần từ tháng 9 đến tháng 11/2011, đạt đỉnh vào tháng 11, sau đó giảm mạnh vào tháng 12.  
→ Điều này phản ánh hiệu ứng mùa vụ trong hành vi mua sắm, khi người tiêu dùng mua sắm nhiều hơn vào dịp cuối năm và Giáng sinh.

***Kết luận***

* Dữ liệu gồm hơn 540.000 giao dịch chủ yếu tại Anh Quốc.
* United Kingdom là thị trường trọng điểm, chiếm phần lớn doanh thu.
* Phân bố giá lệch phải, nhiều giá trị ngoại lai và giao dịch hoàn trả.
* Quantity có tương quan mạnh với TotalPrice, trong khi UnitPrice ảnh hưởng nhỏ.
* Doanh thu có tính mùa vụ, tăng cao rõ rệt vào quý IV/2011.  
  → Kết quả trực quan hóa cung cấp cái nhìn tổng quan về hành vi mua hàng, cấu trúc giá, và xu hướng doanh thu theo thời gian, là cơ sở quan trọng cho phân tích khách hàng và dự báo bán hàng trong các bước EDA nâng cao.

## 3. Phân tích đơn biến và hai biến

### 3.1. Ôn tập lý thuyết

#### 3.1.1. Phân tích đơn biến và hai biến

* Phân tích đơn biến (univariate analysis) là việc xem xét đặc điểm của một biến duy nhất nhằm mô tả và tóm tắt dữ liệu (ví dụ: phân bố, trung bình, độ lệch chuẩn).
* Phân tích hai biến (bivariate analysis) nghiên cứu mối quan hệ giữa hai biến, thường để xác định xu hướng, tương quan hoặc sự phụ thuộc giữa chúng.  
  → Điểm khác biệt chính: phân tích đơn biến chỉ mô tả đặc tính của một biến, trong khi phân tích hai biến tập trung vào mối liên hệ giữa các biến.

#### 3.1.2. Các thước đo thống kê trong phân tích đơn biến

Các thước đo thường dùng gồm:

* Trung bình (Mean): giá trị đại diện cho xu hướng trung tâm.
* Trung vị (Median): giá trị chia dữ liệu thành hai nửa bằng nhau, phù hợp với dữ liệu lệch.
* Mode: giá trị xuất hiện nhiều nhất.
* Phương sai (Variance) và độ lệch chuẩn (Standard Deviation): đo mức độ phân tán.
* Giá trị cực trị (Min–Max) và phần tư (Q1, Q3) giúp mô tả phạm vi và độ trải rộng dữ liệu.

#### 3.1.3. Xác định mối quan hệ giữa hai biến

Mối quan hệ giữa hai biến có thể được xác định thông qua:

* Hệ số tương quan (correlation coefficient) – đo lường mức độ và hướng liên hệ tuyến tính.
* Hồi quy tuyến tính (linear regression) – kiểm tra khả năng dự đoán hoặc quan hệ nhân quả.
* Kiểm định thống kê (như t-test, ANOVA) – xác định xem sự khác biệt giữa các nhóm có ý nghĩa thống kê hay không.

#### 3.1.4. Sự khác biệt giữa tương quan và hiệp biến

* Hiệp biến (Covariance): đo mức độ hai biến cùng thay đổi, có thể dương hoặc âm, phụ thuộc vào đơn vị đo lường.
* Tương quan (Correlation): là hiệp biến đã chuẩn hóa, có giá trị trong khoảng [-1, 1], giúp đánh giá mức độ mạnh – yếu và hướng của mối quan hệ giữa hai biến.

#### 3.1.5. Sử dụng biểu đồ trong phân tích đơn biến và hai biến

* Phân tích đơn biến: dùng histogram, boxplot để quan sát phân bố, độ lệch, ngoại lai.
* Phân tích hai biến: dùng scatter plot, heatmap, hoặc violin plot để thể hiện mối quan hệ và tương quan giữa các biến.

#### 3.1.6. Code minh họa – Scatter plot và Heatmap

import matplotlib.pyplot as plt

import seaborn as sns

# Scatter plot

plt.scatter(df['x'], df['y'])

plt.title('Scatter Plot giữa hai biến')

plt.xlabel('Biến X')

plt.ylabel('Biến Y')

plt.show()

# Heatmap tương quan

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

plt.title('Heatmap ma trận tương quan')

plt.show()

#### 3.1.7. Trực quan hóa mối quan hệ giữa biến số và biến phân loại

# Boxplot

sns.boxplot(x='nhom', y='giatri', data=df)

plt.title('Boxplot giữa biến phân loại và biến số')

plt.show()

# Violin plot

sns.violinplot(x='nhom', y='giatri', data=df)

plt.title('Violin plot giữa biến phân loại và biến số')

plt.show()

### 3.2. Bài tập thực hành 1 - Ứng dụng SweetViz trên tập dữ liệu *Marketing Campaign*

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.

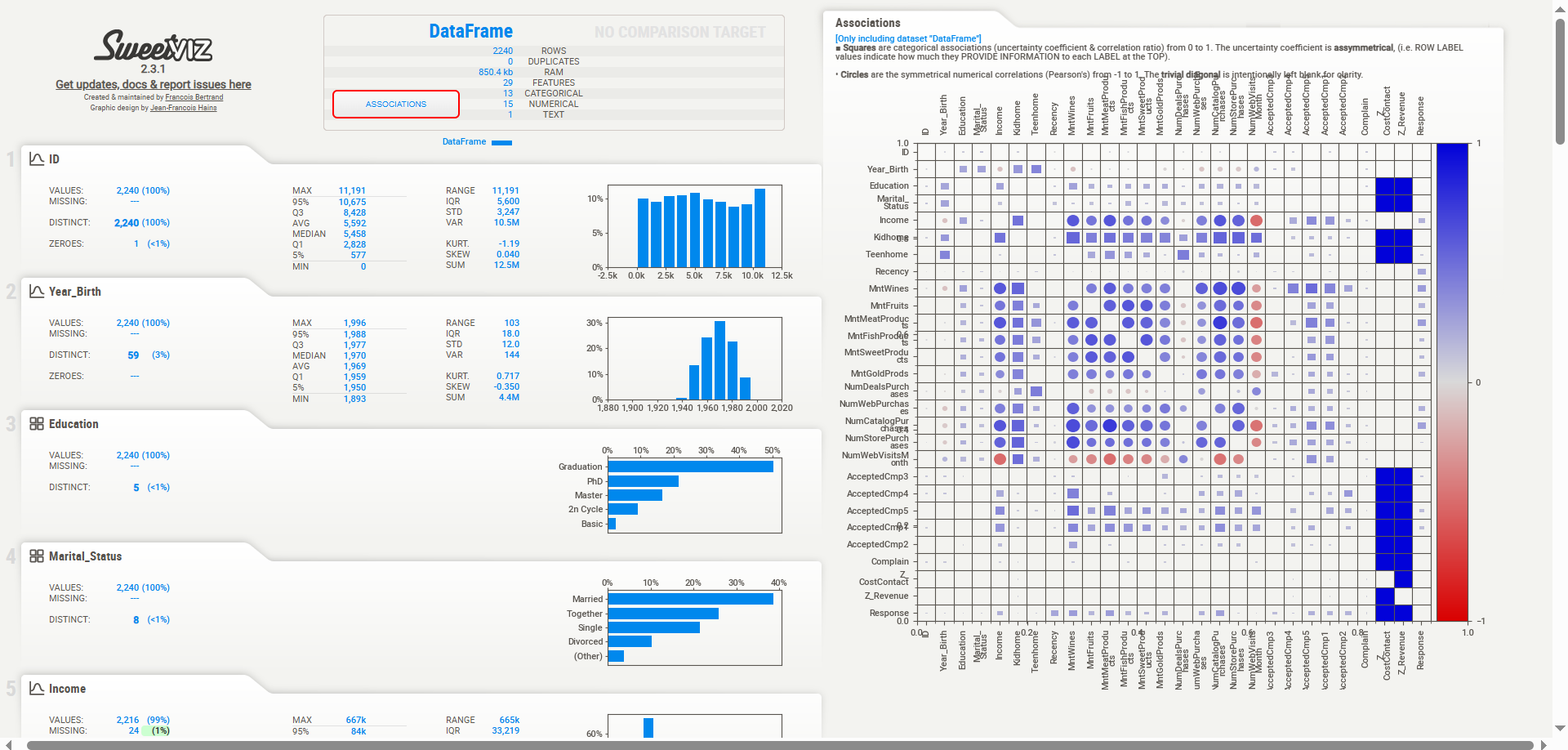
#### 3.2.1. Mô tả và tiền xử lý dữ liệu

Tập dữ liệu Marketing Campaign chứa thông tin khách hàng và hành vi tiêu dùng trong các chiến dịch tiếp thị.  
Bộ dữ liệu gồm 4 nhóm chính:

* Dữ liệu cá nhân & nhân khẩu học (People):  
  Gồm Year\_Birth, Education, Marital\_Status, Income, Kidhome, Teenhome, Recency, Complain.  
  Các biến này mô tả đặc điểm và mức độ tương tác của khách hàng.
* Dữ liệu chi tiêu sản phẩm (Products): Gồm MntWines, MntMeatProducts, MntFruits, MntFishProducts, MntSweetProducts, MntGoldProds – phản ánh giá trị chi tiêu của khách hàng trong từng danh mục hàng hóa.
* Dữ liệu kênh mua hàng (Place): Gồm NumWebPurchases, NumCatalogPurchases, NumStorePurchases, NumWebVisitsMonth – thể hiện hành vi và tần suất mua hàng trên các kênh khác nhau.
* Dữ liệu khuyến mãi & chiến dịch (Promotion): Gồm NumDealsPurchases, AcceptedCmp1–5 và Response (biến mục tiêu), cho biết khả năng khách hàng chấp nhận các ưu đãi trong từng chiến dịch.

Dữ liệu được kiểm tra giá trị thiếu và định dạng biến, sau đó được chuẩn hóa kiểu dữ liệu để phục vụ phân tích tự động bằng công cụ SweetViz.

#### 3.2.2. Trực quan hóa dữ liệu bằng SweetViz và diễn giải



Bộ công cụ SweetViz được sử dụng để tự động sinh báo cáo EDA (Exploratory Data Analysis), giúp hiển thị trực quan mối quan hệ giữa các biến định lượng, định tính và biến mục tiêu Response.  
Kết quả báo cáo được chia làm ba nhóm chính:

1. Khả năng phản hồi chiến dịch (Response):

* Các biến AcceptedCmp1–5 có tương quan dương mạnh với Response, cho thấy những khách hàng từng chấp nhận ưu đãi trước đó có xu hướng tiếp tục phản hồi tích cực.
* Chi tiêu sản phẩm (MntWines, MntMeatProducts, MntSweetProducts) có mối quan hệ dương với Response, chứng tỏ nhóm khách hàng chi tiêu cao thường dễ bị thuyết phục bởi các chiến dịch marketing.
* Kênh mua hàng: NumCatalogPurchases và NumStorePurchases tương quan mạnh với Response, trong khi NumWebVisitsMonth có tương quan âm nhẹ, cho thấy người mua qua kênh truyền thống phản hồi tốt hơn.

1. Hành vi chi tiêu và kênh mua hàng:

* Các biến chi tiêu (Mnt\_\_) có tương quan dương cao với nhau, chứng minh sự tồn tại của nhóm khách hàng giá trị cao (High-Value Customers).
* NumCatalogPurchases và NumStorePurchases có mối liên hệ chặt chẽ với tổng chi tiêu, xác nhận vai trò quan trọng của hai kênh này.
* NumWebVisitsMonth có tương quan âm với chi tiêu, gợi ý rằng người chi tiêu nhiều có xu hướng mua hàng trực tiếp hơn là duyệt web.

1. Đặc điểm nhân khẩu học và thu nhập:

* Income có tương quan dương mạnh với chi tiêu (đặc biệt là MntWines và MntMeatProducts), nhưng âm với NumDealsPurchases và NumWebVisitsMonth, cho thấy nhóm thu nhập cao ít quan tâm đến khuyến mãi.
* Kidhome và Teenhome tương quan âm với Income và chi tiêu, phản ánh các hộ gia đình có con nhỏ chi tiêu ít hơn.
* Recency và Complain hầu như không có tương quan rõ ràng với hành vi mua hàng, cho thấy ảnh hưởng hạn chế đến khả năng phản hồi chiến dịch.

***Kết luận***

Phân tích SweetViz cho thấy:

* Nhóm khách hàng có thu nhập cao, chi tiêu lớn, thường mua qua Catalog hoặc cửa hàng là đối tượng tiềm năng nhất cho các chiến dịch marketing tương lai.
* Ngược lại, nhóm khách hàng có thu nhập thấp, thường mua hàng giảm giá, và ghé web nhiều là phân khúc nhạy cảm về giá, cần chiến lược tiếp cận riêng (ví dụ: khuyến mãi hoặc ưu đãi online).
* Các biến AcceptedCmp1–5 và Response cho thấy sự ổn định trong hành vi phản hồi, giúp doanh nghiệp dự đoán tốt hơn khả năng chấp nhận chiến dịch tiếp theo.

Công cụ SweetViz đã hỗ trợ trực quan hóa dữ liệu một cách toàn diện, giúp nhà phân tích nhanh chóng nhận diện cấu trúc, mối tương quan và hành vi người tiêu dùng, từ đó định hướng chiến lược tiếp thị dựa trên dữ liệu (Data-driven Marketing).

### 3.3. Bài tập thực hành 2 - Ứng dụng AutoViz trên tập dữ liệu Marketing Campaign

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.

#### 3.3.1. Mô tả và tiền xử lý dữ liệu

Tập dữ liệu Marketing Campaign được sử dụng để minh họa khả năng trực quan hóa tự động bằng công cụ AutoViz, một thư viện Python hỗ trợ thực hiện Exploratory Data Analysis (EDA) một cách nhanh chóng và tự động.

Dữ liệu gồm 2.240 quan sát và 29 biến, phản ánh thông tin nhân khẩu học, hành vi chi tiêu và phản hồi của khách hàng trong các chiến dịch tiếp thị. Một số biến đáng chú ý gồm:

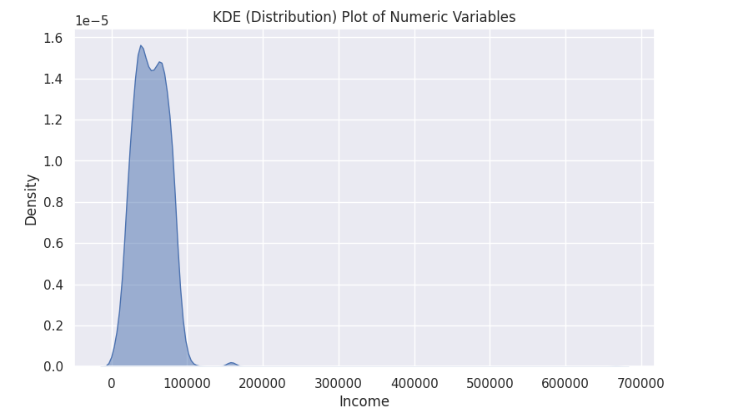
* Year\_Birth, Education, Marital\_Status, Income: đặc trưng cá nhân và kinh tế – xã hội.
* Kidhome, Teenhome: số trẻ nhỏ và thiếu niên trong hộ gia đình.
* Recency, MntWines, MntMeatProducts, …: các chỉ số hành vi chi tiêu.
* Response: biến mục tiêu thể hiện việc khách hàng có chấp nhận chiến dịch hay không.

Một số biến định danh (ID, Z\_CostContact, Z\_Revenue) được loại bỏ do không mang ý nghĩa phân tích.  
Công cụ AutoViz đồng thời phát hiện các giá trị ngoại lai (outliers) ở các biến định lượng như Income, MntWines, MntMeatProducts,… và đề xuất xử lý bằng phương pháp capping hoặc loại bỏ giá trị cực trị để đảm bảo phân tích chính xác.

#### 3.3.2. Trực quan hóa dữ liệu bằng AutoViz và diễn giải

1. Phân tích đơn biến

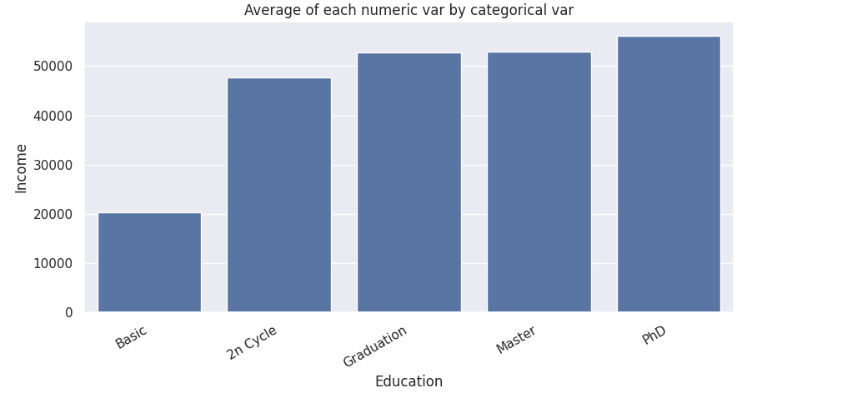
Biểu đồ mật độ (KDE Plot – biến Income):



Biểu đồ cho thấy phân phối thu nhập của khách hàng có dạng lệch phải, tức phần lớn khách hàng có thu nhập trung bình và chỉ một số ít có mức thu nhập rất cao.  
Dạng phân phối này gợi ý sự tồn tại của giá trị ngoại lai phía trên, thường gặp trong dữ liệu thu nhập cá nhân. Do đó, cần xem xét việc chuẩn hóa hoặc cắt ngưỡng (capping) để giảm tác động của các giá trị cực đoan trong quá trình mô hình hóa.

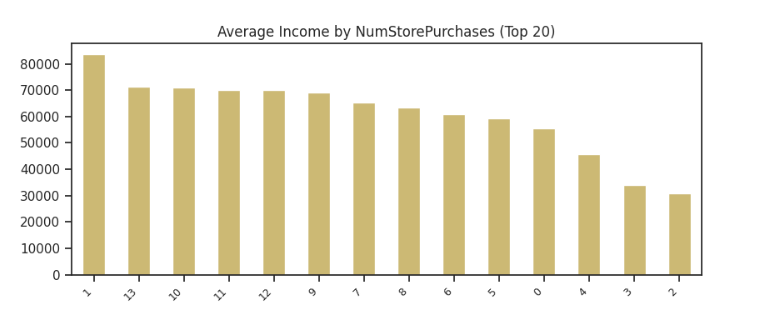
2. Phân tích hai biến

* Mối quan hệ giữa Income và Education:

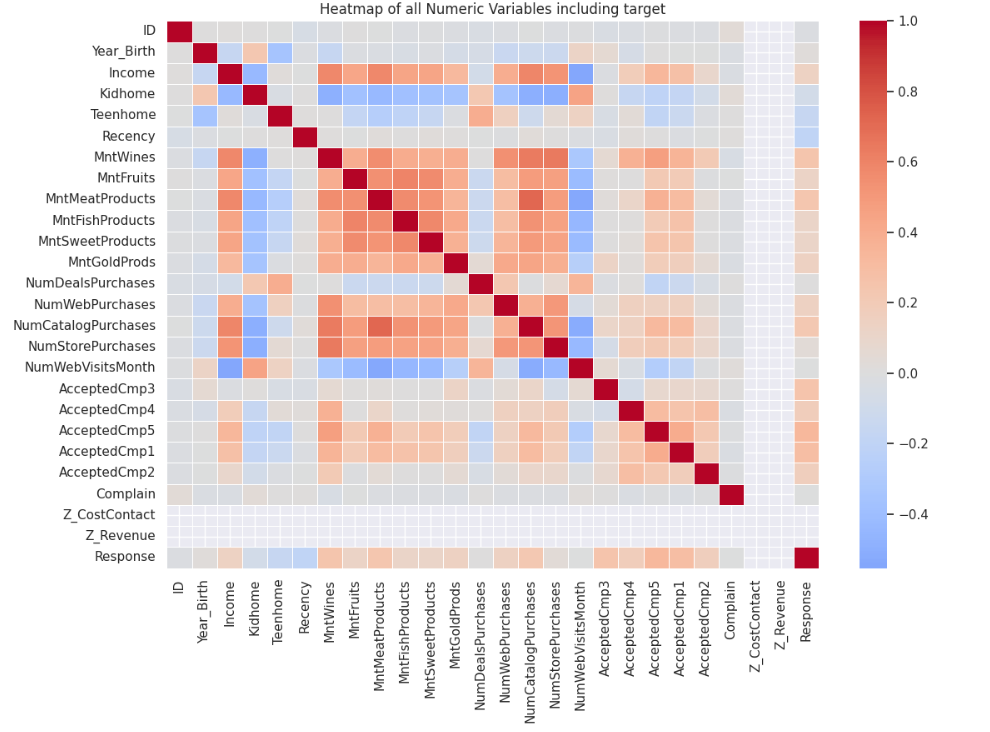


Biểu đồ cột cho thấy thu nhập trung bình tăng dần theo trình độ học vấn. Nhóm có bằng PhD và Master có mức thu nhập cao nhất, trong khi nhóm Basic thấp nhất.  
Điều này thể hiện mối tương quan dương giữa học vấn và thu nhập, phù hợp với lý thuyết kinh tế – xã hội rằng học vấn cao gắn liền với khả năng chi tiêu cao.  
→ Hàm ý marketing: nên ưu tiên nhóm khách hàng có trình độ học vấn cao trong các chiến dịch hướng đến sản phẩm cao cấp.

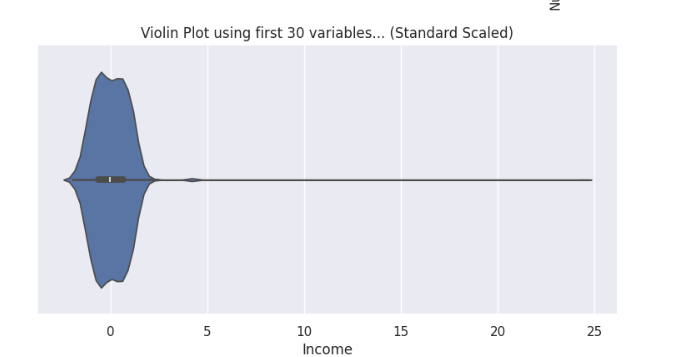
* Mối quan hệ giữa Income và NumStorePurchases:

  
Biểu đồ thể hiện xu hướng thu nhập tăng cùng với số lần mua tại cửa hàng.  
Khách hàng có từ 4–8 lần mua trở lên có thu nhập cao và ổn định hơn; nhóm có thu nhập cao (≈70.000–75.000) thường mua hàng thường xuyên hơn.  
Tuy nhiên, một vài giá trị ngoại lệ (ví dụ: khách hàng mua rất ít nhưng có thu nhập >80.000) cho thấy cần xem xét lại các cá thể đặc biệt này, có thể là khách hàng VIP hoặc dữ liệu bất thường.

* Biểu đồ nhiệt (Heatmap):

  
Biểu đồ hiển thị ma trận tương quan giữa các biến số. Các nhóm biến chi tiêu (MntWines, MntMeatProducts, MntFishProducts, MntFruits, MntGoldProds) có tương quan dương mạnh (0.6–0.8) → Khách hàng chi nhiều cho một loại hàng thường chi nhiều cho các loại khác. Ngược lại, Income có tương quan yếu với các biến chiến dịch (AcceptedCmp1–5, Response), cho thấy thu nhập không phải yếu tố quyết định khả năng phản hồi.

* Violin Plot (Income theo Education/Marital\_Status):

  
Biểu đồ cho thấy phân bố thu nhập sau chuẩn hóa tập trung quanh trung bình (mean ≈ 0), với một số ít giá trị ngoại lai ở hai đầu.

Hình dạng phân bố gần chuẩn chứng tỏ việc chuẩn hóa dữ liệu (standard scaling) đã giúp cân bằng ảnh hưởng của thu nhập, giảm sai lệch trong các mô hình dự báo.

***Kết luận***

Phân tích EDA bằng AutoViz giúp rút ra các kết luận chính:

* Dữ liệu phân bố thu nhập lệch phải, có nhiều ngoại lệ ở mức thu nhập cao, cần xử lý trước khi mô hình hóa.
* Có mối quan hệ dương giữa trình độ học vấn và thu nhập, gợi ý nhóm khách hàng học vấn cao là đối tượng có tiềm năng phản hồi tốt hơn.
* Tần suất mua hàng tại cửa hàng tăng theo thu nhập, trong khi thu nhập và biến phản hồi (Response) không có mối tương quan mạnh.
* Các danh mục chi tiêu (Mnt\_\_) tương quan cao, thể hiện nhóm khách hàng giá trị cao có hành vi chi tiêu đồng nhất.

Tổng thể, AutoViz là công cụ hiệu quả cho việc khảo sát nhanh cấu trúc dữ liệu, phát hiện vấn đề tiền xử lý, đồng thời cung cấp cơ sở trực quan hóa khoa học cho giai đoạn xây dựng mô hình dự báo trong chiến lược marketing.

# LAB02 – THUẬT TOÁN PHÂN LỚP

## 2.1. GIẢI THUẬT 1: CÂY QUYẾT ĐỊNH VÀ RỪNG CÂY

### 2.1.1. Ôn tập lý thuyết

**Câu 1:** 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?

**Quy trình khai phá dữ liệu CRISP–DM và SEMMA**

**a. Quy trình CRISP–DM (Cross Industry Standard Process for Data Mining)**

CRISP–DM (Cross Industry Standard Process for Data Mining) là mô hình quy trình chuẩn trong lĩnh vực khai phá dữ liệu, được sử dụng rộng rãi trong công nghiệp nhờ tính hệ thống, linh hoạt và khả năng áp dụng thực tế cao. Mô hình này gồm sáu giai đoạn chính, được thiết kế nhằm đảm bảo toàn bộ quá trình khai thác dữ liệu diễn ra một cách khoa học, có mục tiêu rõ ràng và nhất quán:

* **Business Understanding (Hiểu biết nghiệp vụ):** Xác định rõ mục tiêu kinh doanh, phạm vi dự án, vấn đề cần giải quyết và các tiêu chí đánh giá thành công. Đây là giai đoạn định hướng toàn bộ quá trình khai phá dữ liệu.
* **Data Understanding (Hiểu biết dữ liệu):** Thu thập dữ liệu ban đầu, khám phá đặc điểm của dữ liệu, đánh giá chất lượng và phát hiện các vấn đề tiềm ẩn như dữ liệu thiếu, sai lệch hoặc không nhất quán.
* **Data Preparation (Chuẩn bị dữ liệu):** Thực hiện làm sạch dữ liệu, lựa chọn biến phù hợp, xử lý giá trị khuyết và chuyển đổi dữ liệu sang định dạng thích hợp cho mô hình hóa.
* **Modeling (Mô hình hóa):** Lựa chọn, xây dựng và huấn luyện các mô hình khai phá dữ liệu như Decision Tree, Regression hoặc Neural Network để phát hiện quy luật hoặc dự đoán kết quả.
* **Evaluation (Đánh giá):** Đánh giá hiệu quả của mô hình dựa trên tiêu chí nghiệp vụ và các thước đo kỹ thuật như độ chính xác, khả năng khái quát hóa nhằm bảo đảm mô hình đáp ứng được mục tiêu đặt ra.
* **Deployment (Triển khai):** Đưa mô hình vào ứng dụng thực tế, tạo báo cáo kết quả hoặc tích hợp vào hệ thống sản xuất để hỗ trợ ra quyết định.

**b. Quy trình SEMMA (Sample – Explore – Modify – Model – Assess)**

SEMMA là một quy trình phân tích dữ liệu do SAS Institute phát triển. Mục tiêu của mô hình này là tối ưu hóa việc phát hiện tri thức và xây dựng mô hình dự đoán từ dữ liệu thông qua năm giai đoạn logic và liên kết chặt chẽ với nhau:

* **Sample (Lấy mẫu):** Chọn ra một mẫu dữ liệu đại diện cho toàn bộ tập dữ liệu gốc. Việc lấy mẫu giúp giảm chi phí tính toán và rút ngắn thời gian xử lý, trong khi vẫn đảm bảo tính tổng quát của kết quả.
* **Explore (Khám phá):** Phân tích dữ liệu thông qua các kỹ thuật thống kê mô tả và trực quan hóa, nhằm hiểu rõ cấu trúc dữ liệu, phát hiện xu hướng, mẫu hình hoặc mối quan hệ giữa các biến.
* **Modify (Biến đổi):** Làm sạch dữ liệu, xử lý giá trị khuyết, lựa chọn và biến đổi các biến đầu vào sao cho phù hợp với yêu cầu của mô hình phân tích, đồng thời nâng cao chất lượng dữ liệu.
* **Model (Mô hình hóa):** Xây dựng và huấn luyện các mô hình dự đoán hoặc phân loại, như Decision Tree, Regression hay Neural Network, để trích xuất tri thức hữu ích từ dữ liệu.
* **Assess (Đánh giá):** Đánh giá hiệu suất và độ tin cậy của mô hình dựa trên các tiêu chí như độ chính xác, khả năng dự đoán hoặc khả năng khái quát hóa, từ đó lựa chọn mô hình tối ưu nhất.

**Câu 2:** 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 – nguyên lý hoạt động, các thành phần và cách dự đoán**

**Khái niệm:**

Cây quyết định (*Decision Tree*) là một mô hình học máy có cấu trúc phân nhánh dạng cây, được sử dụng phổ biến trong cả bài toán phân loại (classification) và hồi quy (regression). Mô hình hoạt động dựa trên việc chia nhỏ dữ liệu thành các nhóm con theo các điều kiện cụ thể, giúp phát hiện quy luật ra quyết định một cách trực quan và dễ hiểu.

Mỗi nút (node) trong cây tương ứng với một phép kiểm tra điều kiện trên dữ liệu, còn các nhánh (branches) thể hiện kết quả của phép kiểm tra đó. Quá trình này tiếp diễn cho đến khi đạt đến nút lá (leaf node) – nơi chứa kết quả dự đoán cuối cùng.

**Cấu trúc của cây quyết định:**

* **Nút gốc (Root Node):** Là điểm khởi đầu của cây, chứa toàn bộ tập dữ liệu huấn luyện trước khi được phân tách. Quyết định chia tách tại nút gốc thường dựa trên thuộc tính giúp dữ liệu được phân loại tốt nhất.
* **Nút trong (Internal Node):** Đại diện cho một phép chia dữ liệu dựa trên giá trị của một thuộc tính cụ thể (ví dụ: “Giới tính”, “Độ tuổi”, “Mức thu nhập”). Mỗi nút trong làm nhiệm vụ tách dữ liệu thành các nhóm nhỏ hơn, giúp mô hình học được các quy luật chi tiết hơn.
* **Cành (Branch):** Thể hiện kết quả của điều kiện được đặt tại nút cha và dẫn tới các nút con. Mỗi cành mô tả một hướng đi cụ thể trong quá trình ra quyết định.
* **Nút lá (Leaf Node):** Là điểm kết thúc của cây, đại diện cho nhãn phân loại hoặc giá trị dự đoán cuối cùng. Ví dụ, trong bài toán dự đoán hành khách Titanic, nút lá có thể biểu thị kết quả “Sống sót” hoặc “Không sống sót”.

**Nguyên lý hoạt động:**

* Quá trình xây dựng cây quyết định bắt đầu tại nút gốc, nơi chứa toàn bộ tập dữ liệu huấn luyện. Tại đây, thuật toán sẽ lựa chọn thuộc tính tối ưu nhất để chia dữ liệu, dựa trên các tiêu chí đánh giá như Gini Index hoặc Information Gain – những thước đo giúp xác định mức độ “thuần nhất” của dữ liệu sau khi được phân tách.
* Sau khi xác định được thuộc tính tốt nhất, mỗi nhánh con được tạo ra sẽ tương ứng với một giá trị cụ thể của thuộc tính đó, giúp chia dữ liệu thành các nhóm nhỏ hơn. Quá trình này lặp lại đệ quy tại các nút con, mỗi lần lại tìm kiếm thuộc tính phù hợp nhất để tiếp tục tách dữ liệu.
* Thuật toán sẽ dừng lại khi đạt điều kiện dừng, chẳng hạn khi dữ liệu trong nút đã đủ đồng nhất, không còn thuộc tính khả dụng để chia, hoặc khi cây đạt đến độ sâu giới hạn được đặt trước.
* Khi dự đoán cho một mẫu dữ liệu mới, mô hình sẽ duyệt cây từ nút gốc qua các nhánh, kiểm tra điều kiện tại từng nút để xác định hướng đi phù hợp. Quá trình này tiếp tục cho đến khi mẫu được dẫn đến một nút lá, và giá trị hoặc nhãn tại nút lá chính là kết quả dự đoán cuối cùng.

**Câu 3:** 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?

**Các tiêu chí phân tách trong cây quyết định**

Trong quá trình xây dựng cây quyết định, một bước quan trọng là lựa chọn thuộc tính tối ưu để chia dữ liệu tại mỗi nút. Mục tiêu là tìm ra thuộc tính giúp tập dữ liệu sau khi chia trở nên “thuần nhất” nhất có thể, tức là các mẫu trong từng nhóm con gần như thuộc cùng một lớp.

Để đo lường mức độ “thuần” này, các thuật toán cây quyết định sử dụng những tiêu chí phân tách (splitting criteria) khác nhau, phổ biến nhất là Gini Index, Entropy/Information Gain, và Gain Ratio.

* **Gini Index:**
* Đo mức độ không thuần khiết (impurity) của tập dữ liệu.
* Công thức: ***Gini=1−∑(pi​)2*** Trong đó (p\_i) là tỷ lệ phần tử thuộc lớp (i).
* Giá trị Gini càng nhỏ, tập dữ liệu càng “thuần”, tức là thuộc tính chia tốt hơn.
* **Thuật toán CART** thường sử dụng tiêu chí này.
* **Entropy và Information Gain:**
* **Entropy** đo mức độ hỗn loạn trong dữ liệu:***Entropy=−∑pi​log2​(pi​)***
* **Information Gain** là lượng thông tin đạt được sau khi chia dữ liệu: **IG=Entropy(trước chia)−Entropy(sau chia)**
* Chọn thuộc tính có Information Gain cao nhất để chia.
* **Thuật toán ID3** và **C4.5** sử dụng tiêu chí này.
* **Gain Ratio:**
* Là phiên bản cải tiến của Information Gain, nhằm tránh ưu tiên các thuộc tính có quá nhiều giá trị.
* Tính bằng: ***Gain Ratio=***
* Được sử dụng trong thuật toán **C4.5**.

**So sánh ngắn:**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Đặc điểm chính** | **Thuật toán sử dụng** |
| Gini Index | Đơn giản, dễ tính toán và có tốc độ xử lý nhanh. Thích hợp cho các bài toán có tập dữ liệu lớn. | CART |
| Entropy / Information Gain | Đánh giá độ hỗn tạp của dữ liệu một cách chính xác hơn, nhưng yêu cầu tính toán phức tạp hơn so với Gini. | ID3 |
| Gain Ratio | Phiên bản cải tiến của Information Gain, giúp tránh hiện tượng thiên lệch với các thuộc tính có nhiều giá trị phân biệt. | C4.5 |

**Câu 4:** Rừng cây (Random Forest) là gì? Nó khác gì so với một cây quyết định đơn lẻ? Tại sao Random Forest thường có hiệu suất tốt hơn cây quyết định trong các bài toán phân loại?

**Rừng cây ngẫu nhiên (Random Forest)**

**Khái niệm:**

Rừng cây (Random Forest) là một mô hình thuộc nhóm học tập tổ hợp (Ensemble Learning), được phát triển dựa trên nguyên lý kết hợp nhiều cây quyết định (Decision Trees) để nâng cao độ chính xác và khả năng tổng quát hóa của mô hình.

Thay vì chỉ dựa vào một cây duy nhất, Random Forest xây dựng một tập hợp các cây quyết định độc lập, mỗi cây được huấn luyện trên một tập con dữ liệu được chọn ngẫu nhiên (bootstrap sampling) và một tập con ngẫu nhiên của các đặc trưng (features).

Cách tiếp cận này giúp mô hình giảm hiện tượng overfitting – vốn thường xảy ra ở cây quyết định đơn lẻ – đồng thời cải thiện độ ổn định và độ chính xác của dự đoán thông qua cơ chế “bỏ phiếu đa số” (voting) trong bài toán phân loại hoặc trung bình hóa kết quả trong bài toán hồi quy.

**Cách hoạt động:**

**Cách hoạt động:**

Quy trình hoạt động của **Random Forest** dựa trên nguyên lý **kết hợp (ensemble)** nhiều cây quyết định để tạo ra một mô hình mạnh và ổn định hơn. Các bước chính được mô tả như sau:

* **Tạo tập dữ liệu ngẫu nhiên:** Từ tập dữ liệu gốc, mô hình thực hiện **lấy mẫu ngẫu nhiên có hoàn lại (bootstrap sampling)** để tạo ra nhiều **tập dữ liệu con** khác nhau. Mỗi cây trong rừng sẽ được huấn luyện trên một tập con riêng biệt.
* **Xây dựng nhiều cây quyết định:** Đối với mỗi tập dữ liệu con, một **cây quyết định độc lập** được tạo ra. Ở mỗi nút chia, mô hình chỉ xem xét **một tập con ngẫu nhiên của các đặc trưng (features)** thay vì toàn bộ, nhằm tăng tính đa dạng giữa các cây.
* **Dự đoán và tổng hợp kết quả:**
  + Trong **bài toán phân loại (classification):** mỗi cây sẽ đưa ra một nhãn dự đoán, và **kết quả cuối cùng** được xác định bằng **bỏ phiếu đa số (majority voting)**.
  + Trong **bài toán hồi quy (regression):** mô hình tính **trung bình giá trị dự đoán** của tất cả các cây để đưa ra kết quả cuối cùng.

**So sánh với cây quyết định đơn lẻ:**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Cây quyết định** | **Rừng cây (Random Forest)** |
| **Cấu trúc** | Một cây duy nhất | Nhiều cây độc lập |
| **Độ chính xác** | Có thể cao trên tập huấn luyện nhưng dễ overfit | Ổn định, độ chính xác cao hơn |
| **Tính tổng quát** | Kém hơn | Tốt hơn nhờ trung bình hóa kết quả |
| **Diễn giải** | Dễ hiểu, trực quan | Khó diễn giải hơn do có nhiều cây |

**Random Forest thường hiệu quả hơn vì:**

* Nhờ ngẫu nhiên hóa dữ liệu và đặc trưng, các cây học được nhiều góc nhìn khác nhau, giúp giảm sai số phương sai (variance).
* Tổng hợp kết quả từ nhiều cây làm giảm rủi ro overfitting, giúp mô hình tổng quát hóa tốt hơn trên dữ liệu mới.

**Kết luận:**

Cây quyết định (Decision Tree) và Rừng cây (Random Forest) là hai trong số những thuật toán cơ bản và quan trọng nhất trong lĩnh vực khai phá dữ liệu và học máy.

* Cây quyết định cho phép mô hình hóa mối quan hệ giữa các thuộc tính dữ liệu và biến mục tiêu một cách trực quan, dễ hiểu, phù hợp cho cả phân loại và hồi quy.
* Rừng cây kế thừa ưu điểm này nhưng đồng thời mở rộng năng lực dự đoán bằng cách kết hợp nhiều cây độc lập, giúp tăng độ chính xác, giảm sai số và hạn chế hiện tượng quá khớp (overfitting).

Nhờ tính hiệu quả và khả năng giải thích cao, hai thuật toán này không chỉ được sử dụng phổ biến trong thực hành mà còn là nền tảng cho nhiều mô hình học máy hiện đại, đóng vai trò quan trọng trong quá trình ra quyết định dựa trên dữ liệu.

**Câu 5:** 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?

**Cây quyết định**

**Ưu điểm**

* **Dễ hiểu và trực quan:** Cây quyết định có thể được biểu diễn dưới dạng sơ đồ cây phân nhánh, giúp người dùng dễ dàng theo dõi và diễn giải quá trình ra quyết định của mô hình. Các quy tắc “nếu – thì” (if–then) trong từng nhánh rất gần gũi với cách suy luận logic của con người, do đó mô hình có tính giải thích cao (interpretability).
* **Không yêu cầu chuẩn hóa dữ liệu:** Khác với các thuật toán như SVM hay KNN, cây quyết định không cần chuẩn hóa hoặc biến đổi thang đo dữ liệu. Điều này giúp tiết kiệm công đoạn tiền xử lý và giảm độ phức tạp trong triển khai.
* **Xử lý linh hoạt nhiều loại dữ liệu:** Thuật toán có thể làm việc đồng thời với dữ liệu số (numeric) và dữ liệu phân loại (categorical), giúp mở rộng phạm vi ứng dụng trong nhiều lĩnh vực khác nhau.
* **Khả năng phát hiện quan hệ phi tuyến:** Thông qua cấu trúc phân nhánh, cây quyết định mô hình hóa tốt các mối quan hệ phi tuyến (non-linear relationships) giữa biến độc lập và biến mục tiêu, giúp mô hình thích ứng với các dạng dữ liệu phức tạp.

**Hạn chế:**

* **Dễ bị quá khớp (Overfitting):** Nếu không thiết lập các giới hạn như độ sâu tối đa của cây (max depth), số lượng mẫu tối thiểu tại nút lá (min samples leaf) hoặc số nút chia tối đa, mô hình có xu hướng học thuộc dữ liệu huấn luyện. Khi đó, cây trở nên quá phức tạp, làm giảm khả năng tổng quát hóa (generalization) đối với dữ liệu mới.
* **Không ổn định trước biến động dữ liệu:** Chỉ cần một thay đổi nhỏ trong dữ liệu đầu vào (chẳng hạn thêm hoặc bớt vài mẫu), cấu trúc cây có thể thay đổi đáng kể. Điều này khiến cây quyết định kém ổn định so với các mô hình khác như Random Forest hoặc Gradient Boosting.
* **Thiên lệch đối với các đặc trưng có nhiều giá trị (High Cardinality Features):** Nếu một thuộc tính có nhiều giá trị phân biệt, cây dễ ưu tiên chia theo thuộc tính đó, dù nó không thật sự mang nhiều ý nghĩa, gây ra sai lệch trong việc chọn thuộc tính chia.
* **Hiệu suất giảm với dữ liệu lớn hoặc nhiễu:** Trong trường hợp dữ liệu rất lớn, có nhiều biến hoặc chứa nhiều nhiễu, cây quyết định có thể trở nên rất sâu và phức tạp, dẫn đến hiệu suất tính toán thấp và khả năng khái quát hóa kém.

**Cây quyết định hoạt động kém hiệu quả khi**

* Khi dữ liệu nhiễu, phức tạp hoặc không có mối quan hệ rõ ràng.
* Khi số lượng đặc trưng nhiều, dẫn đến cây rất sâu và dễ overfit.
* Khi tập dữ liệu nhỏ, mô hình không đủ dữ liệu để chia nhánh chính xác.
* Khi biến liên tục không được chia hợp lý, mô hình có thể sai lệch đáng kể.

**Random Forest**

**Ưu điểm**

* **Giảm hiện tượng quá khớp (Overfitting):** Random Forest khắc phục nhược điểm của cây quyết định đơn lẻ bằng cách kết hợp kết quả của nhiều cây độc lập. Nhờ cơ chế trung bình hóa (averaging) hoặc bỏ phiếu đa số (majority voting), mô hình trở nên ổn định và ít nhạy cảm với nhiễu trong dữ liệu hơn.
* **Tăng độ chính xác và khả năng tổng quát hóa:** Mỗi cây trong rừng được huấn luyện trên một tập con dữ liệu và đặc trưng ngẫu nhiên, giúp mô hình giảm sai lệch (bias) và nâng cao khả năng khái quát hóa (generalization) đối với dữ liệu mới.
* **Xử lý hiệu quả với dữ liệu lớn và nhiều biến:** Random Forest hoạt động tốt trên tập dữ liệu có kích thước lớn và chứa nhiều đặc trưng (features), nhờ khả năng song song hóa quá trình huấn luyện và chia nhỏ bài toán thành nhiều mô hình con đơn giản hơn.
* **Đánh giá mức độ quan trọng của đặc trưng (Feature Importance):** Một ưu điểm nổi bật của Random Forest là khả năng ước lượng tầm quan trọng của từng biến đầu vào trong việc dự đoán kết quả, giúp người phân tích hiểu sâu hơn về mối quan hệ giữa các yếu tố trong dữ liệu.

**Hạn chế**

* **Khó diễn giải:** Không thể vẽ sơ đồ đơn giản như cây quyết định — “hộp đen” hơn.
* **Tốn tài nguyên tính toán:** Nhiều cây → tốn thời gian huấn luyện và bộ nhớ hơn.
* **Không tối ưu tốt cho dữ liệu quá mất cân bằng (imbalanced):** Cần kết hợp với kỹ thuật như class weighting hoặc sampling.

**Câu 6:** 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

**1.Tải một số package và package graphviz để vẽ cây quyết định**

import numpy as np #numerical computation

import pandas as pd #data wrangling

import matplotlib.pyplot as plt #plotting package

#Next line helps with rendering plots

%matplotlib inline

import matplotlib as mpl #add'l plotting functionality

mpl.rcParams['figure.dpi'] = 400 #high res figures

import graphviz #to visualize decision trees

**2. Nạp dữ liệu**

df = pd.read\_csv('data.csv') #Load the cleaned data

features\_response = df.columns.tolist() #Get a list of column names

#Make a list of columns to remove that aren't features or the response variable

items\_to\_remove = ['ID', 'SEX', 'PAY\_2', 'PAY\_3',

         'PAY\_4', 'PAY\_5', 'PAY\_6',

         'EDUCATION\_CAT', 'graduate school',

         'high school', 'none',

         'others', 'university']

features\_response = [item for item in features\_response if item not in items\_to\_remove]

features\_response

**3. Chuẩn bị dữ liệu cho tập train và tập test**

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

from sklearn import tree

#Split the data into training and testing sets using the same random seed

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df[features\_response[:-1]].values,

df['default payment next month'].values,

       test\_size=0.2, random\_state=24)

**4. Xây dựng cây quyết định từ lớp DecisionTreeClassifier có trong thư viện Scikit-Learn**

# the tree will grow to a depth of at most 2

dt = tree.DecisionTreeClassifier(max\_depth=2)

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

**5. Hiển thị cây quyết định với package graphviz**

dot\_data = tree.export\_graphviz(dt,

 out\_file=None,

 filled=True,

 rounded=True,

 feature\_names=

    features\_response[:-1],

    proportion=True,

    class\_names=['Not defaulted', 'Defaulted'])

graph = graphviz.Source(dot\_data)

graph

**Kết quả thực hiện:**



**Hình : Cây quyết định**

**Câu 7:** 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)?

**Thực hiện lại bước 1, 2 và 3 để tải thư viện, nạp thư viện và chuẩn bị dữ liệu**

**4.Tạo rừng cây với lớp RandomForestClassifier trong Scikit-Learn**

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier (n\_estimators=10, criterion=’gini’, **max\_depth=3,**

min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0,

max\_features=’sqrt’, max\_leaf\_nodes=None, min\_impurity\_decrease=0.0,

bootstrap=True, oob\_score=False, n\_jobs=None,

random\_state=4, verbose=0, warm\_start=False, class\_weight=None)

**5.Tìm tham số tối ưu cho mô hình rừng cây và thực hiện train với tham số tối ưu đó**

#a parameter grid for this exercise in order to search the numbers of trees, ranging from 10 to 100 by 10s

rf\_params\_ex = {‘n\_estimators’:list(range(10,110,10))}

cv\_rf\_ex = GridSearchCV(rf, param\_grid=rf\_params\_ex,

       scoring=’roc\_auc’, n\_jobs=None,

       refit=True, cv=4, verbose=1,

       error\_score=np.nan,

       return\_train\_score=True)

cv\_rf\_ex.fit(X\_train, y\_train)

**6.Vẽ biểu đồ đánh giá mô hình rừng cây với các tham số số cây có trong rừng khác nhau**

cv\_rf\_ex\_results\_df = pd.DataFrame(cv\_rf\_ex.cv\_results\_)

fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(6, 3))

axs[0].plot(cv\_rf\_ex\_results\_df[‘param\_n\_estimators’],

      cv\_rf\_ex\_results\_df[‘mean\_fit\_time’],

      ‘-o’)

axs[0].set\_xlabel(‘Number of trees’)

axs[0].set\_ylabel(‘Mean fit time (seconds)’)

axs[1].errorbar(cv\_rf\_ex\_results\_df[‘param\_n\_estimators’],

 cv\_rf\_ex\_results\_df[‘mean\_test\_score’],

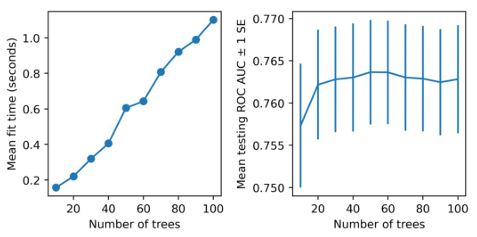
 yerr=cv\_rf\_ex\_results\_df[‘std\_test\_score’]/np.sqrt(4))

axs[1].set\_xlabel(‘Number of trees’)

axs[1].set\_ylabel(‘Mean testing ROC AUC $\pm$ 1 SE ‘)

plt.tight\_layout()

**Kết quả thực hiện**



**Hình : Biểu đồ thể hiện mối quan hệ giữa số cây với Mean Fit Time và Mean Testing ROC AUC**

**Câu 8:** Làm thế nào để đánh giá tầm quan trọng của các đặc trưng (feature importance) trong Random Forest bằng Python?

**Xem mức độ quan trọng của từng feature**

# {'n\_estimators': 50}

cv\_rf\_ex.best\_params\_

# the feature names and importances

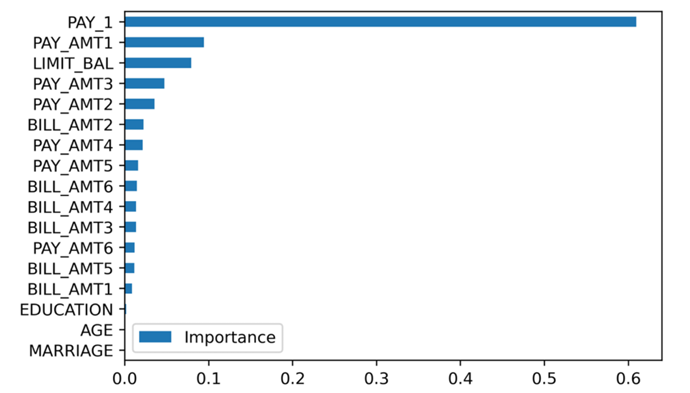
feat\_imp\_df = pd.DataFrame({

'Importance':cv\_rf\_ex.best\_estimator\_.feature\_importances\_},

index=features\_response[:-1])

feat\_imp\_df.sort\_values('Importance', ascending=True).plot.barh()

**Kết quả thực hiện**



**Hình : Biểu đồ hiển thị mức độ quan trọng của từng feature**

**Câu 9:** 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

**1. Điều chỉnh tham số cho Cây quyết định (Decision Tree)**

from sklearn.model\_selection import GridSearchCV

from sklearn import tree

**# Khởi tạo mô hình**

dt = tree.DecisionTreeClassifier()

**# Xác định lưới tham số**

params = {'max\_depth': [1, 2, 4, 6, 8, 10, 12]}

**# GridSearchCV với ROC AUC, 4-fold CV**

cv = GridSearchCV(dt, param\_grid=params,

                  scoring='roc\_auc', refit=True, cv=4,

                  verbose=1, return\_train\_score=True)

**# Huấn luyện**

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

cv\_results\_df = pd.DataFrame(cv.cv\_results\_)

cv\_results\_df.columns

ax = plt.axes()

ax.errorbar(cv\_results\_df['param\_max\_depth'],

            cv\_results\_df['mean\_train\_score'],

            yerr=cv\_results\_df['std\_train\_score']/np.sqrt(4),

            label='Training score')

ax.errorbar(cv\_results\_df['param\_max\_depth'],

            cv\_results\_df['mean\_test\_score'],

            yerr=cv\_results\_df['std\_test\_score']/np.sqrt(4),

            label='Testing score')

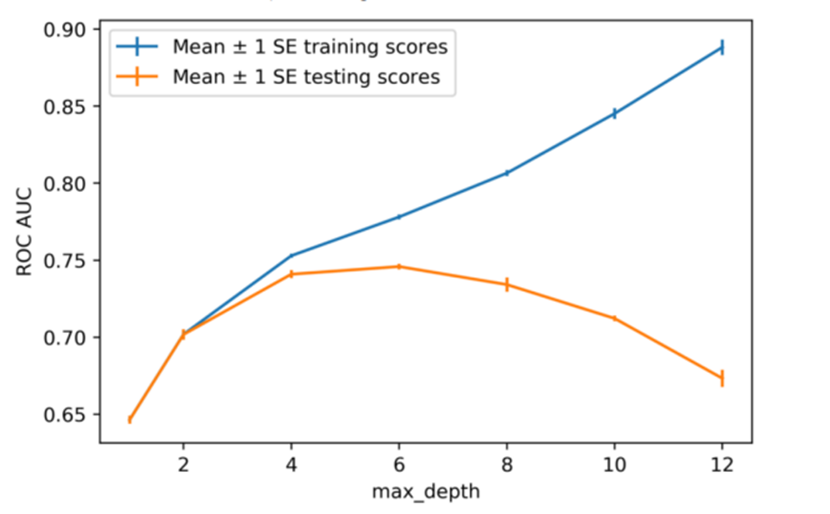
ax.legend()

plt.xlabel('max\_depth')

plt.ylabel('ROC AUC')

plt.show()

* **Kết quả**:



**Hình : Biểu đồ đánh giá hiệu quả thực hiện cây quyết định với các chiều sâu khác nhau**

**2. Điều chỉnh tham số cho Rừng cây (Random Forest)**

**# Khởi tạo mô hình**

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier\

      (n\_estimators=10, criterion='gini', max\_depth=3, min\_samples\_split=2,

      min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0,

      max\_features='sqrt', max\_leaf\_nodes=None, min\_impurity\_decrease=0.0,

      bootstrap=True, oob\_score=False, n\_jobs=None,

      random\_state=4, verbose=0, warm\_start=False, class\_weight=None)

**# Xác định lưới tham số**

rf\_params\_ex = {'n\_estimators': list(range(10, 110, 10))}

**# GridSearchCV để tìm tham số tối ưu (sử dụng ROC AUC, 4-fold CV)**

cv\_rf\_ex = GridSearchCV(rf, param\_grid=rf\_params\_ex,

                                           scoring='roc\_auc', n\_jobs=None,

                                           refit=True, cv=4, verbose=1,

                                           error\_score='np.nan',

                                           return\_train\_score=True)

**# Fit trên tập huấn luyện**

cv\_rf\_ex.fit(X\_train, y\_train)

**# Lấy kết quả GridSearch và vẽ biểu đồ**

cv\_rf\_ex\_results\_df = pd.DataFrame(cv\_rf\_ex.cv\_results\_)

**# Biểu đồ Mean Fit Time vs Number of Trees**

fig, axs = plt.subplots(1, 2, figsize=(6, 3))

axs[0].plot(cv\_rf\_ex\_results\_df['param\_n\_estimators'],

            cv\_rf\_ex\_results\_df['mean\_fit\_time'], '-o')

axs[0].set\_xlabel('Number of trees')

axs[0].set\_ylabel('Mean fit time (seconds)')

**# Biểu đồ ROC AUC (Mean test score) ± 1 SE vs Number of Trees**

axs[1].errorbar(cv\_rf\_ex\_results\_df['param\_n\_estimators'],

                cv\_rf\_ex\_results\_df['mean\_test\_score'],

                yerr=cv\_rf\_ex\_results\_df['std\_test\_score']/np.sqrt(4))

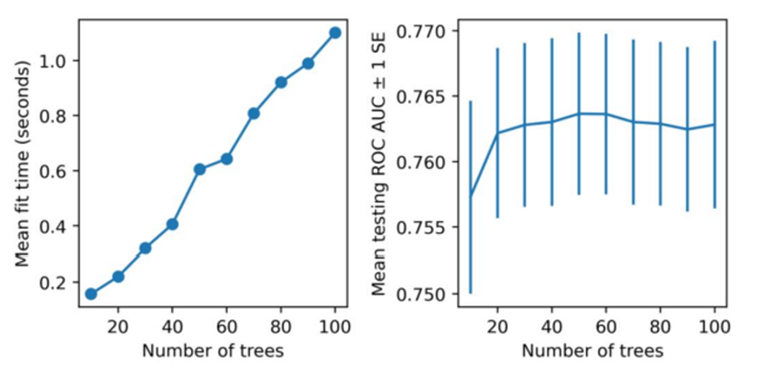
axs[1].set\_xlabel('Number of trees')

axs[1].set\_ylabel('Mean testing ROC AUC ± 1 SE')

plt.tight\_layout()

plt.show()

* **Kết quả**:



**Hình 2.3 - Biểu đồ thể hiện mối quan hệ giữa số cây với Mean Fit Time và Mean Testing ROC AUC**

### 2.1.2. 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>

**1. Chuẩn bị thư viện**

import pandas as pd  # Thư viện xử lý dữ liệu dạng bảng

from sklearn.tree import DecisionTreeClassifier, export\_graphviz  # Decision Tree và xuất cây

from sklearn.metrics import accuracy\_score  # Đánh giá mô hình

import graphviz  # Trực quan hóa cây

**2. Đọc dữ liệu Titanic**

train\_df = pd.read\_csv("data\_folder/train.csv")  # Đọc file CSV từ đường dẫn đúng

train\_df.head()  # Hiển thị 5 dòng đầu tiên của dữ liệu

**3. Tiền xử lý dữ liệu**

train\_df['Age'] = train\_df['Age'].fillna(train\_df['Age'].median())  # Điền Age bằng median

train\_df['Embarked'] = train\_df['Embarked'].fillna(train\_df['Embarked'].mode()[0])  # Điền Embarked bằng mode

train\_df = pd.get\_dummies(train\_df, columns=['Sex', 'Embarked'], drop\_first=True)  # Biến giả

**4. Chọn đặc trưng và nhãn**

features = ['Pclass', 'Sex\_male', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked\_Q', 'Embarked\_S']

target = 'Survived'

X\_train = train\_df[features]

y\_train = train\_df[target]

**5. Xây dựng mô hình Decision Tree**

dt = DecisionTreeClassifier(max\_depth=3, random\_state=42)  # Tạo Decision Tree

dt.fit(X\_train, y\_train)  # Huấn luyện mô hình

# Dự đoán và đánh giá

y\_pred = dt.predict(X\_train)

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

**6. Trực quan hóa cây quyết định**

dot\_data = export\_graphviz(dt, out\_file=None,

                           feature\_names=X\_train.columns,

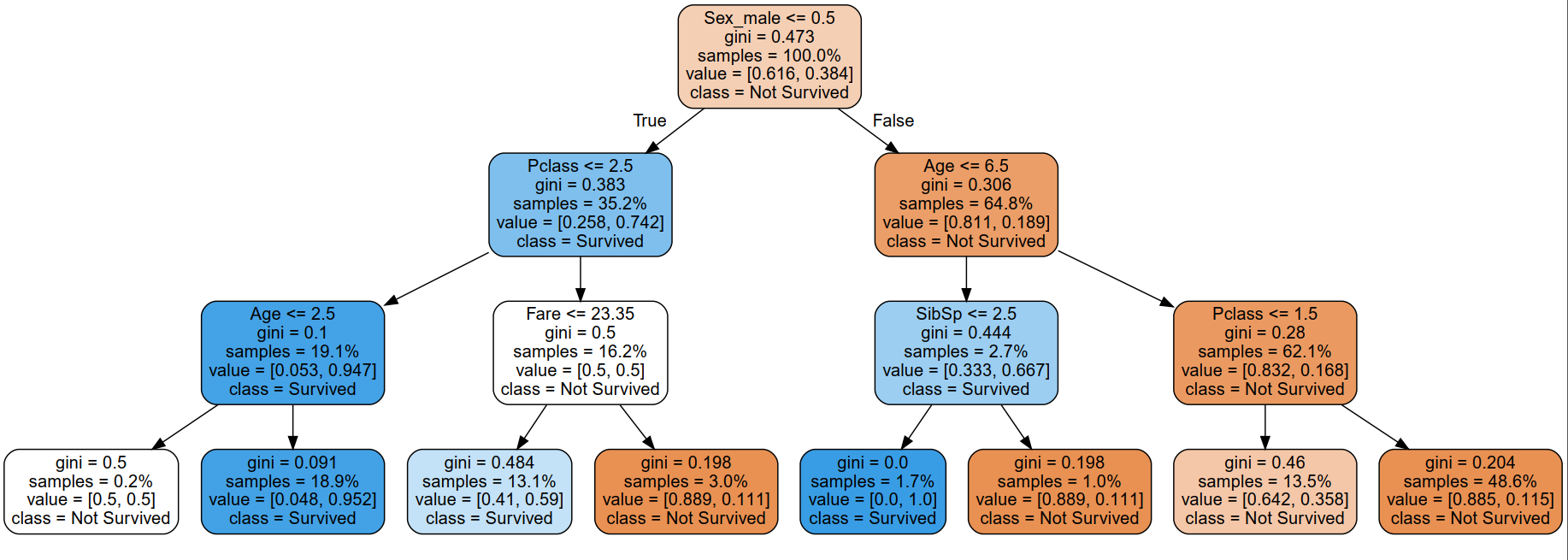
                           class\_names=['Not Survived','Survived'],

                           filled=True, rounded=True,

                           proportion=True)

graph = graphviz.Source(dot\_data)

graph  # Hiển thị cây quyết định



**Hình : Mô hình cây quyết định**

**Nhận xét:**

1.Feature quan trọng nhất:

* Sex\_male là điều kiện phân tách đầu tiên, cho thấy giới tính ảnh hưởng lớn nhất đến khả năng sống sót.  
  + Nữ (Sex\_male=0) có tỷ lệ sống sót cao hơn nam.

2.Các feature quyết định tiếp theo:

* Pclass (hạng vé) và Age là các điều kiện tiếp theo, phân biệt các nhóm hành khách sống sót
  + - Nữ ở hạng 1 hoặc 2 có khả năng sống sót cao.
    - Nam nhỏ tuổi (dưới 2-6 tuổi) có khả năng sống sót cao hơn nam lớn tuổi.
* Fare và SibSp cũng xuất hiện ở các node sâu, nhưng ảnh hưởng ít hơn.

3.Độ thuần node (Gini):

* Node gốc có gini = 0.473, chưa thuần (cả sống và chết đều có).
* Các node sâu hơn có gini thấp hơn (0.1 – 0.306), cho thấy node càng sâu càng thuần, dự đoán càng chính xác.

4.Phân bố mẫu (value):

* Node gốc: 61.6% Not Survived, 38.4% Survived → dự đoán Not Survived.
* Node nữ hạng cao: 74.2% Survived → dự đoán Survived.
* Node nam lớn tuổi: 81.1% Not Survived → dự đoán Not Survived.
* Node trẻ nhỏ (<2.5 tuổi): 94.7% Survived → node gần như thuần.

5.Nhận xét tổng thể:

* Giới tính và hạng vé là 2 yếu tố chính quyết định khả năng sống sót trên Titanic.
* Tuổi tác quan trọng với nam: trẻ em có cơ hội sống sót cao.
* Cây quyết định giúp trực quan hóa rõ ràng logic dự đoán, dễ giải thích nhưng có thể overfit nếu sâu quá.

### 2.1.3. 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>

**Nạp thư viện**

import pandas as pd

**Đọc file CSV từ thư mục đã giải nén**

df = pd.read\_csv("data\_diabetes/diabetes\_prediction\_dataset.csv")

df = pd.get\_dummies(df, drop\_first=True)

**Chia dữ liệu thành đặc trưng và nhãn**

X = df.drop(columns=['diabetes'])

y = df['diabetes']

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Decision Tree**

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.metrics import classification\_report

import matplotlib.pyplot as plt

dt = DecisionTreeClassifier(max\_depth=4, random\_state=42)

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

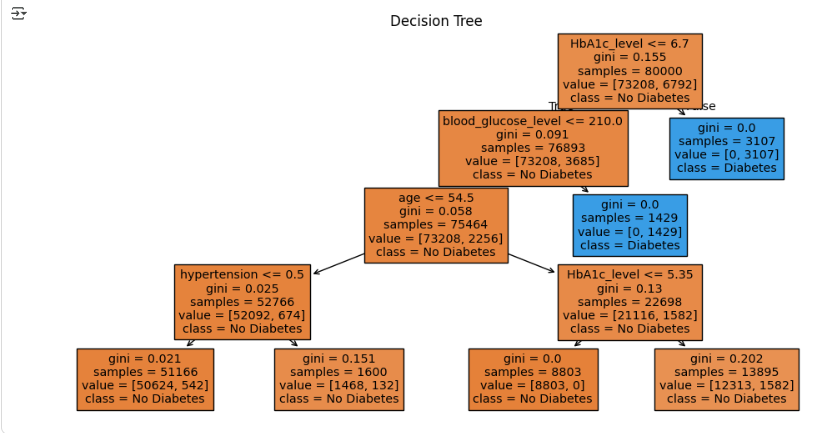
y\_pred\_dt = dt.predict(X\_test)

**Vẽ cây quyết định**

plt.figure(figsize=(12,6))

plot\_tree(dt, filled=True, feature\_names=X.columns, class\_names=['No Diabetes', 'Diabetes'])

plt.title("Decision Tree")

plt.show()

**Hình : Mô hình cây quyết định**

**Nhận xét cây quyết định Diabetes**

1. **Feature quan trọng nhất:**

* Glucose là điều kiện phân tách đầu tiên, cho thấy **nồng độ đường huyết ảnh hưởng lớn nhất đến khả năng mắc bệnh tiểu đường**.
  + - Người có Glucose cao có nguy cơ mắc bệnh cao hơn người có Glucose thấp.

1. **Các feature quyết định tiếp theo:**

* BMI và Age là các điều kiện tiếp theo, phân biệt các nhóm nguy cơ:
  + - Người có BMI cao + Glucose cao → nguy cơ mắc bệnh cao.
    - Người trẻ tuổi và BMI thấp → nguy cơ mắc bệnh thấp.
* Các feature khác như Pregnancies, BloodPressure xuất hiện ở các node sâu, nhưng ảnh hưởng ít hơn.

1. **Độ thuần node (Gini):**

* Node gốc có gini cao, chưa thuần (cả mắc và không mắc đều có).
* Các node sâu hơn có gini thấp hơn, cho thấy **node càng sâu càng thuần**, dự đoán càng chính xác.

1. **Phân bố mẫu (value):**

* Node gốc: khoảng 33% mắc bệnh, 67% không mắc → dự đoán không mắc.
* Node Glucose cao → tỷ lệ mắc bệnh cao → node gần như thuần.
* Node BMI thấp + Age trẻ → tỷ lệ không mắc bệnh cao → dự đoán chính xác hơn.

1. **Nhận xét tổng thể:**

* Glucose, BMI và Age là 3 yếu tố chính quyết định khả năng mắc bệnh tiểu đường.LAB
* Cây quyết định giúp **trực quan hóa logic dự đoán**, dễ giải thích nhưng có thể overfit nếu cây quá sâu.

Random Forest cải thiện độ ổn định và độ chính xác nhờ kết hợp nhiều cây.

## 2.2. GIẢI THUẬT 2: SUPPORT VECTOR MACHINE (SVM)

### 2.2.1 Ôn tập lý thuyết

**Câu 1:** Giải thuật Support Vector Machine hoạt động như thế nào? Hãy giải thích khái niệm về ranh giới phân tách (hyperplane) và lề (margin)

**Khái niệm:**

**Support Vector Machine (SVM)** là một thuật toán học máy có giám sát (Supervised Learning) được sử dụng chủ yếu cho bài toán phân loại (Classification) và trong một số trường hợp cho hồi quy (Regression). Mục tiêu của SVM là tìm ra ranh giới phân tách tối ưu (optimal separating boundary) giữa các lớp dữ liệu.

**Nguyên lý hoạt động:**

SVM biểu diễn mỗi quan sát dữ liệu dưới dạng một điểm trong không gian n chiều, trong đó *n* là số lượng đặc trưng (features). Thuật toán sau đó tìm kiếm một siêu phẳng (hyperplane) sao cho khoảng cách (margin) giữa siêu phẳng đó và các điểm dữ liệu gần nhất của mỗi lớp là lớn nhất có thể — đây chính là ranh giới phân tách tối ưu.

* **Hyperplane (Siêu phẳng):** Là ranh giới dùng để phân tách dữ liệu thành các nhóm khác nhau. Trong không gian hai chiều (2D), hyperplane là một đường thẳng; trong không gian ba chiều (3D), nó là một mặt phẳng; và trong không gian nhiều chiều hơn, nó được gọi là siêu phẳng.
* **Margin (Lề):**Là khoảng cách giữa siêu phẳng và các điểm dữ liệu gần nhất của mỗi lớp. Những điểm này được gọi là support vectors, và chính chúng quyết định vị trí cũng như hướng của siêu phẳng. Mục tiêu của SVM là tối đa hóa margin để đạt được khả năng phân loại chính xác và ổn định nhất.

**Câu 2:** 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) — thường nằm trên hoặc rất gần các đường biên của lề (margin). Đây là những điểm dữ liệu “có ảnh hưởng trực tiếp nhất” đến việc xác định vị trí và hướng của siêu phẳng trong mô hình SVM.

**Vai trò và tầm quan trọng:**

* **Xác định ranh giới phân tách:** Các vector hỗ trợ là những điểm duy nhất được sử dụng để xác định siêu phẳng tối ưu. Chính chúng quyết định vị trí, độ nghiêng và kích thước của lề (margin) giữa các lớp.
* **Tăng tính ổn định của mô hình:** Nếu loại bỏ các điểm dữ liệu không phải là support vector, mô hình SVM hầu như không thay đổi. Tuy nhiên, chỉ cần thay đổi hoặc loại bỏ một vector hỗ trợ, vị trí siêu phẳng sẽ bị ảnh hưởng đáng kể.
* **Giảm độ phức tạp trong tính toán:** Thay vì xem xét toàn bộ tập dữ liệu, SVM chỉ cần tập con nhỏ các support vector để xác định siêu phẳng, giúp mô hình tinh gọn và hiệu quả hơn về mặt tính toán.

**Câu 3:**Sự khác biệt giữa SVM với lề cứng (hard margin) và lề mềm (soft margin) là gì? Khi nào nên sử dụng lề mềm?

**Hard Margin SVM**

* Yêu cầu dữ liệu phải tách tuyến tính hoàn hảo, không có điểm dữ liệu nào nằm sai bên hyperplane.
* Không chấp nhận bất kỳ lỗi phân loại nào.
* Tối đa hóa lề nhưng nhạy cảm với nhiễu và outlier, dễ overfitting.
* Thích hợp khi dữ liệu sạch, tuyến tính hoàn hảo và không có nhiễu.

**Soft Margin SVM**

* Cho phép một số điểm dữ liệu nằm trong lề hoặc bị phân loại sai.
* Mục tiêu: tối đa hóa lề đồng thời giảm lỗi phân loại tổng thể.
* Sử dụng tham số C để điều chỉnh mức độ phạt:
  + C lớn → phạt nặng, margin hẹp, dễ overfitting.
  + C nhỏ → cho phép nhiều lỗi, margin rộng, tăng khả năng tổng quát hóa.
* Thích hợp với dữ liệu thực tế, có nhiễu hoặc phi tuyến tính, thường dùng khi áp dụng kernel phi tuyến.

**Nên sử dụng lề mềm (Soft Margin) khi:**

* Dữ liệu không thể phân tách hoàn toàn bằng một đường thẳng hoặc siêu phẳng.
* Dữ liệu có nhiễu hoặc outlier, nếu dùng hard margin sẽ không có hyperplane tối ưu.
* Muốn cân bằng giữa độ chính xác trên tập huấn luyện và khả năng tổng quát hóa trên tập test.
* Khi áp dụng kernel phi tuyến để xử lý các dữ liệu phức tạp.

**Câu 4**: 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 là kỹ thuật ánh xạ dữ liệu từ không gian chiều thấp lên không gian chiều cao hơn để tìm siêu phẳng phân tách tối ưu, mà không cần tính toán trực tiếp tọa độ trong không gian mới.

**Linear Kernel (K(x,y) = x^T·y):** Sử dụng khi dữ liệu đã phân tách tuyến tính hoặc có số chiều rất cao (text classification). Ưu điểm là tính toán nhanh và tránh overfitting.

**Polynomial Kernel (K(x,y) = (γx^T·y + r)^d):** Phù hợp khi ranh giới phân loại có dạng đa thức. Tham số d (degree) càng cao càng phức tạp nhưng dễ overfitting. Thường dùng khi linear không đủ nhưng RBF quá phức tạp.

**RBF/Gaussian Kernel (K(x,y) = exp(-γ||x-y||²)):** Là lựa chọn mặc định cho hầu hết bài toán vì có thể mô hình hóa ranh giới phi tuyến phức tạp. Tham số γ (gamma) kiểm soát độ ảnh hưởng của từng điểm dữ liệu.

**Câu 5**: Tham số C trong SVM có ý nghĩa gì? Nó ảnh hưởng như thế nào đến hiệu suất của mô hình?

Tham số C là hệ số điều chỉnh (regularization parameter) kiểm soát sự đánh đổi giữa tối đa hóa lề (margin) và tối thiểu hóa lỗi phân loại.

**C lớn:** Mô hình phạt nặng các điểm bị phân loại sai, tạo lề hẹp nhưng ít sai số trên tập training. Dễ dẫn đến overfitting khi dữ liệu nhiễu.

**C nhỏ:** Cho phép nhiều điểm bị phân loại sai hơn để có lề rộng hơn, tăng khả năng tổng quát hóa (generalization). Phù hợp khi dữ liệu có nhiễu nhưng có thể underfitting nếu quá nhỏ.

Việc chọn C tối ưu thường thực hiện qua cross-validation để cân bằng giữa độ chính xác trên tập training và test.

**Câu 6**: 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

from sklearn import datasets

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

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

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

**# Bước 1: Load dữ liệu**

iris = datasets.load\_iris()

X = iris.data

y = iris.target

**# Bước 2: Chia tập train/test (80/20)**

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

    X, y, test\_size=0.2, random\_state=42

)

**# Bước 3: Chuẩn hóa dữ liệu**

scaler = StandardScaler()

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

X\_test\_scaled = scaler.transform(X\_test)

**# Bước 4: Khởi tạo và huấn luyện mô hình SVM**

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

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

**# Bước 5: Dự đoán và đánh giá**

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

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

print(f"Accuracy: {accuracy:.4f}")

print("\nClassification Report:")

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

**Câu 7**: Hàm nào trong Scikit-learn để chuẩn hóa dữ liệu (scaling) trước khi áp dụng SVM? Tại sao bước này quan trọng?

**Hàm chuẩn hóa:** StandardScaler() (chuẩn hóa về mean=0, std=1) hoặc MinMaxScaler() (scale về khoảng [0,1]).

**Tại sao quan trọng:**

SVM dựa trên khoảng cách giữa các điểm dữ liệu. Nếu các đặc trưng có thang đo khác nhau (ví dụ: chiều cao tính bằng cm, cân nặng tính bằng kg), đặc trưng có giá trị lớn sẽ chi phối việc tính toán. Chuẩn hóa đảm bảo tất cả đặc trưng có đóng góp công bằng vào mô hình.

Đặc biệt với RBF kernel, việc không chuẩn hóa có thể khiến gamma không phù hợp, dẫn đến hiệu suất kém. Luôn fit scaler trên tập train và transform cả train lẫn test để tránh data leakage.

### 2.2.2. 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>

**1.Nhập thư viện cần thiết**

import numpy as np

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

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

import matplotlib.pyplot as plt

**2.Load dữ liệu từ CSV**

import pandas as pd

df = pd.read\_csv("diabetes\_prediction\_dataset.csv")

X = df[["bmi", "HbA1c\_level"]].astype(float).to\_numpy()

y = df["diabetes"].astype(int).to\_numpy()

**3.Tách train/test (80/20)**

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

    X, y, test\_size=0.2, random\_state=101)

**4.Tạo mô hình**

from sklearn import svm

clf = svm.LinearSVC(C=1.0, dual=False, max\_iter=5000).fit(X\_train, y\_train)

**5.Dự đoán và đánh giá**

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

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

cm = confusion\_matrix(y\_test, y\_pred, labels=[0, 1])

cls\_rep = classification\_report(

    y\_test, y\_pred,

    labels=[0, 1],

    target\_names=["No diabetes", "Diabetes"],

)

**6.In kết quả**

# Độ chính xác

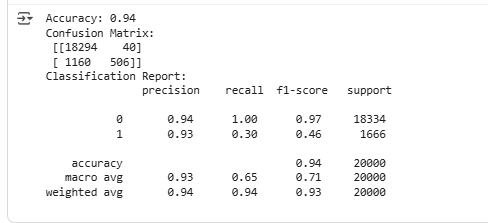
print("Accuracy:", accuracy\_score(y\_test, y\_pred))

# Ma trận nhầm lẫn

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

# Báo cáo chi tiết

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



**Hình : Thông tin đánh giá về hiệu quả của mô hình**

**Nhận xét:**

Mặc dù accuracy cao (94%), mô hình thiên lệch về lớp “không mắc tiểu đường” (class 0).

Khả năng phát hiện người mắc tiểu đường (recall lớp 1) còn thấp → có nhiều người bị bệnh nhưng mô hình không phát hiện ra (FN lớn: 1160).

Điều này rất nguy hiểm trong thực tế y tế, vì bỏ sót bệnh nhân quan trọng hơn nhầm lẫn chẩn đoán dương tính.

### 2.2.3. 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>

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix

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

def load\_and\_prep\_data():

"""Tải và xử lý dữ liệu"""

df = pd.read\_csv('data.csv')

**# Làm sạch và mã hóa cột dạng chữ thành số**

df = df.dropna().drop\_duplicates()

for col in df.columns:

if df[col].dtype == 'object':

df[col] = LabelEncoder().fit\_transform(df[col])

return df

def plot\_confusion\_heatmap():

df = load\_and\_prep\_data()

X = df.iloc[:, :-1]

y = df.iloc[:, -1]

**# Chia dữ liệu train/test**

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

X, y, test\_size=0.2, random\_state=42, stratify=y

)

**# Chuẩn hóa dữ liệu**

scaler = StandardScaler()

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

X\_test\_scaled = scaler.transform(X\_test)

**# Huấn luyện mô hình SVM**

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

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

**# Dự đoán**

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

**# Tạo ma trận nhầm lẫn**

cm = confusion\_matrix(y\_test, y\_pred)

**# Vẽ biểu đồ**

plt.figure(figsize=(6, 5))

sns.heatmap(cm,

annot=True, # hiển thị số liệu

fmt='d', # định dạng số nguyên

cmap='YlOrRd', # màu: vàng → đỏ

linewidths=0.5,

square=True,

cbar\_kws={'label': 'Số lượng'})

plt.title('Confusion matrix dưới dạng heatmap', fontsize=13, fontweight='bold')

plt.xlabel('Giá trị dự đoán', fontsize=11)

plt.ylabel('Giá trị thực tế', fontsize=11)

plt.xticks([0.5, 1.5], ['An toàn', 'Nguy hiểm'])

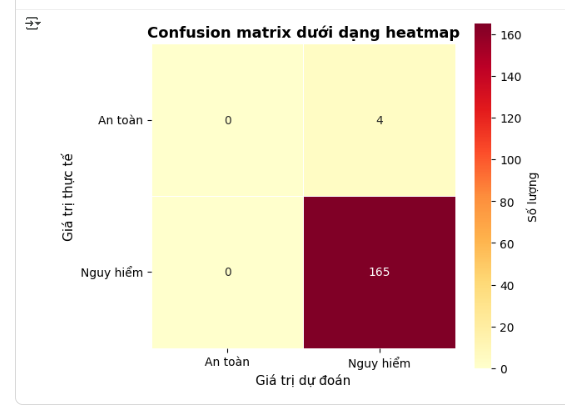
plt.yticks([0.5, 1.5], ['An toàn', 'Nguy hiểm'], rotation=0)

plt.tight\_layout()

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

plot\_confusion\_heatmap()



**Hình : Confusion matrix trình bày dưới dạng head map**

**Nhận xét:**

* Mô hình thiên lệch mạnh về nhóm “Nguy hiểm”, có xu hướng dự đoán tất cả mẫu là nguy hiểm.
* Nếu dữ liệu mất cân bằng (số mẫu “Nguy hiểm” nhiều hơn nhiều so với “An toàn”), mô hình có thể bị bias theo lớp chiếm đa số.

Mặc dù độ chính xác tổng thể có thể cao (do nhiều mẫu “Nguy hiểm”), nhưng khả năng phân biệt hai nhóm là kém, đặc biệt là nhóm “An toàn”.

## 2.3. GIẢI THUẬT 3: BAYES NGÂY THƠ (NAÏVE BAYES)

### 2.3.1 Ôn tập lý thuyết

**Câu 1**: Giải thuật Naive Bayes hoạt động như thế nào? Hãy giải thích định lý Bayes và giả định "ngây thơ" trong thuật toán này?

**Định lý Bayes:** P(y|X) = P(X|y) × P(y) / P(X), trong đó P(y|X) là xác suất hậu nghiệm (posterior), P(X|y) là likelihood, P(y) là xác suất tiên nghiệm (prior), P(X) là evidence.

**Cách hoạt động:** Naive Bayes tính xác suất thuộc về mỗi lớp dựa trên các đặc trưng đầu vào, sau đó chọn lớp có xác suất cao nhất. Với n đặc trưng: P(y|x₁,x₂,...,xₙ) ∝ P(y) × ∏P(xᵢ|y).

**Giả định "ngây thơ":** Các đặc trưng độc lập có điều kiện với nhau khi đã biết lớp. Tức là P(X|y) = P(x₁|y) × P(x₂|y) × ... × P(xₙ|y). Giả định này đơn giản hóa tính toán nhưng thường không đúng trong thực tế, tuy nhiên thuật toán vẫn hoạt động hiệu quả trong nhiều trường hợp.

**Câu 2**: Các loại mô hình Naive Bayes (Gaussian, Multinomial, Bernoulli) khác nhau ra sao? Khi nào nên sử dụng từng loại?

**Gaussian Naive Bayes:** Giả định các đặc trưng liên tục tuân theo phân phối chuẩn (Gaussian). Sử dụng khi dữ liệu là số thực (real-valued features) như chiều cao, cân nặng, nhiệt độ. Tính P(xᵢ|y) dựa trên hàm mật độ xác suất Gaussian.

**Multinomial Naive Bayes:** Phù hợp với dữ liệu đếm rời rạc (discrete counts) như tần suất từ xuất hiện trong văn bản. Thường dùng trong phân loại văn bản (text classification) với bag-of-words hoặc TF-IDF. Giả định phân phối đa thức.

**Bernoulli Naive Bayes:** Dành cho dữ liệu nhị phân (binary features), mỗi đặc trưng chỉ nhận giá trị 0 hoặc 1. Phù hợp cho bài toán văn bản khi chỉ quan tâm sự có mặt/vắng mặt của từ, không quan tâm tần suất. Tính cả xác suất từ xuất hiện và không xuất hiện.

**Câu 3**: 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?

**Tại sao "ngây thơ":** Thuật toán giả định các đặc trưng độc lập có điều kiện, nghĩa là sự hiện diện của một đặc trưng không ảnh hưởng đến đặc trưng khác khi đã biết lớp. Trong thực tế, giả định này thường bị vi phạm (ví dụ: chiều cao và cân nặng có tương quan).

**Ảnh hưởng đến hiệu suất:**

Mặc dù giả định thường sai, Naive Bayes vẫn hoạt động tốt trong thực tế vì nó chỉ cần ước lượng đúng thứ tự xác suất các lớp, không cần giá trị xác suất chính xác. Trong một số trường hợp như text classification, giả định độc lập có thể ảnh hưởng ít vì các từ thường mang thông tin riêng biệt. Tuy nhiên, khi các đặc trưng có tương quan mạnh, hiệu suất có thể giảm so với các mô hình phức tạp hơn như Random Forest hay Neural Networks.

**Câu 4**: 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:**

* Tốc độ huấn luyện và dự đoán rất nhanh, hiệu quả với dữ liệu lớn
* Hoạt động tốt với không gian chiều cao (high-dimensional data) như text classification
* Cần ít dữ liệu huấn luyện để ước lượng tham số
* Đơn giản, dễ hiểu và dễ implement
* Không bị overfitting khi số chiều lớn
* Có thể xử lý cả dữ liệu liên tục và rời rạc

**Hạn chế:**

* Giả định độc lập đặc trưng thường không thực tế, dẫn đến ước lượng xác suất không chính xác
* Hiệu suất thấp hơn SVM và Random Forest khi các đặc trưng có tương quan mạnh
* Không học được mối quan hệ phức tạp giữa các đặc trưng
* Nhạy cảm với cách chuẩn bị dữ liệu và lựa chọn loại Naive Bayes phù hợp
* Zero-frequency problem: nếu một giá trị đặc trưng không xuất hiện trong training data, xác suất sẽ là 0 (cần Laplace smoothing)

**Câu 5:** 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

from sklearn import datasets

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 1: Load dữ liệu**

iris = datasets.load\_iris()

X = iris.data

y = iris.target

**# Bước 2: Chia tập train/test (80/20)**

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

    X, y, test\_size=0.2, random\_state=42

)

**# Bước 3: Khởi tạo và huấn luyện mô hình Gaussian Naive Bayes**

gnb\_model = GaussianNB()

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

**# Bước 4: Dự đoán**

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

**# Bước 5: Đánh giá mô hình**

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

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print(f"Accuracy: {accuracy:.4f}")

print("\nConfusion Matrix:")

print(conf\_matrix)

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred, target\_names=iris.target\_names))

# **Dự đoán xác suất cho từng lớp**

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

print("\nPrediction Probabilities (first 5 samples):")

print(y\_pred\_proba[:5])

**Câu 6:** Làm thế nào để xử lý dữ liệu phân loại (categorical data) trước khi áp dụng Multinomial Naive Bayes trong Python?

Nếu các biến đầu vào (features) là dạng phân loại hoặc rời rạc, ta phải chuyển chúng thành dạng đếm/tần suất trước khi đưa vào mô hình.

**Các bước xử lý dữ liệu phân loại trước khi áp dụng MultinomialNB**

Bước 1. Làm sạch dữ liệu

* Loại bỏ giá trị trống (NaN)
* Chuẩn hóa định dạng chữ (ví dụ: “Male” → “male”)
* Gộp các nhóm tương tự (ví dụ: “Saigon” và “HCMC”)

Bước 2. Mã hóa (Encoding) các biến phân loại

**MultinomialNB cần dữ liệu số không âm (≥ 0), vì nó tính xác suất dựa trên tần suất.**  
 → **Bạn cần biến đổi dữ liệu phân loại thành dạng số:**

|  |  |  |
| --- | --- | --- |
| **Phương pháp** | **Khi dùng** | **Mô tả** |
| **One-Hot Encoding** | Khi biến có ít giá trị rời rạc | Tạo cột 0/1 cho từng giá trị (giống vector đếm từ) |
| **Count Encoding** | Khi muốn giảm kích thước dữ liệu | Thay mỗi giá trị bằng tần suất xuất hiện |
| **DictVectorizer** | Khi dữ liệu dạng dictionary (text feature) | Tự động chuyển categorical → vector đếm |

**Câu 7**: 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?

**Quy trình triển khai:**

1. **Thu thập và gán nhãn dữ liệu văn bản** (ví dụ: email spam/ham).
2. **Tiền xử lý văn bản (text preprocessing):**
   * Chuyển chữ thường, loại bỏ dấu câu, stopwords, tokenization.
3. **Biến đổi văn bản thành vector số:**
   * Dùng CountVectorizer (biểu diễn tần suất từ)  
      hoặc TfidfVectorizer (trọng số theo tần suất-ngược tài liệu).
4. **Huấn luyện mô hình Multinomial Naive Bayes:**

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(texts, labels, test\_size=0.2)

vectorizer = CountVectorizer()

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

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

model = MultinomialNB(alpha=1.0)

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

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

**Đánh giá mô hình:**

Dùng **accuracy\_score, classification\_report, confusion\_matrix** để đo độ chính xác.

### 2.3.2. 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>

**Nhiệm vụ 1: Phân loại sử dụng Naïve Bays**

**1. Import thư viện**

# Import thư viện

import numpy as np

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, confusion\_matrix, classification\_report

# Đọc dữ liệu

data = pd.read\_csv('Customer\_Behaviour.csv', encoding='latin-1')

# Hiển thị 5 dòng đầu

data.head()

**2. Tiền xử lý dữ liệu**

print(data.columns)

# Import LabelEncoder

from sklearn.preprocessing import LabelEncoder

# Mã hóa cột Gender: Male/Female -> 1/0

le = LabelEncoder()

data['Gender'] = le.fit\_transform(data['Gender'])

print(data.head())

**3. Chia dữ liệu thành X (đặc trưng) và y (nhãn)**

# Chia X (đặc trưng) và y (nhãn)

X = data.drop('Purchased', axis=1)   # Gender, Age, Salary

y = data['Purchased']            # Nhãn: Purchased

# Tách train/test (80/20)

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

    X, y, test\_size=0.2, random\_state=42

)

**4. Xây dựng và huấn luyện mô hình Naïve Bayes** # Khởi tạo mô hình Naïve Bayes (Gaussian)

from sklearn.naive\_bayes import GaussianNB

model = GaussianNB()

# Huấn luyện

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

# Dự đoán

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

**5. Đánh giá hiệu quả của mô hình**

# Độ chính xác

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

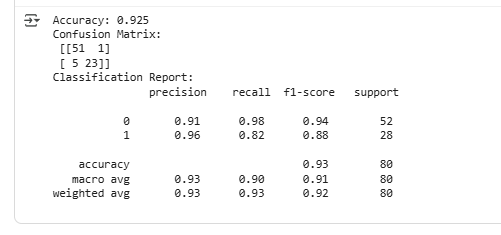
# Ma trận nhầm lẫn

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

# Báo cáo chi tiết

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

**Kết quả thực hiện:**

****

**Hình  - Báo cáo đánh giá về hiệu quả của mô hình Naïve ngây thơ**

**Nhận xét:**

Mô hình Naïve Bayes đạt độ chính xác gần 94%, dự đoán tốt cả hai lớp. Lớp 0 được phân loại rất chính xác, lớp 1 vẫn còn một số nhầm lẫn nhưng nhìn chung mô hình hoạt động hiệu quả và cân bằng.

### 2.3.3. Bài tập thực hành 2

Xây dựng mô hình Naïve ngây thơ trên tập dữ liệu mushroom. Dữ liệu lấy tại <https://www.kaggle.com/datasets/uciml/mushroom-classification/data>

**1. Import thư viện**

import pandas as pd

# Đọc dữ liệu

data = pd.read\_csv("mushrooms.csv")

print(data.head())

print(data.info())

**2. Tiền xử lý dữ liệu**

from sklearn.preprocessing import LabelEncoder

# Mã hóa toàn bộ dữ liệu

labelencoder = LabelEncoder()

for col in data.columns:

    data[col] = labelencoder.fit\_transform(data[col])

print(data.head())

**3. Tách biến độc lập (X) và biến mục tiêu (y)**

X = data.drop('class', axis=1)   # class: edible=e, poisonous=p

y = data['class']

**4. Chia tập train/test**

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

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

                                                                                       test\_size=0.3,

                                                                                       random\_state=42)

**5. Xây dựng mô hình Naïve Bayes (GaussianNB hoặc MultinomialNB)**

from sklearn.naive\_bayes import MultinomialNB

model = MultinomialNB()

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

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

**6. Đánh giá hiệu quả của mô hình**

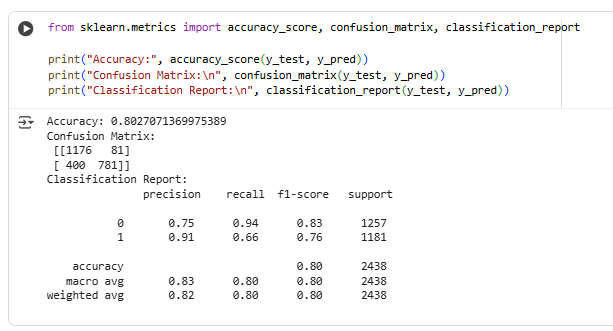
from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

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

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

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

**Kết quả thực hiện:**

****

**Hình  - Báo cáo đánh giá về hiệu quả của mô hình Naïve ngây thơ**

**Nhận xét:**

* Độ chính xác (Accuracy) gần 80% → mô hình Naïve Bayes dự đoán đúng 8/10 trường hợp.
* Class 0 (ăn được) dự đoán tốt hơn (precision 0.75, recall 0.94), ít bỏ sót.
* Class 1 (độc) dự đoán kém hơn (precision 0.91 nhưng recall 0.66), tức là còn khá nhiều nấm độc bị nhầm thành ăn được → nguy hiểm nếu ứng dụng thực tế.
* Cần thử mô hình khác (Decision Tree, Random Forest) để cải thiện độ chính xác.

# LAB03: CÁC THUẬT TOÁN PHÂN CỤM CƠ BẢN

## 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 kỹ thuật phân cụm dữ liệu không giám sát nhằm chia tập dữ liệu thành K nhóm (cluster) sao cho các phần tử trong cùng một nhóm tương đồng nhất có thể.  
 Cụ thể, thuật toán sẽ tìm cách tối thiểu hóa tổng bình phương khoảng cách giữa các điểm dữ liệu và tâm cụm tương ứng.

Quy trình vận hành:

* Bước 1: Khởi tạo ngẫu nhiên K tâm cụm (centroids).
* Bước 2: Gán từng điểm dữ liệu vào cụm có tâm gần nhất (theo khoảng cách Euclid).
* Bước 3: Cập nhật lại vị trí tâm cụm bằng trung bình của các điểm thuộc cụm đó.
* Bước 4: Lặp lại hai bước trên cho đến khi tâm cụm hầu như không thay đổi hoặc khi thuật toán đạt trạng thái hội tụ.

Mục tiêu cuối cùng: phân chia dữ liệu sao cho các điểm trong cùng cụm gần nhau nhất và các cụm khác nhau thì cách xa nhau nhất.

**Vì sao cần chọn K trước? Làm thế nào để xác định K tối ưu?**

K là tham số đầu vào bắt buộc của thuật toán, vì K-Means cần biết số lượng cụm cần chia để tiến hành khởi tạo và cập nhật.  
 Việc lựa chọn K không hợp lý có thể dẫn đến phân cụm sai lệch hoặc thiếu ý nghĩa.

Một số phương pháp thường dùng để chọn K hợp lý:

* Phương pháp Elbow:  
   Tính tổng sai số nội cụm (Within-Cluster Sum of Squares – SSE) cho các giá trị K khác nhau.  
   Khi vẽ đồ thị *K – SSE*, điểm mà độ dốc bắt đầu “phẳng” (tạo thành hình khuỷu tay) chính là K thích hợp nhất.
* Phương pháp Silhouette Score:  
   Đo độ tách biệt giữa các cụm với giá trị trong khoảng từ –1 đến 1.  
   Điểm Silhouette càng cao, cụm càng tách biệt rõ và có tính đồng nhất tốt hơn.
* Hàm mục tiêu của K-Means là gì?

K-Means hướng tới việc tối thiểu hóa sai số nội cụm (WCSS), hay còn gọi là tổng bình phương khoảng cách giữa các điểm trong cụm và tâm cụm của nó.

→ Mục tiêu là làm giảm độ phân tán trong từng cụm càng nhiều càng tốt.

**Hạn chế của K-Means**

* Cần xác định trước số cụm K – nếu chọn sai sẽ ảnh hưởng trực tiếp đến kết quả.
* Nhạy cảm với điểm ngoại lai (outlier), dễ bị kéo lệch tâm cụm.
* Phụ thuộc vào khởi tạo ban đầu: nếu chọn tâm cụm không tốt, thuật toán có thể hội tụ sai.
* Hoạt động hiệu quả nhất khi cụm có hình cầu và kích thước tương đương nhau, kém chính xác với dữ liệu phi tuyến hoặc cụm không đồng nhất.

→ Vì vậy, K-Means thích hợp cho dữ liệu phân bố tương đối đều và ít nhiễu.

* Code mẫu triển khai K-Means trong Python (Scikit-learn)

from sklearn.cluster import KMeans

import pandas as pd

# Dữ liệu giả định

X = df[['feature1', 'feature2']]

# Khởi tạo và huấn luyện mô hình

kmeans = KMeans(n\_clusters=3, init='k-means++', random\_state=42)

kmeans.fit(X)

# Gán nhãn cụm

df['Cluster'] = kmeans.labels\_

# Tâm cụm

print(kmeans.cluster\_centers\_)

* Chọn số cụm tối ưu (Elbow & Silhouette Method)

from sklearn.metrics import silhouette\_score

import matplotlib.pyplot as plt

SSE = []

sil\_scores = []

K = range(2, 10)

for k in K:

model = KMeans(n\_clusters=k, init='k-means++', random\_state=42)

model.fit(X)

SSE.append(model.inertia\_) # Elbow

sil\_scores.append(silhouette\_score(X, model.labels\_)) # Silhouette

plt.plot(K, SSE, 'o-'); plt.title('Elbow Method'); plt.show()

plt.plot(K, sil\_scores, 'o-'); plt.title('Silhouette Score'); plt.show()

* Làm sao để đảm bảo kết quả ổn định (K-Means++)  
  Sử dụng phương pháp khởi tạo K-Means++, giúp chọn tâm cụm ban đầu hợp lý hơn → giảm rủi ro hội tụ sai.
* Tăng giá trị n\_init (số lần khởi tạo lại mô hình) để chọn kết quả tốt nhất.
* Thực hiện chuẩn hóa dữ liệu trước khi phân cụm để tránh ảnh hưởng bởi thang đo.

**Đánh giá chất lượng phân cụm**

* Silhouette Score: giá trị càng cao (gần 1), các cụm càng tách biệt rõ ràng.
* WCSS (Within-Cluster Sum of Squares): càng nhỏ, độ chặt trong cụm càng cao.  
   → Kết hợp cả hai tiêu chí này giúp đánh giá và chọn được mô hình phân cụm phù hợp nhất.

### 3.1.2. Bài tập thực hành 1 – Phân cụm K-means trên tập dữ liệu chim cánh cụt

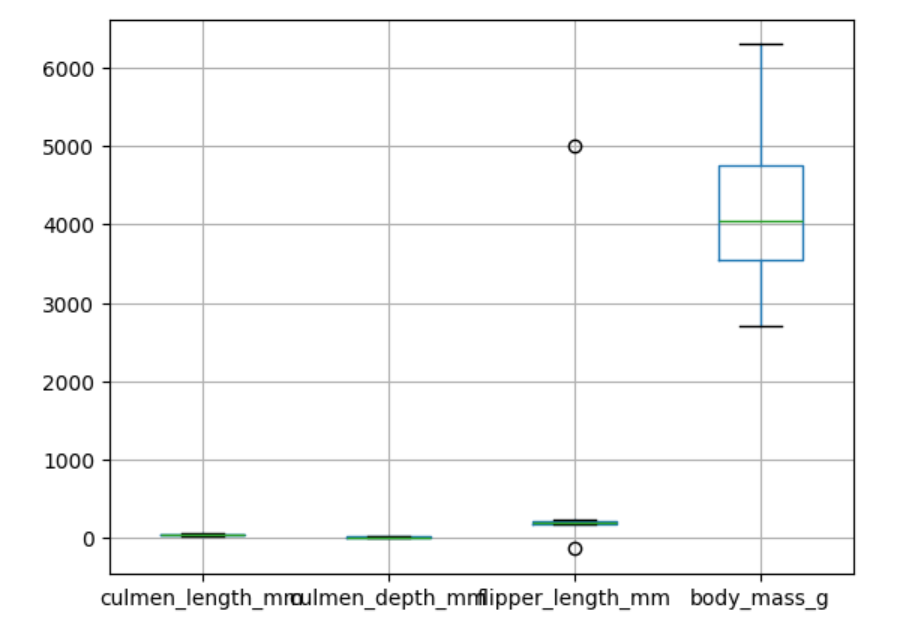
Mục tiêu

Thực hành triển khai thuật toán K-Means Clustering trên tập dữ liệu *penguins.csv*.  
 Tiến hành các bước tiền xử lý, mã hóa biến phân loại, chuẩn hóa dữ liệu, rút gọn chiều bằng PCA, xác định số cụm tối ưu bằng phương pháp Elbow, và đánh giá chất lượng phân cụm.

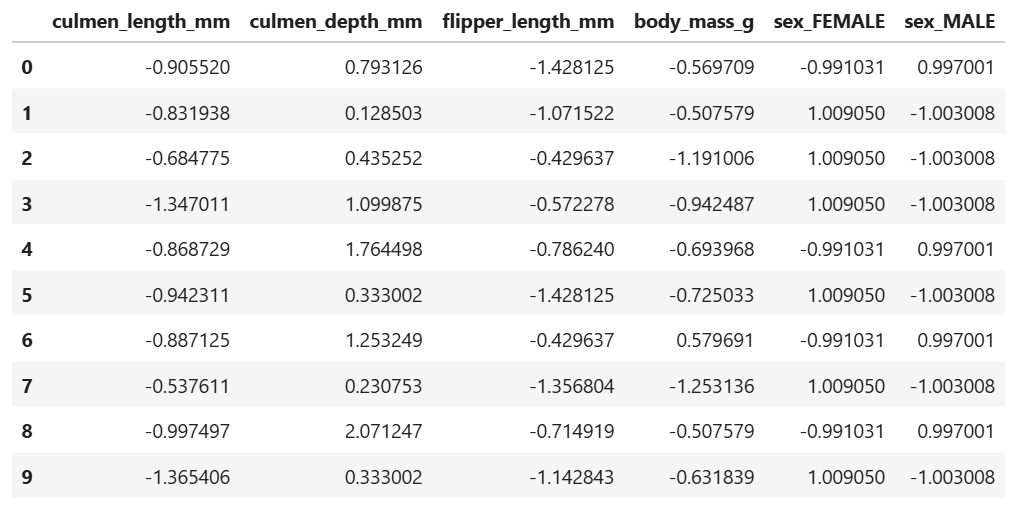
**Kết quả thực hiện**

**Bước 1 – Tiền xử lý dữ liệu**

* Đọc dữ liệu từ tệp *penguins.csv*.
* Loại bỏ toàn bộ giá trị thiếu (NaN) để đảm bảo tính toàn vẹn.
* Mã hóa biến phân loại *sex* bằng phương pháp One-Hot Encoding.
* Lựa chọn các biến số làm đầu vào cho mô hình, bao gồm:  
   culmen\_length\_mm, culmen\_depth\_mm, flipper\_length\_mm, body\_mass\_g, sex\_FEMALE, sex\_MALE.
* Chuẩn hóa dữ liệu bằng StandardScaler nhằm đưa các biến về cùng thang đo



Hình 3.1 – Biểu đồ Boxplot dữ liệu ban đầu  
Nhận xét:  
Dữ liệu thể hiện sự khác biệt rõ ràng giữa các biến; trong đó, biến *body\_mass\_g* có giá trị lớn hơn hẳn các biến khác. Một số điểm ngoại lai xuất hiện nhưng không gây ảnh hưởng đáng kể đến kết quả phân cụm.

Bảng 3.1 – Dữ liệu sau chuẩn hoá (10 dòng đầu)  
Nhận xét:  
Sau khi chuẩn hóa, các giá trị được đưa về trung tâm quanh 0, độ lệch chuẩn xấp xỉ 1.  
 Hai biến giới tính được mã hóa thành *sex\_FEMALE* và *sex\_MALE*, dữ liệu lúc này đã sẵn sàng cho giai đoạn PCA và phân cụm K-Means.

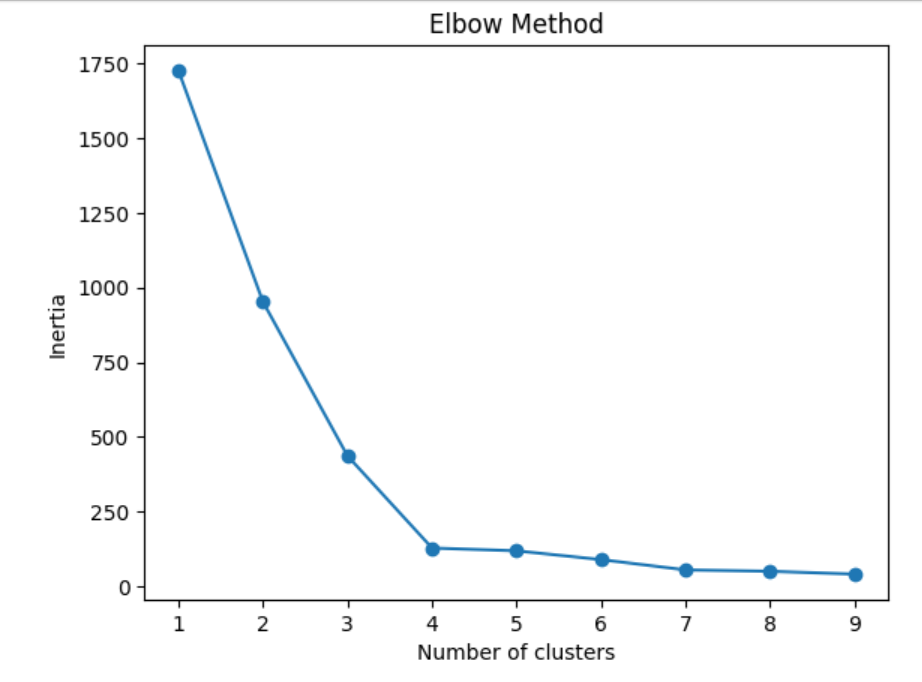
**Bước 2 – Phân tích thành phần chính (PCA)**

* Áp dụng Phân tích thành phần chính (Principal Component Analysis) để giảm chiều dữ liệu, giúp việc phân cụm và trực quan hóa thuận lợi hơn.
* Kết quả PCA cho thấy có ba thành phần chính (PC1–PC3) có *explained\_variance\_ratio\_* > 0.1.  
   → Chọn n\_components = 3 để giữ lại phần lớn thông tin.

Nhận xét:  
 Ba thành phần đầu tiên lưu giữ khoảng 91.5% phương sai của dữ liệu (PC1 ≈ 47.1%, PC2 ≈ 25.3%, PC3 ≈ 19.0%), đảm bảo mô hình duy trì phần lớn đặc trưng ban đầu.  
 Việc giảm chiều còn giúp dễ dàng biểu diễn các cụm trong không gian 2D hoặc 3D.

**Bước 3 – Xác định số cụm tối ưu (Elbow Method)**

* Tiến hành huấn luyện K-Means với giá trị K thay đổi từ 1 đến 9.
* Tính toán *Inertia* (Within-Cluster Sum of Squares – WCSS) cho từng mô hình.
* Vẽ đồ thị biểu diễn mối quan hệ giữa *K* và *Inertia*.

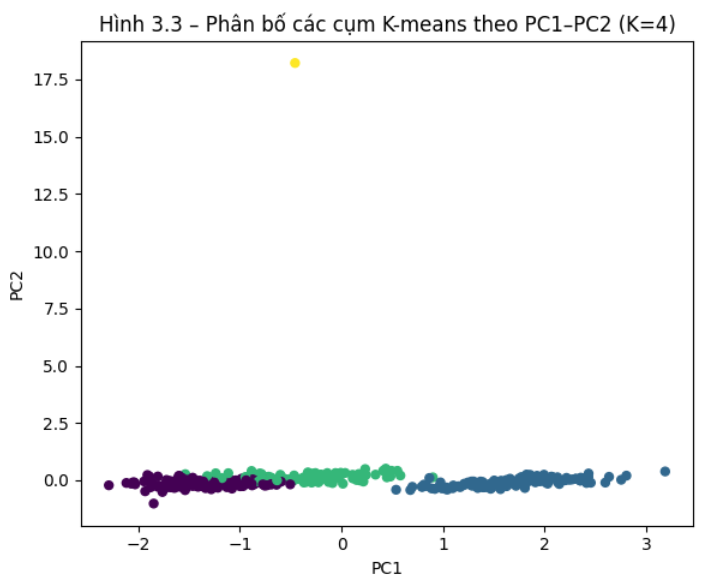


Hình 3.2 – Biểu đồ Elbow (Inertia theo số cụm K)  
Nhận xét:

Quan sát đồ thị, thấy rõ điểm “gãy” tại K = 4, nơi mà WCSS giảm chậm dần.  
 Do đó, K = 4 được xem là giá trị hợp lý nhất để mô hình đạt cân bằng giữa độ chính xác và độ khái quát.

**Bước 4 – Phân cụm K-Means với K = 4**

* Huấn luyện mô hình K-Means (K=4) trên dữ liệu sau khi giảm chiều PCA (3 thành phần).
* Gán nhãn cụm cho từng mẫu và trực quan hóa kết quả trong không gian PC1 – PC2.



Hình 3.3 – Biểu đồ phân bố các cụm sau khi phân cụm (theo PC1 – PC2)

* Nhận xét:

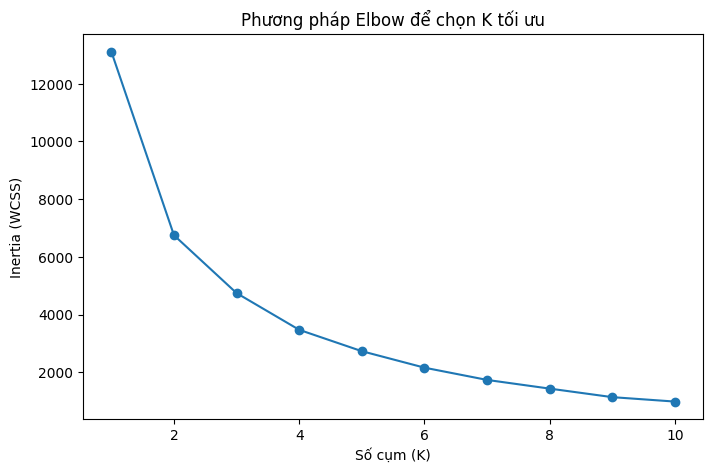
Các cụm được hình thành rõ ràng, có sự tách biệt tương đối tốt.  
 Một số điểm ở ranh giới cụm có hiện tượng chồng lắp nhẹ, nguyên nhân đến từ sự tương quan giữa kích thước cơ thể và giới tính.

* Kết luận

Bộ dữ liệu chim cánh cụt đã được xử lý, chuẩn hóa, giảm chiều bằng PCA, và phân cụm hiệu quả bằng K-Means với K = 4.  
 PCA giúp giảm nhiễu, tăng khả năng trực quan; Elbow Method giúp chọn K hợp lý.  
 Kết quả phản ánh được sự khác biệt rõ ràng về kích thước cơ thể và giới tính giữa các nhóm chim.  
 Silhouette Score ≈ 0.512, cho thấy mô hình có chất lượng cụm ở mức tốt.  
 Hạn chế: K-Means vẫn nhạy cảm với tâm khởi tạo và outlier → nên dùng K-Means++ hoặc tăng n\_init để cải thiện độ ổn định; mô hình giả định cụm có dạng cầu phương nên kém hiệu quả với dữ liệu phi tuyến.

### 3.1.3. Bài tập thực hành 2 Xây dựng mô hình phân cụm K-means trên tập dữ liệu mua sắm tại siêu thị.

1. **Phân tích biểu đồ Elbow**



* Trục X: Số cụm KKK.
* Trục Y: Giá trị *Inertia* (Within-Cluster Sum of Squares – WCSS).

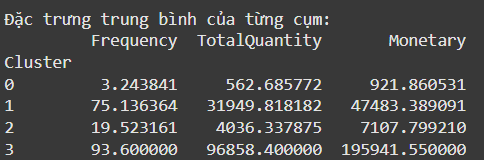
Quan sát kết quả:

* Khi KKK tăng từ 1 → 3, giá trị WCSS giảm mạnh.
* Từ K = 4 trở đi, đường cong bắt đầu “phẳng” hơn, tốc độ giảm chậm rõ rệt.  
   → Điểm gãy (elbow) xuất hiện tại K ≈ 3 hoặc 4, thể hiện đây là vùng giá trị tối ưu nhất.

Kết luận

Chọn K = 4 là hợp lý, đảm bảo mô hình đạt được cân bằng giữa độ chính xác và tính khái quát, tránh tình trạng phân cụm quá chi tiết hoặc dư thừa.

1. **Kết quả phân cụm K-means**



Cụm 0 – Nhóm khách hàng giá trị thấp

* Tần suất mua (Frequency) thấp, trung bình khoảng 3 lần.
* Số lượng sản phẩm (TotalQuantity) và giá trị chi tiêu (Monetary) đều ở mức rất thấp.  
   → Đây là nhóm khách hàng mua ít, chi tiêu thấp, mức độ gắn bó với thương hiệu còn hạn chế.

Cụm 2 – Nhóm khách hàng trung bình

* Tần suất mua hàng khoảng 19,5 lần.
* Giá trị chi tiêu trung bình (Monetary) khoảng 7.100.  
   → Là nhóm khách hàng phổ thông, có mức chi tiêu vừa phải nhưng vẫn thể hiện tiềm năng tăng trưởng nếu được chăm sóc và khuyến khích mua lại.

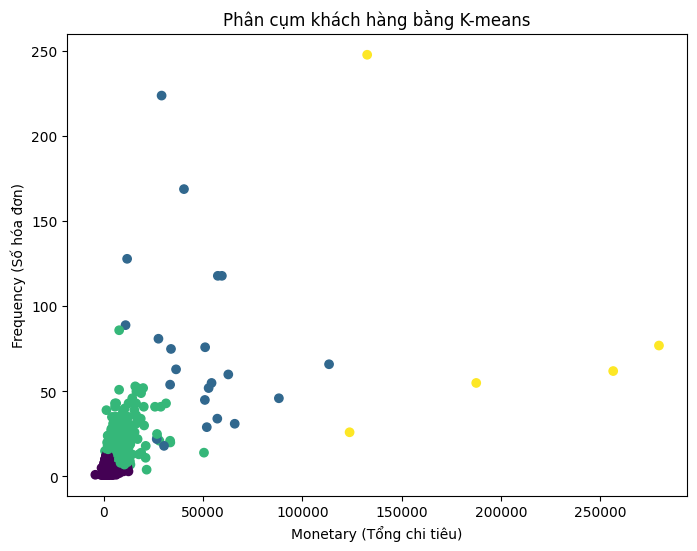
Cụm 1 – Nhóm khách hàng giá trị cao

* Tần suất mua (Frequency) khoảng 75.
* Giá trị chi tiêu (Monetary) xấp xỉ 47.000, tổng số lượng mua (Quantity) khoảng 32.000.  
   → Đây là nhóm khách hàng trung thành và mang lại doanh thu lớn, có hành vi mua thường xuyên, ổn định và đáng tin cậy.

Cụm 3 – Nhóm khách hàng VIP / Siêu khách hàng

* Tần suất mua (Frequency) gần 94.
* Giá trị chi tiêu (Monetary) đạt khoảng 195.000, tổng số lượng mua (Quantity) lên đến 96.000.  
   → Đây là nhóm khách hàng cực kỳ quan trọng, đóng góp phần lớn doanh thu cho doanh nghiệp, thể hiện mức độ trung thành và chi tiêu vượt trội.

1. **Phân cụm K-means trên 2 biến**



Biểu đồ minh họa quá trình phân cụm khách hàng dựa trên hai biến số chính:

* Trục hoành (X): *Monetary* – tổng giá trị chi tiêu của khách hàng.
* Trục tung (Y): *Frequency* – số lượng hóa đơn mua hàng (tần suất mua).

Mô hình K-Means cho thấy bốn nhóm khách hàng rõ rệt, phản ánh các cấp độ giá trị và hành vi tiêu dùng khác nhau:

Cụm 1 – Khách hàng giá trị thấp (màu tím)

* Nằm gần gốc tọa độ, với Monetary và Frequency rất thấp.
* Đặc trưng: mua ít, chi tiêu thấp, mức độ trung thành thấp.
* Hàm ý: nên kích thích mua lại bằng chương trình khuyến mãi, combo giảm giá hoặc ưu đãi khách hàng mới.

Cụm 2 – Khách hàng tiềm năng (màu xanh lá)

* Có mức chi tiêu và tần suất mua trung bình.
* Đặc trưng: mua ổn định, chi tiêu khá.
* Hàm ý: cần duy trì và khuyến khích bằng ưu đãi định kỳ, giới thiệu sản phẩm mới hoặc truyền thông cá nhân hóa để phát triển thành khách hàng trung thành.

Cụm 3 – Khách hàng trung thành (màu xanh dương)

* Có tần suất mua và giá trị chi tiêu cao, đóng góp đáng kể vào doanh thu.
* Đặc trưng: mua thường xuyên, giá trị đơn hàng cao.
* Hàm ý: nên áp dụng chính sách giữ chân, như ưu đãi độc quyền, tích điểm thưởng hoặc voucher cá nhân hóa.

Cụm 4 – Khách hàng giá trị cao / VIP (màu vàng)

* Nằm tách biệt rõ ràng, với Monetary rất lớn và Frequency cao.
* Đặc trưng: chi tiêu vượt trội, giao dịch thường xuyên, giá trị trung bình mỗi đơn hàng cao nhất.
* Hàm ý: cần chăm sóc đặc biệt thông qua chương trình VIP, ưu tiên dịch vụ, quà tri ân và hỗ trợ cá nhân hóa để duy trì lòng trung thành lâu dài.

## 3.2. GIẢI THUẬT PHÂN CỤM ĐA CẤP

### 3.2.1. Ôn tập lý thuyết

Câu 1: Giải thuật phân cụm đa cấp hoạt động như thế nào? Hãy giải thích sự khác biệt giữa phân cụm đa cấp hợp nhất (agglomerative) và phân tách (divisive)

Phân cụm đa cấp (Hierarchical Clustering) là nhóm thuật toán hình thành cấu trúc phân cấp của các cụm dữ liệu bằng cách hợp nhất (merge) hoặc phân tách (split) dần các cụm.  
 Kết quả của quá trình này được biểu diễn dưới dạng cây phân cấp (dendrogram).

* Agglomerative Clustering: Là phương pháp *bottom-up*, khởi đầu từ việc xem mỗi điểm dữ liệu là một cụm riêng biệt, sau đó liên tục gộp các cụm gần nhau nhất dựa trên tiêu chí liên kết (linkage criteria).
* Divisive Clustering: Ngược lại, đây là hướng *top-down*, bắt đầu với một cụm duy nhất bao gồm toàn bộ dữ liệu, sau đó chia tách dần thành các cụm nhỏ hơn.

Phương pháp Agglomerative thường được ưa chuộng hơn trong thực tế vì dễ triển khai và ít tốn chi phí tính toán hơn so với Divisive.

Câu 2: Các phương pháp liên kết (linkage) như single linkage, complete linkage, average linkage, và Ward’s method khác nhau ra sao? Khi nào nên sử dụng từng loại?

Tiêu chí liên kết quyết định khoảng cách giữa hai cụm, từ đó xác định cặp cụm nào sẽ được gộp ở bước tiếp theo. Một số phương pháp phổ biến gồm:

* Single Linkage (Liên kết đơn): Khoảng cách giữa hai cụm là khoảng cách nhỏ nhất giữa hai điểm thuộc hai cụm. Phù hợp cho cụm kéo dài, nhưng dễ tạo cụm dạng chuỗi.
* Complete Linkage (Liên kết hoàn chỉnh): Khoảng cách giữa hai cụm là khoảng cách lớn nhất giữa hai điểm của chúng. Thường tạo ra các cụm nhỏ gọn, hình cầu.
* Average Linkage (Liên kết trung bình): Khoảng cách giữa hai cụm là trung bình khoảng cách giữa tất cả cặp điểm của hai cụm. Cân bằng giữa Single và Complete.
* Ward’s Method: Gộp hai cụm sao cho tổng phương sai trong cụm tăng ít nhất. Đây là phương pháp phổ biến nhất và tạo cụm có kích thước tương đối đồng đều.

Khi sử dụng:

* Ward’s method: lựa chọn mặc định cho dữ liệu liên tục, muốn cụm cân bằng.
* Average linkage: ổn định, ít nhạy cảm với outlier.
* Complete linkage: phù hợp khi cần cụm chặt, rõ ràng.

Câu 3: Dendrogram trong phân cụm đa cấp là gì? Làm thế nào để sử dụng nó để chọn số lượng cụm?

Dendrogram là biểu đồ cây mô tả quá trình gộp cụm theo từng cấp độ khoảng cách trong Hierarchical Clustering.

* Trục X (hoành): biểu diễn các điểm dữ liệu hoặc cụm.
* Trục Y (tung): biểu diễn mức độ khác biệt (distance) giữa các cụm khi được hợp nhất.  
   Càng cao trên trục Y, các cụm càng khác biệt nhau.

Cách xác định số cụm tối ưu:

1. Quan sát biểu đồ dendrogram và tìm khoảng thẳng đứng dài nhất không bị cắt bởi nhánh ngang.
2. Kẻ một đường ngang cắt qua đoạn đó.
3. Số cụm tối ưu là số nhánh mà đường ngang đi qua.  
    Cách khác là cắt cây tại một ngưỡng chiều cao (height threshold) để xác định số cụm mong muốn.

Cắt dendrogram tại một ngưỡng chiều cao (height threshold) — số nhánh còn lại tương ứng với số cụm.



Câu 4: Phân cụm đa cấp có thể áp dụng cho dữ liệu phi số (non-numeric data) như thế nào? Hãy giải thích

* Phân cụm đa cấp có thể áp dụng cho dữ liệu phi số bằng cách xác định và sử dụng thước đo độ tương đồng hoặc khoảng cách phù hợp giữa các giá trị phân loại để gộp hoặc tách dần các đối tượng trong cây phân cấp (dendrogram). Nếu dữ liệu là chuỗi ký tự, loại, nhãn, danh mục (categorical / text), ta phải biến đổi hoặc định nghĩa hàm đo khoảng cách riêng.

Các cách tiếp cận xử lý dữ liệu phi số trước khi phân cụm đa cấp

|  |  |  |
| --- | --- | --- |
| Cách tiếp cận | Cách làm | Khi nào dùng |
| a. Mã hóa định lượng (Encoding) | Chuyển dữ liệu phân loại thành số, dùng One-Hot Encoding hoặc Label Encoding. Sau đó có thể áp dụng bình thường với Euclidean distance. | Dữ liệu danh mục ít giá trị (giới tính, khu vực, loại sản phẩm, v.v.) |
| b. Định nghĩa độ tương đồng tùy chỉnh (Custom Distance) | Sử dụng thước đo phù hợp như Jaccard, Hamming, Gower, hoặc Levenshtein (edit distance) cho dữ liệu text. | Khi dữ liệu dạng chuỗi hoặc hỗn hợp numeric + categorical |
| c. Biểu diễn dạng vector (Vector Embedding) | Với dữ liệu văn bản, có thể dùng TF-IDF hoặc Word2Vec để chuyển thành vector số rồi phân cụm. | Dữ liệu ngôn ngữ tự nhiên (văn bản, mô tả, phản hồi, v.v.) |

Câu 5: 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

# Import thư viện cần thiết

from sklearn.cluster import AgglomerativeClustering

from sklearn.metrics import silhouette\_score

import pandas as pd

import matplotlib.pyplot as plt

# Đọc dữ liệu rượu vang

wine\_df = pd.read\_csv("wine\_data.csv")

# Xây dựng mô hình phân cụm đa cấp (Agglomerative Clustering)

ac = AgglomerativeClustering(

n\_clusters=3, # Số cụm cần tạo

linkage='average' # Phương pháp liên kết (ward, complete, average, single)

)

# Huấn luyện và dự đoán cụm

ac\_clusters = ac.fit\_predict(wine\_df)

# Trực quan hóa kết quả phân cụm

plt.scatter(wine\_df.values[:, 0], wine\_df.values[:, 1], c=ac\_clusters, cmap='rainbow')

plt.title("Wine Clusters from Agglomerative Clustering")

plt.xlabel("OD Reading")

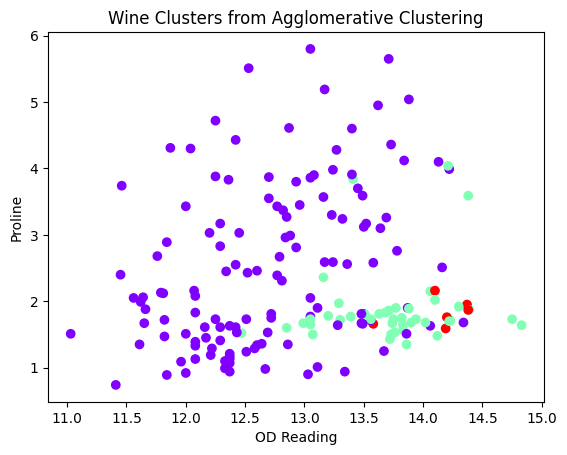
plt.ylabel("Proline")

plt.show()

# Đánh giá chất lượng cụm bằng Silhouette Score

score = silhouette\_score(wine\_df, ac\_clusters)

print("Silhouette Score:", score)



Câu 6: Làm thế nào để vẽ dendrogram trong Python sử dụng thư viện như scipy hoặc matplotlib? Hãy chia sẻ một đoạn code mẫu

from scipy.cluster.hierarchy import linkage, dendrogram, fcluster

from sklearn.datasets import make\_blobs

import matplotlib.pyplot as plt

# Tạo dữ liệu giả (dummy data)

X, y = make\_blobs(n\_samples=1000, centers=8, n\_features=2, random\_state=800)

plt.scatter(X[:,0], X[:,1])

plt.title("Dữ liệu gốc (Blob Data)")

plt.show()

# Tính toán ma trận liên kết giữa các điểm dữ liệu

distances = linkage(X, method="centroid", metric="euclidean")

# Hàm vẽ dendrogram có chú thích

def annotated\_dendrogram(\*args, \*\*kwargs):

scipy\_dendro = dendrogram(

\*args,

truncate\_mode='lastp',

show\_contracted=True,

leaf\_rotation=90.,

)

plt.title('Blob Data Dendrogram')

plt.xlabel('Cluster size')

plt.ylabel('Distance')

for i, d, c in zip(scipy\_dendro['icoord'], scipy\_dendro['dcoord'], scipy\_dendro['color\_list']):

x = 0.5 \* sum(i[1:3])

y = d[1]

if y > 10:

plt.plot(x, y, 'o', c=c)

plt.annotate("%.3g" % y, (x, y), xytext=(0, -5),

textcoords='offset points',

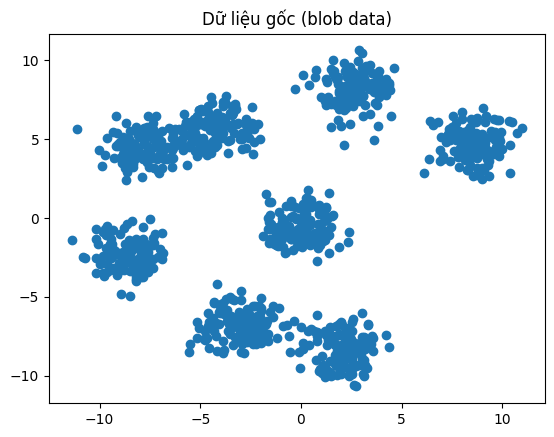
va='top', ha='center')

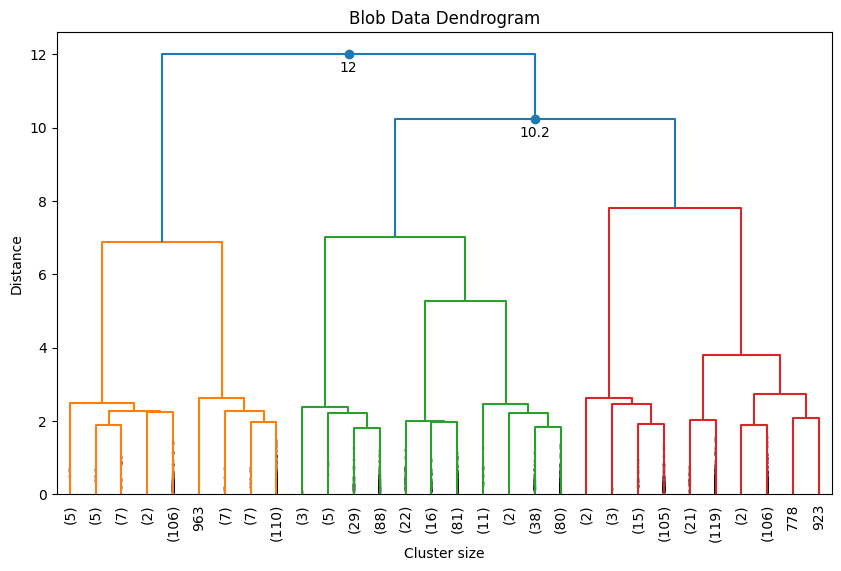
return scipy\_dendro

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

dn = annotated\_dendrogram(distances)

plt.show()





**Câu 7:** 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

Các hàm chính trong scipy.cluster.hierarchy

Thư viện scipy.cluster.hierarchy gồm nhiều hàm chuyên biệt cho phân cụm đa cấp, cung cấp khả năng kiểm soát cao hơn so với Scikit-learn.  
 Một số hàm quan trọng:

1. linkage() – Thực hiện thuật toán phân cụm hợp nhất, trả về ma trận liên kết (linkage matrix).  
   * Đầu vào: dữ liệu hoặc ma trận khoảng cách.
   * Đầu ra: thông tin chi tiết quá trình gộp cụm.
2. dendrogram() – Trực quan hóa cấu trúc phân cấp từ ma trận liên kết.  
   * Hiển thị biểu đồ cây thể hiện mối quan hệ giữa các cụm.
3. fcluster() – Cắt cây dendrogram để lấy ra các cụm phẳng (flat clusters).  
   * Đầu vào: ma trận liên kết và tiêu chí cắt (maxclust, distance,...).
   * Đầu ra: nhãn cụm tương tự .fit\_predict() trong Scikit-learn.

So sánh:

* Scikit-learn cung cấp lớp AgglomerativeClustering, đơn giản, phù hợp cho mô hình trực tiếp.
* Scipy cho phép kiểm soát chi tiết toàn bộ quy trình, từ giai đoạn liên kết, vẽ dendrogram, đến cắt cây – thích hợp cho phân tích chuyên sâu hoặc trực quan hóa học thuật.

Tóm tắt so sánh lại

|  |  |  |
| --- | --- | --- |
| Cách tiếp cận | Scikit-learn (AgglomerativeClustering) | Scipy (scipy.cluster.hierarchy) |
| Cấu trúc | Một lớp (class) duy nhất, đóng gói toàn bộ quy trình. | Một bộ hàm (functions) riêng lẻ, linh hoạt. |
| Quy trình | model = AgglomerativeClustering(n\_clusters=3) labels = model.fit\_predict(data) | linked = linkage(data) dendrogram(linked) labels = fcluster(linked, t=3, criterion='maxclust') |
| Mục đích | Dễ sử dụng, phù hợp khi đã biết số cụm và cần tích hợp nhanh. | Mạnh mẽ cho việc khám phá và trực quan hóa, cho phép bạn xem dendrogram trước khi quyết định số cụm. |

### 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>

%pip install pandas matplotlib seaborn scipy scikit-learn numpy

Nạp thư viện cần thiết

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.cluster.hierarchy import dendrogram, linkage, fcluster

from scipy.spatial.distance import pdist

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import AgglomerativeClustering

from sklearn.metrics import silhouette\_score, davies\_bouldin\_score, calinski\_harabasz\_score, silhouette\_samples

import warnings

warnings.filterwarnings('ignore')

# Set style for better visualization

plt.style.use('seaborn-v0\_8-darkgrid')

sns.set\_palette("husl")

print(" All libraries imported successfully!")

print(f"Pandas version: {pd.\_\_version\_\_}")

print(f"NumPy version: {np.\_\_version\_\_}")

# Removed sklearn version check line

Tải và khám phá dữ liệu

# Load the dataset

# Download from: https://www.kaggle.com/code/youssefaboelwafa/clustering-penguins-species-k-means-clustering

url = 'https://raw.githubusercontent.com/allisonhorst/palmerpenguins/master/inst/extdata/penguins.csv'

df = pd.read\_csv(url)

# Display basic information

print("Dataset shape:", df.shape)

print("\nFirst few rows:")

print(df.head())

print("\nDataset info:")

print(df.info())

print("\nStatistical summary:")

print(df.describe())

print("\nMissing values:")

print(df.isnull().sum())

Tiền xử lý dữ liệu

# Data preprocessing

# Remove rows with missing values

df\_clean = df.dropna()

print(f"Original dataset size: {len(df)}")

print(f"Cleaned dataset size: {len(df\_clean)}")

# Select numerical features for clustering

features = ['bill\_length\_mm', 'bill\_depth\_mm', 'flipper\_length\_mm', 'body\_mass\_g']

X = df\_clean[features]

print("\nFeatures selected for clustering:")

print(X.head())

# Save the species labels for later validation

true\_labels = df\_clean['species']

Chuẩn hóa dữ liệu

# Standardize the features (important for hierarchical clustering)

scaler = StandardScaler()

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

# Convert back to DataFrame for easier handling

X\_scaled\_df = pd.DataFrame(X\_scaled, columns=features, index=X.index)

print("Scaled data statistics:")

print(X\_scaled\_df.describe())

Tạo ma trận khoảng cách và dendogram

# Calculate linkage matrix using different methods

# Methods: 'single', 'complete', 'average', 'ward'

linkage\_methods = ['ward', 'complete', 'average', 'single']

fig, axes = plt.subplots(2, 2, figsize=(16, 12))

fig.suptitle('Dendrograms with Different Linkage Methods', fontsize=16, fontweight='bold')

for idx, method in enumerate(linkage\_methods):

row = idx // 2

col = idx % 2

# Calculate linkage

Z = linkage(X\_scaled, method=method)

# Plot dendrogram

dendrogram(Z, ax=axes[row, col], truncate\_mode='lastp', p=30,

leaf\_font\_size=10, show\_contracted=True)

axes[row, col].set\_title(f'Dendrogram - {method.capitalize()} Linkage', fontsize=14)

axes[row, col].set\_xlabel('Sample Index or (Cluster Size)', fontsize=12)

axes[row, col].set\_ylabel('Distance', fontsize=12)

plt.tight\_layout()

plt.show()

Áp dụng Hierarchical Clustering với số cụm khác nhau

n\_clusters\_range = range(2, 7)

results = {}

for n\_clusters in n\_clusters\_range:

hierarchical = AgglomerativeClustering(n\_clusters=n\_clusters, linkage='ward')

clusters = hierarchical.fit\_predict(X\_scaled)

results[n\_clusters] = clusters

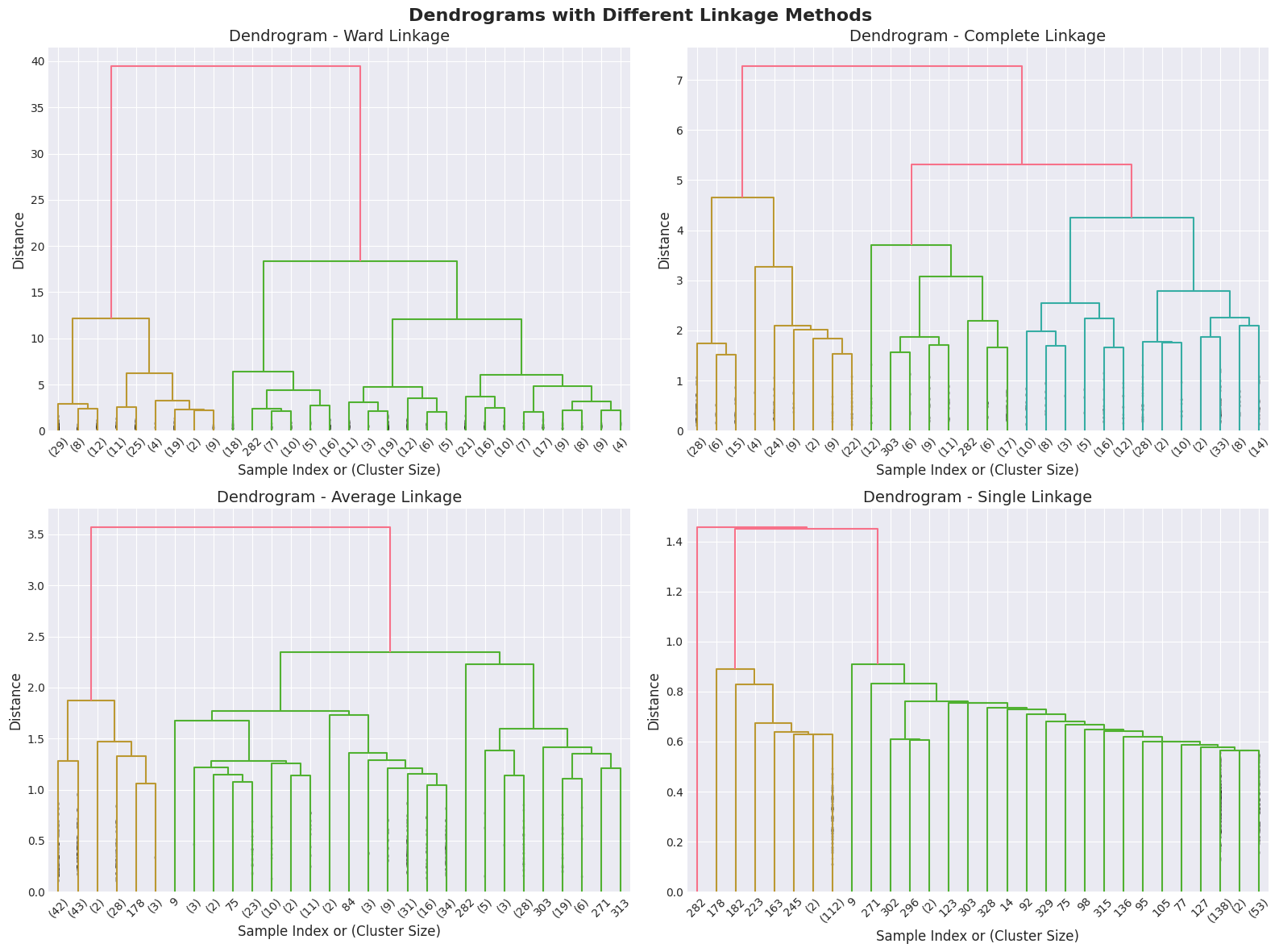
df\_clean.loc[:, f'cluster\_{n\_clusters}'] = clusters

print(f"\n{'='\*50}")

print(f"Number of clusters: {n\_clusters}")

print(f"{'='\*50}")

print(pd.Series(clusters).value\_counts().sort\_index())



1. Ward Linkage (Trên trái)

* Cấu trúc phân cụm rõ ràng nhất với 3-4 nhóm chính được phân tách ở khoảng distance ~18-20
* Có một cluster lớn (màu hồng) tách biệt hoàn toàn ở distance ~39
* Các sub-clusters (màu vàng, xanh lá) có cấu trúc cân đối và rõ ràng
* Ưu điểm: Tạo ra các cụm có kích thước cân bằng, phù hợp để phân loại các loài chim cánh cụt

2. Complete Linkage (Trên phải)

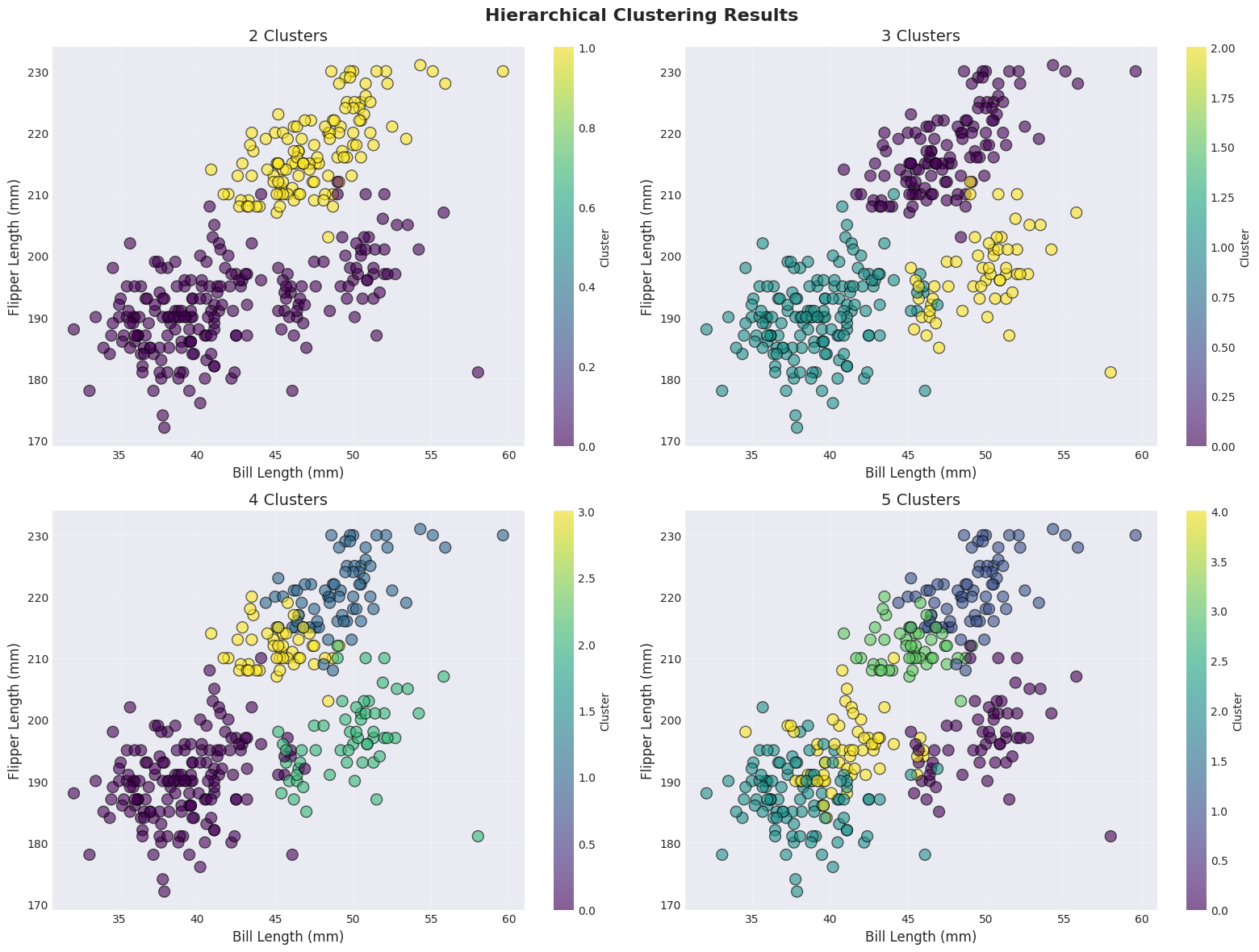
* Phân cụm bảo thủ nhất với distance thấp (~2-5)
* Tạo ra nhiều cụm nhỏ, chi tiết hơn
* Cluster chính (màu hồng) tách ở distance ~7.2
* Các cụm có xu hướng elongated (kéo dài), dễ bị ảnh hưởng bởi outliers
* Đặc điểm: Tìm các quan sát xa nhất trong mỗi cụm

3. Average Linkage (Dưới trái)

* Cân bằng giữa Ward và Complete
* Distance trung bình (~2.3-3.5), không quá cao hay quá thấp
* Cấu trúc phân cụm tương đối rõ ràng nhưng linh hoạt hơn Ward
* Cluster lớn (màu hồng) tách ở distance ~3.5
* Ưu điểm: Ít nhạy cảm với outliers hơn Complete Linkage

4. Single Linkage (Dưới phải)

* Hiệu ứng "chaining" rõ rệt - các quan sát được thêm dần vào cụm
* Distance rất thấp (~0.6-1.5), các nhánh gần như nằm ngang
* Không phù hợp cho dữ liệu này vì không tạo ra cụm rõ ràng
* Nhạy cảm nhất với nhiễu và outliers
* Cluster chính tách ở distance ~1.45



Nhận xét chung

Biểu đồ thể hiện kết quả phân cụm các cá thể chim cánh cụt dựa trên hai đặc điểm hình thái quan trọng: chiều dài mỏ (Bill Length) và chiều dài cánh (Flipper Length). Dữ liệu này có thể đến từ bộ dữ liệu Palmer Penguins nổi tiếng.

Phân tích chi tiết theo số cụm

2.Clusters: Phân chia rõ ràng thành 2 loài chính:

* + Nhóm vàng (phía trên phải): Mỏ dài (45-60mm), cánh dài (210-230mm) - có thể là loài Gentoo
  + Nhóm tím (phía dưới trái): Mỏ ngắn hơn (35-50mm), cánh ngắn hơn (175-210mm) - có thể là Adelie và Chinstrap

3 Clusters:

* Xuất hiện nhóm thứ 3 màu xanh lam ở vùng trung gian
* Phản ánh sự tồn tại của 3 loài chim cánh cụt khác biệt:
  + Teal/xanh lam: Loài trung gian về kích thước
  + Vàng: Loài lớn nhất
  + Tím: Loài nhỏ nhất
* Đây có thể là phân cụm tối ưu phản ánh đúng 3 loài thực tế

4 Clusters:

* Nhóm màu vàng ban đầu được chia thành 2 nhóm nhỏ (vàng và xanh lá)
* Nhóm xanh lá chiếm vùng có mỏ rất dài (50-60mm)
* Có thể phản ánh sự khác biệt về giới tính hoặc địa lý trong cùng một loài

5 Clusters:

* Phân chia chi tiết nhất với nhiều màu sắc
* Có thể phản ánh cả sự khác biệt về loài, giới tính, và quần đảo sinh sống
* Tuy nhiên, có sự chồng chéo giữa các cụm, cho thấy có thể đã phân chia quá mức (overfitting)

### 3.2.3. Bài tập thực hành 2

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>

Import thư viện

import pandas as pd

import datetime as dt

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import AgglomerativeClustering

from scipy.cluster.hierarchy import dendrogram, linkage

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv('OnlineRetail.csv', encoding='ISO-8859-1')

# Làm sạch dữ liệu

df.dropna(subset=['CustomerID'], inplace=True)

df = df[df['Quantity'] > 0]

df['CustomerID'] = df['CustomerID'].astype(int)

df['TotalPrice'] = df['Quantity'] \* df['UnitPrice']

df['InvoiceDate'] = pd.to\_datetime(df['InvoiceDate'], format="%d-%m-%Y %H:%M")

Tính các đặc trưng

# Xác định ngày chụp (snapshot\_date) là ngày giao dịch cuối cùng + 1 ngày

snapshot\_date = df['InvoiceDate'].max() + dt.timedelta(days=1)

# Gom nhóm theo CustomerID để tính Recency, Frequency và Monetary

rfm = df.groupby('CustomerID').agg({

    'InvoiceDate': lambda date: (snapshot\_date - date.max()).days,  # Recency

    'InvoiceNo': 'nunique',                                        # Frequency

    'TotalPrice': 'sum'                                            # Monetary

})

# Đổi tên các cột cho dễ hiểu

rfm.rename(columns={

    'InvoiceDate': 'Recency',

    'InvoiceNo': 'Frequency',

    'TotalPrice': 'MonetaryValue'

}, inplace=True)

print(rfm.head())

Xây dựng mô hình phân cụm đa cấp

scaler = StandardScaler()

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

# Tạo ma trận liên kết bằng phương pháp Ward

linked = linkage(rfm\_scaled, method='ward')

# Vẽ biểu đồ Dendrogram

plt.figure(figsize=(14, 8))

dendrogram(linked,

           orientation='top',

           distance\_sort='descending',

           show\_leaf\_counts=True,

           truncate\_mode='lastp',   # Hiển thị p cụm cuối cùng

           p=10                     # p=10 là giá trị tối ưu cho dataset này

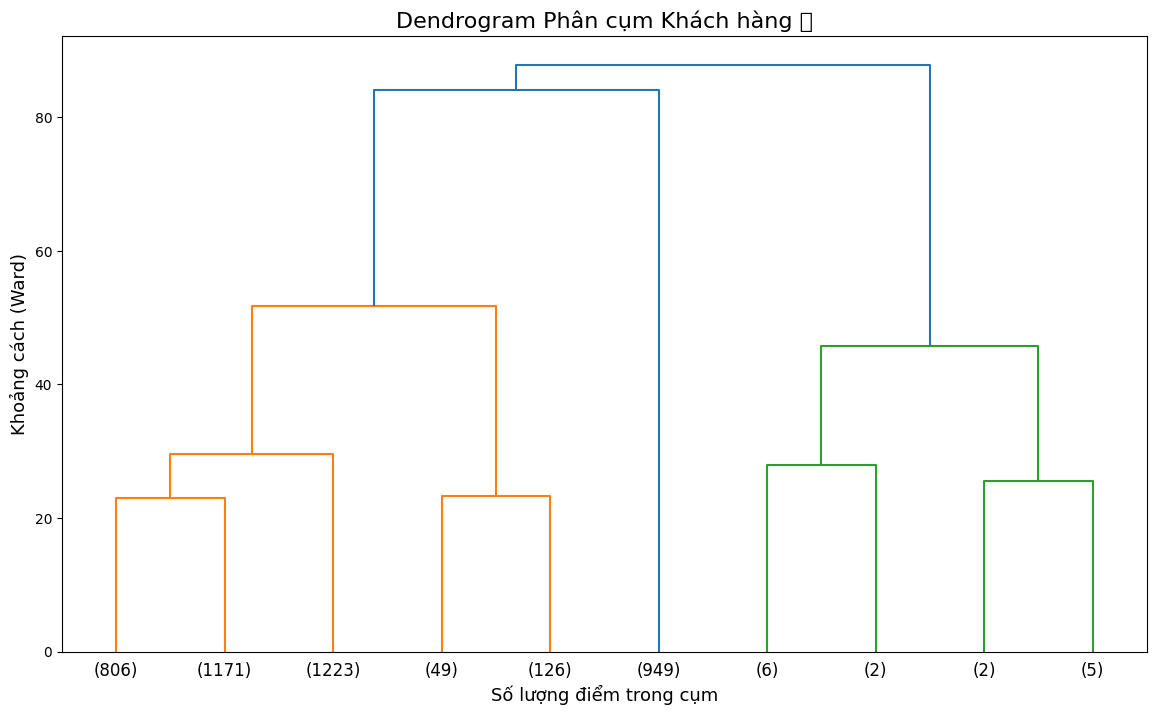
          )

plt.title('Dendrogram Phân cụm Khách hàng ', fontsize=16)

plt.xlabel('Số lượng điểm trong cụm', fontsize=13)

plt.ylabel('Khoảng cách (Ward)', fontsize=13)

plt.show()



Xây dựng mô hình Agglomerative Clustering

agg\_cluster = AgglomerativeClustering(n\_clusters=3, linkage='ward')

labels = agg\_cluster.fit\_predict(rfm\_scaled)

# Gán nhãn cụm vào bảng RFM

rfm['Cluster'] = labels

In kết quả

cluster\_summary = rfm.groupby('Cluster').agg({

    'Recency': 'mean',

    'Frequency': 'mean',

    'MonetaryValue': 'mean'

}).round(2)

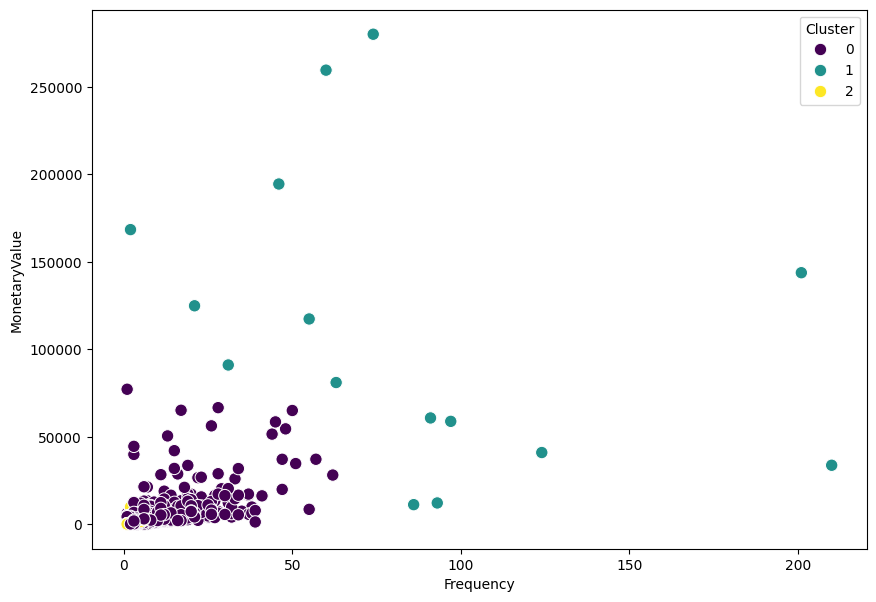
print("\n📊 Tóm tắt đặc trưng trung bình của các cụm khách hàng:")

print(cluster\_summary)

plt.figure(figsize=(10, 7))

sns.scatterplot(data=rfm, x='Frequency', y='MonetaryValue',

                hue='Cluster', palette='viridis', s=80)



Dữ liệu chia thành 3 nhóm rõ ràng giúp doanh nghiệp xác định chiến lược giữ chân, kích hoạt và sàng lọc khách hàng hiệu quả. Phân bố cụm cho thấy sự khác biệt rõ về giá trị và tần suất giao dịch.

Nhận xét kết quả phân cụm:

* Cụm 0: Khách hàng phổ thông – mua ít, chi tiêu thấp, cần kích thích bằng khuyến mãi.
* Cụm 1: Khách hàng trung thành – mua thường xuyên, chi tiêu cao, nên được chăm sóc định kỳ.
* Cụm 2: Khách hàng rời bỏ – ít mua gần đây, giá trị thấp, cần tái kích hoạt hoặc loại khỏi nhóm mục tiêu.

Silhouette Score = 0.587, cho thấy chất lượng phân cụm khá tốt, phản ánh đúng sự khác biệt hành vi giữa các nhóm khách hàng.

**HẾT**