

DECISION TREE

Mô hình cây quyết định

CÂY QUYẾT ĐỊNH LÀ GÌ?

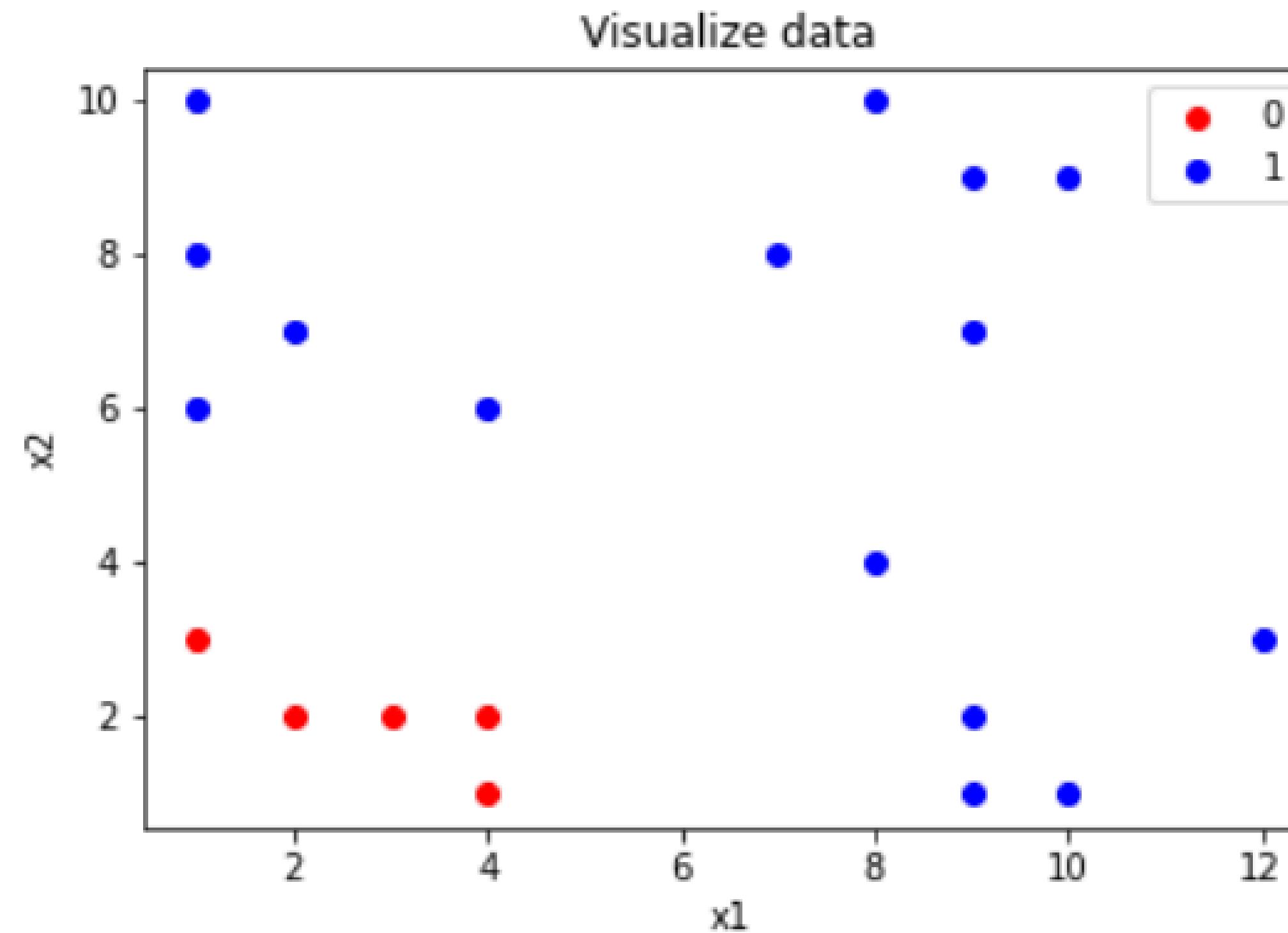
Mô hình cây quyết định là một mô hình được sử dụng khá phổ biến và hiệu quả trong cả hai lớp bài toán phân loại và dự báo của học có giám sát.

Mô hình cây quyết định không tồn tại phương trình dự báo
=>tìm ra một cây quyết định dự báo tốt trên tập huấn luyện và
sử dụng cây quyết định này dự báo trên tập kiểm tra.

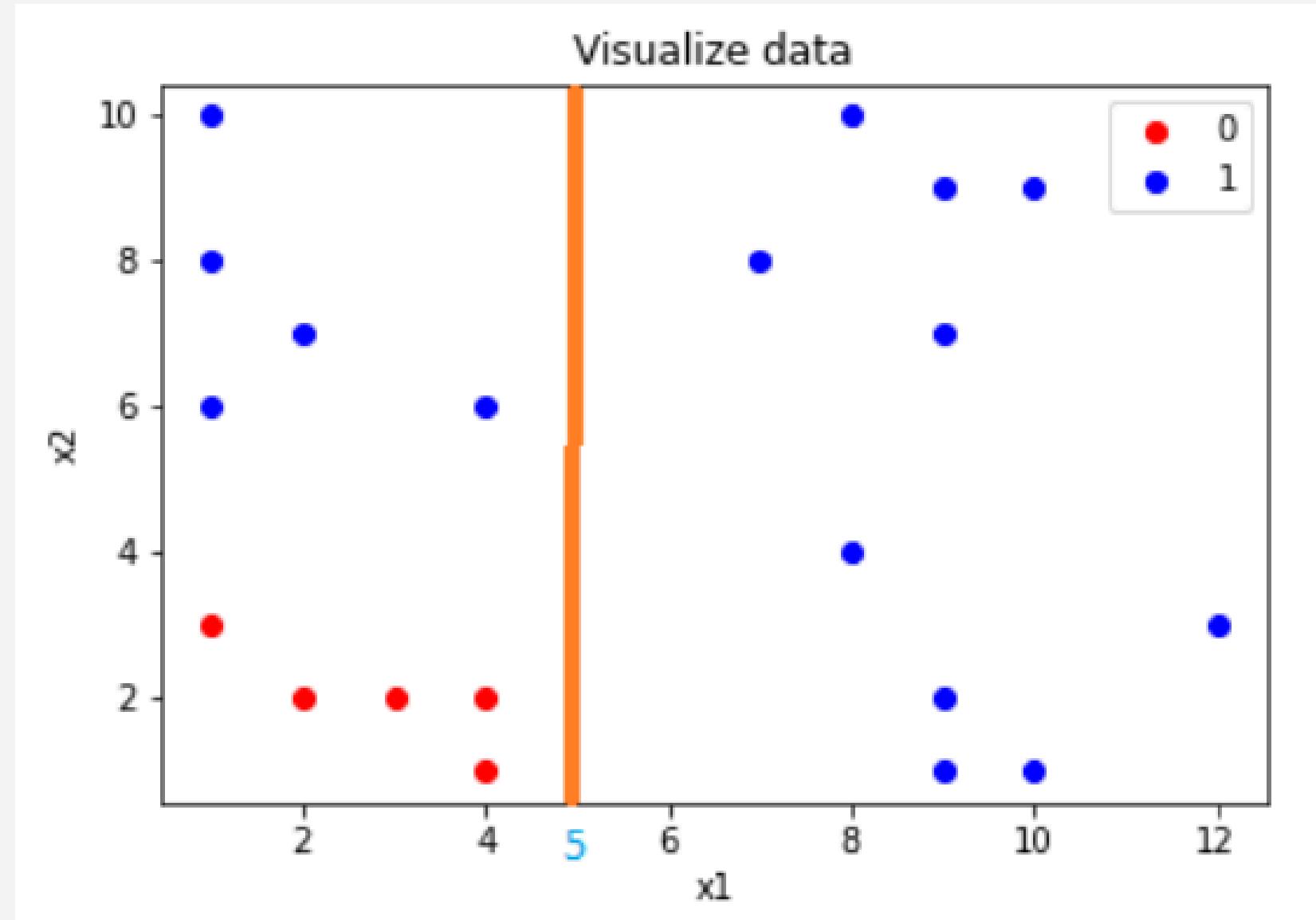
Tác dụng: phân loại, giải thích, dự đoán và phân tích số liệu



XÂY DỰNG CÂY QUYẾT ĐỊNH

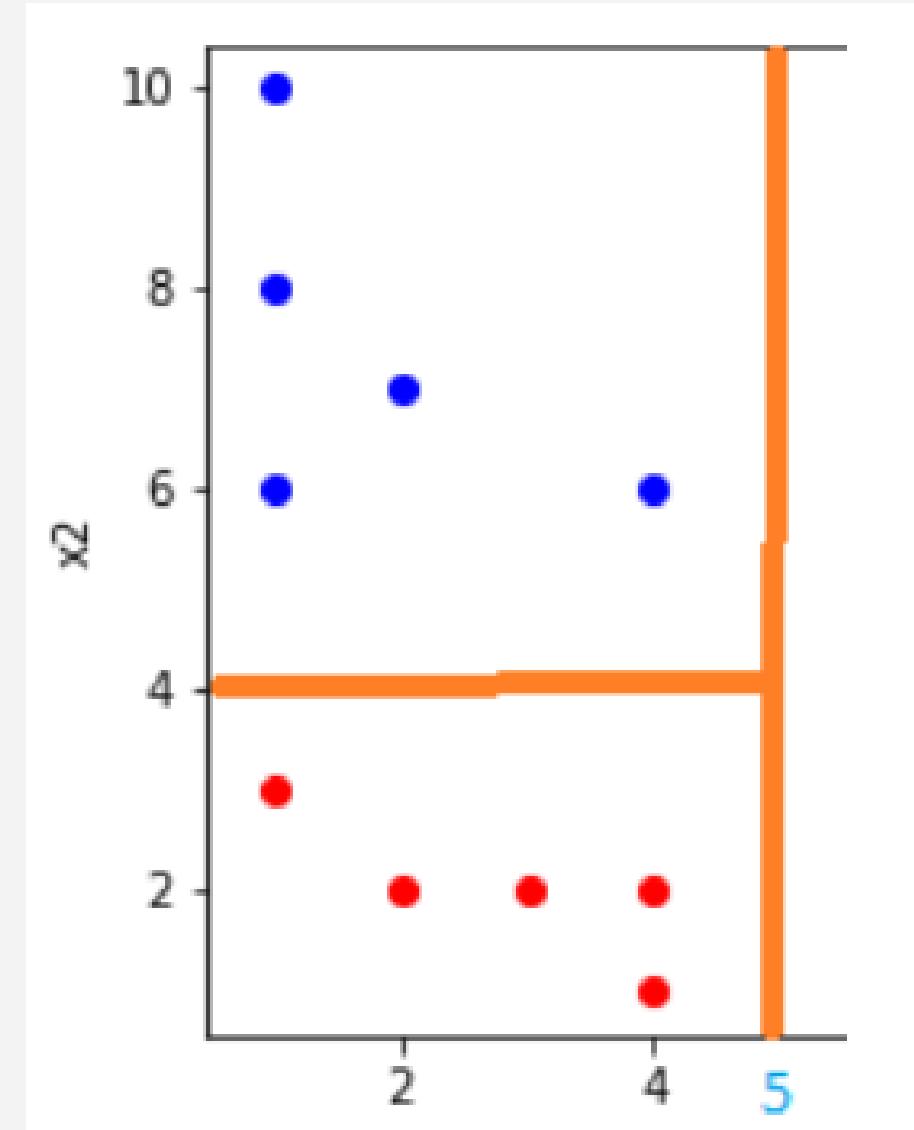


XÂY DỰNG CÂY QUYẾT ĐỊNH



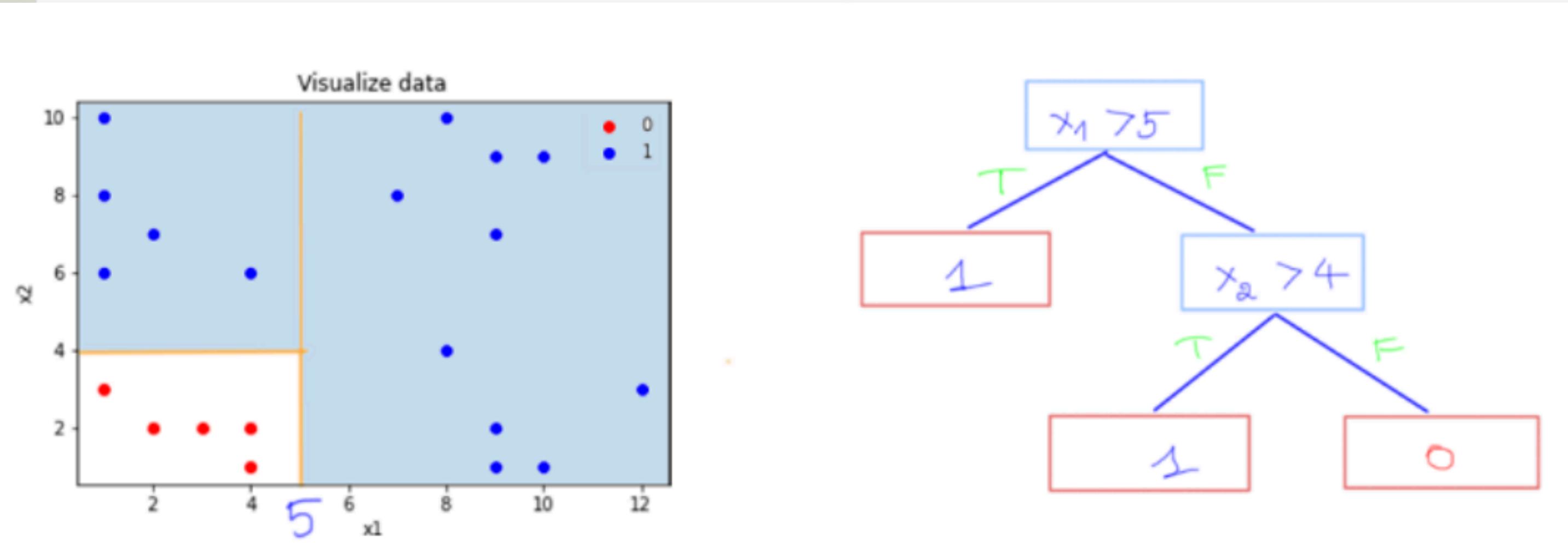
Xét điều kiện $x_1 > 5$, giống như một đường phân chia, chia dữ liệu làm 2 phần, 1 phần thỏa mãn điều kiện và 1 phần không thỏa mãn điều kiện.

XÂY DỰNG CÂY QUYẾT ĐỊNH



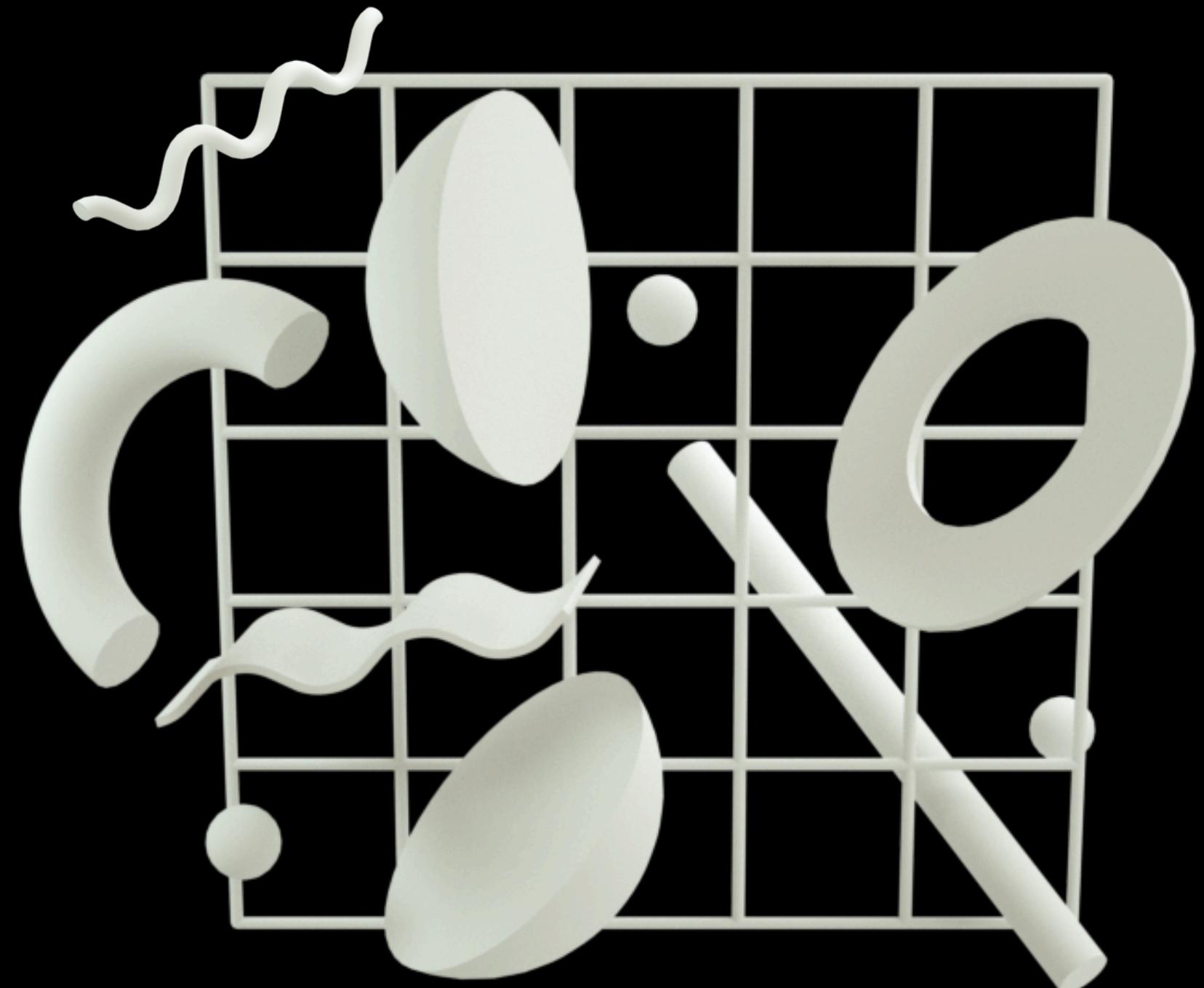
Nếu $x_1 > 5$ đúng thì tất cả các dữ liệu thuộc lớp 1, thế nên mình sẽ dùng lớp lá để dự đoán đây là lớp 1 luôn. Ngược lại thì mình thấy dữ liệu có cả lớp 1 và lớp 0, nên mình tiếp tục thêm điều kiện $x_2 > 4$

XÂY DỰNG CÂY QUYẾT ĐỊNH



CÁCH KHỞI TẠO CÂY QUYẾT ĐỊNH

Chúng ta cần biết được thứ tự câu hỏi là
gi và cách đặt câu hỏi như thế nào?





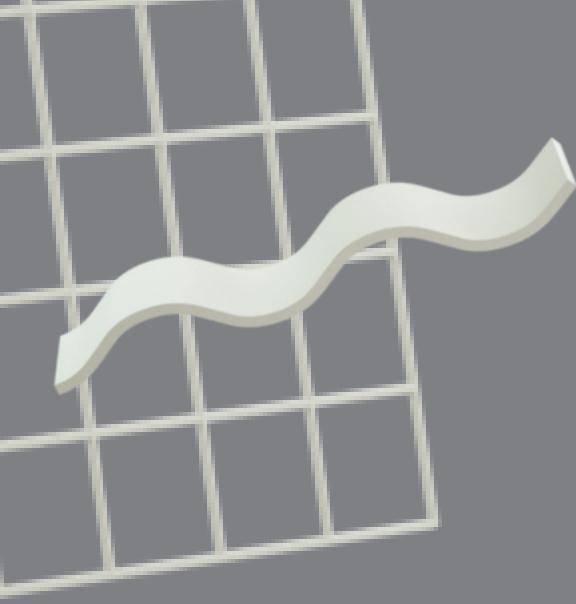
THỨ TỰ LỰA CHỌN CÂU HỎI

Đối với những bộ dữ liệu có số lượng biến đầu vào d lớn, xác suất đúng là $1/d$ và khả năng chọn sai là rất cao

- => cần phải có một tiêu chí nào đó để lựa chọn biến phù hợp
- => hình thành nên các độ đo như entropy, Gini đo lường mức độ tinh khiết (purity) và vẩn đục (impurity) của một biến



Tinh khiết và vẫn đục



TINH KHIẾT VÀ VĂN ĐỤC

Tại một node quyết định, có 50 quan sát rơi vào chúng và có 3 biến để chọn ứng với node lá tiếp theo

Biến 1: 25 nhãn 1, 25 nhãn 0.

Biến 2: 20 nhãn 1, 30 nhãn 0.

Biến 3: 0 nhãn 1, 50 nhãn 0.

==> chọn biến nào cho phù hợp





TINH KHIẾT VÀ VẤN ĐỤC

- Chọn biến 1 đường như là vô nghĩa vì nó tương đương với dự báo ngẫu nhiên
 - Chọn biến 2 có xu hướng dự báo thiên về nhãn 0 nhưng tỷ lệ dự báo sai nhãn 1 vẫn còn cao.
 - Chọn biến 3 là tuyệt vời vì chúng ta đã dự báo đúng hoàn toàn nhãn 0.
==>mục tiêu là kết quả trả về tại node lá chỉ thuộc về một lớp.
==>gọi tên trường hợp này là tinh khiết (purity). Trái ngược lại với tinh khiết sẽ là 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
- 



Thước đo độ tinh khiết

THƯỚC ĐO ĐỘ TINH KHIẾT

Trong thuật toán cây quyết định chúng ta sẽ sử dụng Entropy để đánh giá mức độ tinh khiết của phân phối xác suất của một sự kiện.

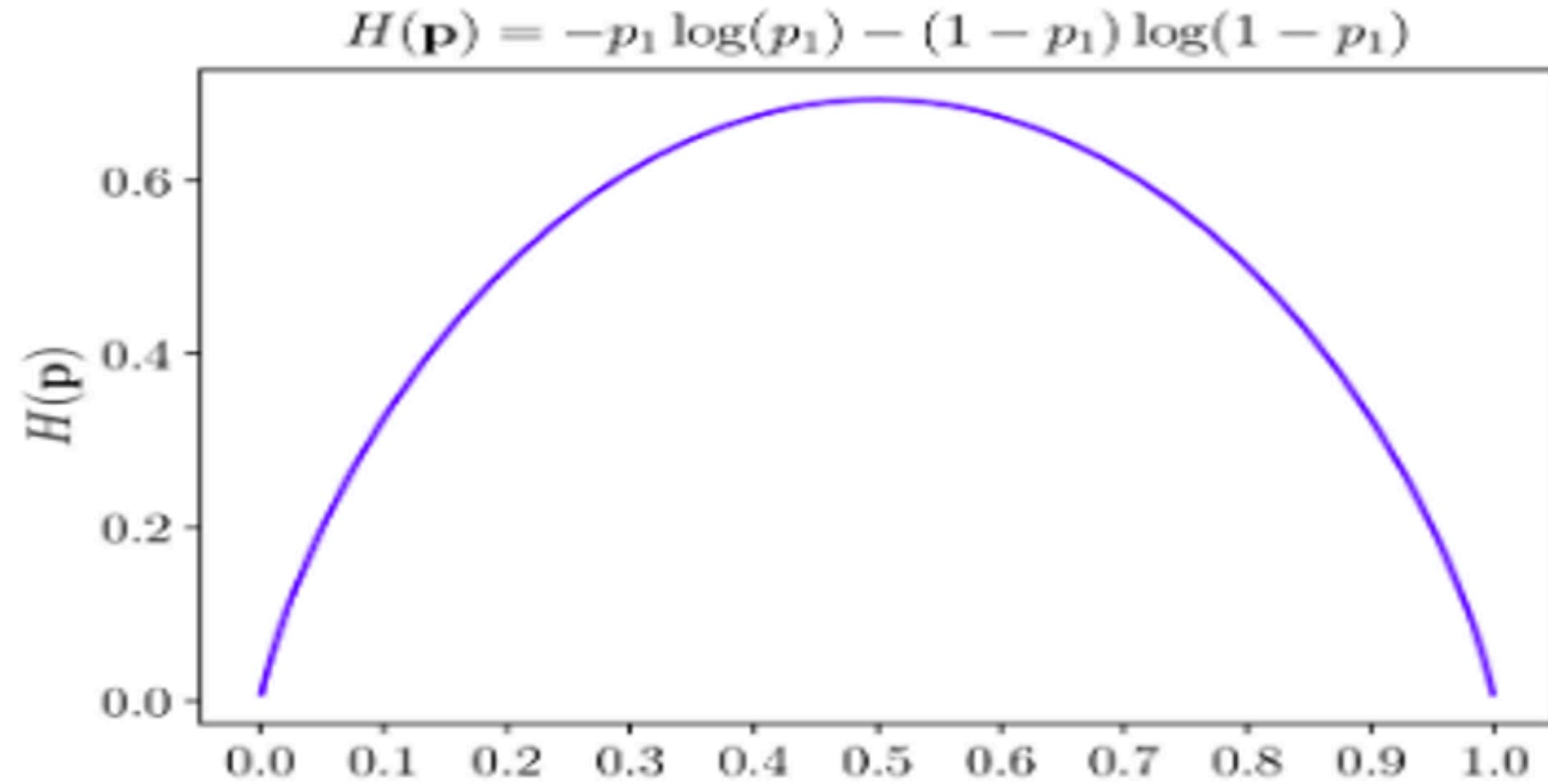
Giả sử một sự kiện xảy ra với phân phối xác suất là $p=(p_1,p_2,\dots,p_n)$ thoả mãn $\sum p_i=1(i=1..n)$. Khi đó hàm entropy đối với sự kiện trên là:

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

Trong trường hợp $p_i=0$ thì hàm entropy là không xác định do log không tồn tại, tuy nhiên chúng ta sẽ qui ước giá trị của entropy trong trường hợp này là 0.

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

THƯỚC ĐO ĐỘ TINH KHIẾT



Giá trị lớn nhất đạt được khi $p_0=p_1=1/2$.

Giá trị nhỏ nhất đạt được khi một trong hai xác suất bằng 1 và xác suất còn lại bằng 0
==>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.

THUẬT TOÁN XÂY DỰNG CÂY QUYẾT ĐỊNH

Giả sử số lượng biến là n rất lớn. Bạn muốn tạo ra một cây nhị phân với độ sâu tối đa là d. Số khả năng lựa chọn d biến (có xét đến tính thứ tự) từ m biến để tạo thành một cây nhị phân là chỉnh hợp A (d^n). Khi d lớn thì đây là một con số rất lớn.

=> rất khó để chúng ta tìm được đúng cây nhị phân tối ưu ngay chỉ trong một lần. => đi từng bước nhỏ và tìm cách lựa chọn câu hỏi tối ưu ở mỗi node(tìm kiếm tham lam)
=> Cách xây dựng cây quyết định theo phương pháp tìm kiếm tham lam và truy hồi dựa trên thuật toán ID3



THUẬT TOÁN ID3

HÀM ENTROPY ĐIỀU KIỆN



$H(Y|X)$ = trung bình các giá trị entropy điều kiện cụ thể $H(Y|X=v)$

$$H(Y|X) = \sum_i P(X=v_i)H(Y|X=v_i)$$

X	Y
Ngành	Thích chơi game
Toán	Có
Lịch sử	Không
CNTT	Có
Toán	Không
Toán	Không
CNTT	Có
Lịch sử	Không
Toán	Có

v _i	P(X = v _i)	H(Y X = v _i)
Toán	0.5	1
Lịch sử	0.25	0
CNTT	0.25	0

$$H(Y|X) = 0.5 \times 1 + 0.25 \times 0 + 0.25 \times 0 = 0.5$$

INFORMATION GAIN

$$G(x, S) = H(S) - H(x, S)$$

X	Y
Ngành Thích chơi game	
Toán	Có
Lịch sử	Không
CNTT	Có
Toán	Không
Toán	Không
CNTT	Có
Lịch sử	Không
Toán	Có

$$H(Y) = 1$$

$$H(Y|X) = 0.5$$

$$IG(Y|X) = 1 - 0.5 = 0.5$$



THUẬT TOÁN ID3

- Sử dụng trong bài toán phân loại, xây dựng cây quyết định dựa trên việc chia tập dữ liệu thành các nhánh con dựa trên các thuộc tính của dữ liệu.
- Sử dụng phương pháp tham lam tìm kiếm từ trên xuống thông qua không gian của các nhánh có thể không có backtracking

==> Các bước :

- 1.Tính entropy ban đầu của tập dữ liệu huấn luyện.
- 2.Thuộc tính có Information Gain cao nhất là thuộc tính được chọn để chia tập dữ liệu.
- 3.Tạo một nút quyết định trên cây dựa trên thuộc tính được chọn ở bước trước và chia tập dữ liệu thành các nhánh con dựa trên giá trị của thuộc tính đó.
- 4.Độ quy áp dụng các bước trên cho mỗi nhánh con, cho đến khi đạt được điều kiện dừng. Các điều kiện dừng có thể là khi tất cả các mẫu trong một nhánh có cùng nhãn hoặc không còn thuộc tính nào để chia.
- 5.Trả về cây quyết định đã xây dựng.

THUẬT TOÁN ID3

- **Ưu điểm**

1. Đơn giản và dễ hiểu: dễ tiếp cận và hiểu, không yêu cầu kiến thức chuyên sâu về toán học hoặc xử lý dữ liệu phức tạp.
2. Xử lý tập dữ liệu lớn: không ảnh hưởng đến hiệu suất và tốc độ tính toán.
3. Tính thời gian huấn luyện nhanh
4. Khả năng làm việc với dữ liệu không hoàn hảo: có thể xử lý các trường hợp dữ liệu bị khuyết thông qua các phép đo độ không chắc chắn như "information gain" để quyết định các quy tắc phân loại.
5. Độ chính xác tương đối cao: cho kết quả phân loại tốt trên các tập dữ liệu có cấu trúc tương đối đơn giản.

- **Hạn chế:**

6. Dễ bị overfitting: ID3 có xu hướng tạo ra các cây quyết định phức tạp và quá khớp (overfitting) với dữ liệu huấn luyện.
7. Không xử lý được dữ liệu liên tục: phải chuyển đổi thành dạng rời rạc trước khi sử dụng.
8. Nhạy cảm với nhiễu và dữ liệu không chính xác:
9. Không tối ưu toàn cục:

THUẬT TOÁN ID3

VÌ SAO LỰA CHỌN IG ĐỂ TÍNH TOÁN???

1. Tăng độ chính xác: Bằng cách sử dụng IG để lựa chọn các đặc trưng quan trọng, ID3 có khả năng xây dựng cây quyết định có khả năng phân loại chính xác cao hơn. Các đặc trưng quan trọng sẽ được sử dụng để xác định các quy tắc phân loại, từ đó cải thiện khả năng dự đoán của mô hình.
2. Tính toán đơn giản: IG có tính toán đơn giản và hiệu quả. Nó chỉ yêu cầu các phép đếm đơn giản và không đòi hỏi quá nhiều tính toán phức tạp ==> thực hiện nhanh chóng trên tập dữ liệu lớn.

Tuy nhiên, IG có một số hạn chế: có xu hướng ưu tiên các đặc trưng có số lượng giá trị rời rạc lớn hơn, và không xử lý trực tiếp các đặc trưng liên tục.

THUẬT TOÁN ID3



id	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no





THUẬT TOÁN ID3

b1: chọn nút gốc

- 9 yes, 5 no -> tính được entropy

$$\text{Entropy}(S) = \text{Entropy}([9+, 5-]) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.94$$

Xét thuộc tính outlook:

- S_{Sunny} : [2+, 3-] (có nghĩa là trong tập dữ liệu hiện tại (S), có 2 kết quả Yes và 3 kết quả No tại Outlook = Sunny). Tương tự:
 - S_{Overcast} : [4+, 0-].
 - S_{Rain} : [3+, 2-].



THUẬT TOÁN ID3

Lấy ví dụ với thuộc tính $A = \text{Outlook}$, ta có $\text{Value}(A) = \{\text{Sunny}, \text{Overcast}, \text{Rain}\}$, và $S_{\text{Sunny}} = [2+, 3-]$ như đã tính ở trên

Từ công thức, dễ dàng tính được:

$$\text{Gain}(S, \text{Outlook}) = \text{Entropy}([9+, 5-]) - \sum_{v \in \text{Value}(\text{Outlook})} \frac{|S_v|}{14} \text{Entropy}(S_v) = 0.246$$

Hoàn toàn tương tự, tính được Information Gain cho 3 thuộc tính còn lại:

$$\text{Gain}(S, \text{Temp}) = 0.029$$

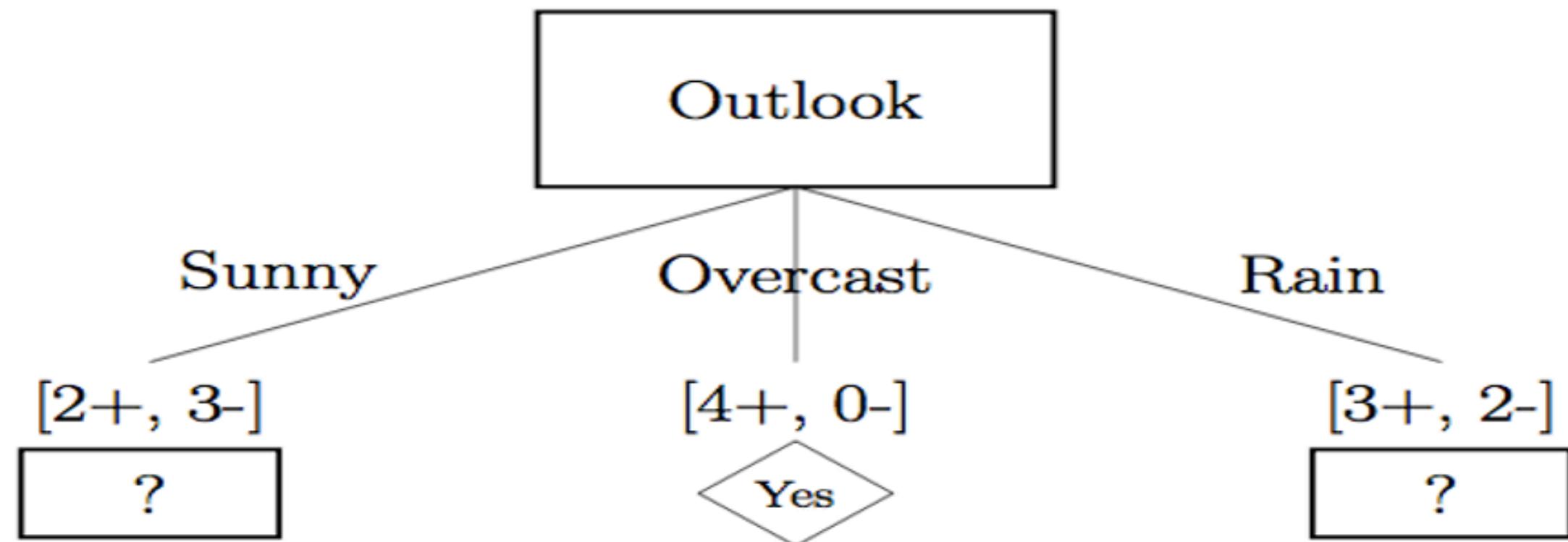
$$\text{Gain}(S, \text{Humidity}) = 0.152$$

$$\text{Gain}(S, \text{Wind}) = 0.048$$

THUẬT TOÁN ID3



Thuộc tính Outlook có Information Gain cao nhất, chọn nó làm nút gốc.





THUẬT TOÁN ID3

b2. Xây dựng cây quyết định

xét thuộc tính sunny:

Hoàn toàn tương tự như cách tìm nút gốc, ta tính Information Gain cho 3 thuộc tính còn lại là `Temp`, `Humidity` và `Wind` (trên tập S_{Sunny}).

Xét thuộc tính `Humidity`, có:

- S_{Normal} : $[2+, 0-]$ (nghĩa là tại những dữ liệu có `Outlook = Overcast` và `Humidity = Normal`, có 2 dữ liệu, tất cả đều cho kết quả Yes).
- S_{High} : $[0, 3-]$.

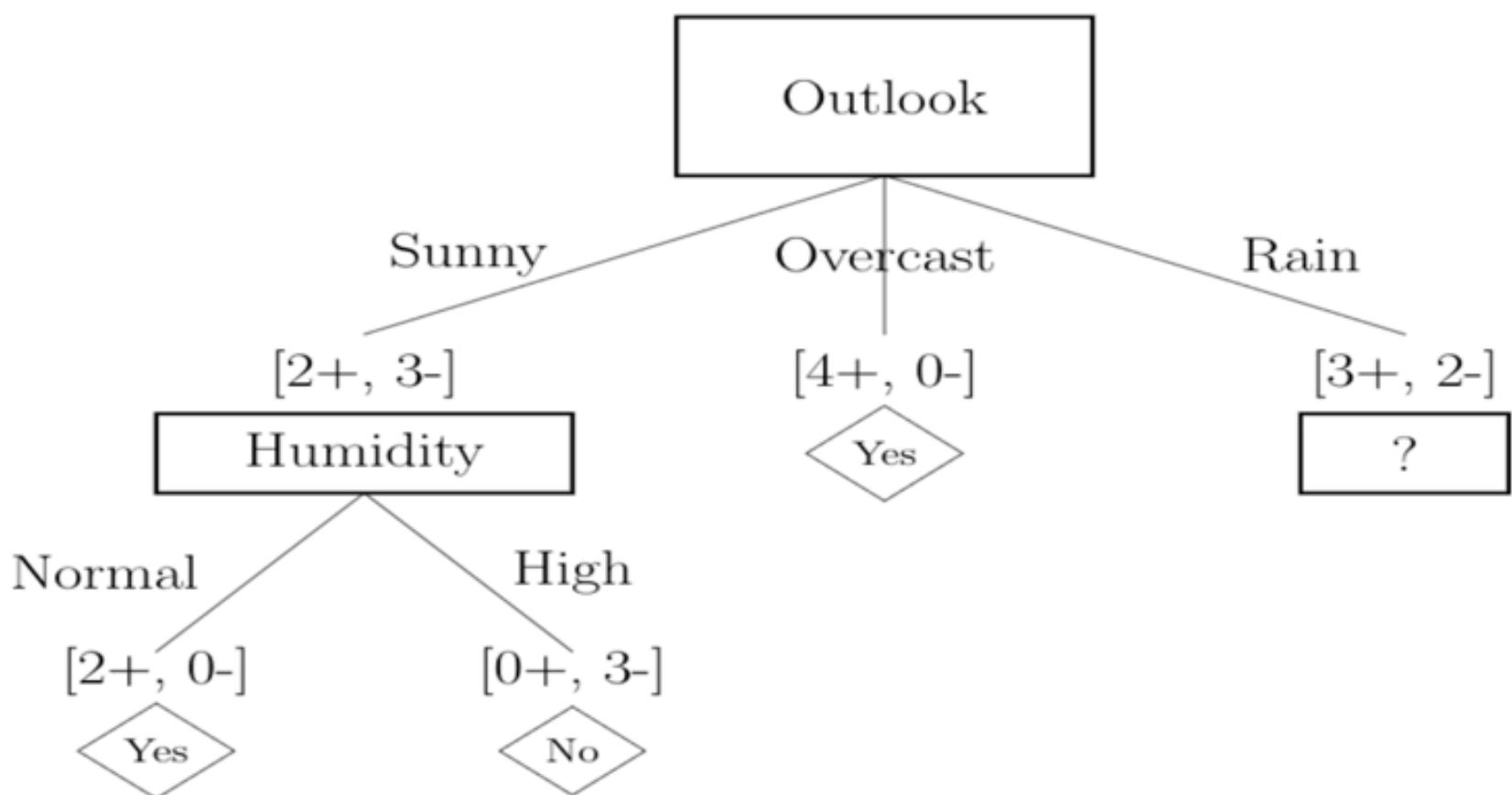
Từ đó:

$$Gain(S_{Sunny}, \text{Humidity}) = Entropy([2+, 3-]) - \sum_{v \in Value(\text{Humidity})} \frac{|S_v|}{5} Entropy(S_v) = 0.971$$

THUẬT TOÁN ID3



Nhận thấy thuộc tính **Humidity** có Information Gain cao nhất, chọn thuộc tính này làm nút ch nhánh trái cùng.

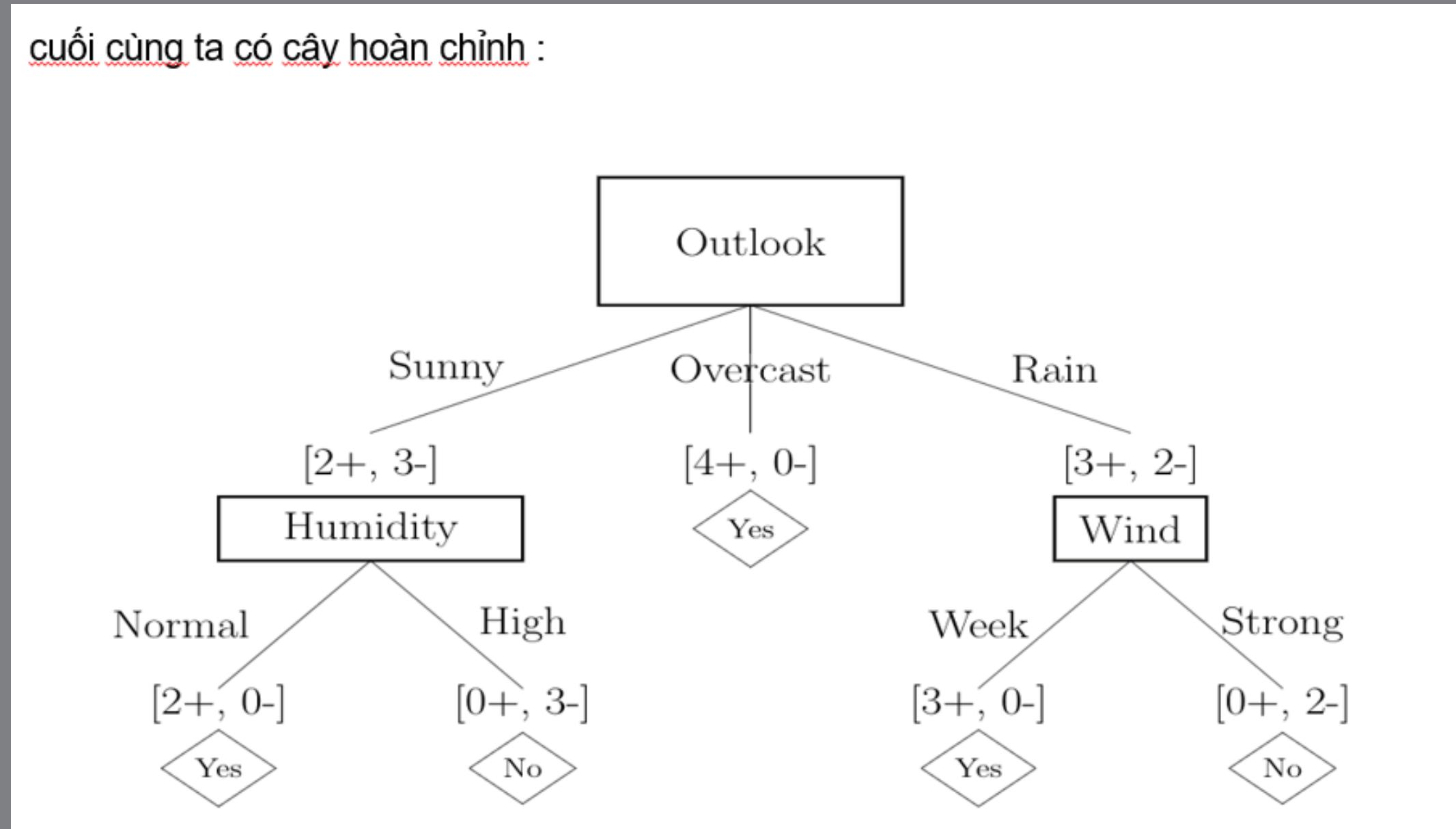


Hình 2. Cây quyết định sau khi chọn nút cho nhánh trái cùng.

THUẬT TOÁN ID3



cuối cùng ta có cây hoàn chỉnh :





THUẬT TOÁN C4.5



THUẬT TOÁN C4.5

- Phù hợp với bộ dữ liệu nhỏ
- Cải tiến của id3, sử dụng GR để chọn thuộc tính thay vì IG như id3 trong bài toán phân loại

Spliting entropy của thuộc tính F_i , ký hiệu $SE(F_i)$:

$$SE_{\mathbf{D}}(F_i) = - \sum_{j=1}^{P_i} \frac{|\mathbf{D}_j|}{|\mathbf{D}|} \log_2 \frac{|\mathbf{D}_j|}{|\mathbf{D}|}$$

Khi đó, Gain Ratio ký hiệu $GR_{\mathbf{D}}(C | F_i)$

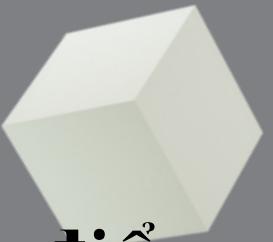
$$GR_{\mathbf{D}}(C | F_i) = \frac{IG_{\mathbf{D}}(C | F_i)}{SE_{\mathbf{D}}(F_i)}$$

- Tiêu chí Information Gain thường "ưu tiên" chọn những thuộc tính có nhiều giá trị (miền xác định lớn)
- Spliting entropy, $SED(F_i)$ sẽ lớn khi thuộc tính F_i có nhiều giá trị. Điều này giúp:
 - Làm giảm Gain Ratio của thuộc tính có nhiều giá trị.
 - Làm tăng Gain Ratio của thuộc tính có ít giá trị.

THUẬT TOÁN C4.5 KHI XỬ LÍ DỮ LIỆU THIẾU



1. Tính toán thông số chọn đặc trưng (Gain Ratio): xử lý các giá trị thiếu bằng cách xem chúng như một giá trị riêng biệt, thường được gọi là "missing value". Điều này cho phép C4.5 tính toán thông số chọn đặc trưng dựa trên tất cả các giá trị có sẵn và giá trị "missing value".
2. Xử lý giá trị thiếu trong quá trình phân loại: C4.5 sẽ xem xét các quy tắc phân loại đã học từ dữ liệu huấn luyện. Nếu một quy tắc yêu cầu giá trị của đặc trưng thiếu, C4.5 sẽ sử dụng giá trị dự đoán từ nhánh phù hợp nhất của cây để phân loại mẫu dữ liệu đó.
3. Xử lý giá trị thiếu trong quá trình xây dựng cây: Trong quá trình xây dựng cây quyết định, khi C4.5 chọn một đặc trưng để chia nhánh, nếu một mẫu dữ liệu có giá trị thiếu cho đặc trưng đó, C4.5 sẽ chia nhánh thành hai nhánh: một nhánh cho các mẫu dữ liệu có giá trị không bị thiếu và một nhánh cho các mẫu dữ liệu có giá trị thiếu. Các mẫu dữ liệu có giá trị thiếu sẽ được xem xét trong từng nhánh riêng biệt hoặc được gắn vào nhánh con khác để tiếp tục xây dựng cây.



THUẬT TOÁN C4.5

Ưu điểm

- Mô hình dễ hiểu và dễ giải thích.
- Cần ít dữ liệu để huấn luyện.
- Có thể xử lý tốt với dữ liệu dạng số (rời rạc và liên tục) và dữ liệu hạng mục.
- Mô hình dạng white box rõ ràng.
- Xây dựng nhanh.
- Phân lớp nhanh.

=> C4.5 là một biến thể cải tiến của ID3, có khả năng xử lý dữ liệu liên tục và dữ liệu bị khuyết, giảm thiểu overfitting, hỗ trợ phân loại đa lớp và cho kết quả phân loại chính xác cao.

Nhược điểm

- Không đảm bảo xây dựng được cây tối ưu.
- Có thể overfitting (tạo ra những cây khớp với dữ liệu huấn luyện hay quá phức tạp).
- Thường ưu tiên thuộc tính có nhiều giá trị (khắc phục bằng các sử dụng Gain Ratio).



THUẬT TOÁN C4.5

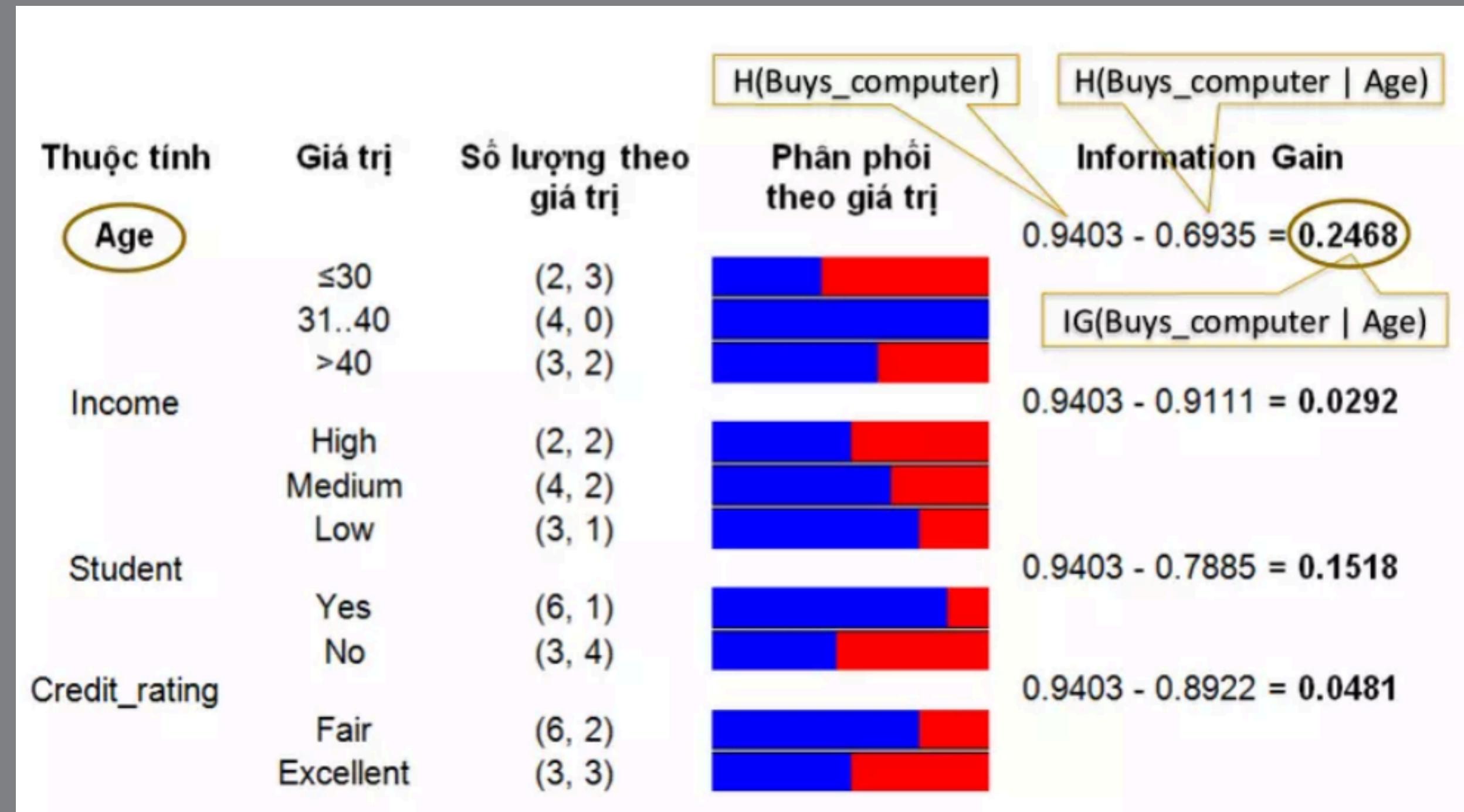
Các thuộc tính điều kiện (dùng để phân loại)

Thuộc tính cần phân loại
(target attribute)

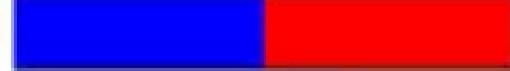
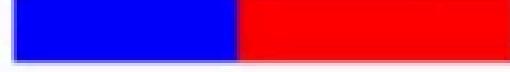
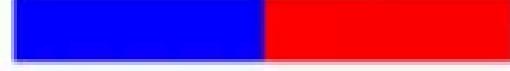
ID	Age	Income	Student	Credit_rating	Buys_computer
1	≤ 30	High	No	Fair	No
2	≤ 30	High	No	Excellent	No
3	31..40	High	No	Fair	Yes
4	>40	Medium	No	Fair	Yes
5	>40	Low	Yes	Fair	Yes
6	>40	Low	Yes	Excellent	No
7	31..40	Low	Yes	Excellent	Yes
8	≤ 30	Medium	No	Fair	No
9	≤ 30	Low	Yes	Fair	Yes
10	>40	Medium	Yes	Fair	Yes
11	≤ 30	Medium	Yes	Excellent	Yes
12	31..40	Medium	No	Excellent	Yes
13	31..40	High	Yes	Fair	Yes
14	>40	Medium	No	Excellent	No



THUẬT TOÁN C4.5



THUẬT TOÁN C4.5

Thuộc tính	Giá trị	Số lượng theo giá trị	Phân phối theo giá trị	Information Gain	Gain ratio
Age	≤ 30	(2, 3)		0.2468	$0.2468 / 1.5774 = 0.1565$
	31..40	(4, 0)			
	>40	(3, 2)			
Income	High	(2, 2)		0.0292	$0.0292 / 1.5567 = 0.0187$
	Medium	(4, 2)			
	Low	(3, 1)			
Student	Yes	(6, 1)		0.1518	$0.1518 / 1 = 0.1518$
	No	(3, 4)			
	Fair	(6, 2)			
Credit_rating	Excellent	(3, 3)		0.0481	$0.0481 / 0.9852 = 0.04882$



SO SÁNH ID3 VÀ C4.5

	ID3	C4.5
Xử lý đặc trưng có giá trị rời rạc và liên tục	ID3 chỉ hỗ trợ xử lý đặc trưng có giá trị rời rạc, và không thể xử lý đặc trưng liên tục trực tiếp.	C4.5 hỗ trợ cả đặc trưng có giá trị rời rạc và liên tục.
Xử lý dữ liệu thiếu	ID3 không hỗ trợ xử lý dữ liệu thiếu một cách trực tiếp.	C4.5 hỗ trợ xử lý dữ liệu thiếu bằng cách sử dụng phép đo "missing value" để tính toán thông số chọn đặc trưng và xử lý các giá trị thiếu trong quá trình phân loại.
Xử lý overfitting	ID3 có xu hướng overfitting khi xây dựng cây quyết định. Nó có xu hướng tạo ra cây phức tạp và có thể quá khớp dữ liệu huấn luyện.	Bằng cách sử dụng kỹ thuật "pruning" (cắt tỉa) trên cây quyết định sau khi xây dựng. Các nhánh không cần thiết sẽ được loại bỏ để giảm kích thước cây và ngăn chặn overfitting.
Tính toán đơn giản	ID3 có tính toán đơn giản và hiệu quả, tuy nhiên, không phải lúc nào cũng tạo ra cây quyết định tối ưu.	C4.5 cũng có tính toán đơn giản và hiệu quả, nhưng thường tạo ra cây quyết định tối ưu hơn



THUẬT TOÁN CART

GINI INDEX

Gini index tương tự như information gain, dùng để đánh giá xem việc phân chia ở node điều kiện có tốt hay không. Trước hết mình sẽ tính chỉ số Gini, chỉ số Gini tính ở từng node.

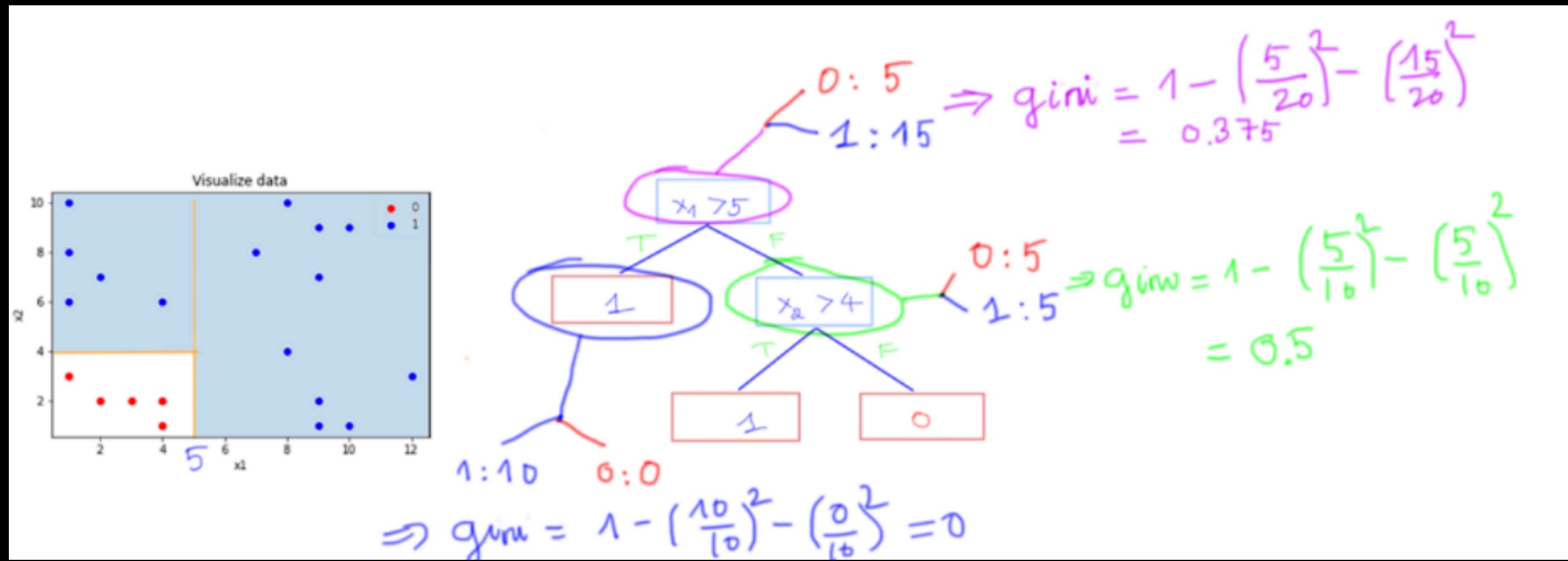
$$\text{Gini} = 1 - \sum_{i=1}^C p_i^2$$

Do $0 \leq p_i \leq 1 \forall i$ và $\sum_{i=1}^N p_i = 1$ nên:

$$\sum_{i=1}^C (p_i)^2 \leq (\sum_{i=1}^C p_i)^2 = 1 \Rightarrow Gini \geq 0, \text{ dấu bằng xảy ra khi } \exists j : p_j = 1 \text{ và } p_k = 0 \forall k \neq j$$

$$\sum_{i=1}^C (p_i)^2 \geq \frac{(\sum_{i=1}^C p_i)^2}{C} = \frac{1}{C} \Rightarrow Gini \leq \frac{C-1}{C}, \text{ dấu bằng xảy ra khi } p_j = \frac{1}{C} \forall j$$

GINI INDEX



gini index là độ lệch gini của node cha với tổng các giá trị gini có đánh trọng số của các node con.

=> Để tìm điều kiện tách, mình thử ở tất cả các thuộc tính, mỗi thuộc tính thử một số giá trị chia, rồi so sánh xem điều kiện nào chỉ số gini index giảm nhiều nhất thì sẽ chọn để chia.

THUẬT TOÁN CART DÙNG GINI??

- 
1. Tính tích cực: Gini impurity là một phép đo tích cực và nhanh chóng. Nó không yêu cầu tính logarit và có thể được tính toán dễ dàng, đặc biệt khi có nhiều lớp phân loại.
 2. Độ tinh khiết đồng nhất: Gini impurity đo lường mức độ hỗn độn hoặc độ tinh khiết của một tập hợp các lớp phân loại. Nếu Gini impurity bằng 0, có nghĩa là tất cả các mẫu trong tập hợp đều thuộc cùng một lớp. Điều này cho thấy rằng Gini impurity là một độ đo hiệu quả để đánh giá độ tinh khiết và đồng nhất của các nhãn lớp trong mỗi nhánh.
 3. Quyết định dựa trên độ tinh khiết: CART tìm kiếm các quy tắc phân chia dựa trên việc giảm Gini impurity. Bằng cách tối thiểu hóa Gini impurity, thuật toán CART tạo ra các nhánh có độ tinh khiết cao và tách biệt giữa các lớp phân loại.
 4. Sự tương đồng với Entropy: Gini impurity thường được ưu tiên sử dụng hơn trong thuật toán CART vì nó có tính chất tương đồng với Entropy và thời gian tính toán ít hơn



THUẬT TOÁN CART

Trong thuật toán CART, hàm mất mát được định nghĩa là tổng có trọng số của entropy trên toàn bộ các node lá. Trọng số ở đây được lấy theo tỷ lệ phần trăm quan sát trên từng node lá

==> những node lá có số lượng quan sát lớn thì ảnh hưởng của nó lên hàm mất mát là lớn hơn so với những node lá có số lượng quan sát nhỏ.

==> Để tối thiểu hoá hàm mất mát thì chúng ta phải lựa chọn biến và ngưỡng sao cho tổng giá trị của hàm mất mát là nhỏ nhất.





THUẬT TOÁN CART

Mức độ tinh khiết sau phân chia sẽ bằng tổng có trọng số của entropy tại mỗi node lá mới. Giá trị này được gọi là entropy sau phân chia:

$$\mathbf{H}(x^{(j)}, t; \mathcal{S}) = \frac{N_0}{N} \mathbf{H}(\mathcal{S}_0) + \frac{N_1}{N} \mathbf{H}(\mathcal{S}_1)$$

Một kịch bản phân chia được coi là mang lại kết quả tối hơn so với không phân chia nếu như kết quả sau phân chia giúp gia tăng độ tinh khiết.

$$\begin{aligned}\mathbf{G}(x^{(j)}, t; \mathcal{S}) &= \mathbf{H}(\mathcal{S}) - \mathbf{H}(x^{(j)}, t; \mathcal{S}) \\ &= \mathbf{H}(\mathcal{S}) - \frac{N_0}{N} \mathbf{H}(\mathcal{S}_0) - \frac{N_1}{N} \mathbf{H}(\mathcal{S}_1)\end{aligned}$$



THUẬT TOÁN CART

- Xây dựng cây quyết định được sử dụng cho cả bài toán phân loại và dự đoán giá trị số.
- Đặc điểm : nó tạo ra cây nhị phân (mỗi nút có tối đa hai nhánh con) và có khả năng xử lý cả dữ liệu rời rạc và dữ liệu liên tục.

==> Các bước :

1.Chọn thuộc tính tốt nhất để chia dữ liệu: sử dụng một phương pháp gọi là "giảm biến thiêng Gini" (Gini impurity) để đo lường sự không thuần khiết của dữ liệu. Thuật toán tìm cách chia dữ liệu thành các nhóm sao cho giảm biến thiêng Gini là lớn nhất.

2.Tạo nút và chia dữ liệu

3.Lặp lại quá trình cho các nhóm con

4.Gán nhãn cho nút lá

5.Dự đoán mới:

==> Ứng dụng : phân loại văn bản, phát hiện gian lận, dự đoán giá nhà, và nhiều bài toán khác.



THUẬT TOÁN CART

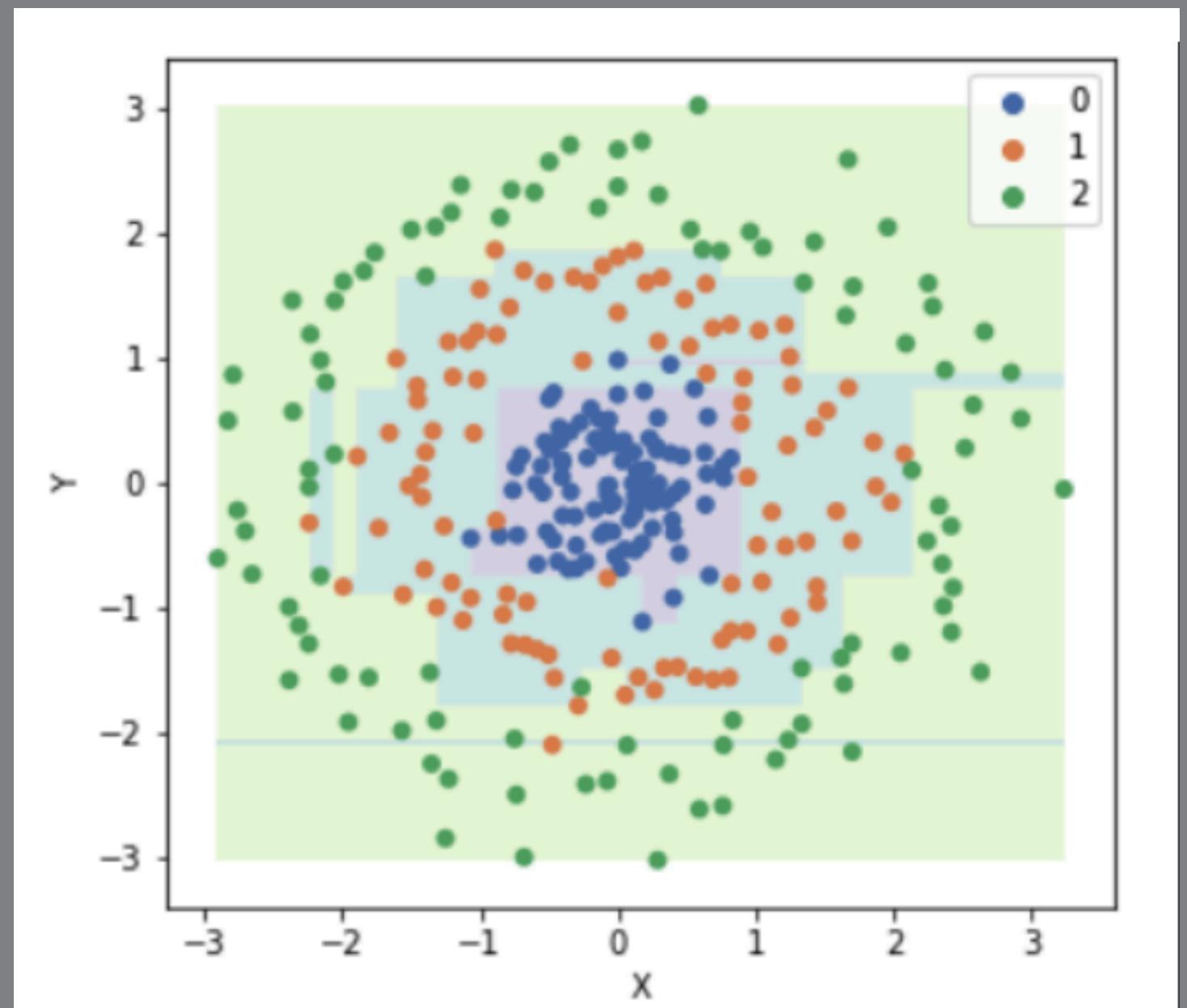
1. Dễ hiểu và dễ giải thích
 2. Xử lý dữ liệu hỗn hợp: CART có khả năng xử lý dữ liệu hỗn hợp, bao gồm các biến định lượng và biến phân loại. Nó tự động tìm kiếm các quy tắc phân chia phù hợp cho cả hai loại biến, giúp xử lý dữ liệu đa dạng mà không cần chuyển đổi biến đầu vào.
 3. Tính tương tác biến: nó có thể tìm ra các quy tắc phân chia phức tạp bằng cách kết hợp nhiều biến lại với nhau. Điều này giúp CART phù hợp với các vấn đề phức tạp và dữ liệu có tính tương tác cao.
 4. Tự động xử lý các giá trị bị khuyết: CART có khả năng tự động xử lý các giá trị bị khuyết trong dữ liệu. Nó có thể tự động tìm ra các quy tắc phân chia dựa trên các biến có giá trị bị khuyết và phân chia dữ liệu dựa trên các giá trị có sẵn.
 5. Khả năng xử lý dữ liệu lớn: CART có thể xử lý các tập dữ liệu lớn mà không yêu cầu quá nhiều tài nguyên tính toán. Quá trình xây dựng cây có thể được thực hiện song song hoặc trên các tập con dữ liệu, giúp tăng tốc độ xử lý.
- Tuy nhiên, CART cũng có một số hạn chế, bao gồm khả năng bị quá khớp dữ liệu huấn luyện khi cây quá phức tạp và khả năng không ổn định khi có những thay đổi nhỏ trong dữ liệu.
- 



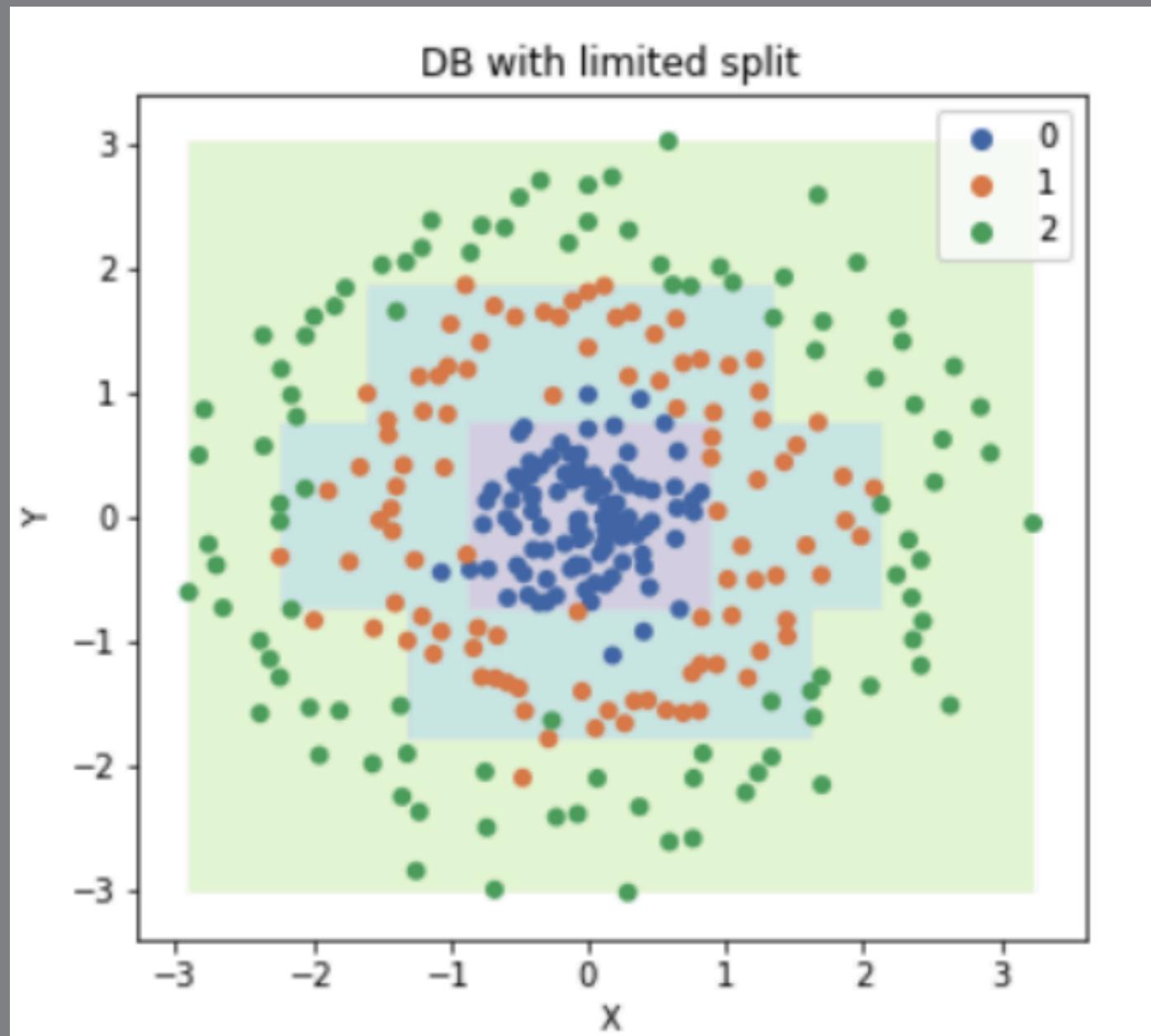
OVERFITTING

Các thuật toán Decision Tree nói chung nếu xây dựng cây quyết định đủ sâu thì sẽ tách được các node lá chỉ chứa dữ liệu một lớp nhất định, nên mô hình rất dễ bị overfitting.





Mô hình Decision Tree trên overfitting với dữ liệu, và tạo ra đường phân chia rất lạ.



khi mình giới hạn độ sâu của cây là 5 và số phần tử tối thiểu trong lớp lá là 5. Mọi người thấy mô hình đỡ bị overfitting hơn, và đường phân chia tổng quát dữ liệu hơn.

Dấu hiệu của overfitting

1. Độ sâu lớn: Cây quyết định có độ sâu lớn, có nhiều tầng và quy tắc phức tạp.
2. Số lượng lá nhiều: Cây có quá nhiều lá, mỗi lá chỉ chứa một số lượng nhỏ các mẫu huấn luyện.
3. Tỉ lệ phân loại chính xác cao trên dữ liệu huấn luyện: Cây có độ chính xác rất cao trên tập dữ liệu huấn luyện, nhưng độ chính xác trên tập dữ liệu kiểm tra hoặc dữ liệu mới là thấp.

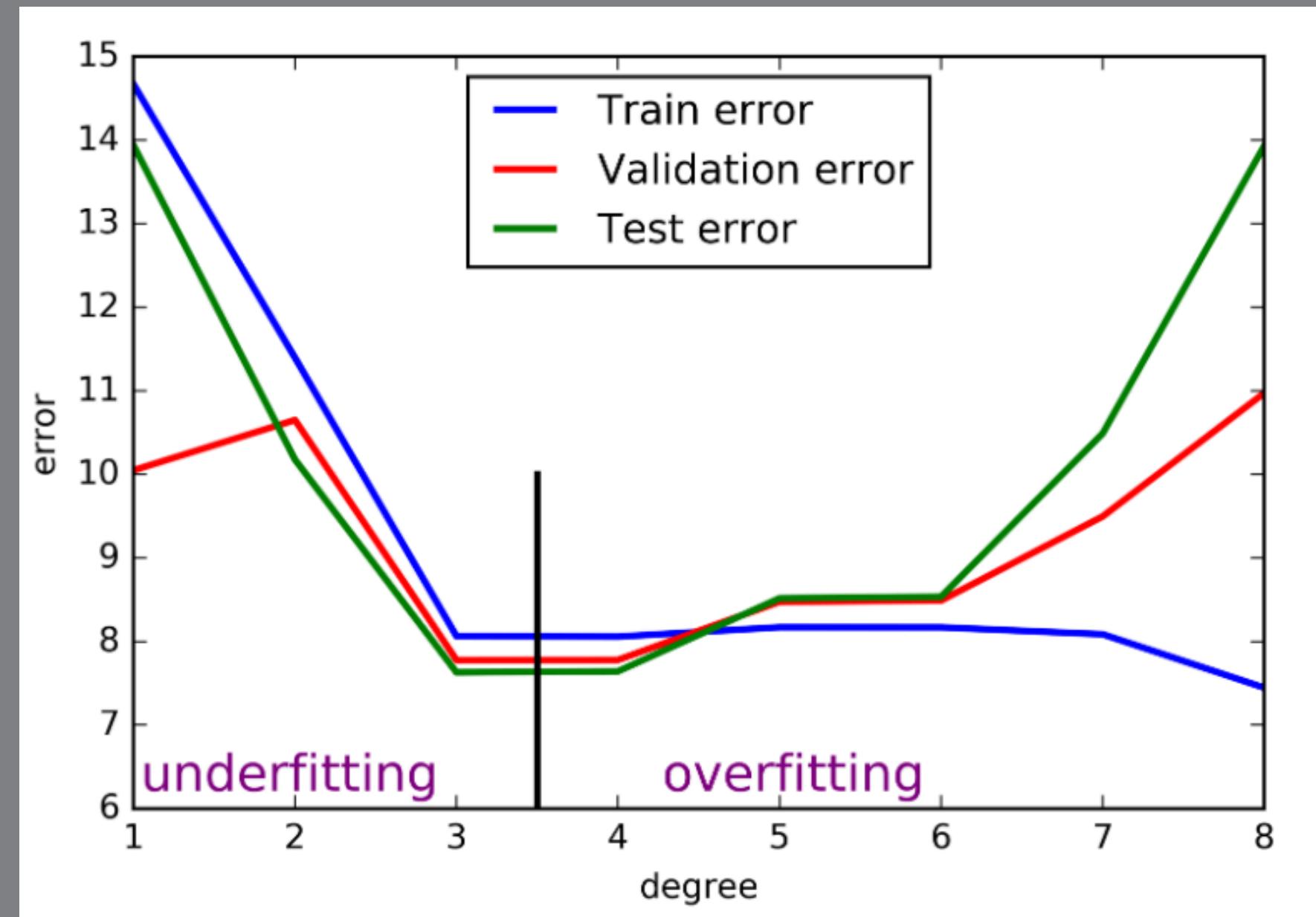
Nguyên nhân overfitting

1. Quá mức phức tạp mô hình: Khi cây quyết định có quá nhiều quy tắc và sử dụng nhiều đặc trưng, nó có khả năng nhớ những chi tiết không cần thiết trong dữ liệu huấn luyện.
2. Thiếu dữ liệu huấn luyện: Khi tập dữ liệu huấn luyện quá nhỏ, cây quyết định có thể dễ dàng nhớ các mẫu riêng lẻ và không thể tổng quát hóa tốt trên dữ liệu mới.
3. Nhiều và dữ liệu không chính xác: Nếu dữ liệu huấn luyện chứa nhiễu hoặc thông tin không chính xác, cây quyết định có thể học những quy tắc không chính xác và gây ra overfitting.

Phương pháp giảm overfitting

1. Tước tỉa (Pruning): Loại bỏ các nhánh không quan trọng trong cây để giảm độ phức tạp và tăng khả năng tổng quát hóa.
2. Điều chỉnh độ sâu cây: Giới hạn độ sâu của cây để tránh quá khớp.
3. Sử dụng các siêu tham số điều chỉnh: Thiết lập các siêu tham số như tỉ lệ tước tỉa, giới hạn độ sâu, số lượng mẫu tối thiểu trong lá để điều chỉnh độ phức tạp của cây.
4. Sử dụng kỹ thuật ensemble learning: Kết hợp nhiều cây quyết định để tạo ra một mô hình mạnh hơn và tránh overfitting.
5. Thu thập thêm dữ liệu huấn luyện để tăng cường sự đa dạng và số lượng dữ liệu huấn luyện.

VALIDATION(thẩm định)



ví dụ phía trên với bậc của đa thức tăng từ 1 đến 8. Tập validation bao gồm 10 điểm được lấy ra từ tập training ban đầu.

VALIDATION(thẩm định)

Trích từ tập training data ra một tập con nhỏ và thực hiện việc đánh giá mô hình trên tập con nhỏ này(validation set). Training set là phần còn lại của training set ban đầu. Train error được tính trên training set mới này, định nghĩa tương tự như trên validation error, tức error được tính trên tập validation.

Tìm mô hình sao cho cả train error và validation error đều nhỏ \Rightarrow dự đoán được rằng test error cũng nhỏ. Phương pháp thường được sử dụng là sử dụng nhiều mô hình khác nhau. Mô hình nào cho validation error nhỏ nhất sẽ là mô hình tốt.

Thông thường, ta bắt đầu từ mô hình đơn giản, sau đó tăng dần độ phức tạp của mô hình. Tới khi nào validation error có chiều hướng tăng lên thì chọn mô hình ngay trước đó. Chú ý rằng mô hình càng phức tạp, train error có xu hướng càng nhỏ đi.

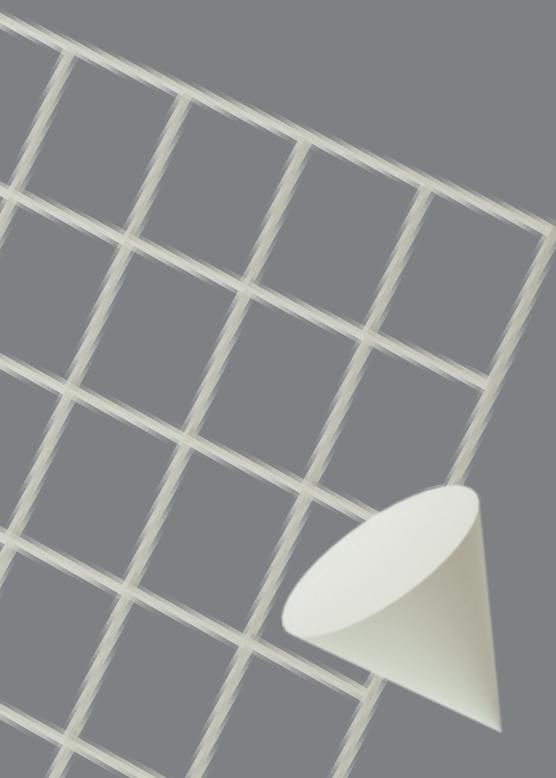


CROSS_VALIDATION

là một cải tiến của validation với lượng dữ liệu trong tập validation là nhỏ nhưng chất lượng mô hình được đánh giá trên nhiều tập validation khác nhau

=> chia tập training ra k tập con không có phần tử chung, có kích thước gần bằng nhau. Tại mỗi lần kiểm thử , được gọi là run, một trong số k tập con được lấy ra làm validate set. Mô hình sẽ được xây dựng dựa vào hợp của $k-1$ tập con còn lại. Mô hình cuối được xác định dựa trên trung bình của các train error và validation error. Cách làm này còn có tên gọi là k-fold cross validation.

Khi k bằng với số lượng phần tử trong tập training ban đầu, tức mỗi tập con có đúng 1 phần tử, ta gọi kỹ thuật này là leave-one-out.

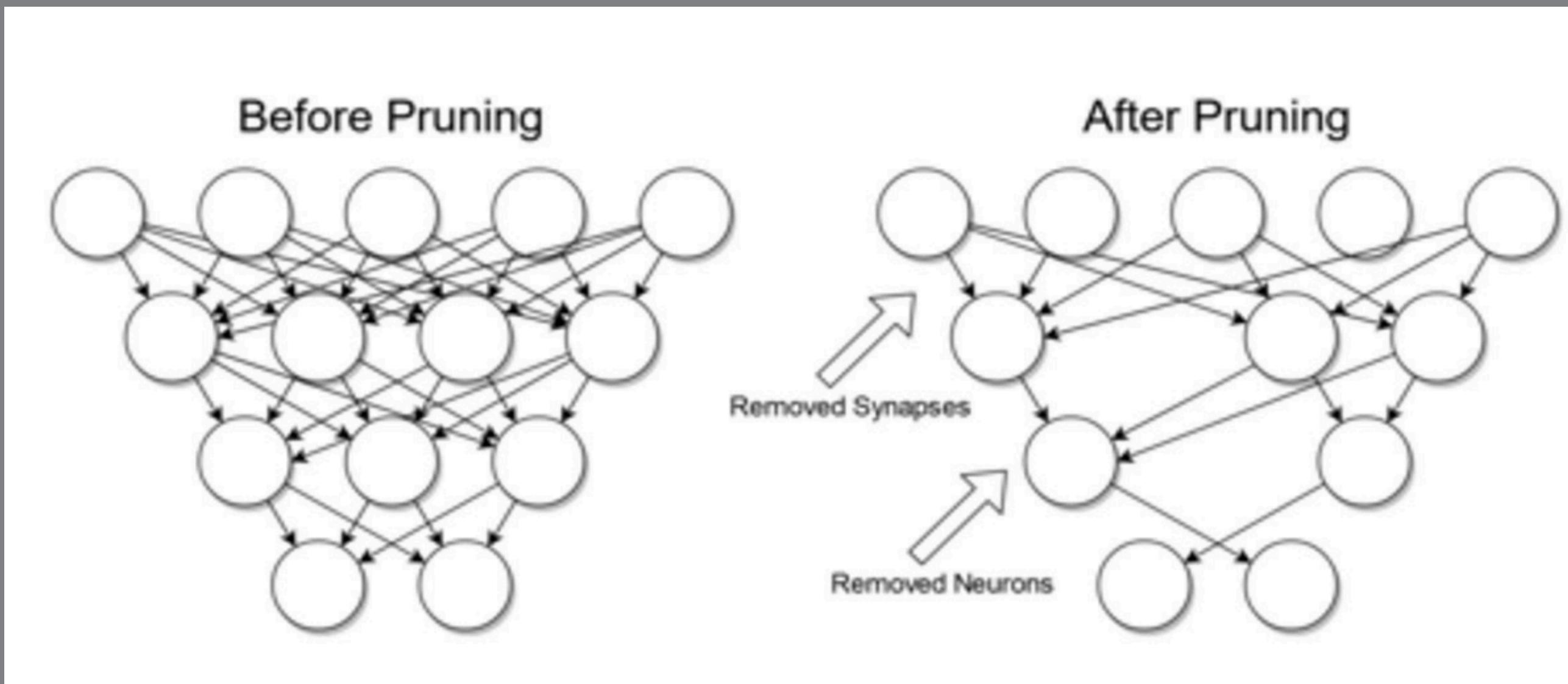




PRUNING



Pruning (cắt tỉa) là quá trình loại bỏ một số nhánh hoặc lá của cây quyết định để cải thiện hiệu suất và khái quát hóa của mô hình.



1. Pre-pruning :

- Dùng sớm : tiêu chuẩn dừng giới hạn độ sâu của cây(max depth) , số lượng mẫu tối thiểu để phân chia (min sample leaf), một ngưỡng về entropy/giniindex
 - Kiểm tra lỗi : xây dựng cây đầy đủ , loại bỏ nhánh so sánh lỗi trên data_train data_test
 - Ktra độ quan trọng thuộc tính : thuộc tính ít quan trọng -> loại bỏ khi chúng k cung cấp nhiều thông tin phân loại
- ==> đơn giản ít tốn thời gian nhưng cần chọn tiêu chuẩn dừng phù hợp

2. Post-pruning :

- Cost complexity pruning :xây dựng cây đầy đủ , sử dụng thuật toán tối ưu tìm cây con tối ưu , kiểm soát kích thước của cây
 - Reduced error pruning: Xây dựng cây đầy đủ và sau đó lược bỏ các nhánh một cách lặp lại dựa trên việc đo lường sự cải thiện của lỗi phân loại trên tập kiểm tra. Nếu việc loại bỏ nhánh không gây tăng lỗi trên tập kiểm tra, nhánh đó sẽ bị loại bỏ.
- > có khả năng tìm ra cây con với hiệu suất cao và ít bị ảnh hưởng bởi tiêu chuẩn dừng nhưng phức tạp hơn do phải xây dựng cây đầy đủ từ đầu

prepruning

tìm siêu tham số tốt nhất cho cây

b1: khởi tạo cây mặc định

siêu tham số :max_depth đại diện cho độ sâu tối đa của cây

min_samples_split là số lượng mẫu tối thiểu yêu cầu để phân chia một nút

min_samples_leaf là số lượng mẫu tối thiểu yêu cầu để tạo thành một lá (leaf) của cây.

b2: tạo danh sách siêu tham số để thử nghiệm.

b3: đánh giá mô hình với từng kết hợp đó bằng cách sử dụng phương thức cross-validation.

b4: xác định siêu tham số tốt nhất dựa trên độ đo hiệu suất được chỉ định (mặc định là điểm chính xác).

postpruning

Reduce error pruning :

là một kĩ thuật cắt tỉa sau khi xây dựng cây bằng tập train

b1: chia dữ liệu train& validation : train tạo cây, val để đánh giá hiệu suất các phiên bản cây khác nhau

b2: xây dựng cây

b3: cắt tỉa cây : Bắt đầu từ gốc của cây quyết định, lặp lại việc đánh giá hiệu suất của mỗi nút trong cây và cây con tương ứng trên tập xác thực. Loại bỏ cây con liên quan đến nút nếu việc cắt tỉa nó dẫn đến cải thiện hiệu suất trên tập xác thực. Hiệu suất cải thiện có thể được đo bằng các chỉ số như độ chính xác, tỷ lệ lỗi hoặc cross-entropy.

b4: **Lặp lại quá trình cắt tỉa:** Lặp lại quá trình cắt tỉa cho mỗi nút trong cây theo hướng từ lá đến gốc. Sau khi cắt tỉa một nút và cây con của nó, đánh giá hiệu suất của cây được cắt tỉa trên tập xác thực. Nếu việc cắt tỉa nút hiện tại không cải thiện hiệu suất, bỏ qua việc cắt tỉa nút đó và chuyển sang nút tiếp theo.

b5: **Chọn cây được cắt tỉa:** Sau khi cắt tỉa tất cả các nút, chọn cây đã được cắt tỉa có hiệu suất tốt nhất trên tập xác thực. Cây được cắt tỉa này dự kiến sẽ có khả năng tổng quát hóa cải thiện so với cây đã được phát triển đến kích thước tối đa.

cost complexity

- tác dụng : tìm ra một mô hình phù hợp và cân bằng giữa độ phức tạp và độ tốt của cây quyết định.
- điều chỉnh độ sâu của cây dựa vào thông số alpha
 - +Alpha càng lớn, cây quyết định càng đơn giản và ít phù hợp với dữ liệu huấn luyện
 - +alpha nhỏ hơn sẽ tạo ra cây quyết định phù hợp hơn với dữ liệu huấn luyện nhưng có nguy cơ overfitting cao.
- Quá trình cắt tỉa này được thực hiện bằng cách tăng giá trị alpha, từ đó giảm độ sâu của cây và làm giảm độ phù hợp với dữ liệu huấn luyện.
- Một kỹ thuật phổ biến để tìm giá trị alpha tối ưu là sử dụng "đường cong độ phức tạp chi phí" (cost complexity pruning path)
 - >Chúng ta có thể tìm ra giá trị alpha tối ưu bằng cách tìm điểm trên đường cong độ phức tạp chi phí có độ tốt cao nhất hoặc độ phức tạp thấp nhất
 - sau khi tìm được alpha phù hợp ta sẽ thực hiện prune với 1 giá trị alpha cụ thể
- ** các bước tìm alpha tối ưu bằng cross validation
 - b1: chia 2 tập datatrain datatest
 - b2: dùng tập train để xây dựng cây
 - b3: tính toán đường cong chi phí, lưu các giá trị alpha
 - b4: huấn luyện cây sử dụng K-Fold ,với mỗi giá trị alpha ta cắt tỉa cây -> đánh giá hiệu suất với tập test
 - b5: chọn alpha tốt nhất thông qua các hiệu suất b4
 - b6: cắt tỉa cây với giá trị alpha tối ưu

Cost complexity pruning use sklearn

```
1 X=iris.drop(columns="Species")
2 y=iris["Species"]
3
4 from sklearn.model_selection import train_test_split
5 X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.20,random_state=45)
```

```
1 from sklearn.tree import DecisionTreeClassifier
2
3 tree = DecisionTreeClassifier(random_state=40)
4 tree.fit(X_train,y_train)
5 y_train_pred=tree.predict(X_train)
6 y_test_pred=tree.predict(X_test)
```

```
1 from sklearn.metrics import accuracy_score
2
3 print(accuracy_score(y_train,y_train_pred),round(accuracy_score(y_test,y_test_pred),2))
```

0.95 0.63

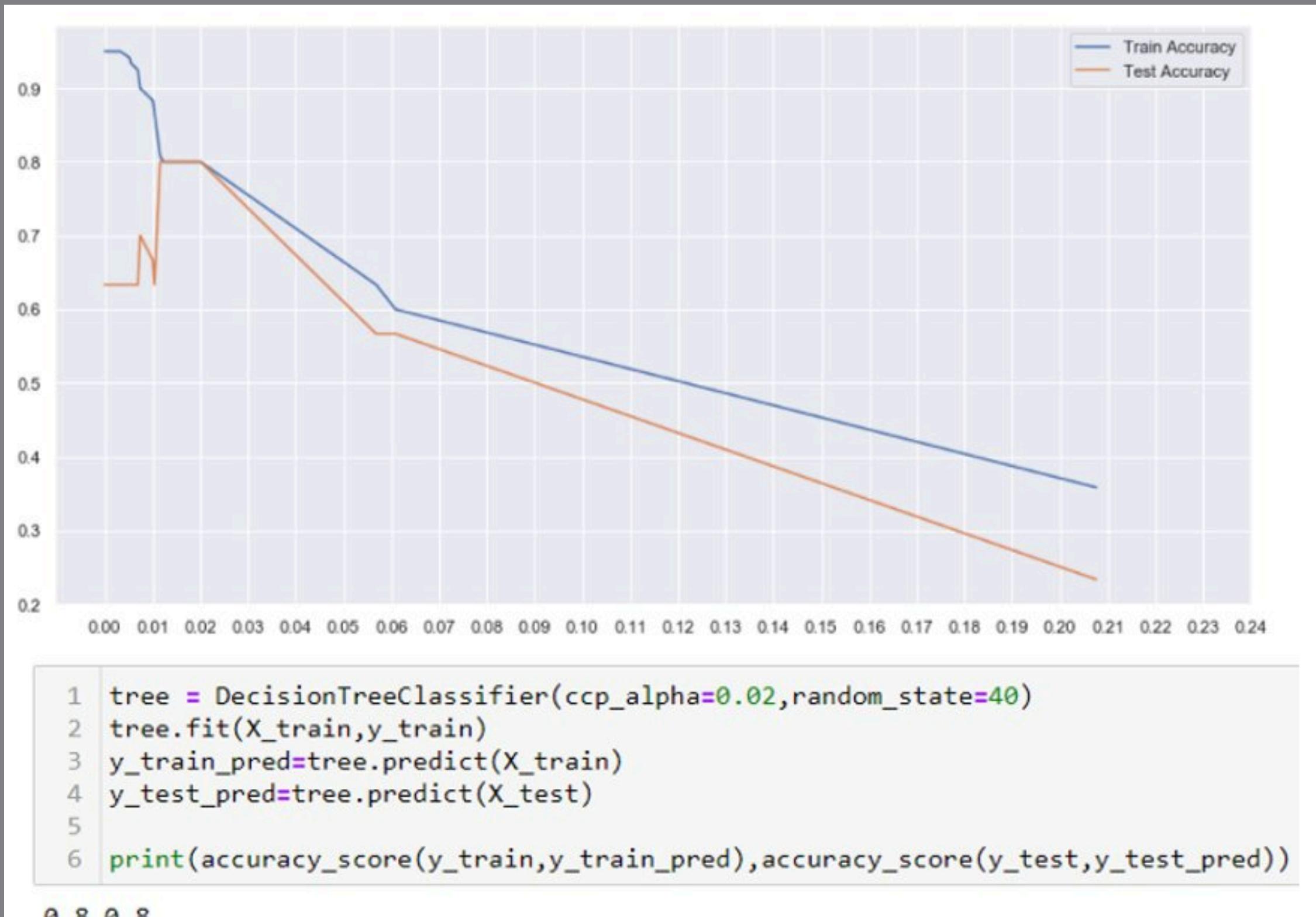
Cost complexity pruning use sklearn

```
1 path=tree.cost_complexity_pruning_path(X_train,y_train)
2 alphas=path['ccp_alphas']
3
4 alphas
5
6 array([0.          , 0.00277778, 0.00277778, 0.00277778, 0.00324074,
7       0.00518519, 0.00555556, 0.00694444, 0.00743464, 0.01006944,
8       0.01041667, 0.01161038, 0.01230159, 0.01581699, 0.02010944,
9       0.05683866, 0.06089286, 0.20756944])
```



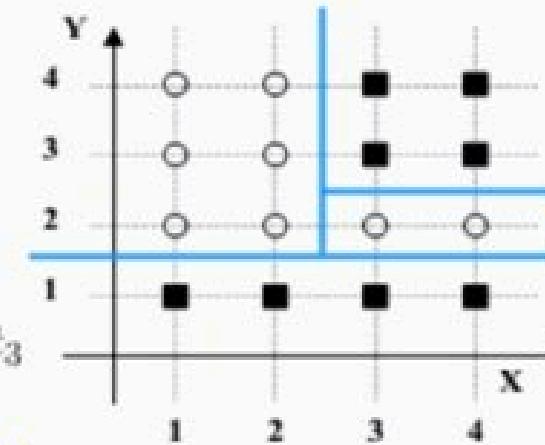
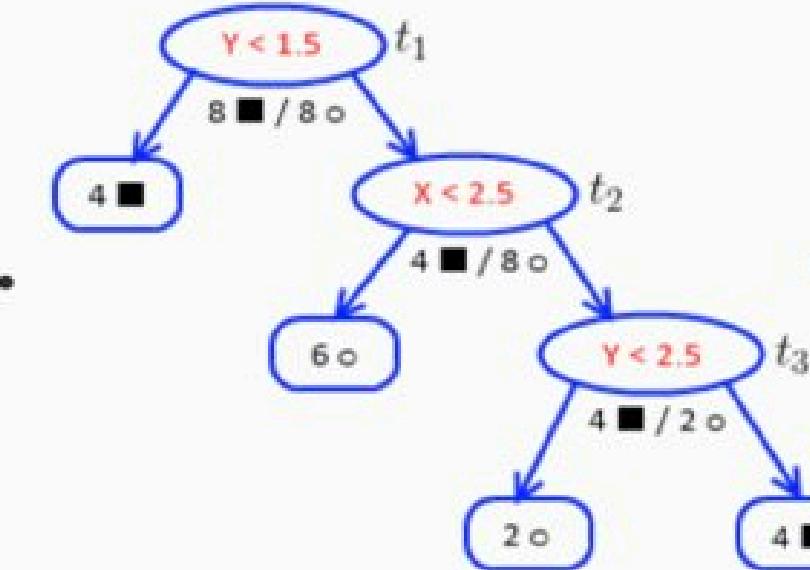
```
1 accuracy_train,accuracy_test=[],[]
2
3 for i in alphas:
4     tree=DecisionTreeClassifier(ccp_alpha=i)
5
6     tree.fit(X_train,y_train)
7     y_train_pred=tree.predict(X_train)
8     y_test_pred=tree.predict(X_test)
9
10    accuracy_train.append(accuracy_score(y_train,y_train_pred))
11    accuracy_test.append(accuracy_score(y_test,y_test_pred))
12
13
14 sns.set()
15 plt.figure(figsize=(14,7))
16 sns.lineplot(y=accuracy_train,x=alphas,label="Train Accuracy")
17 sns.lineplot(y=accuracy_test,x=alphas,label="Test Accuracy")
18 plt.xticks(ticks=np.arange(0.00,0.25,0.01))
19 plt.show()
```

Cost complexity pruning use sklearn



Cost complexity pruning use sklearn

Suppose we have the following tree:



- we want to prune it
- we have 3 inner nodes where we can prune: $t_1 \equiv \text{root}$, t_2 , t_3

Some formulas:

- $R(T_t)$ - training error of a subtree T_t - a tree with root at node t
 - $R(T_t) = \sum_{l \in f(T_t)} R(l)$ - sum of all training errors over all leaves
- $R(t)$ - training error of node t
 - $R(t) = r(t) \cdot p(t)$
 - $r(t)$ - misclassification error at this node (without considering the leaves)
 - $p(t)$ - proportion of data items reached t (i.e. # of items reached t divided by # of training items)
- $g(t) = \frac{R(t) - R(T_t)}{|f(T_t)| - 1}$
 - $|f(T_t)| - 1$ is the number of leaves to prune

Cost complexity pruning use sklearn

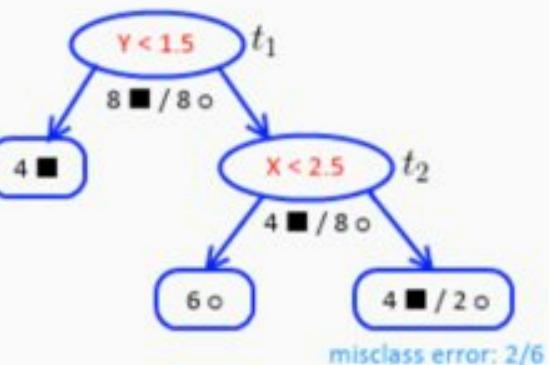
Iteration 1:

- let $\alpha^{(1)} = 0$

t	$R(t)$	$R(T_t)$	$g(t)$
t_1	$\frac{8}{16} \cdot \frac{16}{16}$	T_{t_1} - the entire tree all leaves are pure $R(T_{t_1}) = 0$	$\frac{8/16 - 0}{4 - 1} = \frac{1}{6}$
t_2	$\frac{4}{12} \cdot \frac{12}{16} = \frac{4}{16}$ (there are 12 records, 4 ■ + 8 ○)	$R(T_{t_2}) = 0$	$\frac{4/16 - 0}{3 - 1} = \frac{1}{8}$
t_3	$\frac{2}{6} \cdot \frac{6}{16} = \frac{2}{16}$	$R(T_{t_3}) = 0$	$\frac{2/16 - 0}{3 - 1} = \frac{1}{8}$

We want to find the minimal $g(t)$

- it's $g(t_2)$ and $g(t_3)$
- in case of a tie, we choose the one that prunes fewer nodes
- i.e. $g(t_3)$
- so prune at t_3
- let $\alpha^{(2)} = 1/8$ (the min $g(t)$)



Cost complexity pruning use sklearn

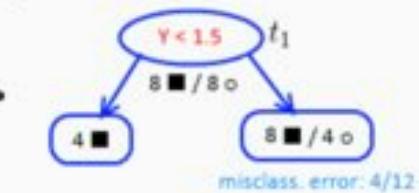
Iteration 2:

- in the tree now we have only candidates: t_1 and t_2

t	$R(t)$	$R(T_t)$	$g(t)$
t_1	$\frac{8}{16} \cdot \frac{16}{16}$	$\frac{2}{16}$	$\frac{8/16 - 2/16}{3-1} = \frac{6}{32}$
t_2	$\frac{4}{12} \cdot \frac{12}{16}$	$\frac{2}{16}$	$\frac{4/16 - 2/16}{2-1} = \frac{1}{8}$

Find minimal $g(t)$:

- it's $g(t_2) = 1/8$
- let $\alpha^{(3)} = 1/8$
- prune at t_2



Selecting the best:

- we have these values: $\alpha^{(0)} = 0, \alpha^{(1)} = 1/8, \alpha^{(2)} = 1/8, \alpha^{(3)} = 1/4$
- by the theorem we want to find tree such T that minimizes the cost-complexity function
 - if $0 \geq \alpha < 1/8$, then T_1 is the best
 - if $\alpha = 1/8$, then T_2 is the best
 - if $1/8 < \alpha < 1/4$, then T_3 is the best
 - if $1/8 < \alpha < 1/4$, then T_3 is the best
- to choose α use [Cross-Validation](#)

Iteration 3:

- only one candidate for pruning: t_1
- $\alpha^{(4)} = g(t_1) = \frac{8/16 - 4/16}{2-1} = \frac{1}{4}$



RẮC CẢM ƠN