M03W03 - Decision Trees

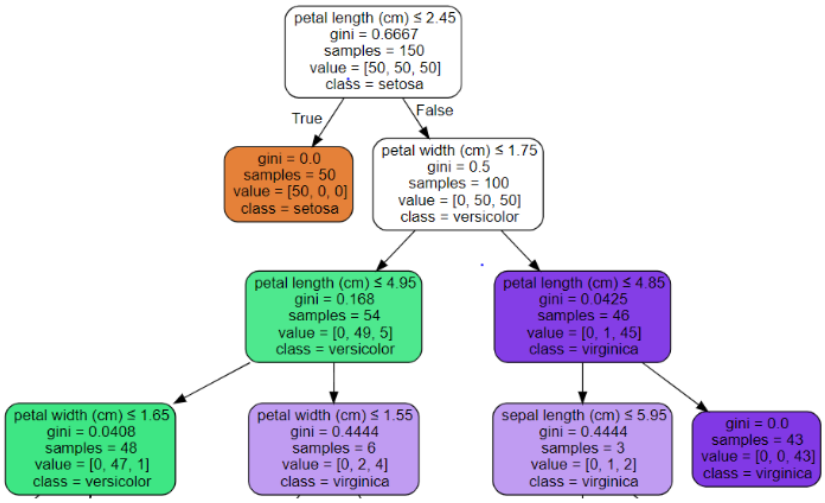
TimeSeries Team

Ngày 18 tháng 8 năm 2025

Mục lục

I. Dẫn nhập

- **Định nghĩa**: Cây quyết định là một mô hình machine learning được sử dụng cho cả bài toán **phân loại (classification)** và **hồi quy (regression)**. Nó có cấu trúc giống một cây, bao gồm các **nút (node)** và **nhánh (branch)**, mô phỏng quá trình đưa ra quyết định dựa trên các điều kiện.



Hình 1: *Cây quyết định (nguồn: Scikit Learn)*

- **Mục tiêu**: Mô hình này chia dữ liệu thành các tập con nhỏ hơn bằng cách đặt các câu hỏi (điều kiện) về các đặc trưng (features) của dữ liệu, từ đó đưa ra dự đoán.

Mặc dù Cây Quyết định trông có vẻ đơn giản và trực quan, nhưng cách thuật toán thực hiện quá trình quyết định phân tách và cắt tỉa cây lại không hề đơn giản. Trong bài viết này, tôi sẽ đưa bạn tìm hiểu rõ hơn về cơ chế hoạt động bên trong của Cây Quyết định.

II. Các Khái Niệm Liên Quan

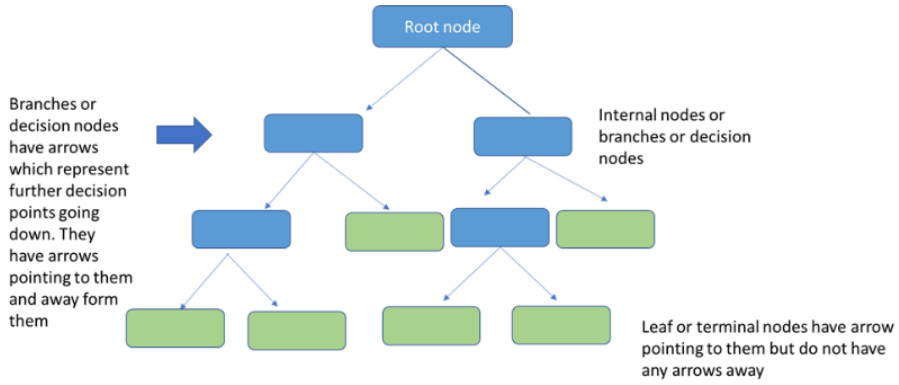
1 Các thuật ngữ cơ bản trong cây quyết định

1.1 Nút (Node)

- **Nút gốc (Root Node)**: Là nút đầu tiên của cây, đại diện cho toàn bộ tập dữ liệu. Từ đây, dữ liệu được phân chia dựa trên các điều kiện.
- **Nút quyết định (Decision Node)**: Các nút trung gian, nơi dữ liệu được phân chia tiếp tục dựa trên các điều kiện.
- **Nút lá (Leaf Node)**: Là nút cuối cùng, đại diện cho kết quả dự đoán (lớp trong bài toán phân loại hoặc giá trị trong bài toán hồi quy).

1.2 Nhánh (Branch)

: Là đường kết nối giữa các nút, đại diện cho kết quả của một điều kiện (ví dụ: "Có" hoặc "Không" với bài toán phân loại. Hoặc một giá trị cụ thể với bài toán hồi quy).



Hình 2: *Cấu trúc của cây quyết định*

Nhìn chung, một cây quyết định lấy một phát biểu, giả thuyết hoặc điều kiện, sau đó đưa ra quyết định xem điều kiện đó có đúng hay không. Các điều kiện được hiển thị dọc theo các nhánh và kết quả của điều kiện, khi áp dụng cho biến mục tiêu, được hiển thị trên nút.Mũi tên hướng ra xa một nút biểu thị điều kiện đang được áp dụng cho nút đó. Mũi tên hướng đến một nút biểu thị điều kiện đang được thỏa mãn.Cuối cùng, biến mục tiêu ngày càng trở nên thuần nhất . Điều này cuối cùng dẫn đến một dự đoán.

1.3 Cách khởi tạo cây quyết định

Để trả lời cho câu hỏi **một cây quyết định** được tạo ra như thế nào? Chúng ta cần biết được **thứ tự câu hỏi** là gì và **cách đặt câu hỏi** như thế nào?

Giả định chúng ta đang xây dựng một cây quyết định mà tại một node quyết định chúng ta có 50 quan sát rơi vào chúng. Có ba phương án lựa chọn node lá tiếp theo tương ứng với 3 biến, chúng có kết quả thống kê tại node lá lần lượt là:

- Biến 1: 25 nhãn 1, 25 nhãn 0.
- Biến 2: 20 nhãn 1, 30 nhãn 0.
- Biến 3: 0 nhãn 1, 50 nhãn 0.

Câu hỏi đặt ra, đâu là biến phù hợp nhất?

- Kịch bản lựa chọn biến 1 dường như là vô nghĩa vì nó tương đương với dự báo ngẫu nhiên nhãn 0 và 1.
- Kịch bản tương ứng với biến 2 có xu hướng dự báo thiên về nhãn 0 nhưng tỷ lệ dự báo sai nhãn 1 vẫn còn cao.
- Lựa chọn biến 3 là tuyệt vời vì chúng ta đã dự báo đúng hoàn toàn nhãn 0.

Như vậy mục tiêu khi đối diện với việc phân loại đó là kết quả trả về tại node lá chỉ thuộc về một lớp. Chúng ta có một thuật ngữ ngắn gọn cho trường hợp này là **tinh khiết (purity)**. Trái ngược lại với tinh khiết sẽ là khái niệm **vẩn đục (impurity)**, tức phân phối của các nhãn tại node lá còn khá mập mờ, không có xu hướng thiên về một nhãn nào cụ thể. Nếu ra quyết định phân loại dựa trên kịch bản dẫn tới node lá sẽ trả về kết quả không đáng tin cậy.

Cây Quyết định mang lại tính linh hoạt cao ở chỗ chúng ta có thể sử dụng cả biến số và biến phân loại để phân chia dữ liệu mục tiêu. Dữ liệu phân loại được phân chia theo các lớp khác nhau trong biến.

Biến số phức tạp hơn một chút vì chúng ta phải phân chia thành các ngưỡng cho điều kiện đang được kiểm tra, chẳng hạn như < 18 và ≥ 18. Một biến số có thể xuất hiện nhiều lần trong dữ liệu với các ngưỡng cắt hoặc ngưỡng khác nhau. Phân loại cuối cùng cũng có thể được lặp lại.

Những điều quan trọng cần lưu ý là:

- Luồng thông tin qua Cây quyết định
- Tiêu chí lựa chọn biến trong cây quyết định là gì?
- Làm thế nào nó quyết định rằng cây đã có đủ cành và ngừng tách?

Để tiếp cận thuật toán này một cách dễ hiểu nhất, ta xét một ví dụ cụ thể như sau: (Lưu ý ví dụ này được dùng xuyên suốt trong bài)

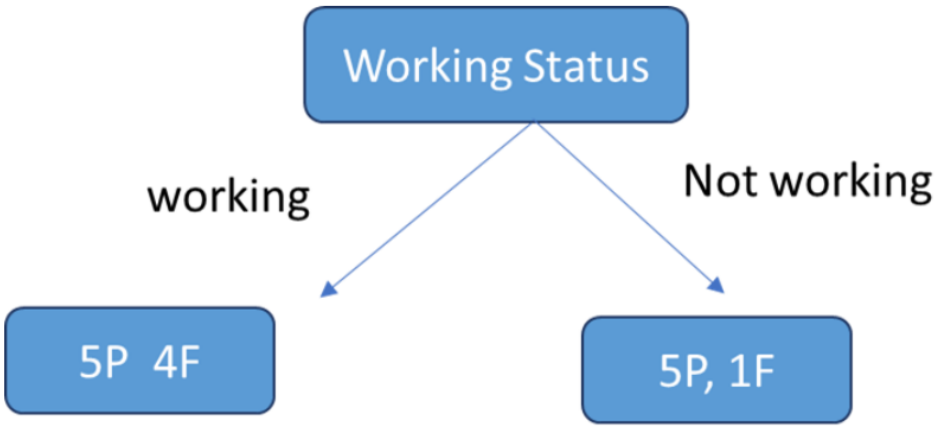
No.	Target variable **Exam result**	Predictor variable **online course**	Predictor variable **major**	Predictor variable **working status**
1	Pass	Y	Maths	N
2	Fail	N	Maths	Y
3	Fail	Y	Maths	Y
4	Pass	Y	CS	N
5	Fail	N	Other	Y
6	Fail	Y	Other	Y
7	Pass	Y	Maths	N
8	Pass	Y	CS	N
9	Pass	N	Maths	Y
10	Pass	N	CS	Y
11	Pass	Y	CS	Y
12	Pass	N	Maths	N
13	Fail	Y	Other	Y
14	Fail	N	Other	N
15	Fail	N	Maths	Y

Bảng 1: *Bộ dữ liệu ví dụ:*
gồm 15 điểm dữ liệu của học sinh về việc đậu hay trượt bài kiểm tra Machine learning

2 Luồng thông tin qua Cây Quyết Định

Cây quyết định bắt đầu với biến mục tiêu. Biến này thường được gọi là nút cha (parent node). Sau đó, Cây Quyết định tạo một chuỗi các phép chia dựa trên mức độ tác động lên biến mục tiêu này. Nút đầu tiên là nút gốc (root node), tức là biến có tác động lớn nhất lên biến mục tiêu, biến đầu tiên chia biến mục tiêu.

Để xác định nút gốc, chúng ta cần đánh giá tác động của tất cả các biến lên biến mục tiêu để xác định biến phân chia các lớp Pass/Fail thành các nhóm đồng nhất nhất. Các biến cần xet bao gồm: online course, major và working status. Chúng ta hy vọng đạt được điều gì với sự phân tách này? Giả sử chúng ta bắt đầu với working status là nút gốc. Nút này được chia thành 2 nút con: Working và Not working. Như vậy, trạng thái Pass/Fail sẽ tương ứng tại các nút con là:



Hình 3: *Ví dụ chia node*

Đây là luồng cơ bản của Cây Quyết định. Miễn là có sự kết hợp giữa Pass và Fail trong cùng một nút con, thì vẫn có thể phân tách thêm để cố gắng đưa nó về một label duy nhất. Điều này được gọi là độ tinh khiết của nút. Ví dụ: Not Working có 5 Pass và 1 Fail, do đó nó tinh khiết hơn nút Working có 5 Pass và 4 Fail. Nút lá sẽ là nút chỉ chứa một lớp duy nhất Pass hoặc Fail.

> Độ tinh khiết của một nút đo lường mức độ đồng nhất của các mẫu trong nút đó. Một nút có độ tinh khiết cao nghĩa là hầu hết các mẫu thuộc cùng một lớp (trong bài toán phân loại) hoặc có giá trị gần nhau (trong bài toán hồi quy).

Như vậy chúng ta đã sơ lược hiểu về luồng hoạt động trong Cây quyết định. Bây giờ, chúng ta hãy chuyển sang tìm hiểu phần cốt lõi của Cây quyết định - Các câu hỏi chính: **Tiêu chí lựa chọn biến trong cây quyết định là gì?** và **Làm thế nào nó quyết định rằng cây đã có đủ cành và ngừng tách?**

3 Nền tảng toán học - Tiêu chí lựa chọn biến

Đây chính là nơi thể hiện sự phức tạp và tinh vi thực sự của quyết định. Các biến được lựa chọn dựa trên một tiêu chí thống kê phức tạp được áp dụng tại mỗi nút quyết định. Hiện nay, tiêu chí lựa chọn biến trong Cây Quyết định có thể được thực hiện thông qua hai phương pháp:

- 1. Entropy and Information Gain
- 2. Gini Index

3.1 Entropy

Entropy là một thuật ngữ bắt nguồn từ vật lý và có nghĩa là thước đo mức độ hỗn loạn. Cụ thể hơn, chúng ta có thể định nghĩa nó như sau:

> **Entropy** là một khái niệm khoa học, đồng thời là một tính chất vật lý có thể đo lường được, thường gắn liền với trạng thái hỗn loạn, ngẫu nhiên hoặc bất định. Thuật ngữ và khái niệm này được sử dụng trong nhiều lĩnh vực khác nhau, từ nhiệt động lực học cổ điển , nơi nó được công nhận lần đầu tiên, đến mô tả vi mô về tự nhiên trong vật lý thống kê , và các nguyên lý của lý thuyết thông tin .

Trong lý thuyết thông tin , entropy của một biến ngẫu nhiên là mức độ trung bình của "thông tin", "sự ngạc nhiên" hoặc "sự không chắc chắn" vốn có trong các kết quả có thể xảy ra của biến đó.

Trong bối cảnh của Cây Quyết định, entropy là thước đo mức độ hỗn loạn hoặc tạp chất trong một nút. Chúng ta sẽ sử dụng Entropy để đánh giá mức độ tinh khiết của phân phối xác suất của một sự kiện. Mức entropy hoặc hỗn loạn tối đa được biểu thị bằng 1 và entropy tối thiểu được biểu thị bằng 0. Các nút lá có tất cả các mẫu thuộc về 1 lớp sẽ có entropy là 0. Trong khi đó, entropy cho một nút có các lớp được chia đều sẽ là 1.

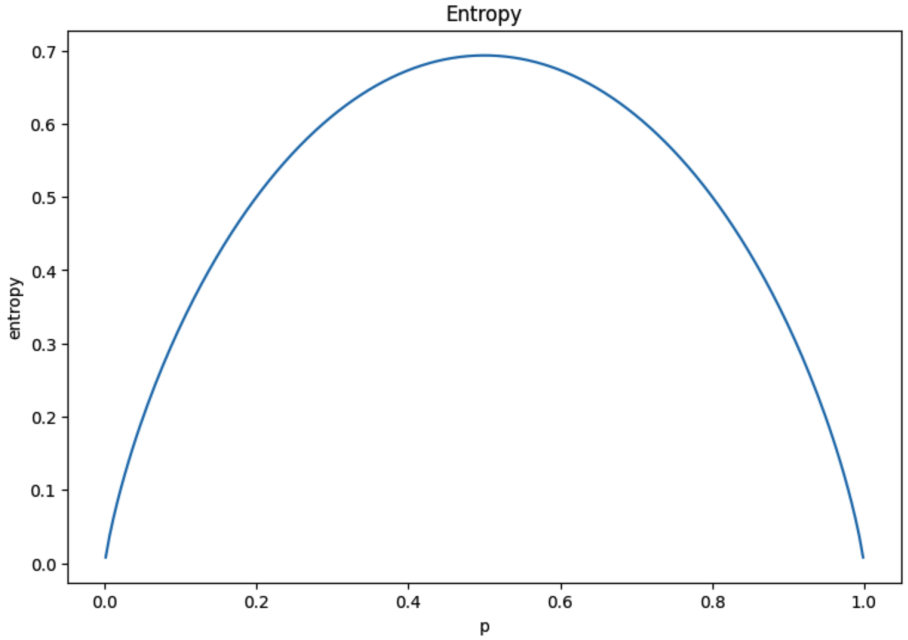
Giả sử một sự kiện xảy ra với phân phối xác suất là $p = (p_1, p_2, ..., p_C)$ thoả mãn $\sum_{i=1}^{C} p_i = 1$. Khi đó hàm *entropy* đối với sự kiện trên là:

$$\mathbf{H}(\mathbf{p}) = -\sum_{i=1}^{C} p_i \log(p_i) \tag{1}$$

Trong trường hợp $p_i = 0$ thì hàm *entropy* là không xác định do $log(p_i)$ không tồn tại, tuy nhiên căn cứ vào giá trị hội tụ thì ta sẽ qui ước giá trị của *entropy* trong trường hợp này là 0.

Trong lý thuyết thông tin thì hàm *log* ở phương trình (1) là hàm logarith với cơ số 2. Tuy nhiên ở đây chúng ta có thể sử dụng hàm logarith cơ số tự nhiên e mà không thay đổi bản chất do giá trị đạt được tương đương với việc nhân với một hằng số.

Khảo sát hàm *entropy* chúng ta sẽ nhận thấy đây là một hàm không âm có giá trị tối đa đạt được khi phân phối xác suất là đồng đều trên toàn bộ các nhãn. Để đơn giản bên dưới chúng ta vẽ đồ thị và khảo sát hàm *entropy* đối với bài toán phân loại nhị phân.



Hình 4: Entropy

Giá trị lớn nhất đạt được khi $p_0 = p_1 = 1/2$. Trong khi giá trị nhỏ nhất đạt được khi một trong hai xác suất bằng 1 và xác suất còn lại bằng 0.

Kết luận giá trị *entropy* cực tiểu đạt được khi phân phối p là tinh khiết nhất, tức phân phối hoàn toàn thuộc về một nhóm. Trái lại, *entropy* cực đại đạt được khi toàn bộ xác suất thuộc về các nhóm là bằng nhau. Một phân phối có *entropy* càng cao thì mức độ tinh khiết của phân phối đó sẽ càng thấp và ngược lại.

Như vậy về bản chất thì *entropy* là một thước đo về độ tinh khiết của phân phối xác suất. Dựa trên *entropy* chúng ta có thể đánh giá tính hiệu quả của câu hỏi ở mỗi *node* và quyết định xem đâu là câu hỏi hiệu quả hơn (có độ tinh khiết lớn hơn, entropy nhỏ hơn).

3.2 Information Gain

Information Gain là một khái niệm quan trọng trong lý thuyết thông tin và machine learning, đặc biệt là trong các mô hình Cây quyết định (Decision Tree). Nó đo lường mức độ giảm *entropy* của tập dữ liệu sau khi phân chia dựa trên một đặc trưng (feature). Information Gain càng cao, việc phân chia dữ liệu dựa trên đặc trưng đó càng hiệu quả trong việc tăng độ tinh khiết của các nút con.

Giả sử một tập dữ liệu S được phân chia thành các tập con $S_1, S_2, ..., S_k$ dựa trên một đặc trưng A Information Gain được tính bằng công thức:

$$\mathbf{IG(S, A)} = \mathbf{H(S)} - \sum_{v \in \mathbf{Value(A)}} \frac{|S_v|}{|H|} \mathbf{H(S_v)} \tag{5}$$

trong đó:

- $H(S)$ là *entropy* của tập dữ liệu S trước khi phân chia.
- $\sum \frac{|S_v|}{|S|} \mathbf{H(S_v)}$:
  - *entropy* trung bình **có trọng số** của các tập con sau khi phân chia. Trọng số $\frac{|S_v|}{|S|}$ phản ánh tỷ lệ mẫu trong mỗi tập con.
  - **Ví dụ**: Nếu một tập con S_v có $\mathbf{H(S_v)} = 0$, tất cả mẫu trong S_v thuộc cùng một lớp $\rightarrow$ Phân chia này lý tưởng.

Tại sao công thức này lại quan trọng?

- **Mục tiêu của cây quyết định**: Tìm cách phân chia dữ liệu sao cho các nút con càng "tinh khiết" (đồng nhất về lớp) càng tốt.
- **Vai trò của IG**: Information Gain đo lường **sự cải thiện** về độ tinh khiết sau khi phân chia. Nếu $\mathbf{IG(S, A)} > 0$, việc phân chia dựa trên đặc trưng A làm giảm entropy, tức tăng độ đồng nhất của các nút con. Đặc trưng có IG cao nhất sẽ được chọn để phân chia.

Tại sao phải dùng entropy trung bình có trọng số?

- **Lý do**: Các tập con S_v có thể có kích thước khác nhau. Trọng số $\frac{|S_v|}{|S|}$ đảm bảo rằng tập con lớn hơn sẽ có ảnh hưởng lớn hơn đến tổng entropy. Điều này giúp tránh thiên vị khi so sánh các đặc trưng có số lượng giá trị khác nhau.

Trường hợp đặc biệt của Information Gain

- **IG = 0**:
  - Xảy ra khi $\mathbf{H(S)} = \sum \frac{|S_v|}{|S|} \mathbf{H(S_v)}$.
  - **Ý nghĩa**: Phân chia dựa trên đặc trưng A **không làm giảm entropy** $\rightarrow$ Đặc trưng A không hữu ích.
- **IG cực đại**:

- Xảy ra khi tất cả các tập con S_v có $\mathbf{H}(\mathbf{S_v}) = 0$.
- **Ý nghĩa**: Phân chia hoàn hảo → Mỗi tập con chỉ chứa một lớp duy nhất.

Nhược điểm của Information Gain

- **Thiên vị các đặc trưng có nhiều giá trị**:
  - Đặc trưng có nhiều giá trị (ví dụ: ID người dùng) dễ tạo ra các tập con nhỏ và "tinh khiết" một cách giả tạo, dẫn đến IG cao. Tuy nhiên, điều này gây **overfitting**.
  - **Giải pháp**: Sử dụng **Information Gain Ratio** (tỷ lệ lợi ích thông tin) để chuẩn hóa IG bằng cách chia cho **Intrinsic Information** (entropy của phân phối giá trị đặc trưng).

Ví dụ minh họa sự thiên vị của Information Gain Giả sử tập dữ liệu S có 10 mẫu, trong đó:

- 5 mẫu thuộc lớp "Có", 5 mẫu thuộc lớp "Không".
- Xét hai đặc trưng:
  - **Đặc trưng A**: Một biến phân loại có 10 giá trị duy nhất (mỗi giá trị ứng với 1 mẫu).
  - **Đặc trưng B**: Một biến phân loại có 2 giá trị (ví dụ: "Nam", "Nữ"), mỗi giá trị chứa 5 mẫu (3 "Có", 2 "Không").

Tính IG cho đặc trưng A

- Mỗi tập con S_v chỉ chứa 1 mẫu → $\mathbf{H}(\mathbf{S_v}) = 0$.
- $\mathbf{IG}(\mathbf{S}, \mathbf{A}) = \mathbf{H}(\mathbf{S}) - 0 = 1 - 0 = 1$ (vì $\mathbf{H}(\mathbf{S}) = -\left(0.5 \log_2 0.5 + 0.5 \log_2 0.5\right) = 1$).

Tính IG cho đặc trưng B

- $\mathbf{H}(\mathbf{S}_{\text{Nam}}) = -\left(\frac{3}{5} \log_2 \frac{3}{5} + \frac{2}{5} \log_2 \frac{2}{5}\right) \approx 0.971$.
- $\mathbf{H}(\mathbf{S}_{\text{Nữ}}) = 0.971$.
- $\mathbf{IG}(\mathbf{S}, \mathbf{B}) = 1 - \left(\frac{5}{10} \times 0.971 + \frac{5}{10} \times 0.971\right) \approx 1 - 0.971 = 0.029$.

Kết luận

- Đặc trưng A có IG cao hơn đặc trưng B, nhưng việc phân chia theo A là vô nghĩa vì mỗi nút lá chỉ chứa 1 mẫu → Overfitting.
- **Bài học**: Information Gain dễ thiên vị các đặc trưng có nhiều giá trị. Cần sử dụng **Information Gain Ratio** để khắc phục. (Thuật toán C4.5)

3.3 Chỉ số Gini (Gini Index)

Chỉ số Gini là một lựa chọn khác bên cạnh hàm entropy được sử dụng để đo lường mức độ bất bình đẳng trong phân phối của các lớp. Chỉ số này được tính bằng cách lấy 1 trừ đi tổng bình phương tỷ lệ phần trăm ở mỗi lớp.

$$\textbf{Gini} = 1 - \sum_{i=1}^{C} p_i^2 \tag{2}$$

Ta có thể chứng minh được giá trị của Gini dao động trong khoảng từ 0 đến $1 - 1/C$. Thật vậy.

$$\begin{aligned} \underbrace{(\sum_{i=1}^{C} p_i)^2}_{1} &= \sum_{i=1}^{C} p_i^2 + 2 \sum_{1 \le j < i \le C} p_i p_j \ge \sum_{i=1}^{C} p_i^2 \\ \Leftrightarrow \quad 1 &\ge \sum_{i=1}^{C} p_i^2 \end{aligned} \tag{3}$$

Dấu '=' xảy ra khi phân phối xác suất p hoàn toàn thuộc về một lớp, tức ta thu được một cách phân chia tại *node* lá là tinh khiết thuần tuý mà không bị vẩn đục.

$$\begin{aligned} \left(\sum_{i=1}^{C} p_i \right)^2 &\le C \sum_{i=1}^{C} p_i^2 \\ \Leftrightarrow \quad \frac{1}{C} &\le \sum_{i=1}^{C} p_i^2 \end{aligned} \tag{4}$$

Đẳng thức thu được khi phân phối xác suất hoàn toàn là đồng đều giữa các lớp. Đây là trường hợp được xem là vẩn đục mà chúng ta không mong đợi xảy ra vì mục tiêu của phân chia vẫn là các quan sát bị dồn về một nhóm.

Như vậy, từ (3) và (4) $\Leftrightarrow 0 \le \textbf{Gini} \le 1 - 1/C$

Gini thường được dùng đối với những biến rời rạc có số lượng các trường hợp là lớn vì nó có tốc độ tính toán nhanh hơn so với hàm *entropy*. Trong thuật toán CART của sklearn thì chỉ số *gini* được sử dụng thay cho hàm *entropy*.

III. Thuật toán phổ biến

Như đã nói ở trên, thuật toán Cây Quyết Định là một họ thuật toán kinh điển trong học máy. Nó có thể được sử dụng cho cả phân loại và hồi quy. Phần này sẽ trình bày các thuật toán phổ biến như ID3 và C4.5, CART. Trong đó, thuật toán CART được nhấn mạnh vì scikit-learn sử dụng phiên bản tối ưu của thuật toán CART làm triển khai cho thuật toán cây quyết định.

1 ID3

1.1 Cơ sở lý luận

Vào những năm 1970, Ross Quinlan đã khám phá ra cách sử dụng entropy từ lý thuyết thông tin để đo lường quá trình ra quyết định của cây quyết định. Phương pháp của ông, đơn giản và hiệu quả, đã gây chấn động dư luận, và Quinlan đã đặt tên cho thuật toán này là ID3Iterative Dichotomiser 3. Hãy cùng xem cách thuật toán ID3 lựa chọn các đặc trưng. Đầu tiên, hãy quay trở lại với công thức entropy đã tìm hiểu ở phần II:

$$\mathbf{H}(\mathbf{X}) = -\sum_{i=1}^{n} p_i \log(p_i) \tag{1}$$

Trong đó n biểu diễn n giá trị rời rạc khác nhau của X. Và p_i biểu diễn xác suất X nhận giá trị i, và log là logarit cơ số 2 hoặc e. Ví dụ, nếu X có hai giá trị có thể xảy ra và xác suất 1 giá trị xảy ra bằng 1/2, thì X có entropy lớn nhất, đồng nghĩa X có độ không chắc chắn lớn nhất.

Khi bạn đã quen với entropy của một biến X, việc khái quát hóa nó thành entropy chung của nhiều biến sẽ rất dễ dàng. Dưới đây là biểu thức cho entropy chung của hai biến X và Y:

$$\mathbf{H}(\mathbf{X}, \mathbf{Y}) = -\sum_{x_i \in X} \sum_{y_i \in Y} p(x_i, y_i) \log(p(x_i, y_i)) \tag{1}$$

Từ entropy kết hợp, ta có thể thu được biểu thức của entropy có điều kiện H(X|Y). Entropy có điều kiện tương tự như xác suất có điều kiện. Nó đo lường độ bất định của X sau khi biết Y. Biểu thức như sau:

$$H(X|Y) = -\sum_{x_i \in X} \sum_{y_i \in Y} p(x_i, y_i) \log p(x_i|y_i) = \sum_{j=1}^{n} p(y_j) H(X|y_j) \tag{1}$$

Chúng ta vừa đề cập rằng H(X) đo lường sự không chắc chắn của X và entropy có điều kiện H(X|Y) đo lường sự không chắc chắn còn lại về X sau khi chúng ta biết Y. Vậy còn H(X)-H(X|Y) thì sao? Như bạn có thể thấy từ mô tả ở trên, nó đo lường mức độ mà sự không chắc chắn của X giảm đi sau khi chúng ta biết Y. Trong lý thuyết thông tin, số liệu này được gọi là thông tin tương hỗ, ký hiệu là I(X,Y). Trong thuật toán cây quyết định ID3, điều này được gọi là **độ lợi thông tin**. Thuật toán ID3 sử dụng **độ lợi thông tin** để xác định những đặc điểm nào sẽ được sử dụng để xây dựng cây quyết định cho nút hiện tại. Độ lợi thông tin càng lớn thì càng phù hợp để phân loại.

Thuật toán ID3 sử dụng độ lợi thông tin để xác định các đặc trưng nào làm nut phân chia hiện tại. Đặc trưng có độ lợi thông tin được tính toán cao nhất sau đó được sử dụng để thiết lập nút hiện tại trong cây quyết định.

Quá trình thuật toán ID3 diễn ra như sau:

Đầu vào: m mẫu, tập hợp đầu ra của các mẫu là D, mỗi mẫu có n thuộc tính rời rạc, tập hợp thuộc tính là A, và đầu ra là cây quyết định T.

Algorithm 1: Thuật toán ID3

```
induce_tree(D, A) begin
    if mọi ví dụ trong D đều thuộc cùng một lớp đơn c then
        return một nút lá được gán nhãn bởi lớp c
    end
    else if A là rỗng then
        return một nút lá được gán nhãn bởi lớp phổ biến nhất trong D
    end
    else
        chọn thuộc tính A_g ∈ A có thông tin tăng cường lớn nhất
        tạo một nút gốc cho cây hiện tại với nhãn là A_g
        xóa A_g ra khỏi A
        for mỗi giá trị v của thuộc tính A_g do
            tạo một nhánh mới từ nút gốc với nhãn là v
            Đặt D_v là tập hợp các ví dụ trong D có giá trị v tại thuộc tính A_g
            Gọi induce_tree(D_v, A) và gán kết quả vào nhánh mới
        end
    end
end
```

Hãy thử ID3 cho ví dụ Pass/Fail trong phần II nhé:

1.2 Hạn chế

Mặc dù thuật toán ID3 cung cấp điểm rất mới trong bài toán Cây Quyết Định, nhưng nó vẫn có nhiều điểm hạn chế.

1. ID3 không xem xét các đặc điểm liên tục, chẳng hạn như độ dài và mật độ, là các giá trị liên tục và do đó không thể áp dụng cho ID3.

2. ID3 ưu tiên các đặc điểm có mức tăng thông tin cao để thiết lập các nút trong cây quyết định. Người ta sớm phát hiện ra rằng, trong cùng điều kiện, các đặc điểm có nhiều giá trị hơn có mức tăng thông tin cao hơn các đặc điểm có ít giá trị hơn. Ví dụ, một biến có hai giá trị, mỗi giá trị là 1/2 và một biến khác có ba giá trị, mỗi giá trị là 1/3. Trên thực tế, đây là các biến hoàn toàn không chắc chắn, nhưng mức tăng thông tin của biến có ba giá trị cao hơn biến có hai giá trị.

3. Thuật toán ID3 không tính đến tình trạng thiếu giá trị.

4. ID3 không xem xét vấn đề quá khớp.

Dựa trên những thiếu sót trên, Quinlan, tác giả của thuật toán ID3, đã cải tiến thuật toán ID3, đó là thuật toán C4.5.Bạn có thể hỏi, tại sao không gọi nó là ID4, ID5 hay tên gì đó tương tự? Đó là bởi vì cây quyết định đã quá phổ biến. Ngay khi ID3 của ông ra đời, những người khác đã thực hiện các cải tiến thứ cấp ID4 và ID5. Vì vậy, ông đã chọn một cách tiếp cận khác và đặt tên cho nó là thuật toán C4.0. Phiên bản phát triển sau đó là thuật toán C4.5. Chúng ta hãy cùng tìm hiểu về thuật toán C4.5 bên dưới.

2 C4.5

2.1 Cơ sở lý luận

Như đã nói ở phần trên, C4.5 ra đời nhằm giải quyết những hạn chế của ID3.

2.1.1 Xử lý các thuộc tính liên tục

Vấn đề đầu tiên, C4.5 xử lý các thuộc tính liên tục bằng cách rời rạc hóa chúng. Giả sử có m mẫu với một thuộc tính liên tục A có các giá trị được sắp xếp từ nhỏ đến lớn là $a_1, a_2, \ldots, a_m$. C4.5 tính giá trị trung bình của hai mẫu liền kề làm điểm cắt, thu được $m-1$ điểm cắt, được biểu diễn là $T_i = \frac{a_i + a_{i+1}}{2}$. Sau đó, tính thông tin tăng cường cho từng điểm cắt. Điểm cắt có thông tin tăng cường lớn nhất sẽ được chọn làm điểm chia nhị phân. Ví dụ, chia thành hai lớp: lớn hơn t_1 và nhỏ hơn hoặc bằng t_1. Bằng cách này, chúng ta đã rời rạc hóa một thuộc tính liên tục. Điều quan trọng cần lưu ý là đây là phép chia nhị phân. Nếu thuộc tính hiện tại là liên tục, các nút con cũng có thể tiếp tục được chọn để chia.

2.1.2 Tiêu chuẩn thông tin tăng cường thiên vị

Vấn đề thứ hai, thông tin tăng cường có xu hướng ưu tiên các thuộc tính có nhiều giá trị hơn. C4.5 sử dụng **tỷ lệ thông tin tăng cường** ($I_R(D, A)$) thay cho thông tin tăng cường ($I(D, A)$). Tỷ lệ này là tỷ lệ giữa thông tin tăng cường và thông tin phân tách. Công thức được biểu diễn như sau:

$$I_R(D, A) = \frac{I(D, A)}{H_A(D)}$$

Trong đó D là tập hợp đầu ra của các mẫu, A là thuộc tính, $H_A(D)$ là Entropy của thuộc tính A, được tính bằng công thức:

$$H_A(D) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|}$$

Trong đó n là số loại giá trị của thuộc tính A, $|D|$ là số mẫu trong D, và $|D_i|$ là số mẫu có giá trị thứ i của thuộc tính A.

Tỷ lệ thông tin tăng cường càng lớn, thuộc tính càng có khả năng được chọn. Do đó, nó có thể hiệu chỉnh xu hướng thiên vị của thông tin tăng cường đối với các thuộc tính có nhiều giá trị.

2.1.3 Xử lý dữ liệu bị thiếu

Đối với vấn đề thứ ba, có hai vấn đề chính cần giải quyết. Thứ nhất, làm thế nào để chọn thuộc tính chia khi một số mẫu có giá trị bị thiếu? Thứ hai, làm thế nào để chia các mẫu bị thiếu khi đã chọn được thuộc tính chia?

2.1.3.1. Làm thế nào chọn thuộc tính khi bị thiếu dữ liệu

Giả sử một thuộc tính A nào đó có giá trị bị thiếu. C4.5 chia các mẫu thành hai phần: một phần có giá trị của thuộc tính A và một phần không có. Đầu tiên, gán trọng số 1 cho mỗi mẫu (ban đầu). Sau đó, chia các mẫu có giá trị thành các nhóm con và tính tổng trọng số của chúng. Phần còn lại không có giá trị của thuộc tính A sẽ không tham gia vào việc tính toán thông tin tăng cường, do đó chỉ sử dụng các mẫu có giá trị để tính thông tin tăng cường và tổng trọng số của chúng. Cuối cùng, trọng số của mỗi nhóm con được điều chỉnh lại để tổng trọng số bằng 1.

2.1.3.2. làm thế nào để chia các mẫu bị thiếu khi đã chọn được thuộc tính chia

Giả sử chúng ta đã chọn được thuộc tính chia. Các mẫu có giá trị bị thiếu sẽ được chia vào tất cả các nhánh con của nút. Tuy nhiên, trọng số của chúng sẽ được điều chỉnh lại theo tỷ lệ. Ví dụ, nếu mẫu bị thiếu giá trị của thuộc tính A mà A có ba giá trị $A1, A2, A3$, các giá trị này tương ứng với các trọng số 2, 3, 4. Thì mẫu này sẽ được sao chép vào ba nhánh con với trọng số được điều chỉnh lại thành $2/9, 3/9, 4/9$ tương ứng.

2.1.4 Cắt tỉa (Pruning)

Đối với vấn đề thứ tư, C4.5 giới thiệu pruning (cắt tỉa cây), đây là một bước sơ bộ trong các thuật toán cây. Chúng ta sẽ không đi sâu vào chi tiết ở đây. Phần nói về CART sẽ giải thích chi tiết hơn về các chiến lược cắt tỉa.

Ngoài bốn điểm trên, C4.5 và ID3 không khác nhau nhiều.

Quá trình thuật toán C4.5 diễn ra như sau:

Đầu vào: m mẫu, tập hợp đầu ra của các mẫu là D, mỗi mẫu có n thuộc tính rời rạc, tập hợp thuộc tính là A, và đầu ra là cây quyết định T.

Algorithm 2: Thuật toán C4.5

```
C4.5_BuildTree(D, A) begin
    if tất cả các mẫu trong D thuộc cùng một lớp đơn c then
        return một nút lá được gán nhãn bởi lớp c
    end
    if A là rỗng then
        return một nút lá được gán nhãn bởi lớp phổ biến nhất trong D
    end
    // Tìm thuộc tính tốt nhất để phân chia chọn thuộc tính A_g ∈ A có tỷ lệ thông tin
     tăng cường lớn nhất
    if tỷ lệ thông tin tăng cường của A_g dưới một ngưỡng ε hoặc không đáng kể then
        return một nút lá được gán nhãn bởi lớp phổ biến nhất trong D
    end
    tạo một nút gốc cho cây hiện tại với nhãn là A_g
    xóa A_g ra khỏi A
    for mỗi giá trị v của thuộc tính A_g do
        tạo một nhánh mới từ nút gốc với nhãn là v
        if A_g là thuộc tính liên tục then
            // Tạo 2 nhánh nhị phân đặt D_left là các mẫu có giá trị < v
            đặt D_right là các mẫu có giá trị ≥ v
            Gọi C4.5_BuildTree(D_left, A) và gán kết quả vào nhánh trái
            Gọi C4.5_BuildTree(D_right, A) và gán kết quả vào nhánh phải
        end
        else if A_g là thuộc tính rời rạc then
            Đặt D_v là các mẫu trong D có giá trị v tại thuộc tính A_g
            if D_v là rỗng then
                gán nhãn cho nút lá là lớp phổ biến nhất trong D
            end
            else
                Gọi C4.5_BuildTree(D_v, A) và gán kết quả vào nhánh mới
            end
        end
    end
    return nút gốc đã hoàn chỉnh
end
```

2.2 Hạn chế

Mặc dù C4.5 cải thiện một số vấn đề chính của thuật toán ID3, nhưng vẫn còn vài hạn chế như:

1. Vì các thuật toán cây quyết định dễ bị quá khớp, nên các cây quyết định được tạo ra phải được cắt tỉa.

2. C4.5 tạo ra một cây nhiều nhánh, nghĩa là một nút cha có thể có nhiều nút. Trong nhiều trường hợp, cây nhị phân hiệu quả hơn cây nhiều nhánh trong các hệ thống máy tính.

3. C4.5 chỉ có thể được sử dụng để phân loại; Việc áp dụng cây quyết định vào hồi quy sẽ mở rộng phạm vi của nó.

4. C4.5 sử dụng mô hình entropy, đòi hỏi một lượng lớn các phép toán logarit tốn thời gian. Đối với các giá trị liên tục, nó cũng đòi hỏi một lượng lớn các phép toán sắp xếp, điều này khiến độ phức tạp tính toán tăng cao.

3 CART

Như chúng ta đã biết, thuật toán ID3 sử dụng độ lợi thông tin để chọn các đặc trưng, ưu tiên các đặc trưng có độ lợi thông tin cao. Thuật toán C4.5 sử dụng tỷ lệ độ lợi thông tin để chọn các đặc trưng, giảm thiểu vấn đề độ lợi thông tin ưu tiên các đặc trưng có trị riêng cao. Tuy nhiên, cả ID3 và C4.5 đều dựa trên mô hình entropy của lý thuyết thông tin, bao gồm một số lượng lớn các phép toán logarit. Liệu có thể đơn giản hóa mô hình mà không làm mất hoàn toàn các ưu điểm của mô hình entropy không? Có! Thuật toán cây phân loại CART sử dụng hệ số Gini thay vì tỷ lệ độ lợi thông tin. Hệ số Gini biểu thị độ tạp chất của mô hình. Hệ số Gini càng nhỏ thì độ tạp chất càng thấp và đặc trưng càng tốt. Điều này ngược lại với tỷ lệ độ lợi thông tin.

3.1 Cơ sở lý luận

3.1.1 Tiêu chí lựa chọn biến của CART

Trong bài toán phân loại, giả sử có K labels và xác suất của phạm trù thứ k là p_k thì biểu thức của hệ số Gini là:

$$Gini(p) = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2 \tag{2}$$

Nếu là bài toán phân loại nhị phân, phép tính thậm chí còn đơn giản hơn. Nếu xác suất của kết quả mẫu đầu tiên là p, biểu thức của hệ số Gini là:

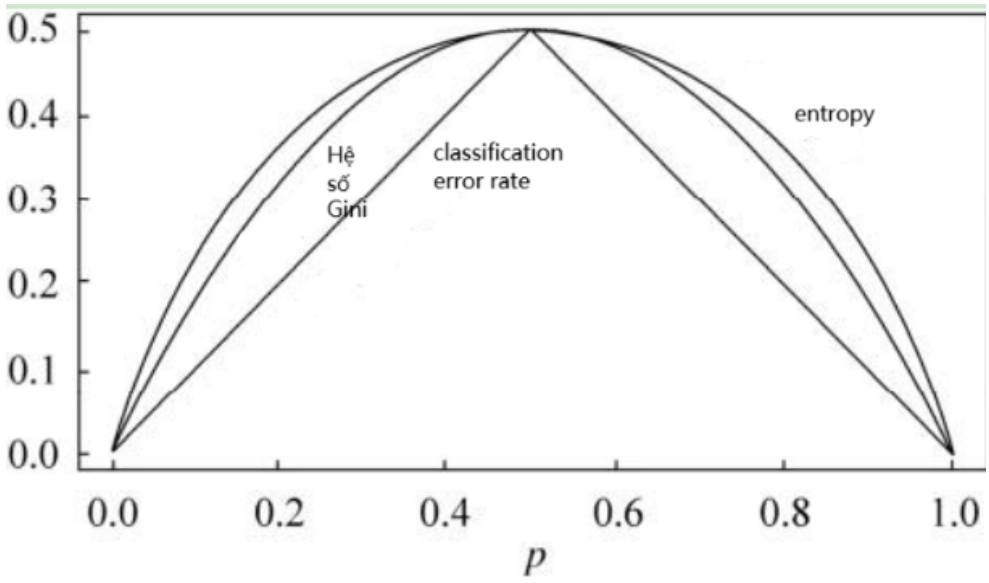
$$Gini(p) = 2p(1 - p) \tag{3}$$

Đối với một mẫu D cho trước, giả sử có K loại và số loại thứ k là C_k, thì biểu thức hệ số Gini của mẫu D là:

$$Gini(D) = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|} \right)^2 \tag{4}$$

Nếu với mẫu D, đặc trưng A theo một giá trị a nào đó chia D thành 2 phần D1 và D2 thì với điều kiện của đặc trưng A, biểu thức hệ số Gini của D là:

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \tag{5}$$



Như có thể thấy từ hình trên, đường cong hệ số Gini và đường cong entropy rất gần nhau, chỉ chênh lệch một chút ở góc 45 độ. Do đó, hệ số Gini có thể được sử dụng như một phương pháp thay thế gần đúng cho mô hình entropy. Thuật toán cây phân loại CART sử dụng hệ số Gini để lựa chọn các đặc trưng cho cây quyết định. Hơn nữa, để đơn giản hóa mô hình hơn nữa, thuật toán cây phân loại CART chỉ thực hiện phân tách nhị phân trên giá trị của một đặc trưng, thay vì phân tách đa phân vùng. Điều này tạo ra một cây nhị phân, thay vì một cây đa phân vùng. Điều này giúp đơn giản hóa việc tính toán hệ số Gini và tạo ra một mô hình cây nhị phân tinh tế hơn.

3.1.2 CART xử lý biến liên tục và rời rạc

Cây phân loại CART sử dụng cùng khái niệm với C4.5 để xử lý các giá trị liên tục: cả hai đều rời rạc hóa các đặc trưng liên tục. Điểm khác biệt duy nhất nằm ở phép đo được sử dụng để chọn các điểm phân tách: C4.5 sử dụng tỷ lệ tăng thông tin, trong khi cây phân loại CART sử dụng hệ số Gini.

Ý tưởng cụ thể như sau, ví dụ, tính năng liên tục A của m mẫu có m giá trị liên tục, sắp xếp từ nhỏ đến lớn thành $a_1, a_2, a_3..., a_m$. Thuật toán CART lấy giá trị trung bình của hai giá trị mẫu liền kề và thu được tổng cộng m-1 điểm phân vùng, trong đó điểm phân vùng thứ i,T_i được biểu diễn: $T_i = \frac{a_i+a_{i+1}}{2}$. Với mỗi điểm m-1, tính thông tin tăng cường cho từng điểm.Điểm cắt nào có Gini nhỏ nhất sẽ được chọn làm điểm chia nhị phân.Ví dụ, nếu Gini index nhỏ nhất là ở t_1 , thì chia thành hai lớp: nhỏ hơn t_1 là lớp 1 và lớn hơn hoặc bằng t_1 là lớp 2. Bằng cách này, chúng ta đã rời rạc hóa một thuộc tính liên tục. Điều cần chú ý là nó khác với ID3 hoặc C4.5 khi xử lý các thuộc tính rời rạc, nhưng nếu thuộc tính hiện tại là liên tục, thì thuộc tính này vẫn có thể tham gia vào quá trình chọn nút con. Đối với vấn đề phân loại nhị phân của thuộc tính rời rạc trong CART, tư duy là liên tục thực hiện phân loại nhị phân các thuộc tính.

Hồi tưởng lại ID3 hoặc C4.5, nếu một thuộc tính A nào đó được chọn để xây dựng cây quyết định, và nó có ba loại giá trị A1,A2,A3, chúng ta sẽ xây dựng ba nhánh trên cây quyết định. Điều này dẫn đến việc cây quyết định có nhiều nhánh. Nhưng cách CART sử dụng thì khác, nó liên tục thực hiện phân loại nhị phân. Ví dụ, CART sẽ xem xét chia thuộc tính A thành A1 và A2,A3, A2 và A1,A3, và A3 và A1,A2. Ba trường hợp này, tìm ra tổ hợp có Gini index nhỏ nhất, ví dụ, A1 và A2,A3, sau đó xây dựng hai nút con, một nút là tập hợp tương ứng với A2, nút còn lại là tập hợp tương ứng với A1,A3. Đồng thời, vì lần này không chia hoàn toàn các giá trị thuộc tính của A, nên sau đó, trong các nút con vẫn có cơ hội tiếp tục chọn thuộc tính A để chia A1 và A3. Điều này khác với ID3 hoặc C4.5. Trong ID3 hoặc C4.5, đối với một thuộc tính rời rạc, chỉ xây dựng một lần.

Quá trình thuật toán CART diễn ra như sau:

Algorithm 2: Thuật toán CART

```
CART_BuildTree(D) begin
    if tất cả các mẫu trong D đều cùng một lớp HOẶC không còn thuộc tính để chia then
        return một nút lá, kết thúc đệ quy
    end
    if Gini Index của D nhỏ hơn ϵ HOẶC số mẫu trong D nhỏ hơn k then
        return một nút lá, kết thúc đệ quy
    end
    // Tìm thuộc tính và giá trị tốt nhất để phân chia tính Gini Index cho tất cả các
     thuộc tính và giá trị của chúng trên tập dữ liệu D
    chọn cặp thuộc tính-giá trị (A_g, a) có Gini Index nhỏ nhất
    // Phân chia dữ liệu và xây dựng cây tạo một nút gốc cho cây với nhãn là A_g và điều
     kiện phân chia a
    chia D thành hai tập con D_left và D_right dựa trên điều kiện này
    // Đệ quy cho các nút con gọi CART_BuildTree(D_left) để xây dựng cây con trái
    gọi CART_BuildTree(D_right) để xây dựng cây con phải
    return nút gốc đã hoàn chỉnh
end
```

3.2 So sánh ID3, C4.5 và CART

Thuật toán	Hỗ trợ mô hình	Cấu trúc cây	Lựa chọn thuộc tính	Xử lý giá trị liên tục	Xử lý giá trị thiếu	Cắt tỉa (Pruning)
ID3	Phân loại	Đa nhánh	Thông tin tăng cường	Không hỗ trợ	Không hỗ trợ	Không hỗ trợ
C4.5	Phân loại	Đa nhánh	Tỷ lệ thông tin tăng cường	Hỗ trợ	Hỗ trợ	Hỗ trợ
CART	Phân loại, Hồi quy	Nhị phân	Gini Index, Phương sai	Hỗ trợ	Hỗ trợ	Hỗ trợ

Hình 5: *So sánh ID3, C4.5 và CART*

IV. Triển khai mô hình và Đánh giá

1 Triển khai mô hình

Mô hình Decision Tree (Cây quyết định) là một thuật toán học máy được sử dụng rộng rãi trong bài toán phân loại (classification) và hồi quy (regression). Dưới đây là các bước triển khai mô hình Decision Tree trong Python: Ví dụ: Dự đoán khả năng sống sót sau phẫu thuật tim 1. Bối cảnh thực tế Bệnh nhân phẫu thuật tim có nguy cơ biến chứng và tử vong. Dựa trên các yếu tố sức khỏe, bác sĩ có thể dự đoán khả năng sống sót sau phẫu thuật và đề xuất phương án điều trị phù hợp.

2. Dữ liệu đầu vào

- Tuổi (Age)
- Chỉ số khối cơ thể (BMI)
- Tiền sử bệnh tim mạch (Heart disease history
- Huyết áp (Blood Pressure)
- Lượng đường trong máu (Blood Sugar Level)
- Tình trạng phổi (Lung Condition)
- Chỉ số ejection fraction (đánh giá khả năng bơm máu của tim)

3. Dữ liệu đầu ra: Kết quả (Outcome): Sống sót (1) hoặc không (0)

Cách triển khai:

Bước 1: Chuẩn bị dữ liệu Dữ liệu có thể được thu thập từ các nguồn như bệnh án điện tử, kết quả xét nghiệm, hoặc thông tin sức khỏe của bệnh nhân.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

Bước 2: Giả lập dữ liệu (trong thực tế có thể dùng dữ liệu bệnh viện) Tạo bộ dữ liệu giả lập

```
data = {
    'Age': [65, 72, 58, 80, 45, 50, 77, 69, 55, 62],
    'BMI': [28, 30, 25, 27, 23, 26, 31, 29, 24, 22],
    'Heart_Disease_History': [1, 1, 0, 1, 0, 0, 1, 1, 0, 0],
    'Blood_Pressure': [140, 150, 120, 160, 115, 130, 155, 145, 125, 118],
    'Blood_Sugar_Level': [100, 120, 90, 110, 85, 95, 130, 125, 88, 92],
    'Lung_Condition': [1, 0, 0, 1, 0, 0, 1, 1, 0, 0],
    'Ejection_Fraction': [35, 40, 55, 30, 60, 50, 38, 42, 58, 65],
    'Outcome': [0, 0, 1, 0, 1, 1, 0, 0, 1, 1]
}
df = pd.DataFrame(data)
```

Bước 3: Chia dữ liệu huấn luyện và kiểm tra

```
X = df.drop(columns=['Outcome'])
y = df['Outcome']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
    =42)
```

Bước 4: Huấn luyện mô hình Decision Tree

```
model = DecisionTreeClassifier(max_depth=3, random_state=42)
model.fit(X_train, y_train)
```

Bước 5: Dự đoán và đánh giá mô hình

```
#       ch nh  x  c
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Ma  t r n   n h m   l n
cm = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:')
print(cm)

# Precision, Recall, F1-Score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1-Score: {f1:.2f}')

# AUC-ROC
auc = roc_auc_score(y_test, y_prob)
print(f'AUC-ROC: {auc:.2f}')
```

Bước 6: Trực quan hóa cây quyết định

```
plt.figure(figsize=(12, 8))
plot_tree(model, feature_names=X.columns, class_names=['Khong song sot', 'Song sot'],
    filled=True)
plt.show()
```

2 Chỉ số đánh giá

Sau khi xây dựng mô hình Decision Tree, cần đánh giá hiệu suất để xem mô hình hoạt động tốt như thế nào. Dưới đây là một số chỉ số quan trọng thường được sử dụng:

2.1 Độ chính xác (Accuracy)

- Xác định tỷ lệ dự đoán đúng so với tổng số dự đoán.
- Công thức:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Nhược điểm**: Không phù hợp khi dữ liệu mất cân bằng (imbalanced data).

2.2 Ma trận nhầm lẫn (Confusion Matrix)

- Gồm 4 giá trị:
  - **TP (True Positive)**: Dự đoán đúng lớp dương (1).
  - **TN (True Negative)**: Dự đoán đúng lớp âm (0).
  - **FP (False Positive - Type I Error)**: Dự đoán sai lớp dương (dự đoán 1 nhưng thực tế là 0).
  - **FN (False Negative - Type II Error)**: Dự đoán sai lớp âm (dự đoán 0 nhưng thực tế là 1).

2.3 Precision (Độ chính xác theo lớp dương)

- Đo lường tỷ lệ dự đoán đúng trong số các mẫu được mô hình dự đoán là dương.
- Công thức:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Ý nghĩa**: Giúp tránh các cảnh báo sai, hữu ích khi sai lầm FP có tác động lớn (ví dụ: dự đoán ung thư).

2.4 Recall (Độ nhạy, TPR - True Positive Rate)

- Đo lường khả năng phát hiện đúng các mẫu thực sự dương.
- Công thức:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **Ý nghĩa**: Quan trọng trong các bài toán y tế, vì bỏ sót một ca bệnh nghiêm trọng (FN) có thể gây hậu quả lớn.

2.5 F1-Score

- Trung bình điều hòa giữa Precision và Recall.
- Công thức:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Ý nghĩa**: Cân bằng giữa Precision và Recall, đặc biệt hữu ích khi dữ liệu không cân bằng.

2.6 ROC Curve và AUC (Area Under Curve)

- **ROC Curve** (Receiver Operating Characteristic): Đường cong hiển thị mối quan hệ giữa **TPR (Recall)** và **FPR (False Positive Rate)**.
- **AUC** (Diện tích dưới đường cong ROC): Đo độ phân tách giữa hai lớp. AUC càng cao, mô hình càng tốt.

2.7 Ví dụ:

Nếu Accuracy = 0.80, nghĩa là mô hình dự đoán đúng 80Nếu Precision = 0.75, mô hình dự đoán đúng 75Nếu Recall = 0.83, nghĩa là mô hình phát hiện đúng 83Nếu F1-Score = 0.79, mô hình có độ cân bằng tốt giữa Precision và Recall. Nếu AUC = 0.85, mô hình phân biệt khá tốt giữa bệnh nhân có bệnh và không có bệnh.

V. Code

1 Thư viện sử dụng

Thư viện cây quyết định scikit-learn triển khai nội bộ một thuật toán cây CART đã được tinh chỉnh, có thể thực hiện cả phân loại và hồi quy. Lớp tương ứng cho cây quyết định phân loại là DecisionTreeClassifier, trong khi lớp tương ứng cho cây quyết định hồi quy là DecisionTreeRegressor. Định nghĩa tham số cho hai lớp này gần như giống hệt nhau, nhưng ý nghĩa của chúng lại khác nhau. Phần sau đây tóm tắt các tham số chính của DecisionTreeClassifier và DecisionTreeRegressor, tập trung vào sự khác biệt trong cách sử dụng và những điểm chính cần lưu ý khi tinh chỉnh chúng.

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
```

2 DecisionTreeClassifier vs. DecisionTreeClassifier

Để dễ so sánh, ở đây chúng tôi sử dụng bảng để so sánh các tham số quan trọng của DecisionTreeClassifier và DecisionTreeRegressor.

Tham số	DecisionTreeClassifier	DecisionTreeRegressor
criterion	sử dụng "gini" hoặc "entropy". "gini" biểu thị hệ số Gini và "entropy" biểu thị độ lợi thông tin	sử dụng "mse" hoặc "mae". "mse" là sai số bình phương trung bình và "mae" là tổng các sai số tuyệt đối so với giá trị trung bình.
splitter	Sử dụng "best" hoặc "random". "best" phù hợp với kích thước mẫu nhỏ, "random" phù hợp kích thước mẫu rất lớn.	
max_features	Mặc định "None" - tất cả các đặc điểm đều được xem xét trong quá trình phân tách. "log2" - tối đa log2N đặc điểm được xem xet. "sqrt" hoặc "auto nghĩa làtối đa $\sqrt{N}$ đặc điểm được xem xet	
max_depth	Độ sâu tối đa của cây quyết định. Theo mặc định, tham số này không được thiết lập. Nếu không được thiết lập, cây quyết định sẽ không giới hạn độ sâu của các cây con khi xây dựng chúng. Thông thường, giá trị này có thể bị bỏ qua khi dữ liệu hoặc tính năng ít. Tuy nhiên, nếu mô hình có kích thước mẫu lớn và nhiều tính năng, nên giới hạn độ sâu tối đa. Giá trị cụ thể phụ thuộc vào phân phối dữ liệu. Các giá trị phổ biến nằm trong khoảng từ 10 đến 100.	
min_samples_split	Giá trị mặc định là 2 - giới hạn các điều kiện cho việc phân tách cây con tiếp theo. Nếu số lượng mẫu tại một nút nhỏ hơn min_samples_split, sẽ không thực hiện chọn đặc trưng tối tiếp theo cho việc phân tách. Nếu kích thước mẫu nhỏ, giá trị này không quan trọng. Nếu kích thước mẫu rất lớn, nên tăng giá trị này.	
presort	Giá trị này là giá trị Boolean, và mặc định là False, nghĩa là không sắp xếp. nếu kích thước mẫu nhỏ hoặc nếu cây quyết định bị giới hạn ở độ sâu rất nhỏ, việc đặt giá trị này thành True có thể tăng tốc độ lựa chọn điểm phân chia và xây dựng cây quyết định. Tuy nhiên, nếu kích thước mẫu quá lớn, lợi ích này không đáng kể.	

Tham số	DecisionTreeClassifier	DecisionTreeRegressor
min_samples_leaf	số lượng mẫu tối thiểu mà một nút lá có thể có, Giá trị mặc định là 1. Nếu kích thước mẫu nhỏ, giá trị này không quan trọng. Nếu kích thước mẫu rất lớn, nên tăng giá trị này.	
min_weight_fraction	tổng tối thiểu của tất cả trọng số mẫu cho một nút lá. Nếu tổng nhỏ hơn giá trị này, nút lá sẽ bị cắt tỉa. Giá trị mặc định là 0. Nếu chúng ta có nhiều mẫu bị thiếu giá trị hoặc nếu phân phối mẫu trong cây phân loại bị lệch đáng kể, trọng số mẫu nên được đưa vào.	
max_leaf_nodes	số lượng nút lá tối đa, giá trị mặc định là "None". Nếu áp dụng giới hạn, thuật toán sẽ xây dựng một cây quyết định tối ưu trong phạm vi số lượng nút lá tối đa.	
class_weight	Sử dụng "balanced" - thuật toán sẽ tự tính toán trọng số, và các mẫu tương ứng với các hạng mục có ít mẫu hơn sẽ có trọng số cao hơn. Nếu phân phối mẫu không quá lệch, sử dụng mặc định "None"	Không phù hợp trong cây quyết định hồi quy.
min_impurity_split	Giá trị này giới hạn sự phát triển của cây quyết định. Nếu độ tinh khiết (hệ số Gini, độ lợi thông tin, sai số bình phương trung bình, hiệu số tuyệt đối) của một nút nhỏ hơn ngưỡng này, nút đó sẽ không còn tạo ra các nút con nữa. Nó trở thành một nút lá.	

Hình 6: *DecisionTreeClassifier vs. DecisionTreeClassifier*

Ngoài các tham số này, những điểm khác cần lưu ý khi điều chỉnh tham số bao gồm:

- Khi số lượng mẫu nhỏ nhưng số lượng đặc trưng cao, cây quyết định dễ bị quá khớp. Khi đó, nên thực hiện giảm chiều, chẳng hạn như PCA, lựa chọn đặc trưng (Losso) hoặc ICA trước khi khớp mô hình cây quyết định. Điều này sẽ làm giảm đáng kể chiều của các đặc trưng, mang lại kết quả tốt hơn khi khớp mô hình cây quyết định.
- Nên thường xuyên sử dụng trực quan hóa cây quyết định (được thảo luận trong phần tiếp theo) và giới hạn độ sâu của cây quyết định (ví dụ: tối đa 3 lớp). Điều này cho phép bạn quan sát độ khớp ban đầu của cây quyết định được tạo ra với dữ liệu trước khi quyết định có nên tăng độ sâu hay không.
- Trước khi huấn luyện mô hình, hãy chú ý đến phân phối lớp của các mẫu (chủ yếu đề cập đến cây phân loại). Nếu phân phối lớp rất không đồng đều, hãy cân nhắc sử dụng class_weight để tránh mô hình bị thiên vị quá mức về các lớp có nhiều mẫu hơn.
- Mảng cây quyết định sử dụng kiểu numpy float32. Nếu dữ liệu huấn luyện không ở định dạng này, thuật toán sẽ tạo một bản sao trước khi chạy mô hình.
- Nếu ma trận mẫu đầu vào thưa thớt, nên gọi csc_matrix để làm cho nó thưa thớt trước khi khớp và csr_matrix để làm cho nó thưa thớt trước khi dự đoán.

3 Visualize kết quả cây quyết định

Trực quan hóa cây quyết định có thể giúp chúng ta quan sát mô hình một cách trực quan và tìm ra các vấn đề trong mô hình. Ở đây, chúng tôi giới thiệu phương pháp trực quan hóa cây quyết định trong scikit-learn.

3.1 Xây dựng môi trường trực quan hóa cây quyết định

Việc trực quan hóa cây quyết định trong scikit-learn rất dễ thực hiện và cực kỳ hữu ích để bạn hiểu cách mô hình đưa ra quyết định. Dưới đây là các cách phổ biến:

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# Load data
X, y = load_iris(return_X_y=True)

# Train model
clf = DecisionTreeClassifier(max_depth=3, random_state=0)
clf.fit(X, y)

# Plot tree
plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=load_iris().feature_names, class_names=
    load_iris().target_names)
plt.show()
```

Bạn cũng có thể dùng export_graphviz() + Graphviz như sau:

```
from sklearn.tree import export_graphviz
import graphviz

dot_data = export_graphviz(clf, out_file=None,
                           feature_names=load_iris().feature_names,
                           class_names=load_iris().target_names,
```

```
                            filled=True, rounded=True,
                            special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("iris_tree")
graph.view()
```

4 Code mẫu

```
data = pd.read_csv("heart_disease.csv")

X = data.drop("target", axis=1)
y = data["target"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
    =42)

model = DecisionTreeClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

VI. Ứng dụng của Cây quyết định (Decision Tree) trong lĩnh vực Y tế

1 Bài toán trong Y Tế

Cây quyết định là một mô hình Machine Learning được sử dụng rộng rãi trong y tế nhờ khả năng dễ giải thích, linh hoạt với nhiều loại dữ liệu và hiệu quả trong việc hỗ trợ ra quyết định. Dưới đây là phân tích chi tiết về ứng dụng của cây quyết định trong y tế.

1.1 Bài toán phân loại (ví dụ)

1.1.1 Phân loại bệnh tim mạch

Bối cảnh: Xác định khả năng mắc bệnh tim mạch của bệnh nhân dựa trên các chỉ số lâm sàng và đặc điểm cá nhân.

Input (Đầu vào):

- Tuổi (x_1)
- Giới tính (x_2)
- Mức cholesterol (x_3)
- Huyết áp tâm thu (x_4)
- Tình trạng hút thuốc ($x_5 \in \{0, 1\}$)
- Bệnh tiểu đường ($x_6 \in \{0, 1\}$)
- BMI (x_7)

Output (Đầu ra):

- Nhãn phân loại:
  - $y = 1$: Bệnh nhân có nguy cơ mắc bệnh tim mạch
  - $y = 0$: Bệnh nhân không có nguy cơ mắc bệnh tim mạch

Ví dụ minh họa: Một bệnh nhân 60 tuổi, là nam, có mức cholesterol 260 mg/dL, huyết áp 150 mmHg, hút thuốc và có BMI = 30 có thể được phân loại là nguy cơ cao mắc bệnh tim mạch ($y = 1$).

1.1.2 Phân loại u ác tính và u lành tính

Bối cảnh: Chẩn đoán u từ các dữ liệu sinh thiết hoặc hình ảnh y tế, giúp phân biệt giữa u ác tính và u lành tính.

Input (Đầu vào):

- Kích thước khối u (x_1)
- Hình dạng (độ đối xứng, biên dạng) (x_2)
- Mật độ tế bào (x_3)
- Tỷ lệ phân chia tế bào (x_4)

Output (Đầu ra):

- Nhãn phân loại:
  - $y = 1$: U ác tính (Malignant)
  - $y = 0$: U lành tính (Benign)

Ví dụ minh họa: Một khối u có kích thước lớn, biên dạng không đều, mật độ tế bào cao và tỷ lệ phân chia tế bào nhanh có thể được phân loại là ác tính ($y = 1$).

1.1.3 Phân loại bệnh tiểu đường

Bối cảnh: Dự đoán khả năng mắc bệnh tiểu đường của bệnh nhân dựa trên các yếu tố lâm sàng.

Input (Đầu vào):

- Mức đường huyết lúc đói (x_1)
- BMI (x_2)
- Tuổi (x_3)
- Tiền sử gia đình ($x_4 \in \{0, 1\}$)
- Huyết áp (x_5)
- HbA1c (x_6)

Output (Đầu ra):

- Nhãn phân loại:
  - $y = 1$: Có nguy cơ mắc bệnh tiểu đường
  - $y = 0$: Không có nguy cơ mắc bệnh tiểu đường

Ví dụ minh họa: Một bệnh nhân 55 tuổi, có mức đường huyết lúc đói cao, BMI vượt mức bình thường và có tiền sử gia đình mắc tiểu đường sẽ được phân loại là có nguy cơ cao ($y = 1$).

1.2 Bài toán hồi quy (ví dụ)

1.2.1 Dự đoán mức độ hiệu quả của thuốc điều trị Covid-19

Bối cảnh: Dự đoán mức độ hiệu quả của một loại thuốc điều trị Covid-19 dựa trên đặc điểm bệnh nhân và các yếu tố lâm sàng.

Input (Đầu vào):

- **Liều lượng thuốc** (mg/ngày) (x_1)
- **Tuổi bệnh nhân** (x_2)
- **Giới tính** (x_3) (0: Nữ, 1: Nam)
- **Mức độ nghiêm trọng ban đầu của bệnh** (x_4) (thang điểm 1-10)
- **Bệnh nền** (x_5) (0: Không, 1: Có)
- **Chỉ số BMI** (x_6)
- **Mức độ tổn thương phổi qua CT scan** (x_7) (thang điểm 0-100%)
- **Số ngày điều trị** (x_8)
- **Các chỉ số sinh học khác** (CRP, SpO_2, D-dimer, v.v.) ($x_9, x_{10}, x_{11}, \dots$)

Output (Đầu ra):

- **Mức độ hiệu quả của thuốc** (y), được tính theo thang điểm từ 0 đến 100:
  - 0-40: Không có tác dụng hoặc rất ít tác dụng
  - 41-70: Có hiệu quả trung bình
  - 71-100: Hiệu quả cao

Ví dụ minh họa: Giả sử một bệnh nhân có các thông số như sau:

- **Liều thuốc**: 600 mg/ngày
- **Tuổi**: 55
- **Giới tính**: Nam (1)
- **Mức độ nghiêm trọng ban đầu**: 7
- **Bệnh nền**: Có (1)
- **BMI**: 28.5
- **Mức độ tổn thương phổi**: 45%
- **Số ngày điều trị**: 10

Sử dụng mô hình cây quyết định, mức độ hiệu quả của thuốc được dự đoán là:

$$y = 75 \quad \text{(Hiệu quả cao)}$$

1.2.2 Dự đoán mức đường huyết trong tương lai

Bối cảnh: Dự đoán mức đường huyết của bệnh nhân sau 3 tháng dựa trên chỉ số hiện tại và lối sống.
Input (Đầu vào):

- Mức đường huyết hiện tại (mg/dL) (x_1)
- Chỉ số HbA1c (x_2)
- Chế độ ăn (thang điểm 1-10) (x_3)
- Tần suất tập thể dục (số buổi tập/tuần) (x_4)
- Thuốc điều trị tiểu đường ($x_5 \in \{0, 1\}$)

Output (Đầu ra):

- Dự đoán mức đường huyết sau 3 tháng (y mg/dL)

Ví dụ minh họa: Một bệnh nhân có mức đường huyết hiện tại là 150 mg/dL, HbA1c = 7.5, chế độ ăn đạt 6 điểm, tập thể dục 2 buổi/tuần và đang dùng thuốc tiểu đường có thể được dự đoán:

$$y = 135 \text{ mg/dL}$$

1.2.3 Dự đoán chi phí điều trị bệnh nhân

Bối cảnh: Dự đoán tổng chi phí điều trị bệnh nhân dựa trên đặc điểm bệnh và dịch vụ y tế.
Input (Đầu vào):

- Loại bệnh (mã ICD-10) (x_1)
- Tuổi bệnh nhân (x_2)
- Thời gian nằm viện (ngày) (x_3)
- Số lần phẫu thuật/thủ thuật (x_4)
- Số loại thuốc sử dụng (x_5)
- Bệnh viện điều trị (hạng bệnh viện) (x_6)

Output (Đầu ra):

- Tổng chi phí điều trị (y, đơn vị: USD)

Ví dụ minh họa: Một bệnh nhân nhập viện điều trị viêm phổi, 45 tuổi, nằm viện 7 ngày, có 1 lần thủ thuật, sử dụng 5 loại thuốc và điều trị tại bệnh viện hạng 1 có thể được dự đoán có chi phí:

$$y = 8,500 \text{ USD}$$

2 Input

Ta nhận thấy rằng các ví dụ trên **Input** đều là dữ liệu dạng bảng(tabular data).

Mặc dù dữ liệu đầu vào của cây quyết định không nhất thiết phải là dữ liệu dạng bảng, nhưng trong thực tế, dữ liệu dạng bảng là loại dữ liệu phổ biến nhất được sử dụng cho cây quyết định. Dưới đây là phân tích chi tiết về các loại dữ liệu đầu vào và cách cây quyết định xử lý chúng:

2.1 Dữ liệu dạng bảng (Tabular Data)

Là dữ liệu được tổ chức dưới dạng bảng, với các hàng đại diện cho mẫu dữ liệu và các cột đại diện cho đặc trưng (features).

2.1.1 Ví dụ

- **Bệnh nhân**: Tuổi, giới tính, huyết áp, glucose máu, kết quả chẩn đoán.
- **Giao dịch**: Thời gian, số tiền, loại giao dịch, kết quả (gian lận/không gian lận).

2.1.2 Lý do phổ biến

- Cây quyết định được thiết kế để xử lý các đặc trưng rời rạc hoặc liên tục, phù hợp với cấu trúc dạng bảng.
- Dữ liệu dạng bảng dễ dàng biểu diễn và xử lý bằng các thư viện như Pandas (Python).

2.2 Các loại dữ liệu khác

Cây quyết định cũng có thể xử lý các loại dữ liệu khác, nhưng thường cần tiền xử lý để chuyển đổi về dạng phù hợp.

2.2.1 Dữ liệu văn bản (Text Data)

Ví dụ Báo cáo y tế, đánh giá bệnh nhân, mô tả triệu chứng.
Tiền xử lý

- Chuyển đổi văn bản thành các đặc trưng số (TF-IDF, Word Embedding).
- Sử dụng các kỹ thuật như Bag of Words hoặc TF-IDF để tạo vector đặc trưng.

Ứng dụng Phân loại văn bản (ví dụ: Phân loại báo cáo y tế thành các loại bệnh).

2.2.2 Dữ liệu hình ảnh (Image Data)

Ví dụ Hình ảnh X-quang, MRI, CT scan.
Tiền xử lý

- Trích xuất đặc trưng từ hình ảnh (sử dụng mạng CNN hoặc các phương pháp thủ công như HOG, SIFT). Chuyển ảnh có hàng triệu pixel thành một tập nhỏ các giá trị đặc trưng (ví dụ: 50 đặc trưng), sau đó sử dụng Decision Tree để phân loại.
- Chuyển đổi hình ảnh thành vector đặc trưng.

Ví dụ:

Ứng dụng Phân loại hình ảnh (ví dụ: Phát hiện khối u trong ảnh X-quang).

2.2.3 Dữ liệu chuỗi thời gian (Time Series Data)

Ví dụ Dữ liệu nhịp tim, huyết áp theo thời gian.
Tiền xử lý

- Trích xuất các đặc trưng thống kê (trung bình, phương sai, cực đại, cực tiểu).
- Sử dụng các kỹ thuật như Fourier Transform hoặc Wavelet Transform.

Ứng dụng Dự đoán bệnh tim mạch dựa trên dữ liệu nhịp tim.

2.2.4 Dữ liệu đồ thị (Graph Data)

Ví dụ Mạng lưới tương tác protein, mạng xã hội bệnh nhân.
Tiền xử lý

- Trích xuất các đặc trưng từ đồ thị (số lượng nút, cạnh, độ tập trung).
- Sử dụng các phương pháp như Graph Embedding.

Ứng dụng Phân tích mạng lưới lây nhiễm bệnh.

2.3 Tại sao dữ liệu dạng bảng phổ biến?

2.3.1 Dễ hiểu và dễ xử lý

Dữ liệu dạng bảng có cấu trúc rõ ràng, phù hợp với cách thức hoạt động của cây quyết định (phân chia dựa trên giá trị đặc trưng).

2.3.2 Phù hợp với bài toán phân loại và hồi quy

Cây quyết định được thiết kế để xử lý các bài toán với đầu vào là các đặc trưng rời rạc hoặc liên tục.

2.3.3 Tương thích với các công cụ phân tích

Các thư viện như Scikit-learn (Python) và các công cụ như Excel, SQL đều hỗ trợ tốt dữ liệu dạng bảng.

2.4 Kết luận

- **Dữ liệu không phải dạng bảng** (văn bản, hình ảnh, ...) **bắt buộc phải được tiền xử lý** thành dạng bảng trước khi sử dụng cây quyết định.
- **Lựa chọn mô hình**: Nếu dữ liệu phức tạp (ảnh, chuỗi thời gian), nên cân nhắc các mô hình chuyên dụng thay vì cây quyết định.

3 Output

- **Với bài toán phân loại (Classification)**: Nhãn 0 - 1
- **Với bài toán hồi quy (Regression)**: Giá trị hồi quy

VII. Tổng Kết

Hạn chế của Decision Tree

- Dễ bị overfiring nếu cây quá phức tạp.
- Nhạy cảm với dữ liệu nhiễu.
- Không hiệu quả với dữ liệu có mối quan hệ phi tuyến tính.

Tài liệu

[1] Scikit-learn Documentation, "Decision Trees", https://scikit-learn.org/stable/modules/tree.html

[2] UCI Machine Learning Repository, "Heart Disease Dataset", https://archive.ics.uci.edu/ml/datasets/Heart+Disease

article amsmath amsfonts graphicx

Ví dụ về Decision Tree với Nhiều Điều Kiện trong Y Tế

Giả sử ta có một tập dữ liệu với các thuộc tính sau:

- **BMI** (Chỉ số khối cơ thể): Cao (High), Thấp (Low)
- **HBP** (Huyết áp cao): Có (Yes), Không (No)
- **Tuổi**: Trẻ (Young), Già (Old)
- **Tiểu đường**: Có (Yes), Không (No)

Dữ liệu Mẫu

Bệnh nhân	BMI	HBP	Tuổi	Tiểu đường
1	Cao	Có	Già	Có
2	Cao	Có	Trẻ	Không
3	Cao	Không	Già	Có
4	Thấp	Không	Trẻ	Không
5	Thấp	Có	Già	Có
6	Thấp	Không	Già	Không
7	Cao	Có	Trẻ	Không
8	Cao	Không	Già	Có
9	Thấp	Có	Trẻ	Có
10	Cao	Không	Trẻ	Không

Bước 1: Tính Entropy và Information Gain (IG)

1.1. Entropy tổng của tập dữ liệu: - **Có Tiểu đường**: 5 bệnh nhân - **Không Tiểu đường**: 5 bệnh nhân

Xác suất:

$$P(\text{Có}) = \frac{5}{10} = 0.5, \quad P(\text{Không}) = \frac{5}{10} = 0.5$$

Entropy tổng:

$$H(S) = -\left(0.5\log_2 0.5 + 0.5\log_2 0.5\right) = 1$$

1.2. Entropy của từng thuộc tính:

- **BMI:** - **Cao (6 bệnh nhân)**: 3 có tiểu đường, 3 không - **Thấp (4 bệnh nhân)**: 2 có tiểu đường, 2 không

Tính Entropy của các nhóm:

$$H(\text{BMI} = \text{Cao}) = 1, \quad H(\text{BMI} = \text{Thấp}) = 1$$

- **HBP:** - **Có (5 bệnh nhân)**: 3 có tiểu đường, 2 không - **Không (5 bệnh nhân)**: 2 có tiểu đường, 3 không

Tính Entropy của các nhóm:

$$H(\text{HBP} = \text{Có}) = 0.971, \quad H(\text{HBP} = \text{Không}) = 0.971$$

Bước 2: Kết hợp nhiều điều kiện tại một nút

Giả sử ta quyết định phân chia dữ liệu không chỉ dựa vào một thuộc tính duy nhất mà sử dụng **BMI và HBP kết hợp** để làm điều kiện tại nút quyết định.

2.1. Cắt dữ liệu theo kết hợp của BMI và HBP:

- **BMI = Cao và HBP = Có** (4 bệnh nhân): - 3 có tiểu đường, 1 không
- **BMI = Cao và HBP = Không** (2 bệnh nhân): - 2 có tiểu đường, 0 không
- **BMI = Thấp và HBP = Có** (2 bệnh nhân): - 2 có tiểu đường, 0 không
- **BMI = Thấp và HBP = Không** (2 bệnh nhân): - 0 có tiểu đường, 2 không

2.2. Tính Entropy cho mỗi nhóm kết hợp:

1. **BMI = Cao và HBP = Có** (4 bệnh nhân): - **Có Tiểu đường**: 3/4 - **Không Tiểu đường**: 1/4

$$H(\text{BMI} = \text{Cao, HBP} = \text{Có}) = -\left(\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right) = 0.811$$

2. **BMI = Cao và HBP = Không** (2 bệnh nhân): - **Có Tiểu đường**: 2/2 - **Không Tiểu đường**: 0/2

$$H(\text{BMI} = \text{Cao, HBP} = \text{Không}) = 0$$

3. **BMI = Thấp và HBP = Có** (2 bệnh nhân): - **Có Tiểu đường**: 2/2 - **Không Tiểu đường**: 0/2

$$H(\text{BMI} = \text{Thấp, HBP} = \text{Có}) = 0$$

4. **BMI = Thấp và HBP = Không** (2 bệnh nhân): - **Có Tiểu đường**: 0/2 - **Không Tiểu đường**: 2/2

$$H(\text{BMI} = \text{Thấp, HBP} = \text{Không}) = 0$$

2.3. Tính Entropy trung bình cho các nhóm kết hợp:

Với 10 bệnh nhân, ta tính Entropy trung bình của các nhóm:

$$H(S|BMI\&HBP) = \frac{4}{10} \times 0.811 + \frac{2}{10} \times 0 + \frac{2}{10} \times 0 + \frac{2}{10} \times 0 = 0.3244$$

Bước 3: Tính Information Gain

Information Gain khi sử dụng **BMI và HBP** kết hợp là:

$$IG(\text{BMI, HBP}) = H(S) - H(S|BMI\&HBP) = 1 - 0.3244 = 0.6756$$

Kết luận

- **Sử dụng kết hợp nhiều thuộc tính** (như BMI và HBP) có thể giúp cải thiện quá trình phân chia và tăng độ chính xác của mô hình Decision Tree. - **Thông qua việc kết hợp các điều kiện**, chúng ta có thể tạo ra các nhánh nhỏ hơn và đặc trưng hơn để dự đoán bệnh tiểu đường, giúp mô hình dễ dàng phân loại chính xác hơn. - **Information Gain** cho thấy rằng việc kết hợp BMI và HBP cho một độ **Information Gain** khá cao (0.6756), chứng tỏ đây là một lựa chọn tốt để phân chia dữ liệu.