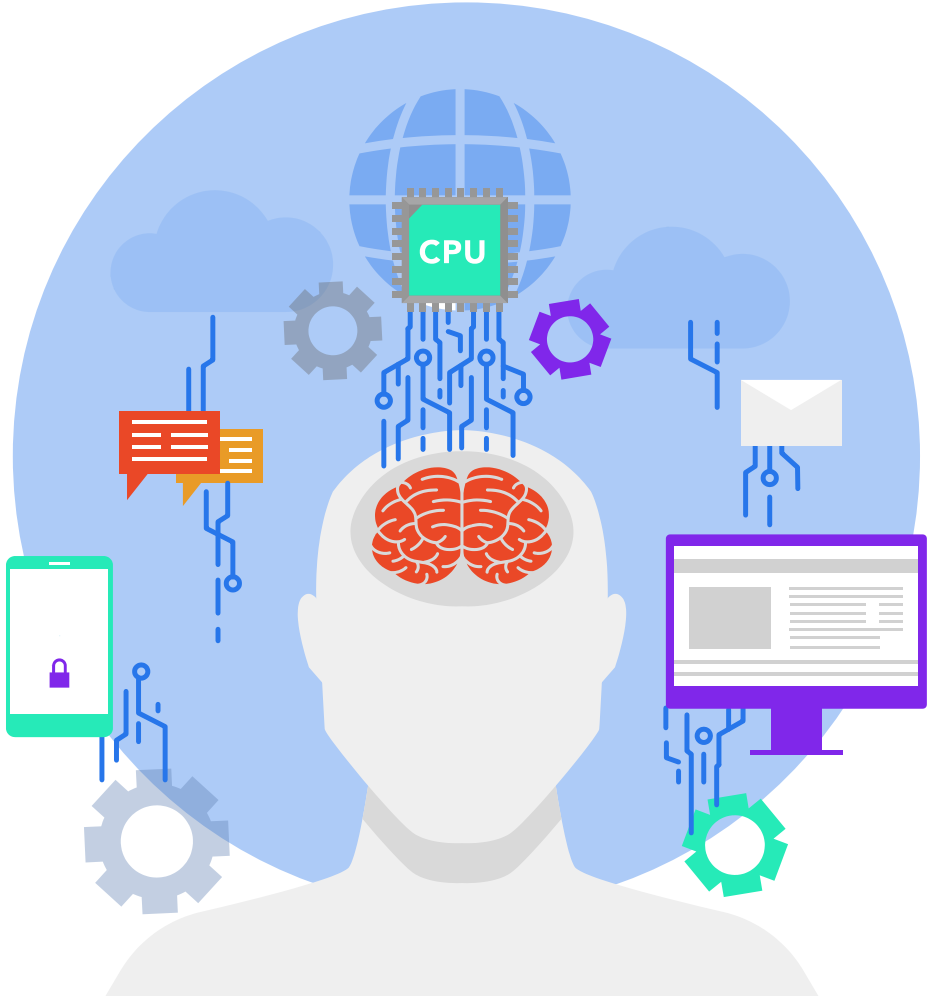
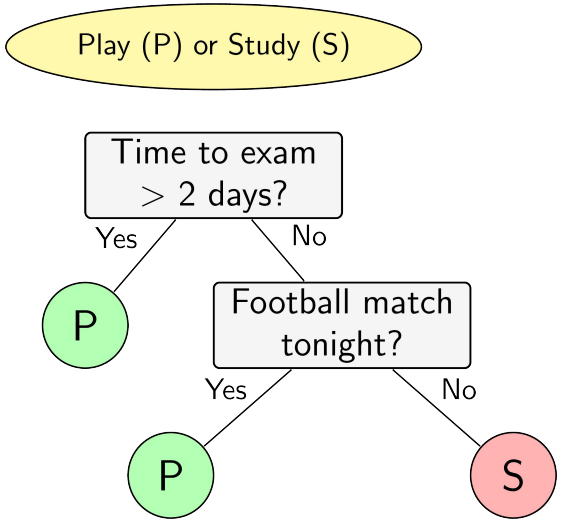


Chương 2: Phương pháp cây quyết định

2.1 Giới thiệu – Cây quyết định là gì?

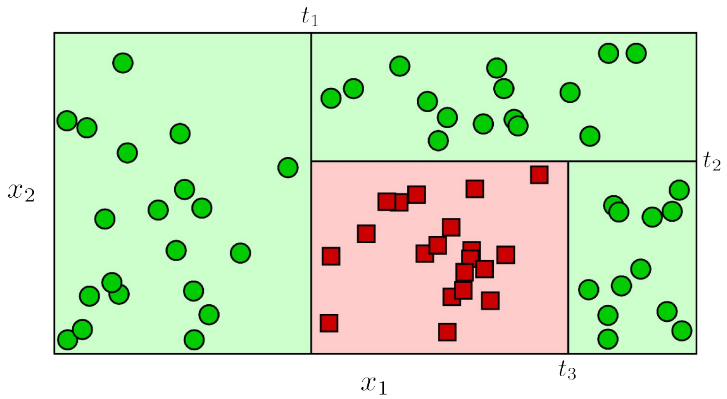
Sắp đến kỳ thi, một sinh viên tự đặt ra quy tắc *học* hay *chơi* của mình như sau: Nếu còn nhiều hơn hai ngày tới ngày thi, cậu ra sẽ đi chơi. Nếu còn không quá hai ngày và đêm hôm đó có một trận bóng đá, cậu sẽ sang nhà bạn chơi và cùng xem bóng đêm đó. Cậu sẽ chỉ học trong các trường hợp còn lại. Việc ra quyết định của cậu sinh viên này có thể được mô tả trên sơ đồ trong Hình 1. Hình ellipse nền vàng thể hiện quyết định cần được đưa ra. Quyết định này phụ thuộc vào các câu trả lời của các câu hỏi trong các ô hình chữ nhật màu xám. Dựa trên các câu trả lời, quyết định cuối cùng được cho trong các hình tròn màu lục (*chơi*) và đỏ (*học*). Sơ đồ trong Hình 1 còn được gọi là một *cây quyết định*.



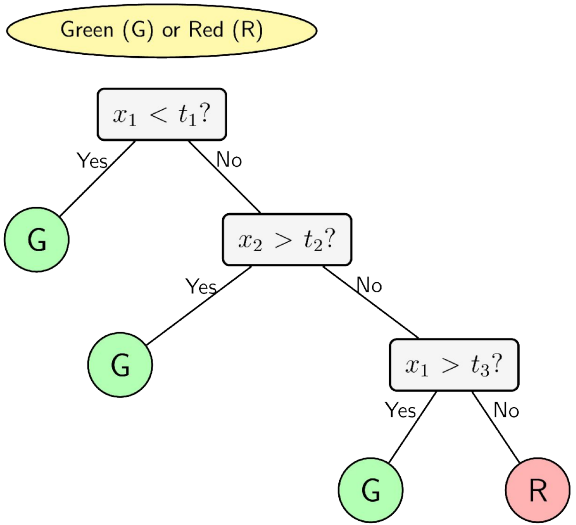
2.1 Giới thiệu – Cây quyết định là gì?

Việc quan sát, suy nghĩ và ra các quyết định của con người thường được bắt đầu từ các câu hỏi. Machine learning cũng có một mô hình ra quyết định dựa trên các câu hỏi. Mô hình này có tên là *cây quyết định* (*decision tree*).

Trong bài toán phân loại, cây quyết định giúp tìm ranh giới đơn giản để phân chia các lớp dữ liệu, xác định một điểm dữ liệu mới thuộc về lớp nào



(a)



(b)

2.2 Cấu trúc cây quyết định?

Các thành phần của cây quyết định:

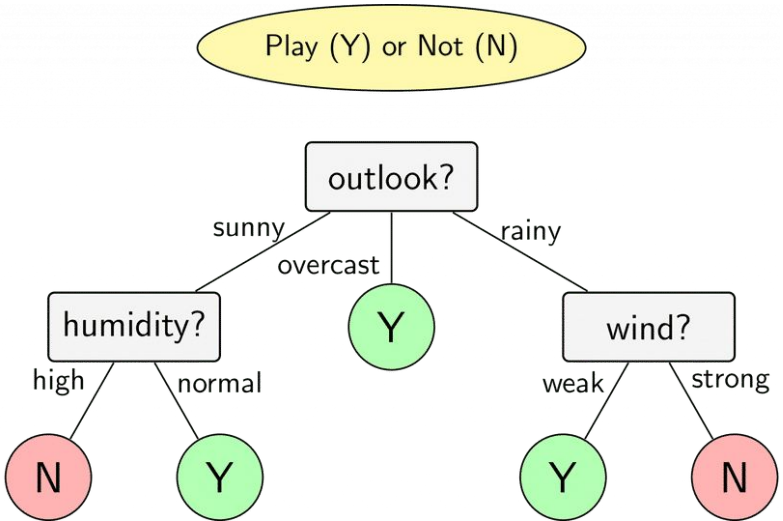
- Các hình chữ nhật màu xám, hình tròn màu lục, và đỏ trong sơ đồ được gọi là **node**
- **Node lá** (leaf node hoặc terminal node) là các node thể hiện đầu ra (quyết định cuối cùng, ví dụ: màu lục hoặc đỏ)
- **Node không lá** (non-leaf node) là các node thể hiện các câu hỏi
- **Node gốc** (root node) là non-leaf node nằm trên cùng, đại diện cho câu hỏi đầu tiên
- Các non-leaf node thường có hai hoặc nhiều **node con** (child node), Các node con có thể là node lá hoặc một non-leaf node khác
- Các node con có cùng bố mẹ được gọi là **sibling node**
- Nếu tất cả các non-leaf node chỉ có hai node con, đó được gọi là **cây quyết định nhị phân** (binary decision tree), trong đó các câu hỏi có thể được đưa về dạng đúng/sai

2.2 Cấu trúc cây quyết định?

Quy trình xây dựng cây quyết định:

1. Chọn thuộc tính tốt nhất để chia dữ liệu (dựa trên tiêu chí như Gini, Entropy)
2. Phân chia tập dữ liệu thành các tập con
3. Lặp lại cho đến khi đạt điều kiện dừng (độ sâu, số mẫu tối thiểu)

Ví dụ: Quyết định chơi tennis dựa trên thời tiết, nhiệt độ, độ ẩm



2.2 Cấu trúc cây quyết định?

Đặc điểm và ứng dụng của cây quyết định:

- Cây quyết định là một **mô hình học có giám sát** (supervised learning)
- Nó có thể được áp dụng cho cả hai bài toán **phân loại** (classification) và **hồi quy** (regression)
- Việc xây dựng một cây quyết định trên dữ liệu huấn luyện đòi hỏi việc xác định các **câu hỏi** và **thứ tự của chúng**
- Cây quyết định có khả năng làm việc với các đặc trưng (thường gọi là **thuộc tính** hoặc attribute) ở dạng **phân loại** (categorical), ví dụ như "mưa, nắng" hay "xanh, đỏ", thường là rời rạc và không có thứ tự
- Nó cũng hoạt động được với dữ liệu có vector đặc trưng bao gồm cả thuộc tính dạng phân loại và **liên tục** (numeric)
- Một điểm đáng chú ý nữa là cây quyết định **ít yêu cầu chuẩn hóa dữ liệu**

2.2 Cấu trúc cây quyết định?

Đặc điểm và ứng dụng của cây quyết định:

- Các câu hỏi trong cây quyết định thường được áp dụng lên từng thuộc tính hoặc một tổ hợp tuyến tính của các thuộc tính. Cách áp dụng lên từng thuộc tính phổ biến hơn vì tính đơn giản
- Với thuộc tính phân loại, câu hỏi có thể là “Nó rơi vào category nào?” hoặc “Nó có rơi vào category nào đó không?”
- Với thuộc tính liên tục, câu hỏi có thể là “Nó nằm vào khoảng giá trị nào?” hoặc “Nó có lớn hơn một ngưỡng nào đó không?”

Câu hỏi:

1. Cây quyết định thuộc loại học máy náo?
2. Nêu một ví dụ thực tế sử dụng cây quyết định

2.2 Cấu trúc cây quyết định – Hàm Entropy

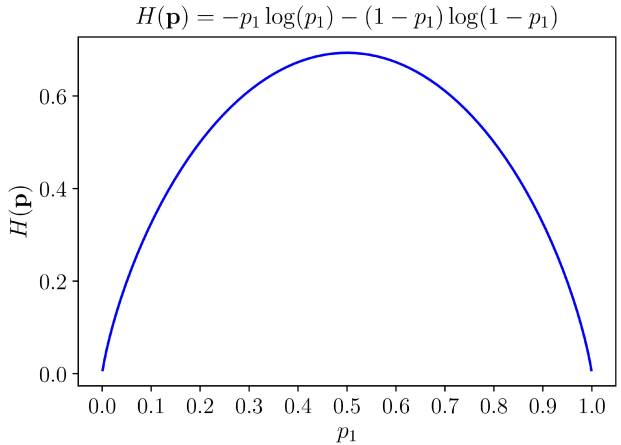
Cho một phân phối xác suất của một biến rời rạc x có thể nhận n giá trị khác nhau x1,x2,…,xn. Giả sử rằng xác suất để x nhận các giá trị này là pi=p(x=xi) với 0≤pi≤1,∑ni=1pi=1. Ký hiệu phân phối này là p=(p1,p2,…,pn). Entropy của phân phối này được định nghĩa là

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

trong đó log là logarit tự nhiên (*Một số tài liệu dùng logarit cơ số 2, nhưng giá trị của* H(p) *chỉ khác đi bằng cách nhân với một hằng số.*) và quy ước 0log(0)=0.

2.2 Cấu trúc cây quyết định – Hàm Entropy

Xét một ví dụ với n=2 được cho trên Hình 3. Trong trường hợp p là *tinh khiết* nhất, tức một trong hai giá trị pi bằng 1, giá trị kia bằng 0, entropy của phân phối này là H(p)=0. Khi p là *vẩn đục* nhất, tức cả hai giá trị pi=0.5, hàm entropy đạt giá trị cao nhất.



Tổng quát lên với n>2, hàm entropy đạt giá trị nhỏ nhất nếu có một giá trị pi=1, đạt giá trị lớn nhất nếu tất cả các pi bằng nhau ((việc này có thể được chứng minh bằng phương pháp nhân tử Lagrange).

2.2 Cấu trúc cây quyết định – Hàm Entropy

Bài tập: Xây dựng cây quyết định thủ công

Dữ liệu:

- **Thuộc tính:** Thời tiết (Nắng/Mưa), Nhiệt độ (Nóng/Mát), Độ ẩm (Cao/Thấp)
- **Nhãn:** Chơi tennis (Có/Không)
- **Dữ liệu mẫu:** [Nắng, Nóng, Cao, Không], [Mưa, Mát, Thấp, Có]

Yêu cầu: Vẽ cây quyết định trên giấy với tiêu chí Entropy

2.3 Một số thuật toán cơ bản học cây quyết định

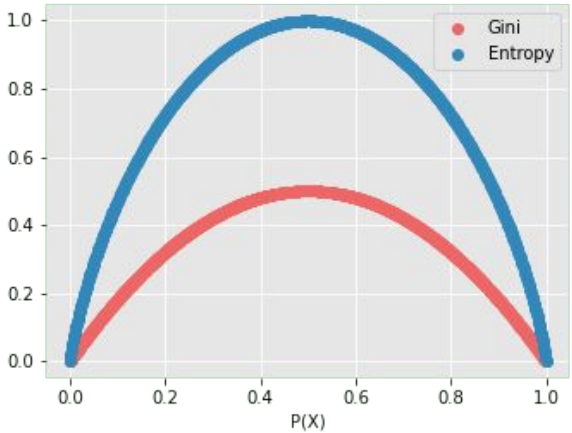
Mục tiêu: Tự động hoá việc xây dựng cây từ dữ liệu

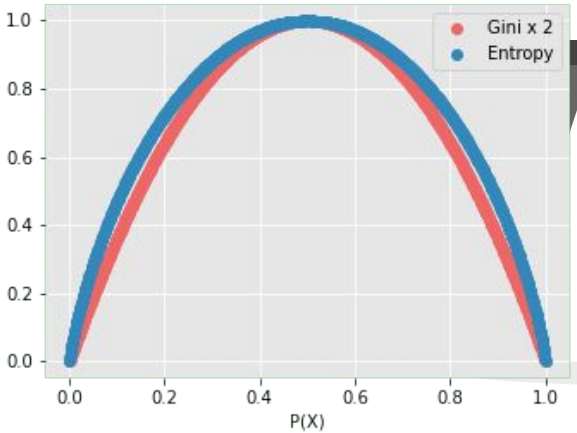
Tiêu chí chia:

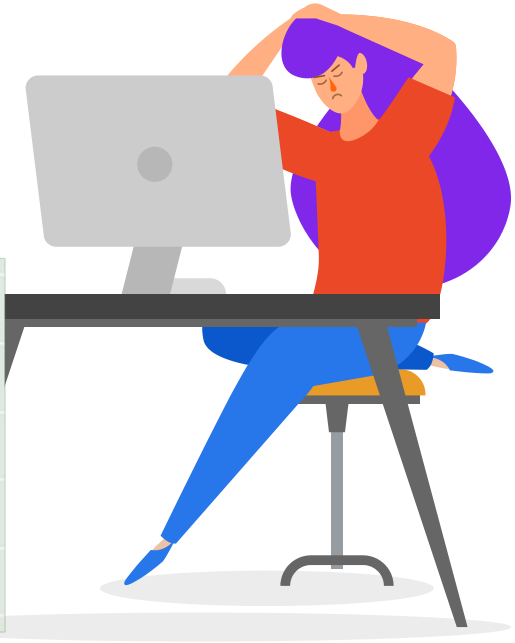
Entropy: Đo độ bất định của dữ liệu

Gini Index: Đo độ tinh khiết của tập con

Thuật toán phổ biến: C4.5, CART, ID3







2.3.1 Cây C4.5 – Cách hoạt động

01

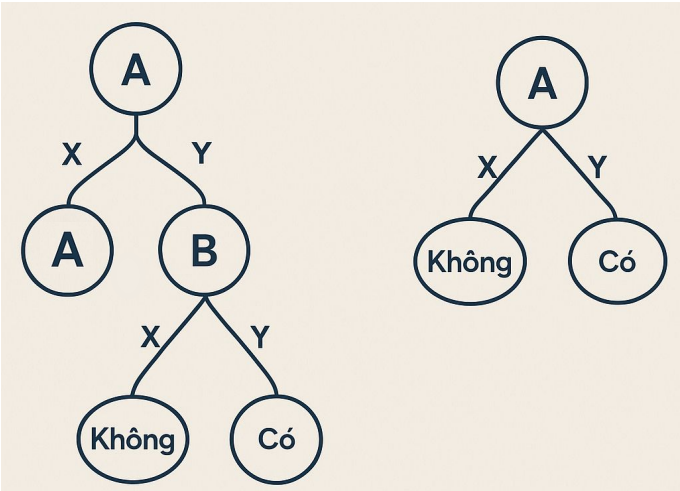
Gain Ratio:

- Công thức: Gain Ratio = Information Gain / Split Information
- Giảm ưu tiên các thuộc tính có nhiều giá trị

Cắt tỉa:

- Loại bỏ nhánh không cải thiện độ chính xác trên tập kiểm tra

Trước khi cắt tỉa



Sau khi cắt tỉa

2.3.1 Cây C4.5 – Cách hoạt động

Bài tập: Phân loại hoa Iris (Setosa, Versicolor, Virginica)

Dữ liệu:

- **Thuộc tính:** Chiều dài / dài cánh hoa, chiều rộng / rộng cánh đài

Quy trình:

1. Tính Entropy cho tập dữ liệu
2. Chọn thuộc tính có Gain Ratio cao nhất
3. Chia dữ liệu và lặp lại

2.3.1 Cây C4.5 – Cách hoạt động

Bài tập: Tính Entropy và Gain Ratio

Dữ liệu:

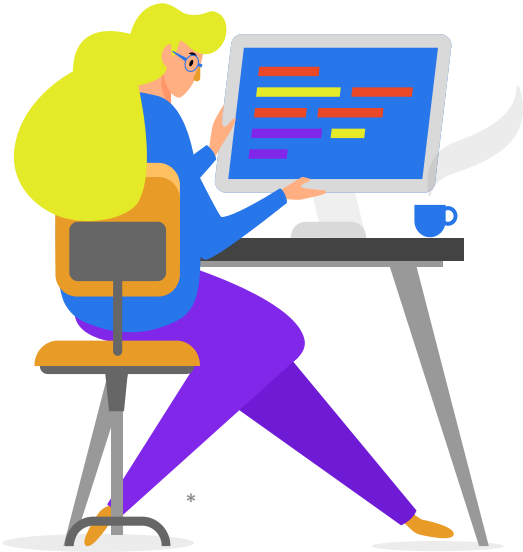
- **10 mẫu:** 6 mẫu lớp A, 4 mẫu lớp B
- **Thuộc tính X chia thành: Nhóm 1 (3A, 1B), Nhóm 2 (3A, 3B)**

Yêu cầu:

1. Tính Entropy cho tập dữ liệu
2. Tính Gain Ratio cho thuộc tính X

2.3.1 Cây C4.5 – Giới thiệu

- Là sự phát triển từ CLS và ID3.
- ID3 (Quinlan, 1979)- 1 hệ thống đơn giản ban đầu chứa khoảng 600 dòng lệnh Pascal
- Năm 1993, J. Ross Quinlan phát triển thành C4.5 với 9000 dòng lệnh C.
- Hiện tại: phiên bản See5/C5.0.
- Tư tưởng thuật toán: Hunt, chiến lược phát triển theo độ sâu.



2.3.1 Cây C4.5 – Mã giả

Pseudocode:

- Kiểm tra case cơ bản
- Với mỗi thuộc tính A tìm thông tin nhờ việc tách thuộc tính A
- Chọn a_best là thuộc tính mà độ đo lựa chọn thuộc tính “tốt nhất”
- Dùng a_best làm thuộc tính cho **node** chia cắt cây.
- Đệ quy trên các danh sách phụ được tạo ra bởi việc phân chia theo *a_best*, và thêm các node này như là con của **node**

```
(1) ComputerClassFrequency(T);
(2) if OneClass or FewCases
return a leaf;
Create a decision node N;
(3) ForEach Attribute A
ComputeGain(A);
(4) N.test=AttributeWithBestGain;
(5) if (N.test is continuous)
find Threshold;
(6) ForEach T' in the splitting of T
(7) If ( T' is Empty )
Child of N is a leaf
else
(8) Child of N=FormTree(T');
(9) ComputeErrors of N;
return N
```

2.3.1 Cây C4.5 – Công thức

$$I(S) = -\sum_{j=1}^{x} RF(C_j, S)\log(RF(C_j, S)).$$

Độ đo lựa chọn thuộc tính "tốt nhất":

- *Information gain: Test B chia* S={S1,S2,…,St)

$$G(S, B) = I(S) - \sum_{i=1}^{t} \frac{|S_i|}{|S|} I(S_i).$$

- Test B sẽ được chọn nêu có **G(S, B) đạt giá trị lớn nhất**.

- Thông tin tiềm năng (*potential information*) của bản thân mỗi phân hoạch:

$$P(S, B) = -\sum_{i=1}^{t} \frac{|S_i|}{|S|} \log\left(\frac{|S_i|}{|S|}\right).$$

- *Gain ratio* = G(S, B) / P(S, B) lớn nhất => chọn test B

2.3.1 Cây C4.5 – Ví dụ

Ví dụ:

- s1 (yes) 9 case,s2 (no) 5 case
- I(S) = I(s1,s2) = I(9, 5) = 0.940
- A = age =>S={S1,S2,S3}
- S1 (age<30), S2(30-40), S3 (>30).
- I (S1): s11(yes &<30) =2, s22(no&<30) =3
- I (S1) = (s11,s21) =0,971
- **Gain (S, age)** =

=I(s1,s2) – Σ|Si| / |S|* I(Si) = 0.246

- A = income: Gain (S, A) = 0.029
- A = student: Gain (S, A) = 0.151
- A = credit_rating: Gain (S, A) = 0.048
- A= age có Gain lớn nhất

=>chọn làm thuộc tính phát triển tại node đang xét

rid	age	income	student	credit_rating	Class: buys_computer
1	<30	high	no	fair	no
2	<30	high	no	excellent	no
3	30-40	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	fair	yes
6	>40	low	yes	excellent	no
7	30-40	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	30-40	medium	no	excellent	yes
13	30-40	high	yes	fair	yes
14	>40	medium	no	excellent	no

2.3.1 Cây C4.5

Xử lý "quá vừa" dữ liệu:

- Cho phép cây quá vừa dữ liệu, sau đó cắt tỉa cây.

Xử lý những giá trị thiếu*:*

- Là case có thuộc tính không có giá trị.
- information gain và potential information: S0 = case thuộc S có giá trị thuộc tính = null.

$$G(S,B) = \frac{|S - S_0|}{|S|} G(S - S_0, B).$$

$$P(S,B) = -\frac{|S_0|}{|S|} \log\left(\frac{|S_0|}{|S|}\right) - \sum_{i=1}^{t} \frac{|S_i|}{|S|} \log\left(\frac{|S_i|}{|S|}\right).$$

- Nếu test B được chọn, C4.5 phân chia các case trong S0 về các tập con Si

*

2.3.1 Cây C4.5

Chuyển đổi cây quyết định sang tập luật (pruned rules)

•Dạng luật:

If A and B and C then class X

→ Nếu không thỏa điều kiện → phân lớp mặc định.

- Quy trình xây dựng luật (4 bước):

  1.Tạo luật mẫu: Mỗi đường từ gốc đến lá tạo thành một luật.

  2.Đơn giản hóa luật: Bỏ bớt điều kiện không ảnh hưởng độ chính xác.

  3.Cắt tỉa & nhóm luật: Luật cùng phân lớp được nhóm lại, chọn luật tối ưu theo độ chính xác.

  4.Tạo lớp mặc định: Với các trường hợp không khớp luật nào → chọn lớp phổ biến nhất.

- Sắp xếp luật theo tần suất lỗi, đánh giá độ chính xác trên toàn bộ tập dữ liệu.
- Loại bỏ các luật làm giảm độ chính xác.

→ Kết quả: Một tập luật đơn giản, hiệu quả, đại diện cho từng lớp.

2.3.1 Cây C4.5 – Đặc điểm

Đặc điểm C4.5:

- Chiếm thời gian sử dụng CPU và bộ nhớ lớn:
  - **Ví dụ:** với 10k tới 100k case, tạo cây quyết định tăng từ 1,4s lên 61s, tạo luật tăng từ 32s lên 9,715s.
- Sử dụng cơ chế lưu dữ liệu thường trú trong bộ nhớ => ứng dụng với database nhỏ (tần số lỗi lặp lại 4% với database 20000 cases).
- Có cơ chế xử lý thiếu, lỗi hoặc quá vừa dữ liệu.
- Luật tạo ra đơn giản.

2.3.1 Cây C4.5 - Ứng dụng

Ứng dụng vào bài toán phân lớp dữ liệu:

- Bước 1 (Học): xây dựng mô hình mô tả tập dữ liệu; khái niệm đã biết
  - Input: tập dữ liệu có cấu trúc được tạo mô tả bằng các thuộc tính
  - Output: Các luật If…Then
- Bước 2 (Phân loại): dựa trên mô hình đã xây dựng để phân lớp dữ liệu mới: đi từ gốc đến các nút lá nhắm rút ra lớp của đối tượng cần xét.

2.3.1 Cây C4.5 - Ứng dụng

Ứng dụng vào bài toán phân lớp dữ liệu:

- Xử lý với dữ liệu thuộc tính liên tục:

- Sử dụng kiểm tra dạng nhị phân: value(V) < h với h là hằng số ngưỡng (threshold)
- h được tìm bằng cách:
  - Quick sort sắp xếp các case trong S theo các giá trị của thuộc tính liên tục V đang xét =>V = {v1, v2, …, vm}
  - hi = (vi + v(i+1))/2. Test phân chia dữ liệu:V <= hi hay V>hi => chia V thành V1={v1,v2,…, vi} và V2 = {vi+1, vi+2, …, vm} và có hi (i=1…m-1)
  - Tính *Information gain* hay G*ain ratio* với từng hi. Ngưỡng có giá trị của *Information gain* hay *Gain ratio* lớn nhất sẽ được chọn làm ngưỡng phân chia của thuộc tính đó.

2.3.2 Cây CART – Giới thiệu

CART (Classification and Regression Tree)

- Thuật toán CART (cây phân loại và hồi quy - Classification and Regression Tree), là một thuật toán được Leo Breiman giới thiệu năm 1984, dùng để xây dựng cây quyết định đa năng, với mục đích phân loại và hồi quy dữ liệu.
- Thuật toán CART là nền tảng cho những thuật toán khác mạnh mẽ hơn như bagged decision trees (tập hợp nhiều cây quyết định để cùng đoán nhận), random forest (rừng ngẫu nhiên) và boosted decision trees (tạp hợp các cây khác nhau, là các model; và một siêu model - meta model - để tập hợp kết quả đoán nhận của các model con)

2.3.2 Cây CART – Biểu diễn mô hình

- Biểu diễn của mô hình CART chính là một cây nhị phân.
- Cây nhị phân này cũng chính là cây nhị phân thường thấy trong môn cấu trúc dữ liệu và giải thuật, với mỗi nút là một biến độc lập riêng lẻ x (trong học máy, ta gọi đây là các đặc trưng), và điểm tách (hay là ngưỡng phân loại) của đặc trưng đó.
- Nút lá của cây chứa biến đầu ra y, dùng để thực hiện đoán nhận.

2.3.2 Cây CART – Huấn luyện mô hình

Huấn luyện mô hình CART từ dữ liệu

Huấn luyện mô hình CART yêu cầu ta phải lựa chọn các biến đầu vào và các ngưỡng phân loại tương ứng cho đến khi ta xây dựng được cây phù hợp. Thường quá trình xây cây sẽ dừng lại khi ta đủ điều kiện thoả mãn tiêu chí dừng. Ví dụ của tiêu chí dừng là "Số lượng bản ghi tối thiểu tương ứng với nút lá của cây".

2.3.2 Cây CART – Phân chia tham lam

- Quá trình ta xây cây quyết định nhị phân là quá trình ta phân chia không gian dữ liệu vào (input space). Một cách tiếp cận vét cạn để chia không gian đó chính là chia đôi đệ quy (recursive binary splitting).
- Chia đôi đệ quy là thuật toán heuristics, từng đặc trưng một được lựa chọn, từng giá trị ngưỡng được thử với đặc trưng. Các lựa chọn này được đánh giá thông qua hàm mất mát, và thông số tối ưu hàm mất mát được lựa chọn.
- Với bài toán phân loại, ta sử dụng độ thuần khiết Gini. Nhắc lại, độ thuần khiết Gini được tính bằng cách lấy tổng xác suất pi của bản ghi i được chọn nhân với xác suất $\sum k \neq ipk=1-pi$, tức xác suất khi phân loại sai bản ghi i. Độ thuần khiết đạt giá trị nhỏ nhất, tức bằng 0 khi tất cả các trường hợp tại điểm nút cuối đều rơi vào cùng một giá trị.

2.3.2 Cây CART – Phân chia tham lam

- Độ thuần khiết Gini cũng là độ đo lý thuyết thông tin và tương ứng với entropy dạng Tsallis với hệ số biến dạng q=2, trong Vật lý, công thức entropy này đại diện cho tính mất thông tin trong các hệ thống lượng tử mất cân bằng, tắt dần.
- Khi có q→1, công thức trở về dạng Boltzmann - Gibbs hoặc công thức entropy của Shannon. Với các điều kiện này, độ thuần khiết Gini có tác dụng như một biến thể của phép đo entropy cho các cây quyết định.
- Để tính độ thuần khiết Gini cho một tập các điểm dữ liệu có J lớp, với i∈{1,2,...,J}. và để pi là lượng bản ghi được đánh dấu với lớp i trong tập hợp, thì:

$$I_g(p) = \sum_{i=1}^{J}(p_i \sum_{k \neq i} p_k) = \sum_{i=1}^{J}(p_i - p_i^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} p_i^2 = 1 - \sum_{i=1}^{J} p_i^2$$

Hàm mất mát của CART được mô tả như sau:

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

2.3.2 Cây CART – Phân chia tham lam

Hàm mất mát của CART được mô tả như sau:

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

trong đó:

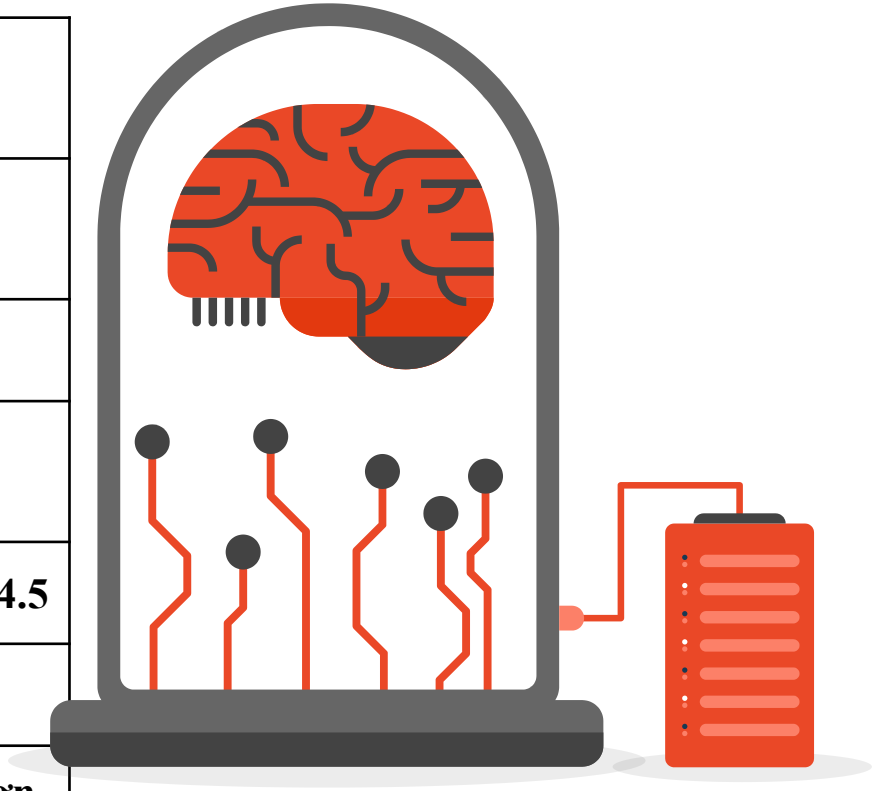
$G_{left/right}$:mô tả tính thuần khiết (hoặc vẩn đục) cùa tập con bên trái/phải

$m_{left/right}$:là số lượng bản ghi có trong tập con trái/phải

- Khi thành công tách bộ dữ liệu làm 2 phần, ta tiếp tục tách các tập con một cách đệ quy cho đến khi đạt được cây quyết định.
- Thuật toán CART là một giải thuật tham lam. theo đó. thuật toán liên tục thử chọn để lấy

Bài tập lý thuyết

01	Nêu 2 ưu điểm và 1 nhược điểm của cây quyết định
02	Sự khác biệt giữa information Gain và Gain Ratio trong C4.5 là gì?
03	Tại sao cần cắt tỉa cây quyết định?
04	Cây CART khác gì với C4.5 về cấu trúc cây?
05	Nêu một ví dụ mà CART phù hợp hơn C4.5
06	Khi nào nên chọn C4.5 thay vì CART?
07	Tại sao cây nhị phân của CART có thể đơn giản hơn C4.5?



Bài tập thực hành

Xây dựng cây CART đơn giản bằng Python

Yêu cầu

- Sử dụng thư viện scikit-learn
- Tập dữ liệu: Iris hoặc tập tự tạo (5 mẫu)
- In ra cấu trúc cây và độ chính xác

Hướng dẫn

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris
X, y = load_iris(return_X_y=True)
clf = DecisionTreeClassifier(criterion='gini')
clf.fit(X, y)
```

Bài tập thực hành

So sánh hiệu quả cây quyết định trước và sau cắt tỉa.

Yêu cầu

- Sử dụng thư viện scikit-learn trên tập Iris
- Thay đổi tham số max_depth (3 và 10)
- So sánh độ chính xác trên tập huấn luyện và kiểm tra