BỘ GIÁO DỤC & ĐÀO TẠO

TRƯỜNG ĐẠI HỌC KINH TẾ - ĐẠI HỌC ĐÀ NẴNG

KHOA THỐNG KÊ – TIN HỌC

BÁO CÁO BÀI TẬP NHÓM

MÔN: PHÂN TÍCH DỮ LIỆU BẰNG PYTHON

CHỦ ĐỀ: DECISION TREE

Giảng viên hướng dẫn : **TS. Lê Diên Tuấn**

Nhóm : **04**

Lớp : **48K21.2**

Sinh viên thực hiện : **Nguyễn Trọng Khang (0703026097)**

Nguyễn Thị Thùy Linh

Nguyễn Châu My

Nguyễn Nhuận Tiến

Nguyễn Lê Thảo Vi

Đà Nẵng, 05/2025

Đóng góp của các thành viên

Họ và tên	Phần trăm đóng góp
Nguyễn Trọng Khang	**100%**
Nguyễn Thị Thùy Linh	**100%**
Nguyễn Châu My	**100%**
Nguyễn Nhuận Tiến	**100%**
Nguyễn Lê Thảo Vi	**100%**

MỤC LỤC

I. Tổng quan và Cơ sở lý thuyết

1. Tổng quan về Decision Tree

Decision Tree bắt nguồn từ thời kỳ đầu của việc phát triển các ghi chép bằng văn bản. Lịch sử phát triển cho thấy điểm mạnh nổi bật của Decision Tree là kết quả có tính diễn giải cao, được biểu diễn dưới dạng giống như mô hình cây từ đó giúp nâng cao sự hiểu biết và truyền đạt kết quả.

Nguồn gốc tính toán của Decision Tree — đôi khi còn được gọi là classification trees hoặc regression trees — là các mô hình của quá trình sinh học và nhận thức. Từ đây đã thúc đẩy sự phát triển song song giữa cây quyết định phục vụ thống kê và cây quyết định phục vụ cho học máy.

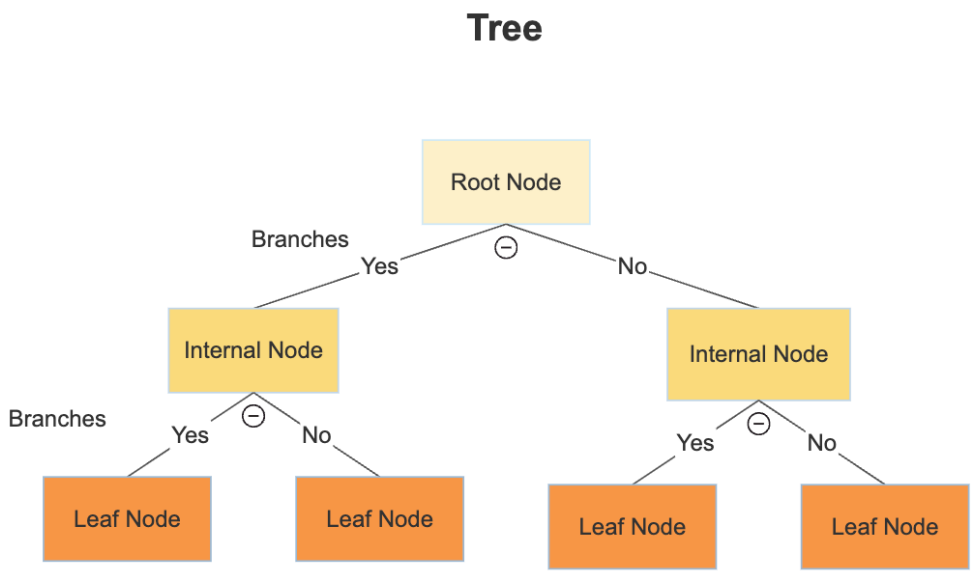
Đơn giản hơn, Decision Tree là một loại lưu đồ cho thấy một lộ trình rõ ràng để đưa ra quyết định. Trong phân tích dữ liệu, đây là một loại thuật toán bao gồm các câu lệnh điều kiện để kiểm soát việc phân lớp dữ liệu. Decision Tree bắt đầu tại một điểm (hoặc "node"), sau đó phân nhánh (hoặc "chia tách") theo từng lựa chọn khác nhau. Mỗi nhánh dẫn đến các đầu ra khả thi khác nhau, thông qua nhiều bước lựa chọn hoặc sự kiện ngẫu nhiên, cho đến khi đạt được kết quả cuối cùng.

Decision Tree cực kỳ hữu dụng trong phân tích dữ liệu và học máy vì nó chia nhỏ dữ liệu phức tạp để dễ dàng quản lý hơn. Nó thường được sử dụng trong các lĩnh vực này để phân tích dự đoán, phân loại dữ liệu và hồi quy. Decision Tree đóng vai trò như một công cụ trực quan giúp cấu trúc và phân tích các vấn đề ra quyết định trong điều kiện không chắc chắn. Nhờ vào cách biểu diễn đơn giản và dễ hiểu, nó cho phép người phân tích dễ dàng đánh giá các chiến lược quyết định khác nhau thông qua tính toán giá trị kỳ vọng, phân tích độ nhạy và xác định chiến lược tối ưu. Điều này đặc biệt hữu ích trong môi trường kinh doanh phức tạp, nơi các yếu tố rủi ro và cơ hội luôn hiện diện.

Cây quyết định (Decision Tree) là một cấu trúc dữ liệu phân cấp nhằm biểu thị dữ liệu thông qua chiến lược phân chia. Đây là thuật toán học có giám sát phi tham số được sử dụng cho nhiệm vụ phân loại (classification) và hồi quy (regression). Cấu trúc phân cấp của cây quyết định bao gồm các nút gốc (root node), các nhánh (branches), nút bên trong(internal node) và nút lá (leaf node).

- Nút gốc (Root Node): Cơ sở của cây quyết định.
- Nhánh (Branch): Một tiểu mục của cây quyết định bao gồm nhiều nút.

- Nút lá (Leaf Node): Khi một nút con không tách thành các nút con bổ sung; đại diện cho các kết quả có thể có.
- Nút bên trong (internal node): là một điểm trong thời gian mà tại đó một sự kiện diễn ra, và sự kiện này dẫn đến các nhánh tiếp theo trong cây.
- Cắt tỉa (pruning): là một kỹ thuật quan trọng trong việc xây dựng cây quyết định, giúp giảm kích thước của cây để cải thiện độ tổng quát của mô hình.



Hình 1. Ví dụ về Decision Tree

2. Các bước tạo ra cây quyết định

- **Bước 1: Xác định vấn đề**

Mỗi cây quyết định bắt đầu bằng việc hiểu rõ vấn đề cần giải quyết. Xác định mục tiêu, mục đích và các yếu tố quan trọng sẽ ảnh hưởng đến quyết định. Bước này đặt nền tảng cho toàn bộ phân tích. Xác định vấn đề càng chính xác thì cây quyết định càng hiệu quả.

- **Bước 2: Bắt đầu xây dựng cây quyết định**

Sau khi vấn đề đã được xác định rõ, bước tiếp theo là tạo cây quyết định. Cây bắt đầu từ một nút quyết định, từ đó các nhánh mở rộng, đại diện cho các lựa chọn khác nhau. Tiếp theo, các nút con sẽ được thêm vào để thể hiện kết quả tiềm năng của từng lựa chọn.

- **Bước 3: Xác định các lựa chọn quyết định**

Tiếp tục phát triển cây quyết định bằng cách liệt kê tất cả các lựa chọn hoặc hướng hành động có thể thực hiện. Các lựa chọn này sẽ là các nhánh xuất phát từ nút quyết định trung tâm. Những lựa chọn này đại diện cho các con đường khác nhau trong quá trình ra quyết định. Đảm bảo bao gồm đầy đủ các lựa chọn cũng như kết quả tiềm năng của từng lựa chọn.

- **Bước 4: Ước tính lợi ích hoặc chi phí**

Gán giá trị lợi ích hoặc chi phí cho từng kết quả. Những giá trị này thể hiện tác động hoặc hậu quả của từng kết quả đối với quyết định tổng thể. Khi ước tính, cần xem xét cả yếu tố định lượng và định tính.

- **Bước 5: Gán xác suất**

Việc gán xác suất cho từng kết quả tiềm năng là rất quan trọng. Các xác suất này có thể được suy ra từ dữ liệu lịch sử, nghiên cứu thị trường hoặc đánh giá của chuyên gia. Chúng thể hiện khả năng xảy ra của mỗi kết quả, tạo cơ sở định lượng cho quá trình ra quyết định.

- **Bước 6: Xác định kết quả tiềm năng**

Mỗi kết quả đều có một giá trị đi kèm, có thể là lợi nhuận tài chính, tác động đến khách hàng hoặc bất kỳ chỉ số quan trọng nào. Kết quả này nên bao gồm cả mặt tích cực và tiêu cực, cũng như các yếu tố rủi ro hoặc bất định. Nhân giá trị mỗi kết quả với xác suất tương ứng để tính toán giá trị kỳ vọng của từng nhánh quyết định.

- **Bước 7: Phân tích và chọn quyết định tối ưu**

Đây là bước phân tích. Bằng cách cộng các giá trị kỳ vọng của từng nhánh quyết định, có thể xác định phương án tối ưu nhất. Quyết định có giá trị kỳ vọng cao nhất sẽ là lựa chọn được khuyến nghị, dựa trên dữ liệu phân tích.

- **Bước 8: Xem xét và cập nhật cây quyết định**

Cây quyết định không phải là một mô hình cố định. Khi có thêm thông tin mới hoặc tình huống thay đổi, nên điều chỉnh cây quyết định Ở bước này, có thể thực hiện phân tích độ nhạy, kiểm tra các giả định, xác suất hoặc giá trị lợi ích/chi phí. Việc này giúp đánh giá tính ổn định của quyết định và xác định các yếu tố rủi ro tiềm ẩn.

3. Các giá trị phân chia

- **Entropy:** Là chỉ số đo lường độ không tinh khiết hoặc không chắc chắn trong một nhóm các quan sát. Nó xác định cách một cây quyết định chọn để phân chia dữ liệu. Entropy tại 1 node được tính theo công thức sau:

$$\text{Entropy(S)} = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Hình 2. Công thức Entropy

  - Trong đó:
    - S: tập dữ liệu huấn luyện
    - pi: xác suất các mẫu thuộc lớp i
    - c: tập các giá trị của thuộc tính phân loại
- Information Gain (Thông tin thu được): Nó đo lường lượng thông tin thu được khi chúng ta phân chia một tập dữ liệu dựa trên một thuộc tính cụ thể

$$IG(Q) = S_O - \sum_{i=1}^{q} \frac{N_i}{N} S_i$$

Hình 3. Công thức Information Gain

  - Trong đó:
    - IG(Q): Độ tăng thông tin khi phân chia dữ liệu theo thuộc tính Q.
    - Q: là điều kiện để chia data.
    - So:Entropy của tập dữ liệu ban đầu (trước khi phân chia).
    - q: Số lượng phân chia (nhánh) sau khi phân chia theo thuộc tính Q.
    - Ni: Số lượng mẫu trong phân chia thứ i.
    - N: Tổng số mẫu trong tập dữ liệu.
    - Si: Entropy của phân chia thứ i.
- Gini impurity: Chỉ số gini đo lường mức độ hoặc xác suất của một biến cụ thể bị

phân loại sai khi nó được chọn ngẫu nhiên. Chỉ số gini của một node được tính theo công thức sau:

$$G = \sum_k p_k * (1 - p_k)^2 = 1 - \sum_k (p_k)^2$$

Hình 4. Công thức Gini impurity

- Trong đó:
  - G: là giá trị Gini Impurity.
  - k: số các lớp có trong tập data.
  - Pk: là xác suất mà một phần tử ngẫu nhiên thuộc lớp.

- Gini gain: là độ giảm của Gini Impurity, có công thức tính tương tự như information gain, chỉ khác là ta sẽ sử dụng giá trị Gini Impurity thay vì Entropy:

$$GG(Q) = G_O - \sum_{i=1}^{q} \frac{N_i}{N} G_i$$

Hình 5. Công thức Gini Gain

- Trong đó:
  - IG(Q): là giá trị Gini gain.
  - Q: là điều kiện để chia data.
  - Go: Giá trị Gini Impurity của tập dữ liệu.
  - q: Số lượng phân chia (nhánh) sau khi phân chia theo thuộc tính Q.
  - Ni: Số lượng mẫu trong phân chia thứ i.
  - N: Tổng số mẫu trong tập dữ liệu.
  - Gi: Gini Impurity của phân chia thứ i.

4. Ưu và nhược điểm của Decision Tree

A. Ưu điểm:

- Mô hình dễ hiểu và dễ giải thích: Mỗi quyết định (phân nhánh) đều dựa trên điều kiện rõ ràng và có thể vẽ dạng sơ đồ cây như trong hình, ai cũng dễ theo

dõi.

- Cần ít dữ liệu để huấn luyện.
- Xây dựng nhanh: Việc chia tách theo thuộc tính được thực hiện bằng các tính toán đơn giản như entropy hay Gini → mô hình được huấn luyện rất nhanh
- Phân lớp nhanh: Khi đã huấn luyện xong, việc phân lớp một mẫu mới chỉ cần đi theo các nhánh điều kiện đến node lá.

B. Nhược điểm:

- Không đảm bảo xây dựng được cây tối ưu: Chỉ tối ưu tại mỗi bước tách, không xét toàn cục → có thể không ra được cây nhỏ gọn và chính xác nhất.
- Có thể overfitting (tạo ra những cây quá khớp với dữ liệu huấn luyện hay quá phức tạp): Cây có thể phân chia quá sâu, học kỹ cả nhiễu của dữ liệu huấn luyện → giảm khả năng tổng quát hóa khi áp dụng cho dữ liệu mới.
- Thường ưu tiên thuộc tính có nhiều giá trị (khắc phục bằng các sử dụng Gain Ratio). Ví dụ: một thuộc tính có ID sinh viên (mỗi ID là một giá trị riêng) sẽ được ưu tiên do phân chia tốt về mặt chỉ số thông tin, nhưng lại không có ý nghĩa thực tế.

II. Thuật toán ID3

1. Giới thiệu về thuật toán ID3

ID3 (Iterative Dichotomiser 3) là một thuật toán xây dựng cây quyết định (Decision Tree) được sử dụng cho các bài toán phân loại (classification) với các thuộc tính dạng phân loại (categorical). Thuật toán này được phát triển bởi Ross Quinlan và hoạt động dựa trên việc chọn thuộc tính phân chia dữ liệu dựa trên độ lợi thông tin (Information Gain).

2. Ý tưởng chính của ID3

Mục tiêu của ID3 là xây dựng một cây quyết định bằng cách chọn thuộc tính tốt nhất để phân chia dữ liệu tại mỗi nút, nhằm giảm thiểu độ hỗn loạn (entropy) và tăng độ thuần nhất của các nhóm dữ liệu con.

3. Hàm số Entropy

Entropy đo lường độ hỗn loạn của một tập dữ liệu. Công thức tính entropy cho một tập dữ liệu S với c lớp là:

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

Hình 6. Công thức Entropy

- Trong đó:
  - **S** là tập dữ liệu
  - **c** là số lượng lớp (nhãn phân loại)
  - **pi** là xác suất xuất hiện của lớp thứ **i**

Giải thích:

- Nếu entropy = **0** → Tập dữ liệu hoàn toàn thuần nhất (chỉ có một lớp).
- Nếu entropy = **1** → Tập dữ liệu có phân bố đồng đều giữa các lớp (rất hỗn loạn).

4. Độ lợi thông tin (Information Gain)

Độ lợi thông tin đo lường mức giảm entropy khi phân chia tập dữ liệu theo một thuộc tính cụ thể. Công thức tính độ lợi thông tin của thuộc tính A đối với tập dữ liệu S

là:

$$IG(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Hình 7. Công thức Information Gain

- Trong đó:
  - **A** là thuộc tính,
  - **Values(A)** là tập giá trị có thể có của **A**,
  - **Sv** là tập con của **S** khi thuộc tính **A** có giá trị **v**,
  - **|Sv| / |S|** là tỷ lệ của **Sv** trong tổng số mẫu dữ liệu.

Ý nghĩa:

- Nếu **IG(A)** cao → Thuộc tính **A** phân chia dữ liệu tốt → Nên chọn làm nút phân nhánh.
- Nếu **IG(A)** thấp → Thuộc tính **A** không giúp ích nhiều trong phân loại.

5. Thuật toán ID3

Thuật toán ID3 được thực hiện theo các bước sau:

- Bước 1: Nếu tất cả các mẫu trong tập dữ liệu thuộc cùng một lớp, tạo một nút lá với lớp đó.
- Bước 2: Nếu tập dữ liệu rỗng hoặc không còn thuộc tính nào để phân chia, tạo một nút lá với lớp phổ biến nhất trong tập dữ liệu cha.
- **Bước 3:** Ngược lại:
  - Chọn thuộc tính A có độ lợi thông tin cao nhất.
  - Tạo một nút quyết định dựa trên A.
  - Chia tập dữ liệu thành các tập con dựa trên các giá trị của A.
  - Áp dụng đệ quy thuật toán ID3 cho các tập con.

6. Ví dụ minh họa

Thời tiết	Nhiệt độ	Độ ẩm	Gió	Chơi thể thao
Nắng	Nóng	Cao	Yếu	Không
Nắng	Nóng	Cao	Mạnh	Không
Nhiều mây	Nóng	Cao	Yếu	Có
Mưa	Mát	Cao	Yếu	Có
Mưa	Lạnh	Bình thường	Yếu	Có
Mưa	Lạnh	Bình thường	Mạnh	Không
Nhiều mây	Lạnh	Bình thường	Mạnh	Có
Nắng	Mát	Cao	Yếu	Không
Nắng	Lạnh	Bình thường	Yếu	Có

Hình 8. Dữ liệu minh họa

→ Thuật toán ID3 sẽ tính toán entropy và độ lợi thông tin cho từng thuộc tính, sau đó chọn thuộc tính có độ lợi thông tin cao nhất để phân chia dữ liệu tại mỗi nút, xây dựng cây quyết định giúp dự đoán liệu có nên chơi thể thao hay không dựa trên các điều kiện đã cho.

Cách tính như sau:

- **Bước 1:** Tính Entropy ban đầu:

  Có 9 mẫu, trong đó:

  - 5 mẫu "Có" (chơi thể thao)
  - 4 mẫu "Không"

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

$$Entropy(S) = -\left(\frac{5}{9}\log_2\frac{5}{9} + \frac{4}{9}\log_2\frac{4}{9}\right)$$

  Tính toán:

$$Entropy(S) = -\left(0.555 \times (-0.485) + 0.444 \times (-0.530)\right) = 0.991$$

- **Bước 2:** Tính Information Gain cho từng thuộc tính:

  Xét thuộc tính "Thời tiết". Các giá trị có thể có: Nắng, Nhiều mây, Mưa

Phân chia tập dữ liệu theo "Thời tiết":

Thời tiết = "Nắng" (4 mẫu):

- Số mẫu "Có" = 1
- Số mẫu "Không" = 3

$$Entropy(Nắng) = -\left(\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}\right)$$

$$= -(0.25 \times -2 + 0.75 \times -0.415) = 0.811$$

Thời tiết = "Nhiều mây" (2 mẫu):

- Số mẫu "Có" = 2
- Số mẫu "Không" = 0

$$Entropy(Nhiềumây) = -(1\log_2 1) = 0$$

Thời tiết = "Mưa" (3 mẫu):

- Số mẫu "Có" = 2
- Số mẫu "Không" = 1

$$Entropy(Mưa) = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right)$$

$$= -(0.667 \times -0.585 + 0.333 \times -1.585) = 0.918$$

- **Bước 3:** Tính Information Gain của "Thời tiết":

$$IG(S, \text{Thời tiết}) = Entropy(S) - \sum_{v \in \text{Values(Thời tiết)}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= 0.991 - \left(\frac{4}{9} \times 0.811 + \frac{2}{9} \times 0 + \frac{3}{9} \times 0.918\right)$$

$$= 0.991 - (0.360 + 0 + 0.306) = 0.325$$

[...] Tính Information Gain cho các thuộc tính khác

Tương tự, ta tính IG cho các thuộc tính khác và so sánh:

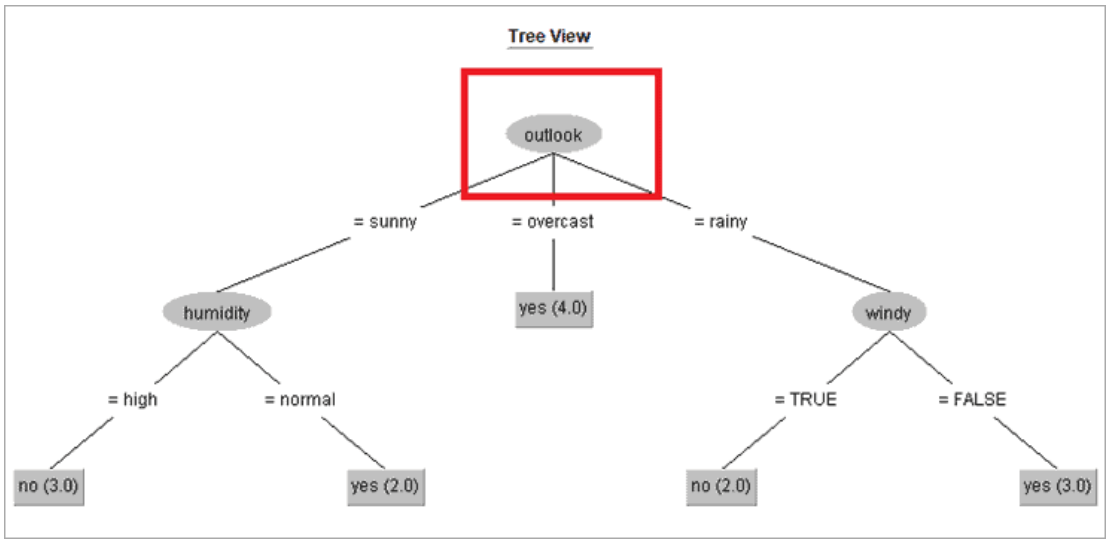
- IG(Thời tiết) = 0.325

- IG(Nhiệt độ) = 0.029
- IG(Độ ẩm) = 0.152
- IG(Gió) = 0.048

→ Thuộc tính có IG cao nhất là "Thời tiết", nên ta chọn nó làm nút gốc của cây quyết định.

- **Bước 4:** Xây dựng cây quyết định:

Từ các bước trên, cây quyết định có thể có dạng như sau:



Hình 9. Xây dựng cây quyết định

7. Ưu và nhược điểm của thuật toán ID3

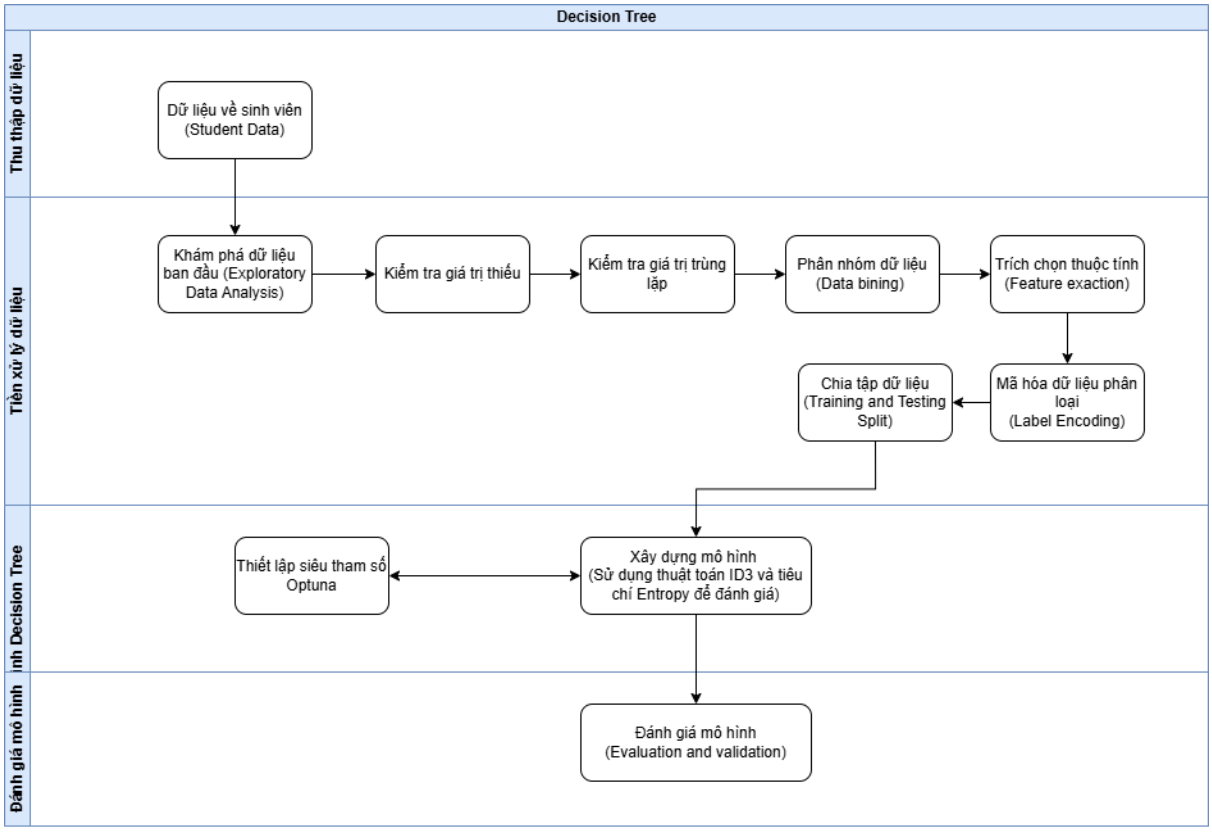
Ưu điểm	Nhược điểm
Dễ hiểu và trực quan: ID3 tạo ra cây quyết định dễ đọc và dễ giải thích.	**Thiên vị thuộc tính có nhiều giá trị**: ID3 có xu hướng chọn các thuộc tính có nhiều giá trị phân biệt, ngay cả khi chúng không thực sự quan trọng.
Tính toán nhanh: ID3 sử dụng Entropy và Information Gain, giúp xây dựng cây quyết định một cách nhanh chóng.	**Dễ bị overfitting**: Nếu dữ liệu có nhiều nhiễu hoặc không đủ tổng quát, cây quyết định có thể trở nên quá phức tạp và không hoạt động tốt trên dữ liệu mới.

Không cần giả định về phân phối dữ liệu: ID3 không yêu cầu dữ liệu phải tuân theo một mô hình thống kê cụ thể.	**Không xử lý tốt dữ liệu liên tục**: ID3 chỉ làm việc tốt với dữ liệu phân loại (categorical) mà không hỗ trợ trực tiếp dữ liệu số (continuous). Cần phải phân loại trước khi sử dụng.
Có thể mở rộng: ID3 là nền tảng cho các thuật toán mạnh hơn như C4.5 và CART.	**Không tối ưu khi có quá nhiều thuộc tính**: Nếu tập dữ liệu có nhiều thuộc tính, cây quyết định có thể trở nên rườm rà và khó tối ưu.
Xử lý tốt dữ liệu thiếu: Nếu có một số giá trị bị thiếu, ID3 có thể hoạt động mà không cần loại bỏ toàn bộ dữ liệu đó.	**Nhạy cảm với dữ liệu nhiễu**: Nếu tập dữ liệu chứa lỗi hoặc ngoại lệ, ID3 có thể tạo ra cây không chính xác.

Bảng 1. Ưu và nhược điểm của thuật toán ID3

III. Triển khai dữ liệu

1. Framework



Hình 10. Framework

2. Môi trường và công cụ

- Môi trường: Visual Studio Code

- Công cụ: Jupiter Notebook

- Ngôn ngữ: Python

IV. Tiền xử lý dữ liệu

1. Nguồn gốc

Bộ dữ liệu sinh viên được thu thập từ một hệ thống quản lý sinh viên của một trường đại học, bao gồm các thông tin cá nhân và học tập như độ tuổi, giới tính, chuyên ngành, điểm trung bình, số giờ học mỗi tuần, mức độ chuyên cần, và việc làm thêm.

Dữ liệu này có thể được sử dụng cho các mục tiêu phân tích như phân loại khả năng đậu/rớt môn học, phân cụm sinh viên theo hiệu suất học tập, hoặc hỗ trợ ra quyết định trong quản lý giáo dục.

2. Kiểm tra và xử lý dữ liệu

- Import các thư viên cần dùng:

```
import pandas as pd
import seaborn as sns
import math
```

- Đọc dữ liệu:

```
df = pd.read_csv(r"student_data.csv")
```

- Hình dạng gồm:

```
df
```

→ Kết quả:

	Age	Gender	Major	GPA	Attendance	Part_Time_Job	Study_Hours	Pass
0	25	Male	IT	2.38	89	Yes	39	No
1	20	Male	IT	2.08	89	Yes	26	Yes
2	22	Male	IT	2.34	100	Yes	28	Yes
3	23	Female	Education	2.56	88	Yes	21	Yes
4	18	Female	Engineering	2.35	100	No	14	No
...	...	...	...	...	...	...	...	...
145	21	Male	Engineering	3.01	92	No	34	Yes
146	25	Female	Engineering	2.46	71	No	17	Yes
147	18	Male	Education	3.80	75	Yes	17	Yes
148	23	Female	Business	2.77	77	No	22	No
149	22	Male	Education	3.09	99	Yes	36	No

150 rows × 8 columns

Hình 11. Kết quả import dữ liệu

Dữ liệu đa dạng:

- Bao gồm cả biến định lượng (Age, GPA, Attendance, Study_Hours) và biến định tính (Gender, Major, Part_Time_Job, Pass).
- Điều này phù hợp cho các bài toán phân loại (dự đoán Pass) hoặc phân tích thống kê.

- Khám phá kích thước:

```
print("Kích thước dữ liệu:", df.shape)
```

→Kết quả:

- Tổng số dòng: 150
- Số cột: 8

```
Kích thước dữ liệu: (150, 8)
```

Hình 12. Kết quả kích thước dữ liệu

- Thông tin chi tiết:

```
print("\nThông tin chi tiết:")
print(df.info())
```

→ Kết quả:

- Tất cả 8 cột đều có **150 non-null**, tức là không có dữ liệu bị thiếu → thuận lợi cho phân tích và huấn luyện mô hình.

```
Thông tin chi tiết:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Age            150 non-null    int64
 1   Gender         150 non-null    object
 2   Major          150 non-null    object
 3   GPA            150 non-null    float64
 4   Attendance     150 non-null    int64
 5   Part_Time_Job  150 non-null    object
 6   Study_Hours    150 non-null    int64
 7   Pass           150 non-null    object
dtypes: float64(1), int64(3), object(4)
memory usage: 9.5+ KB
None
```

Hình 13. Kết quả thông tin dữ liệu

- Thống kê mô tả dữ liệu ban đầu:

```
print("\nThống kê mô tả:")
print(df.describe())
```

→ Kết quả:

- Dữ liệu cho thấy có **sự khác biệt rõ** giữa sinh viên về thời gian học, chuyên cần và GPA.
- Điều này phù hợp để phân tích xem những yếu tố nào ảnh hưởng đến việc **Pass** hay **Không**.

```
Thống kê mô tả:
              Age         GPA  Attendance  Study_Hours
count  150.000000  150.000000  150.000000   150.000000
mean    21.313333    2.904600   76.480000    24.073333
std      2.331968    0.551007   14.959776     9.749959
min     18.000000    2.040000   50.000000     5.000000
25%     19.000000    2.412500   62.000000    16.000000
50%     21.000000    2.835000   77.000000    25.000000
75%     23.000000    3.355000   89.000000    31.000000
max     25.000000    3.980000  100.000000    40.000000
```

Hình 14. Kết quả thống kê mô tả

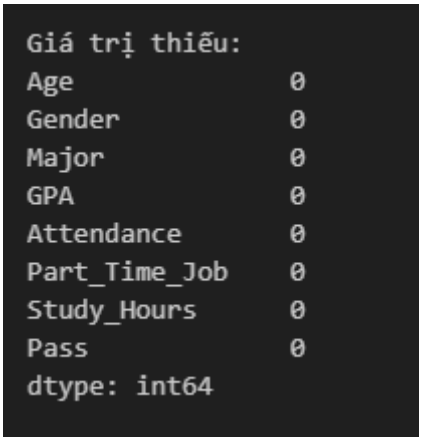
- Kiểm tra giá trị null:

```
print("\nGiá trị thiếu:")
```

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

→ Kết quả:

- Kết quả cho thấy **không có giá trị nào bị thiếu** trong toàn bộ 8 cột dữ liệu.
- Đây là một **điểm rất tích cực**, vì **không cần xử lý dữ liệu thiếu (null)** — một bước thường tốn thời gian trong tiền xử lý dữ liệu.

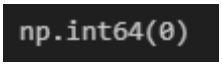


Hình 15. Kết quả kiểm tra giá trị thiếu

- Kiểm tra giá trị trùng lặp:

```
df.duplicated().sum()
```

→ Kết quả: không có dòng nào bị trùng lặp



Hình 16. Kết quả kiểm tra giá trị trùng lặp

- Xem phân phối từng cột bằng biểu đồ histogram:

```
# Chọn các cột số, loại bỏ cột Pass (vì đây là cột mục tiêu phân loại)
numerical_columns = df.select_dtypes(include=['float', 'int']).columns
numerical_columns = [col for col in numerical_columns if col != 'Pass']  # Loại bỏ cột Pass
num_plots = len(numerical_columns)

# Setup số lượng hàng và cột
num_rows = 3
num_columns = math.ceil(num_plots / num_rows)

# Tạo khung lớn với số lượng hàng và cột phù hợp
fig, axes = plt.subplots(num_rows, num_columns, figsize=(num_columns * 5, num_rows * 5))

# Lặp qua từng cột số và vẽ biểu đồ histogram
```

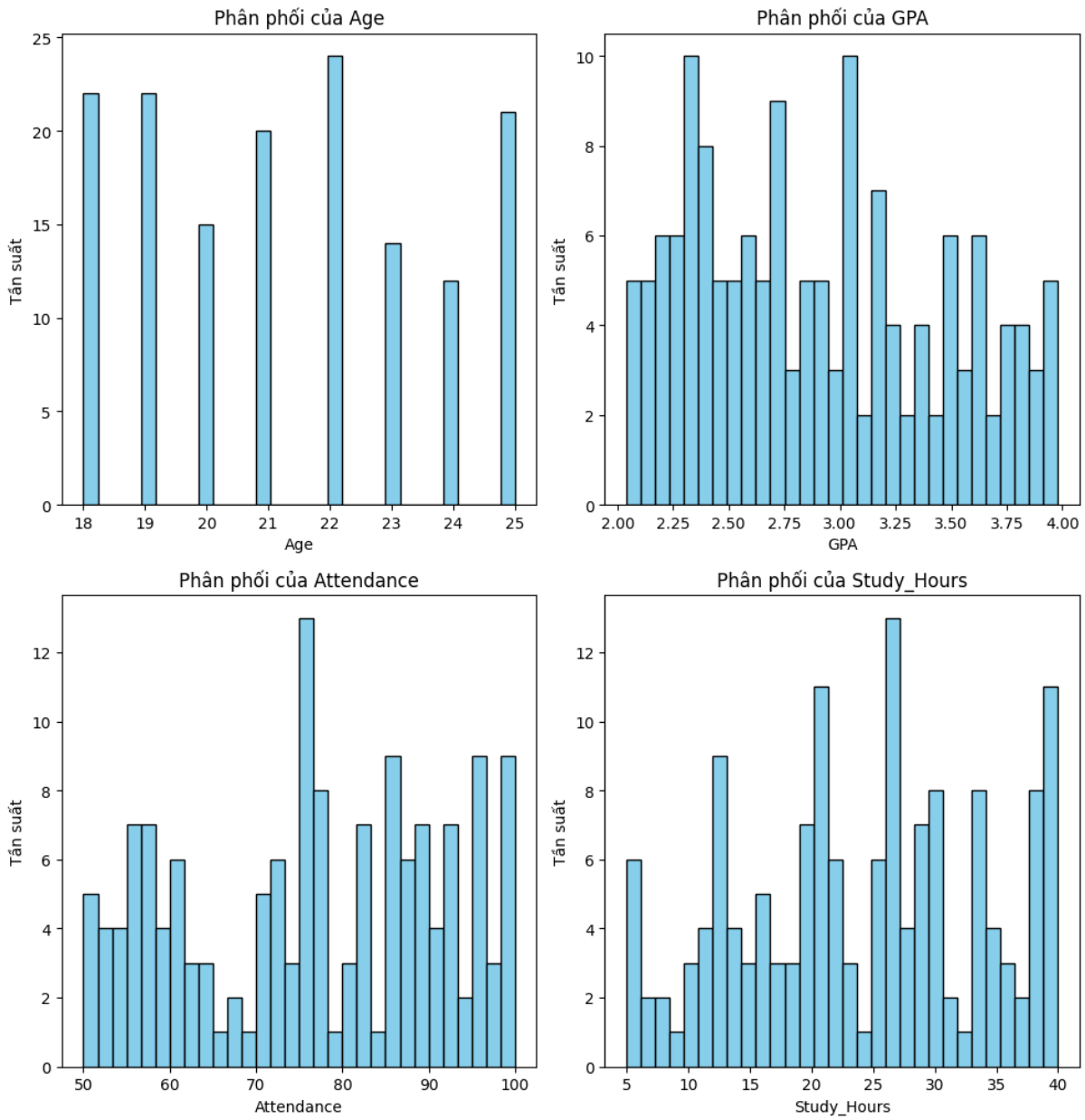
```
for i, column in enumerate(numerical_columns):
    row = i // num_columns
    col = i % num_columns

    # Vẽ biểu đồ histogram
    if num_rows > 1:
        axes[row, col].hist(df[column], bins=30, color='skyblue', edgecolor='black')
        axes[row, col].set_title(f'Phân phối của {column}')
        axes[row, col].set_xlabel(column)
        axes[row, col].set_ylabel('Tần suất')
    else:
        axes[col].hist(df[column], bins=30, color='skyblue', edgecolor='black')
        axes[col].set_title(f'Phân phối của {column}')
        axes[col].set_xlabel(column)
        axes[col].set_ylabel('Tần suất')

# Xóa các ô trống nếu có
if num_rows > 1:
    for j in range(i + 1, num_rows * num_columns):
        fig.delaxes(axes[j // num_columns, j % num_columns])
else:
    for j in range(i + 1, num_columns):
        fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```

→ Kết quả:



Hình 17. phân phối từng cột bằng biểu đồ histogram

Phân phối của Age (Tuổi):

- Độ tuổi sinh viên chủ yếu dao động từ 18 đến 25 tuổi.
- Có vẻ như các nhóm tuổi 18, 19, 22 và 25 chiếm tỷ lệ cao hơn.
- Phân phối không đồng đều, nhưng không quá lệch → hợp lý với sinh viên đại học.

Phân phối của GPA:

- GPA dao động từ khoảng 2.0 đến 4.0.
- Phân phối khá rộng và đa dạng, với nhiều sinh viên có điểm từ 2.5 đến 3.5.

- Có xu hướng hơi nghiêng về bên trái → nhiều sinh viên có GPA khá.

Phân phối của Attendance (Chuyên cần):

- Tần suất chuyên cần phân bố khá đều, tuy có một số điểm tập trung nhiều hơn ở vùng gần 75–100%.
- Số lượng sinh viên chuyên cần thấp (dưới 60%) vẫn có, nhưng ít hơn.

Phân phối của Study_Hours (Giờ học):

- Sinh viên học từ 5 đến 40 giờ mỗi tuần.
- Không có đỉnh rõ ràng, nhưng nhiều sinh viên học khoảng 25–30 giờ/tuần.
- Có sự phân tán lớn → phản ánh mức độ chăm học khác nhau rõ rệt.

- Phân tổ dữ liệu sử dụng hàm qcut từ thư viện pandas. Chia dữ liệu dựa trên phân vị:

```
# Phân tổ các cột số (trừ Age) thành 5 nhóm bằng pd.qcut
df['GPA_LABEL'] = pd.qcut(df['GPA'], 5, labels=range(0, 5)).astype(int)
df['ATTENDANCE_LABEL'] = pd.qcut(df['Attendance'], 5, labels=range(0, 5)).astype(int)
df['STUDY_HOURS_LABEL'] = pd.qcut(df['Study_Hours'], 5, labels=range(0, 5)).astype(int)
```

- Trích xuất các cột cần sử dụng trong mô hình cây quyết định:

```
feature_columns = df[[
    'Age',              # Số (giữ lại nếu thấy có ảnh hưởng)
    'Gender',           # Phân loại
    'Major',            # Phân loại
    'Part_Time_Job',    # Phân loại
    'GPA_LABEL',        # Phân loại thay cho GPA
    'ATTENDANCE_LABEL', # Phân loại thay cho Attendance
    'STUDY_HOURS_LABEL' # Phân loại thay cho Study_Hours
]]
```

- Mã hóa các cột phân loại:

```
# Mã hóa Gender và Part_Time_Job bằng replace
df['Gender'] = df['Gender'].replace({'Male': 1, 'Female': 0})
df['Part_Time_Job'] = df['Part_Time_Job'].replace({'Yes': 1, 'No': 0})

# Mã hóa Major bằng cách gán thủ công 0, 1, 2, 3, 4
df['Major'] = df['Major'].replace({
```

```
    'IT': 0,
    'Education': 1,
    'Engineering': 2,
    'Arts': 3,
    'Business': 4
})

# Mã hóa cột mục tiêu Pass
le = LabelEncoder()
df['Pass'] = le.fit_transform(df['Pass'])  # Yes/No -> 1/0
```

V. Xây dựng cây quyết định sử dụng thư viện scikit-learn của python

1. Import các thư viện cần sử dụng để xây dựng cây quyết định

```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn import tree
import optuna
import matplotlib.pyplot as plt
import seaborn as sns
```

Từ thư viện sklearn, các công cụ được nhập bao gồm:

- train_test_split: Chia dữ liệu thành tập huấn luyện và tập kiểm tra
- cross_val_score: Đánh giá hiệu suất mô hình thông qua kiểm định chéo
- DecisionTreeClassifier: Xây dựng mô hình Cây Quyết định để dự đoán kết quả học tập
- plot_tree: Trực quan hóa cấu trúc của Cây Quyết định
- LabelEncoder: Chuyển đổi các giá trị phân loại (như giới tính, chuyên ngành) thành dạng số
- accuracy_score, classification_report, confusion_matrix: Tính toán các chỉ số đánh giá hiệu suất mô hình, bao gồm độ chính xác, độ phủ, điểm F1, và ma trận nhầm lẫn

Thư viện optuna được sử dụng để tối ưu hóa các tham số của mô hình, giúp tìm ra cấu hình tốt nhất nhằm nâng cao độ chính xác.

2. Chia dữ liệu để thành tập huấn luyện và kiểm tra

```
X = df.drop('Pass', axis=1)  # Đặc trưng
y = df['Pass']  # Nhãn

# Chia dữ liệu: 80% huấn luyện, 20% kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Chia dữ liệu: Việc chia dữ liệu thành tập huấn luyện (80%) và kiểm tra (20%) là một thực hành tiêu chuẩn trong học máy, nhằm:

- **Huấn luyện mô hình:** Tập huấn luyện (120 sinh viên) cung cấp đủ dữ liệu để mô hình học cách dự đoán dựa trên các đặc trưng.
- **Đánh giá mô hình:** Tập kiểm tra (30 sinh viên) cho phép kiểm tra hiệu suất của mô hình trên dữ liệu chưa từng thấy, từ đó đánh giá khả năng tổng quát hóa của mô hình trong thực tế.

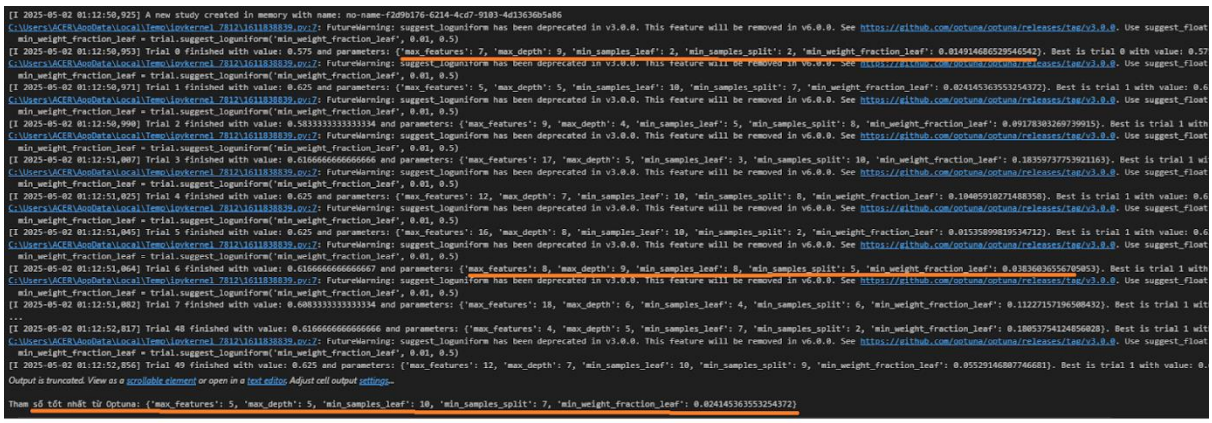
3. Tối ưu hóa tham số bằng Optuna

```
def objective(trial): #Tạo hàm có tên objective và hàm này nhận 1 tham số là trial
    #Xét giá trị cho các tham số của mô hình trong một khoảng chỉ định
    max_features = trial.suggest_int('max_features',4,20)
    max_depth = trial.suggest_int('max_depth', 4, 10)
    min_samples_leaf = trial.suggest_int('min_samples_leaf', 2, 10)
    min_samples_split = trial.suggest_int('min_samples_split', 2,10)
    min_weight_fraction_leaf = trial.suggest_loguniform('min_weight_fraction_leaf', 0.01, 0.5)

    #Tạo mô hình cây quyết định với tham số thử nghiệm
    dt = tree.DecisionTreeClassifier(
        criterion='entropy',
        max_depth=max_depth,
        min_samples_split=min_samples_split,
        min_samples_leaf=min_samples_leaf,
        min_weight_fraction_leaf=min_weight_fraction_leaf,
        random_state=42
    )
    #Đánh giá hiệu suất mô hình bằng kiểm định chéo
    scores = cross_val_score(dt, X_train, y_train, cv=5, scoring='accuracy')
    return scores.mean()
 #Tạo và chạy quá trình tối ưu hóa bằng Optuna
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=50)

# Lấy và hiển thị tham số tốt nhất
best_params = study.best_params
print("\nTham số tốt nhất từ Optuna:", best_params)
```

Kết quả:



Hình 18. Kết quả tối ưu hóa tham số bằng Optuna

- **Tối ưu hóa tham số:** Việc sử dụng Optuna giúp tìm ra cấu hình tốt nhất cho mô hình Cây quyết định, thay vì chọn tham số thủ công. Các tham số như độ sâu tối đa (max_depth), số đặc trưng tối đa (max_features), và số mẫu tối thiểu (min_samples_leaf, min_samples_split) ảnh hưởng trực tiếp đến hiệu suất và độ phức tạp của mô hình.
- **Đánh giá hiệu suất bằng kiểm định chéo:** Sử dụng cross_val_score với 5 lần lặp trên tập huấn luyện (120 sinh viên) giúp đánh giá độ chính xác một cách đáng tin cậy, tránh phụ thuộc vào một lần chia dữ liệu duy nhất.

4. Xây dựng và huấn luyện mô hình với tham số tốt nhất

```
#Tạo mô hình cây quyết định với tham số tối ưu
dt = tree.DecisionTreeClassifier(
    criterion='entropy',
    max_depth=best_params['max_depth'],
    max_features=best_params['max_features'],
    min_samples_split=best_params['min_samples_split'],
    min_samples_leaf=best_params['min_samples_leaf'],
    min_weight_fraction_leaf = best_params['min_weight_fraction_leaf'],
    random_state=42
)
#Huấn luyện mô hình trên tập dữ liệu huấn luyện
dt.fit(X_train, y_train)
```

Kết quả:

```
DecisionTreeClassifier(criterion='entropy', max_depth=5, max_features=5,
                       min_samples_leaf=10, min_samples_split=7,
                       min_weight_fraction_leaf=0.024145363553254372,
                       random_state=42)
```

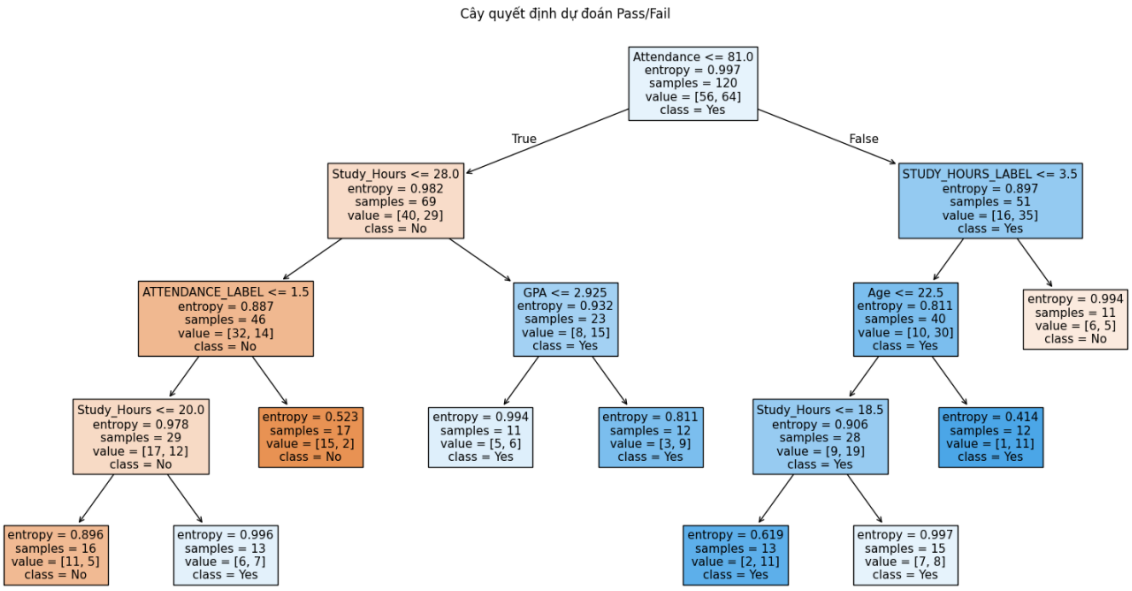
Hình 19. Kết quả xây dựng và huấn luyện mô hình với tham số tốt nhất

- **Tạo mô hình với tham số tối ưu:** Việc sử dụng tham số từ best_params (được xác định bằng Optuna) đảm bảo rằng mô hình được cấu hình tối ưu, cân bằng giữa độ chính xác và độ phức tạp. Các tham số như max_depth=10 và min_samples_leaf=10 giúp mô hình không quá đơn giản (dẫn đến dự đoán kém) hoặc quá phức tạp (dẫn đến hiện tượng quá khớp - overfitting).
- **Huấn luyện mô hình:** Bước huấn luyện trên tập dữ liệu X_train và y_train (120 sinh viên) cho phép mô hình học các mẫu (patterns) từ dữ liệu. Mô hình sẽ sử dụng các mẫu này để dự đoán kết quả học tập trên tập kiểm tra (30 sinh viên) hoặc dữ liệu mới trong thực tế.

5. Trực quan hóa cây quyết định bằng plot-tree

```
#Thiết lập kích thước biểu đồ
plt.figure(figsize=(20, 10))
tree.plot_tree(
    dt,
    fontsize=11,
    filled=True,
    feature_names=X.columns,
    class_names=['No', 'Yes']
)
#Đặt tên biểu đồ
plt.title('Cây quyết định dự đoán Pass/Fail')
plt.show()
```

Kết quả:



Hình 20. Kết quả trực quan hóa cây quyết định bằng plot-tree

Giải thích cây quyết định:

Cây quyết định được huấn luyện để dự đoán kết quả học tập của sinh viên (Pass/Fail) dựa trên các đặc trưng từ dữ liệu student_data.csv (150 sinh viên, với các thuộc tính như tuổi, giới tính, chuyên ngành, GPA, tỷ lệ chuyên cần, công việc bán thời gian, số giờ học, và kết quả học tập Pass: No - Không đạt, Yes - Đạt).

Các thông tin tại mỗi nút bao gồm:

- **Entropy:** Đo độ hỗn loạn của dữ liệu (gần 0 nếu dữ liệu thuần nhất, gần 1 nếu dữ liệu cân bằng giữa các lớp).
- **Samples:** Số lượng mẫu (sinh viên) tại nút đó.
- **Value:** Số lượng mẫu thuộc từng lớp ([No, Yes], tức là [Không đạt, Đạt]).
- **Class:** Lớp dự đoán tại nút, dựa trên lớp chiếm đa số.

Mô hình này được huấn luyện trên tập huấn luyện (120 sinh viên) với các tham số tối ưu đã được xác định trước đó bằng Optuna.

- **Ở node gốc:**
  - Có 120 mẫu (Tập huấn luyện).
  - Bao gồm:
    - 56 mẫu có nhãn Không đạt (No).
    - 64 mẫu có nhãn Đạt (Yes).
  - Điều kiện phân nhánh: Attendance <= 81.0
    - Nếu thỏa điều kiện → đi nhánh trái.
    - Ngược lại → đi nhánh phải.
- **Nhánh trái của node gốc (Attendance <= 81.0):**
  - Có 69 mẫu, bao gồm:
    - 40 mẫu Không đạt (No)
    - 29 mẫu Đạt (Yes)
  - Điều kiện phân nhánh: STUDY_HOURS_LABEL <= 2.5
- **Nhánh phải của node gốc (Attendance > 81.0):**
  - 51 mẫu
    - 16 Không đạt (No), 35 Đạt (Yes)
  - Điều kiện phân nhánh: Age <= 22.5

VI. Đánh giá mô hình thông qua kỹ thuật kiểm tra chéo 5 lần (k-fold cross-validation)

```
print(cross_val_score(dt, X_train, y_train)) #Tham số cross validation mặc định là 5
```

Kết quả:

```
[0.625      0.41666667 0.79166667 0.66666667 0.66666667]
```

Hình 21. Kết quả đánh giá mô hình thông qua kỹ thuật kiểm tra chéo 5 lần

Hàm cross_val_score:

- Hàm này thực hiện kiểm tra chéo (cross-validation) để đánh giá mô hình dt
- X_train và y_train là tập dữ liệu huấn luyện.
- Kết quả trả về là một mảng chứa các điểm số (scores) của mô hình trên các tập kiểm tra (folds) trong quá trình cross-validation.

Kết quả chạy:

- Kết quả là một mảng: [0.625, 0.41666667, 0.79166667, 0.66666667, 0.66666667].
- Mỗi giá trị trong mảng đại diện cho điểm số (score) của mô hình trên một fold. Ở đây, điểm số dao động từ 0.4167 đến 0.7917.

Nhận xét:

- **Sự ổn định:** Điểm số có sự dao động (từ 0.4167 đến 0.7917), con số này cho thấy mô hình không ổn định trên các fold khác nhau. Điều này có thể do:
  - Dữ liệu không đồng đều.
  - Số lượng dữ liệu nhỏ, dẫn đến các fold có sự khác biệt lớn.
  - Mô hình dt có thể bị overfitting hoặc chưa được tối ưu.

Dự đoán trên tập kiểm tra:

```
y_pred = dt.predict(X_test)
```

Ta sử dụng mô hình dt để dự đoán nhãn (y_pred) cho tập dữ liệu kiểm tra X_test.

X_test là tập đặc trưng (features), và y_pred là tập nhãn dự đoán tương ứng.

- **Độ chính xác:**

```
accuracy = accuracy_score(y_test, y_pred)
print(f"\nĐộ chính xác của mô hình: {accuracy:.2f}")
```

Kết quả:

```
Độ chính xác của mô hình: 0.57
```

Hình 22. Độ chính xác của mô hình

Hàm accuracy_score từ scikit-learn được sử dụng để tính độ chính xác của mô hình, so sánh nhãn thực tế (y_test) với nhãn dự đoán (y_pred).

Kết quả: Độ chính xác của mô hình: 0.57.

- Mô hình dự đoán đúng 57% số mẫu trong tập kiểm tra.
- Độ chính xác 57% là khá thấp, cho thấy mô hình chưa thực sự tốt và có thể cần cải thiện

- **Báo cáo phân loại: (Classification Report) (precision, recall, f1-score)**

```
print("\nBáo cáo phân loại:")
print(classification_report(y_test, y_pred, target_names=['No', 'Yes']))
```

Kết quả:

```
Báo cáo phân loại:
              precision    recall  f1-score   support

          No       0.42      0.45      0.43        11
         Yes       0.67      0.63      0.65        19

    accuracy                           0.57        30
   macro avg       0.54      0.54      0.54        30
weighted avg       0.57      0.57      0.57        30
```

Hình 23. Báo cáo phân loại

Hàm classification_report từ scikit-learn tạo ra một báo cáo chi tiết về các chỉ số đánh giá mô hình, bao gồm:

- Precision (Độ chính xác): Tỷ lệ dự đoán đúng trong số các dự đoán là positive.
- Recall (Độ phủ): Tỷ lệ các mẫu positive thực sự được dự đoán đúng.
- F1-score: Trung bình hài hòa giữa precision và recall.

- Support: Số lượng mẫu thuộc mỗi lớp trong tập kiểm tra.

Kết quả:

Lớp "No" (11 mẫu):

- Precision: 0.42 → Trong số các mẫu được dự đoán là "No", chỉ 42% thực sự là "No".
- Recall: 0.45 → Mô hình chỉ phát hiện được 45% số mẫu thực sự là "No".
- F1-score: 0.43 → Hiệu suất tổng thể của lớp "No" khá thấp.

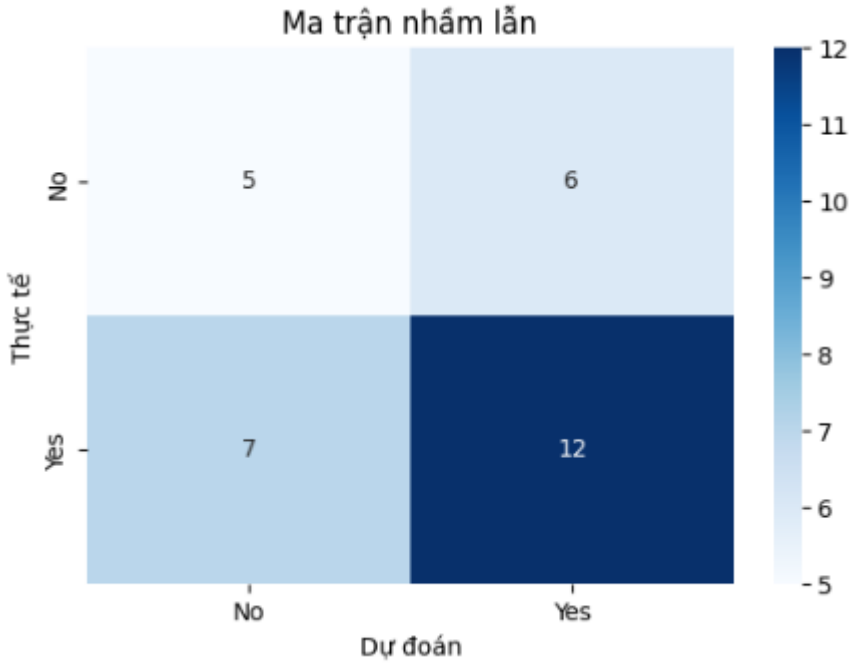
Lớp "Yes" (19 mẫu):

- Precision: 0.67 → Trong số các mẫu được dự đoán là "Yes", 67% thực sự là "Yes".
- Recall: 0.63 → Mô hình phát hiện được 63% số mẫu thực sự là "Yes".
- F1-score: 0.65 → Hiệu suất của lớp "Yes" tốt hơn lớp "No", nhưng vẫn chưa cao.

- **Ma trận nhầm lẫn (Confusion Matrix):**

```
-    cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No', 'Yes'], yticklabels=['No',
'Yes'])
plt.title('Ma trận nhầm lẫn')
plt.xlabel('Dự đoán')
plt.ylabel('Thực tế')
plt.show()
```

Kết quả:



Hình 24. Ma trận nhầm lẫn

cm = confusion_matrix(y_test, y_pred): Tạo ma trận nhầm lẫn từ nhãn thực tế (y_test) và nhãn dự đoán (y_pred).

sns.heatmap(...): Vẽ ma trận nhầm lẫn dưới dạng heatmap với các tham số như kích thước, màu sắc ("Blues"), và nhãn trục.

plt.xlabel, plt.ylabel, plt.title: Đặt nhãn cho trục x, trục y, và tiêu đề.

- **True Negative (TN):** 5 (dự đoán "No" đúng khi thực tế là "No")
- **False Positive (FP):** 6 (dự đoán "Yes" sai khi thực tế là "No")
- **False Negative (FN):** 7 (dự đoán "No" sai khi thực tế là "Yes")
- **True Positive (TP):** 12 (dự đoán "Yes" đúng khi thực tế là "Yes")

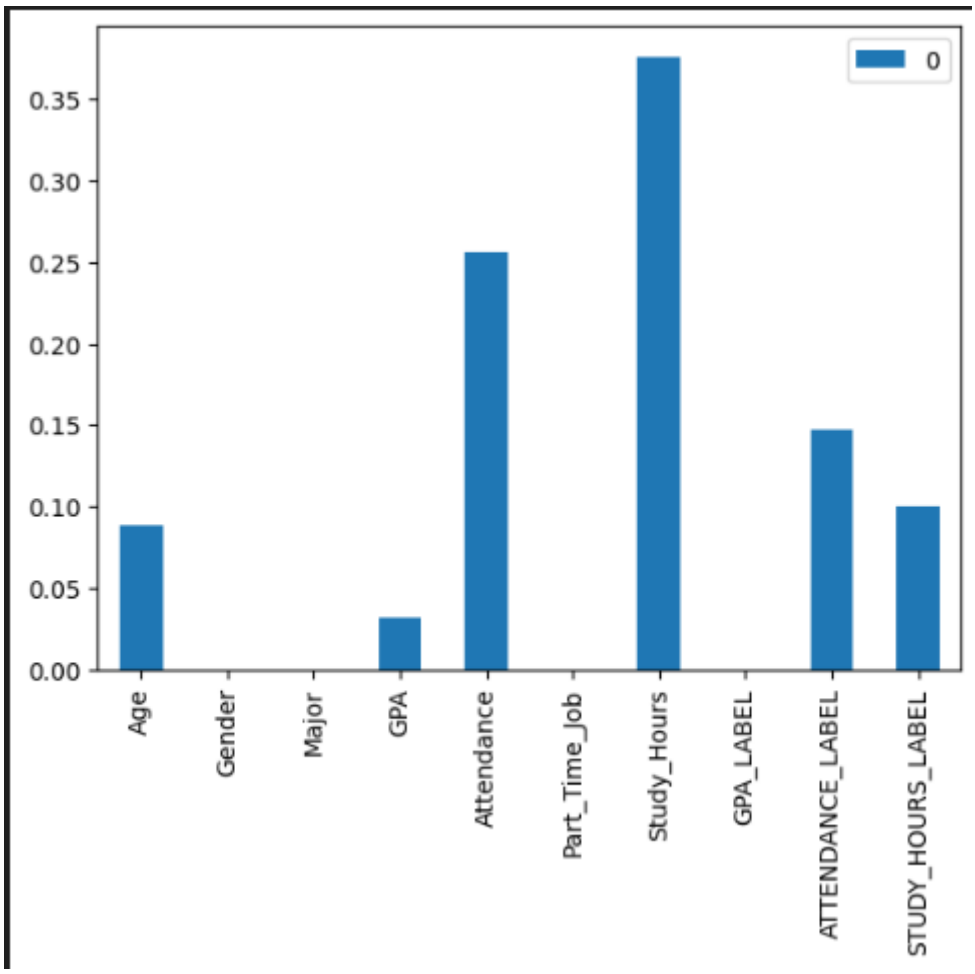
Mô hình dự đoán đúng 5 + 12 = 17/30 mẫu (56.7%)

Lớp "Yes" có số dự đoán đúng (12) cao hơn lớp "No" (5), nhưng vẫn có nhiều sai sót (6 FP và 7 FN), => mô hình chưa phân loại tốt.

- **Đánh giá độ quan trọng của các biến đầu vào trong mô hình:**

```
feature_importance=pd.DataFrame(dt.feature_importances_,index=X.columns)
feature_importance.plot(kind='bar')
```

Kết quả:



Hình 25. Đánh giá độ quan trọng của các biến đầu vào trong mô hình

feature_importance = pd.DataFrame(dt.feature_importances_, index=X.columns): Lấy độ quan trọng của các đặc trưng từ mô hình dt và tạo DataFrame.

feature_importance.plot(kind='bar'): Vẽ biểu đồ cột để hiển thị.

Kết quả: Các đặc trưng (features) như GPA, Study_Hours, Attendance, v.v., được đánh giá độ quan trọng:

- **GPA và Study_Hours có độ quan trọng cao nhất (gần 0.35 và 0.3)**
- **Các đặc trưng như Age, Gender, Major có độ quan trọng rất thấp (gần 0)**

Ý nghĩa:

- Mô hình dt (Decision Tree) coi GPA và Study_Hours là yếu tố quan trọng nhất trong việc dự đoán.
- Các đặc trưng khác như Age hay Gender hầu như không ảnh hưởng, có

thể loại bỏ để đơn giản hóa mô hình.

- **Xem các tham số đã dùng trong mô hình:**

```
- dt.get_params()
```

dt.get_params(): Hiển thị tất cả các tham số của mô hình dt

Kết quả:

```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'entropy',
 'max_depth': 5,
 'max_features': 5,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 10,
 'min_samples_split': 7,
 'min_weight_fraction_leaf': 0.024145363553254372,
 'monotonic_cst': None,
 'random_state': 42,
 'splitter': 'best'}
```

Hình 26. Hiển thị các tham số đã dùng trong mô hình

- criterion='entropy': Sử dụng entropy làm tiêu chí phân tách
- max_depth=5: Độ sâu tối đa của cây là 5
- max_features=5: Tối đa 5 đặc trưng được xem xét tại mỗi nút
- min_samples_split=7: Ít nhất 7 mẫu để phân tách một nút
- min_samples_leaf=10: Ít nhất 10 mẫu tại mỗi lá
- random_state=42: Đảm bảo tái hiện kết quả

Ý nghĩa: Các tham số này kiểm soát cách cây quyết định được xây dựng, với giới hạn về độ sâu và số mẫu để tránh overfitting.

VII. Kết luận

1. Tổng quan dữ liệu

- Tập dữ liệu gồm các thông tin sinh viên: Tuổi, Giới tính, Chuyên ngành, GPA, Chuyên cần, Làm thêm, Giờ học mỗi ngày và Kết quả (Pass/Fail).
- Các biến định lượng (tuổi, GPA, giờ học…) và định tính (giới tính, chuyên ngành…).

2. Mục tiêu bài toán

- Dự đoán khả năng sinh viên vượt qua kỳ thi dựa trên các đặc điểm cá nhân và

hành vi học tập.

3. Xử lý dữ liệu thực tế

- Phân tích dữ liệu sinh viên từ student_data.csv.
- Trực quan hóa các thuộc tính để hiểu xu hướng.
- Chuẩn bị dữ liệu để huấn luyện mô hình ID3.

4. Thư viện và công cụ sử dụng

- Scikit-learn: Dùng để xử lý dữ liệu, xây dựng mô hình, đánh giá hiệu suất và trực quan hóa.
- Optuna: Dùng để tự động tối ưu hóa siêu tham số của mô hình.
- Matplotlib, Seaborn: Hỗ trợ vẽ biểu đồ và trực quan hóa cây quyết định.

5. Xử lý dữ liệu và chia tập

- Dữ liệu gồm 150 sinh viên, với biến mục tiêu là Pass (No: Không đạt, Yes: Đạt).
- Chia dữ liệu:
  - 80% (120 mẫu) cho huấn luyện.
  - 20% (30 mẫu) cho kiểm tra.
- Dùng LabelEncoder để mã hóa các đặc trưng phân loại.

6. Tối ưu hóa siêu tham số với Optuna:

- Tối ưu các tham số:
  - max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_features
- Kiểm định chéo (cross-validation) 5 lần được dùng để đánh giá từng tổ hợp tham số.
- Kết quả: Chọn tổ hợp tham số tốt nhất để xây dựng mô hình.

7. Huấn luyện và trực quan hóa cây quyết định:

- Mô hình được huấn luyện với tham số tối ưu.
- Sử dụng plot_tree để vẽ cấu trúc cây:
  - **Attendance** là đặc trưng quan trọng nhất ở node gốc.
  - Các điều kiện như Study_Hours, GPA, Age tiếp tục phân nhánh.
  - Cây giúp rút ra **quy tắc dễ hiểu**, ví dụ:
    - Nếu Attendance > 81 → Khả năng cao là **Pass**.
    - Nếu Attendance <= 81 và Study_Hours <= 28 → Khả năng **Fail** cao.

8. Kiểm định chéo (Cross-validation):

- Hàm cross_val_score cho thấy độ chính xác dao động từ 0.42 đến 0.79.

Kết luận: Mô hình chưa ổn định, có thể do dữ liệu nhỏ hoặc phân bố không đều.

9. Đánh giá mô hình trên tập kiểm tra:

- Độ chính xác (Accuracy): 0.57 → Hiệu suất thấp.
- Báo cáo phân loại (Classification Report):
  - Lớp "No": Precision: 0.42, Recall: 0.45, F1-score: 0.43.
  - Lớp "Yes": Precision: 0.67, Recall: 0.63, F1-score: 0.65.
- Nhận xét:
  - Mô hình phân biệt lớp tốt hơn với lớp "Yes".
  - Cần cải thiện mô hình để tăng độ chính xác và độ ổn định.

VIII. Các đường dẫn:

- Link video youtube: https://youtu.be/Vo1NSWNaIBA
- Link google drive: https://drive.google.com/drive/folders/182t3VWMSNwDKg-wCRfqCkkkhT4Wi7mcW?usp=sharing

Danh mục tài liệu tham khảo

blog.vietnamlab.vn. (2019, October 09). *Decision Tree và ý nghĩa của các chỉ số Gini Impurity và Entropy*. Retrieved from Vietnamlab: https://blog.vietnamlab.vn/decistion-tree/

Klingler, N. (2024, April 14). *What is a Decision Tree?* Retrieved from Viso.ai: https://viso.ai/deep-learning/decision-trees/

miro.com. (n.d.). *What are the Steps in Decision Tree Analysis?* Retrieved from Miro: https://miro.com/diagramming/decision-tree-analysis-steps/

Rizvi, S. M. (2024, August 09). *Gini Index & Entropy: 2 Impurity Measures*. Retrieved from datasciencedojo.com: https://datasciencedojo.com/blog/gini-index-and-entropy/