KHAI THÁC DỮ LIỆU
(Data Mining)

Lê Văn Hạnh
levanhanhvn@gmail.com

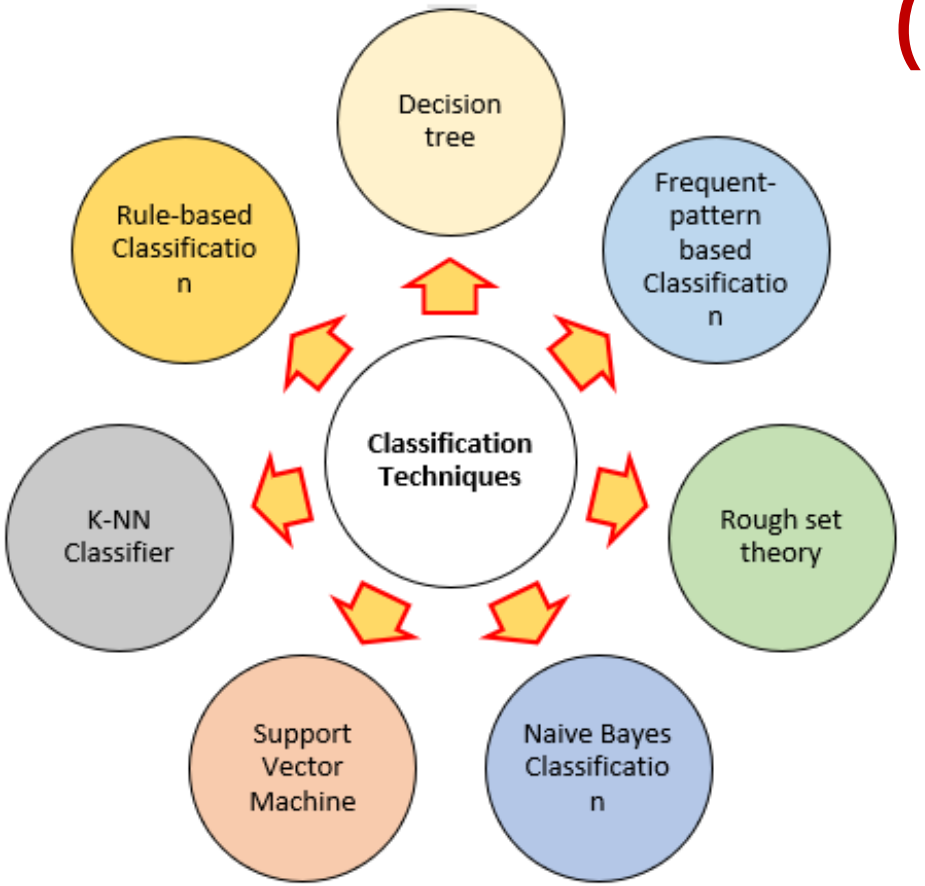
NỘI DUNG MÔN HỌC

1. Tổng quan về Data Science
2. Tìm hiểu dữ liệu
3. Tiền xử lý dữ liệu
4. Khai thác các mẫu phổ biến, mối kết hợp và mối tương quan
5. Phân loại (*Classification*)
6. Phân tích cụm (*Cluster analysis*)

Chương 5

PHÂN LOẠI

(Classification)



Lê Văn Hạnh

levanhanhvn@gmail.com

NỘI DUNG CHƯƠNG 5

1. Một số khái niệm cơ bản
2. Cây quyết định (*Decision Tree*)
3. Phương pháp phân loại Bayes (*Bayes Classification Methods*)
4. Phân loại dựa trên quy tắc (*Rule-Based Classification*)
5. Đánh giá và lựa chọn mô hình (*Model Evaluation and Selection*)
6. Kỹ thuật cải thiện độ chính xác của phân loại (*Techniques to Improve Classification Accuracy*)
7. Bài tập

- Phân loại là một hình thức phân tích dữ liệu trích xuất các mô hình mô tả các lớp dữ liệu quan trọng. Những mô hình như vậy, được gọi là bộ phân loại (*classifiers*), dự đoán nhãn lớp phân loại (*predict categorical class labels* - rời rạc, không có thứ tự).

  Ví dụ: có thể xây dựng một mô hình phân loại để phân loại các khoản vay ngân hàng là an toàn hoặc rủi ro. Phân tích như vậy có thể giúp ta hiểu rõ hơn về dữ liệu nói chung.
- Nhiều phương pháp phân loại đã được các nhà nghiên cứu đề xuất trong machine learning (*machine learning*), nhận dạng mẫu (*pattern recognition*) và thống kê (*statistics*). Hầu hết các thuật toán đều nằm trong bộ nhớ, thường giả định kích thước dữ liệu nhỏ.
- Phát triển các kỹ thuật dự đoán và phân loại có thể mở rộng có khả năng xử lý một lượng lớn dữ liệu.
- Phân loại (*Classification*) có nhiều ứng dụng, bao gồm phát hiện gian lận, tiếp thị mục tiêu, dự đoán hiệu suất, sản xuất và chẩn đoán y tế.

1. MỘT SỐ KHÁI NIỆM CƠ BẢN

1.1. Classification là gì?

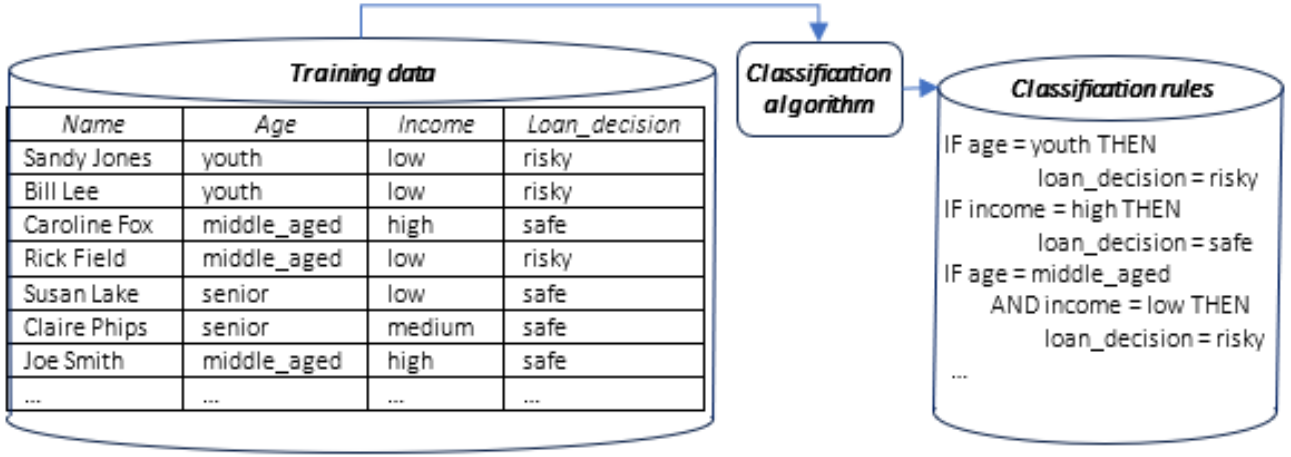
- Một số tình huống thường gặp:
  - Một nhân viên tín dụng của ngân hàng cần phân tích dữ liệu của mình để tìm hiểu xem người nộp đơn xin vay nào là "an toàn" và ai là "rủi ro" cho ngân hàng.
  - Người quản lý tiếp thị cần phân tích dữ liệu để giúp đoán xem liệu khách hàng có hồ sơ nhất định có mua máy tính mới hay không hoặc muốn dự đoán số tiền mà một khách hàng nhất định sẽ chi tiêu trong đợt giảm giá. .
  - Một nhà nghiên cứu y học muốn phân tích dữ liệu về bệnh ung thư vú để dự đoán xem bệnh nhân nên áp dụng một trong ba phương pháp điều trị cụ thể nào.
  - …

Trong mỗi tình huống này, nhiệm vụ phân tích dữ liệu là phân loại, trong đó mô hình hoặc bộ phân loại được xây dựng để dự đoán nhãn lớp (phân loại), chẳng hạn như "***an toàn***" hoặc "***rủi ro***" đối với dữ liệu ứng dụng khoản vay; "***có***" hoặc "***không***" đối với dữ liệu tiếp thị; hoặc "***điều trị A***", "***điều trị B***" hoặc "***điều trị C***" cho dữ liệu y tế.

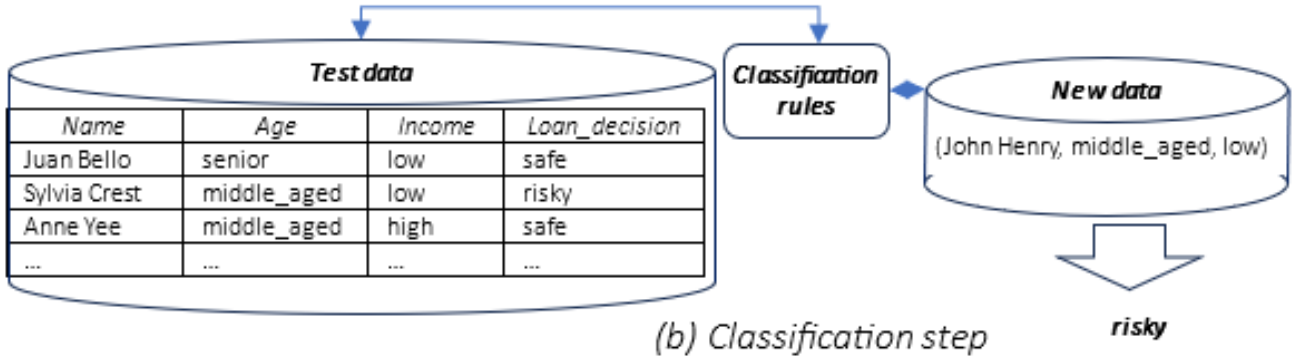
Các danh mục này có thể được biểu diễn bằng các giá trị rời rạc, trong đó thứ tự giữa các giá trị không có ý nghĩa. Ví dụ: các giá trị 1, 2 và 3 có thể được sử dụng để biểu thị các phương pháp điều trị A, B và C, trong đó không có thứ tự ngụ ý nào trong nhóm chế độ điều trị này.

1.2. Cách tiếp cận chung để phân loại *(General Approach to Classification)*

- Phân loại dữ liệu là một quá trình gồm hai bước:
  - Bước học tập (*learning step):* xây dựng mô hình phân loại.
  - Bước phân loại (*classification step*): sử dụng mô hình để dự đoán nhãn lớp (*class labels*).



(a) Learning step

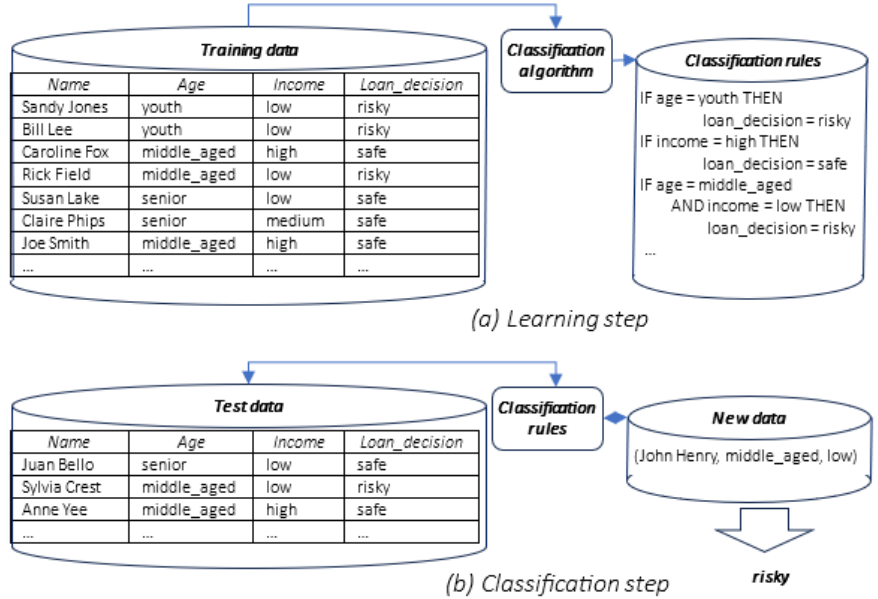


(b) Classification step

Quá trình phân loại dữ liệu:

1.2. Cách tiếp cận chung để phân loại *(General Approach to Classification)*

- Các bộ dữ liệu riêng lẻ tạo nên tập huấn luyện được gọi là các bộ dữ liệu huấn luyện và được lấy mẫu ngẫu nhiên từ CSDL đang được phân tích.
- Thuộc tính nhãn lớp có giá trị rời rạc và không có thứ tự. Nó mang tính phân loại (*categorical* - hoặc danh nghĩa - *nominal*) ở chỗ mỗi giá trị đóng vai trò là một danh mục hoặc lớp.



Do nhãn lớp của mỗi bộ dữ liệu huấn luyện đã được cung cấp nên bước này còn được gọi là ***học có giám sát*** (*supervised learning* - nghĩa là việc học của bộ phân loại được "giám sát" ở chỗ nó được cho biết mỗi bộ dữ liệu huấn luyện thuộc về lớp nào). Nó trái ngược với ***học không giám sát*** (*unsupervised learning* - hoặc **phân cụm** - ***clustering***), trong đó nhãn lớp của mỗi bộ dữ liệu huấn luyện không được biết trước và số lượng hoặc tập hợp các lớp được học có thể không được biết trước.

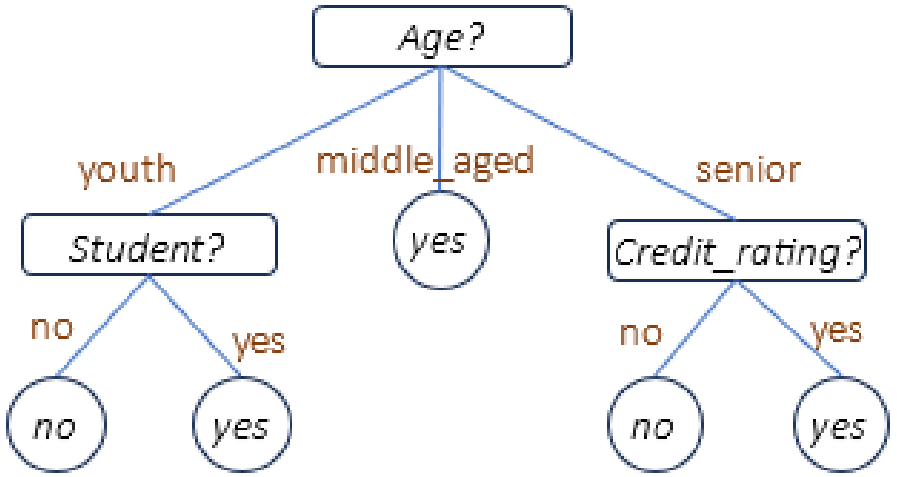
NỘI DUNG CHƯƠNG 5

1. Một số khái niệm cơ bản
2. Cây quyết định (*Decision Tree*)
3. Phương pháp phân loại Bayes (*Bayes Classification Methods*)
4. Phân loại dựa trên quy tắc (*Rule-Based Classification*)
5. Đánh giá và lựa chọn mô hình (*Model Evaluation and Selection*)
6. Kỹ thuật cải thiện độ chính xác của phân loại (*Techniques to Improve Classification Accuracy*)
7. Bài tập

2. CÂY QUYẾT ĐỊNH *(Decision Tree)*

- Cây quyết định *(Decision Tree Induction)*
- Các thước đo hỗ trợ việc lựa chọn thuộc tính *(Attribute Selection Measures)*
- Cắt tỉa cây *(Tree Pruning)*
- Khả năng mở rộng của cây quyết định *(Scalability and Decision Tree Induction)*
- Trực quan hóa quá trình tạo cây quyết định *(Visual Mining for Decision Tree Induction)*

2. Cây quyết định *(Decision Tree)*



Minh họa Cây quyết định cho khái niệm mua máy tính, cho biết liệu khách hàng có khả năng mua máy tính hay không. Mỗi nút nội (không phải nút lá) đại diện cho một thử nghiệm trên một thuộc tính. Mỗi nút lá đại diện cho một lớp (mua máy tính = có hoặc mua máy tính = không).

- Cây quyết định là một cấu trúc cây giống như sơ đồ, trong đó:
  - Nút trên cùng của cây là nút gốc.
  - Mỗi nút nội (*internal node* - nút không phải lá) biểu thị một phép thử trên một thuộc tính, mỗi nhánh biểu thị một kết quả của phép thử. Các nút nội được biểu thị bằng hình chữ nhật.
  - Mỗi nút lá (*leaf node*) hoặc nút đầu cuối (*terminal node*) giữ một nhãn lớp và được biểu thị bằng hình bầu dục.
- Một số thuật toán cây quyết định chỉ tạo ra cây nhị phân (trong đó mỗi nút bên trong phân nhánh thành chính xác hai nút khác), trong khi các thuật toán khác có thể tạo ra cây không nhị phân (*nonbinary trees*).

2.1. Cây quyết định (Decision Tree Induction)

- Vào cuối những năm 1970, J. Ross Quinlan, một nhà nghiên cứu về machine learning, đã phát triển một thuật toán cây quyết định được gọi là ID3 (*Iterative Dichotomiser*). Công trình này mở rộng dựa trên công trình trước đó về hệ thống học tập khái niệm, được mô tả bởi E. B. Hunt, J. Marin và P. T. Stone.
- Quinlan sau đó đã trình bày C4.5 (phiên bản kế thừa của ID3), nó đã trở thành chuẩn mực để so sánh với các thuật toán học có giám sát mới hơn.
- Năm 1984, một nhóm các nhà thống kê (L. Breiman, J. Friedman, R. Olshen và C. Stone) đã xuất bản cuốn sách Cây phân loại và hồi quy (*Classification and Regression Trees* - CART), trong đó mô tả việc tạo ra cây quyết định nhị phân. ID3 và CART được phát minh độc lập với nhau cùng một lúc, nhưng vẫn tuân theo một cách tiếp cận tương tự để học cây quyết định từ các bộ dữ liệu huấn luyện. Hai thuật toán nền tảng này đã tạo ra một loạt công việc về quy nạp cây quyết định.

2.1. Cây quyết định (Decision Tree Induction)

- ID3, C4.5 và CART áp dụng cách tiếp cận tham lam (tức là không quay lui) trong đó cây quyết định được xây dựng theo phương pháp chia để trị đệ quy từ trên xuống (*top-down recursive divide-and-conquer*).
- Hầu hết các thuật toán tạo cây quyết định cũng tuân theo cách tiếp cận từ trên xuống, bắt đầu bằng tập huấn luyện gồm các bộ dữ liệu và nhãn lớp liên quan của chúng. Tập huấn luyện được phân chia đệ quy thành các tập con nhỏ hơn khi cây đang được xây dựng.

2.1. Cây quyết định (Decision Tree Induction)

- Thuật toán tạo cây quyết định

- Cây bắt đầu bằng 1 nút đại diện cho 1 thuộc tính (có trong bộ dữ liệu D) được chọn làm tiêu chí phân tách.
- **Input:**
  - *selection method*: một thủ tục để xác định tiêu chí phân chia "tốt nhất" phân chia các bộ dữ liệu thành các lớp riêPhân vùng dữ liệu D: là một tập hợp các bộ dữ liệu huấn luyện và các nhãn lớp liên quan của chúng;
  - *attribute list*: tập hợp các thuộc tính ứng viên;
  - *Attribute selection method*: một thủ tục để xác định tiêu chí phân chia "tốt nhất" phân chia các bộ dữ liệu thành các lớp riêng lẻ. Tiêu chí này bao gồm một thuộc tính phân tách và có thể là điểm phân tách hoặc tập hợp con phân tách.
- **Output:** Cây quyết định

2.1. Cây quyết định (Decision Tree Induction)

- Thuật toán tạo cây quyết định

```
(1) tạo nút N;
(2) if các bộ dữ liệu trong D đều thuộc cùng một lớp C then
(3)    return N dưới dạng nút lá được gắn nhãn lớp C;
(4) if nếu danh sách thuộc tính (attribute_list) rỗng then
(5)    return N dưới dạng nút lá được gắn nhãn lớp chiếm đa số trong D;
(6) Gọi method  Attribute_selection (D, attribute_list) để tìm ra tiêu chí phân tách
    "tốt nhất";
(7) gán nhãn nút N với tiêu chí phân chia (splittingcriterion);
(8)  if thuộc tính phân tách (splitting_attribute) có giá trị rời rạc và cho phép
     phân chia theo nhiều lớp then
                                                  // không bị giới hạn ở cây nhị phân
(9)    attribute_list ← attribute_list - splitting _attribute; /* loại bỏ
                                                  thuộc tính phân tách*/
(10) for each kết quả j của tiêu chí phân chia (splitting_criterion)
      // phân vùng các bộ dữ liệu và phát triển cây con cho mỗi phân vùng
(11)    Cho Dj là tập hợp các bộ dữ liệu trong D thỏa mãn kết quả j;
                                                  // một phân vùng
(12)    if Dj rỗng then
(13)          gắn nhãn lớp đa số trong D cho nút N;
(14)    else gắn nút được trả về bởi hàm Generate_decision_tree (Dj, attribute_list)
             vào nút N
(15) endfor
(16) return N;
```

2.1. Cây quyết định (Decision Tree Induction)

- **Thuật toán tạo cây quyết định:** giải thích thuật toán
  - Nếu các bộ trong D đều thuộc cùng một lớp thì nút N sẽ trở thành một lá và được gắn nhãn với lớp đó.
  - Thuật toán gọi phương thức *Attribute_selection_method* để xác định tiêu chí phân tách (*splitting criterion*).
    - Tiêu chí phân tách cho ta biết thuộc tính nào cần kiểm tra tại nút N bằng cách xác định cách "tốt nhất" để phân tách hoặc phân chia các bộ dữ liệu trong D thành các lớp riêng lẻ.
    - Tiêu chí phân tách cũng cho biết nhánh nào sẽ phát triển từ nút N cụ thể là chỉ ra thuộc tính phân tách (*splitting attribute*) và cũng có thể chỉ ra điểm phân tách (*split-point*) hoặc tập hợp con phân tách (*splitting subset*).
    - Tiêu chí phân tách được xác định sao cho lý tưởng nhất là các phân vùng kết quả ở mỗi nhánh càng "thuần" (*pure*) càng tốt.

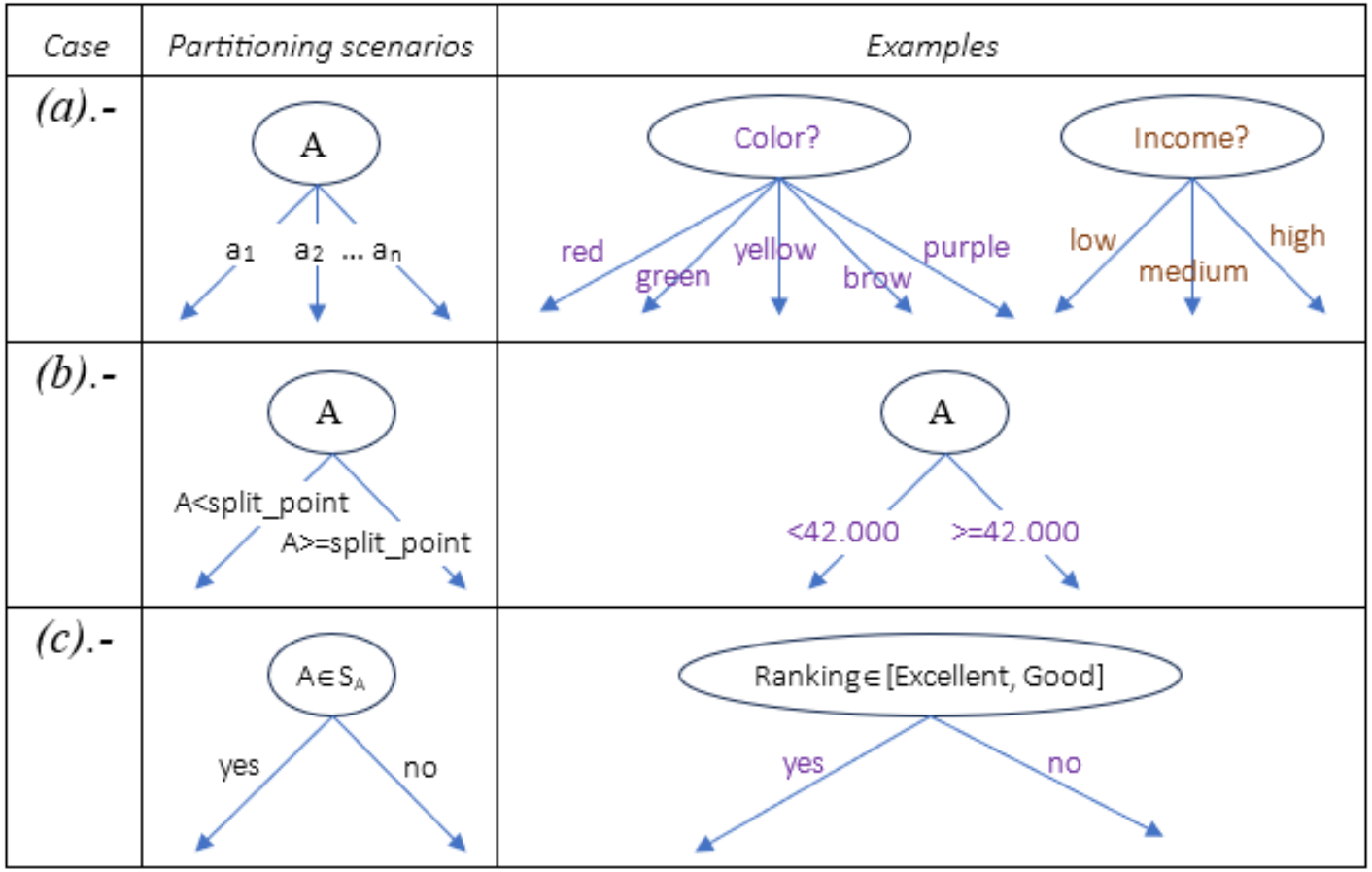
2.1. Cây quyết định (Decision Tree Induction)

- **Thuật toán tạo cây quyết định:** giải thích thuật toán
  - Một nhánh được phát triển từ nút N cho mỗi kết quả của tiêu chí phân tách. Các bộ dữ liệu trong D được phân chia tương ứng. Có ba kịch bản có thể xảy ra:

a) Nếu A có giá trị rời rạc thì một nhánh được phát triển cho mỗi giá trị đã biết của A.

b) Nếu A có giá trị liên tục thì hai nhánh được phát triển, tương ứng với điểm phân chia A<= điểm phân tách và A > điểm phân tách. Việc xác định số lượng điểm phân tách và giá trị tại mỗi điểm phân tách cần được đánh giá dựa trên $Info_A(D)$.

c) Nếu A có giá trị rời rạc và phải tạo ra cây nhị phân thì thử nghiệm có dạng $A \in S_A$, trong đó S_A là tập con phân tách của A.



2. CÂY QUYẾT ĐỊNH *(Decision Tree)*

- Cây quyết định *(Decision Tree Induction)*
- Các thước đo hỗ trợ việc lựa chọn thuộc tính *(Attribute Selection Measures)*
- Cắt tỉa cây *(Tree Pruning)*
- Khả năng mở rộng của cây quyết định *(Scalability and Decision Tree Induction)*
- Trực quan hóa quá trình tạo cây quyết định *(Visual Mining for Decision Tree Induction)*

2.2. Các thước đo hỗ trợ việc lựa chọn thuộc tính (Attribute Selection Measures)

- Nếu chia D thành các phân vùng nhỏ hơn theo kết quả của tiêu chí phân chia thì lý tưởng nhất là **mỗi phân vùng sẽ thuần túy** (*pure* - nghĩa là tất cả các bộ dữ liệu rơi vào một phân vùng nhất định sẽ thuộc cùng một lớp). Về mặt khái niệm, tiêu chí phân chia "tốt nhất" là tiêu chí có kết quả gần đúng nhất trong tình huống như vậy.
- Các thước đo hỗ trợ việc lựa chọn thuộc tính còn được gọi là **quy tắc phân tách** (*splitting rules*) vì chúng xác định cách phân chia các bộ dữ liệu tại một nút nhất định.
- Thuộc tính có điểm đo **tốt nhất** được chọn làm thuộc tính phân tách cho các bộ dữ liệu đã cho.
- Ba thước đo lựa chọn thuộc tính phổ biến là:
  - Độ lợi thông tin (*information gain*)
  - Tỷ lệ khuếch đại (*gain ratio*)
  - Chỉ số Gini (*Gini index*).

2.2. Các thước đo hỗ trợ việc lựa chọn thuộc tính (Attribute Selection Measures)

Cho D, phân vùng dữ liệu, là tập huấn luyện gồm các bộ dữ liệu được gắn nhãn lớp. Giả sử thuộc tính nhãn lớp có m giá trị riêng biệt xác định m lớp riêng biệt, C_i (với i = 1, ..., m). Đặt $C_{i,D}$ là tập các bộ dữ liệu thuộc lớp C_i trong D. Đặt |D| và $|C_{i,D}|$ lần lượt biểu thị số bộ dữ liệu trong D và $C_{i,D}$.

2.2.1. Độ lợi thông tin (Information Gain)

- ID3 sử dụng độ lợi thông tin làm thước đo lựa chọn thuộc tính của nó.
- Đặt nút N đại diện hoặc giữ các bộ dữ liệu của phân vùng D. Thuộc tính có độ lợi thông tin cao nhất được chọn làm thuộc tính phân tách cho nút N.
- Công thức tính thông tin dự kiến cần thiết để phân loại một bộ dữ liệu trong D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

trong đó:

- p_i là xác suất khác 0 để một bộ tùy ý trong D thuộc về lớp C_i và được ước tính bởi $|C_{i,D}|/|D|$.
- Hàm ***log*** cơ số 2 được sử dụng vì thông tin được mã hóa theo bit.
- ***Info(D)*** chỉ là lượng thông tin trung bình cần thiết để xác định nhãn lớp của một bộ dữ liệu trong D. ***Info(D)*** còn được gọi là ***entropy*** của D.

Lưu ý: tại thời điểm này, thông tin ta có chỉ dựa trên tỷ lệ các bộ dữ liệu của mỗi lớp.

2. Cây quyết định *(Decision Tree)*

2.2. Các thước đo hỗ trợ việc lựa chọn thuộc tính (Attribute Selection Measures)

2.2.1. Độ lợi thông tin (Information Gain)

- Để đi đến phân loại chính xác, vẫn cần xác định thêm bao nhiêu thông tin (sau khi phân vùng) để đi đến phân loại chính xác? Số lượng này được đo bằng công thức sau:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

trong đó:

- $\frac{|D_j|}{|D|}$ đóng vai trò là trọng số của phân vùng thứ j.
- ***Info$_A$(D)*** là thông tin dự kiến cần có để phân loại một bộ dữ liệu từ D dựa trên phân vùng theo A. Thông tin dự kiến **càng nhỏ** thì **độ tinh khiết** của các **phân vùng càng cao**.

- Công thức tính Độ lợi thông tin:

$$Gain(A) = Info(D) - Info_A(D).$$

thuộc tính X nào có Gain(X) có giá trị cao nhất sẽ được chọn làm thuộc tính phân tách tại nút N.

2.2.1. Độ lợi thông tin (Information Gain)

- Ví dụ: cho tập dữ liệu huấn luyện D như sau

RID	age	income	student	credit-rating	class: buys_computer
1.	youth	high	no	fair	no
2.	youth	high	no	excellent	no
3.	middle_aged	high	no	fair	yes
4.	senior	medium	no	fair	yes
5.	senior	low	yes	fair	yes
6.	senior	low	yes	excellent	no
7.	middle_aged	low	yes	excellent	yes
8.	youth	medium	no	fair	no
9.	youth	low	yes	fair	yes
10.	senior	medium	yes	fair	yes
11.	youth	medium	yes	excellent	yes
12.	middle_aged	medium	no	excellent	yes
13.	middle_aged	high	yes	fair	yes
14.	senior	medium	no	excellent	no

Thuộc tính nhãn lớp *buys_computer*, có hai giá trị riêng biệt (là {yes, no}); do đó, có hai lớp riêng biệt (tức là m=2). Gọi lớp C_1 tương ứng với ***yes*** và lớp C2 tương ứng với ***no***. Có chín bộ thuộc lớp ***yes*** và năm bộ thuộc lớp ***no***. Nút (gốc) N được tạo cho các bộ trong D.

2. Cây quyết định *(Decision Tree)*

2.2. Các thước đo hỗ trợ việc lựa chọn thuộc tính

*2.2.1. **Độ lợi thông tin** (Information Gain)*

- Ví dụ:

RID	age	income	student	credit-rating	class: buys_computer
1.	youth	high	no	fair	no
2.	youth	high	no	excellent	no
3.	middle_aged	high	no	fair	yes
4.	senior	medium	no	fair	yes
5.	senior	low	yes	fair	yes
6.	senior	low	yes	excellent	no
7.	middle_aged	low	yes	excellent	yes
8.	youth	medium	no	fair	no
9.	youth	low	yes	fair	yes
10.	senior	medium	yes	fair	yes
11.	youth	medium	yes	excellent	yes
12.	middle_aged	medium	no	excellent	yes
13.	middle_aged	high	yes	fair	yes
14.	senior	medium	no	excellent	no

  - Tính thông tin dự kiến cần thiết Info(D) để phân loại một bộ trong D:

$$Info(D) = -\sum_{i=1}^{m} p_i \, log_2(p_i) = -\frac{9}{14} log_2\left(\frac{9}{14}\right) - \frac{5}{14} log_2\left(\frac{5}{14}\right) = 0.940 \; bits$$

  - Tính thông tin mong đợi cho từng thuộc tính. Hãy bắt đầu với thuộc tính age. Cần xem xét sự phân bổ các bộ dữ liệu ***yes*** và ***no*** cho từng loại độ tuổi. Đối với loại tuổi "*youth*", có hai bộ dữ liệu ***yes*** và ba bộ dữ liệu ***no***. Đối với loại tuổi "*middle_aged*", có bốn bộ dữ liệu ***yes*** và không có bộ dữ liệu ***no***. Đối với loại tuổi "*senior*", có ba bộ dữ liệu ***yes*** và hai bộ dữ liệu ***no***.

2.2.1. Độ lợi thông tin (Information Gain)

- Ví dụ:
  - Tính thông tin mong đợi cho từng thuộc tính.
    i. Thuộc tính ***age***:

RID	age	income	student	credit-rating	class: buys_computer
3.	middle_aged	high	no	fair	yes
7.	middle_aged	low	yes	excellent	yes
12.	middle_aged	medium	no	excellent	yes
13.	middle_aged	high	yes	fair	yes
6.	senior	low	yes	excellent	no
14.	senior	medium	no	excellent	no
4.	senior	medium	no	fair	yes
5.	senior	low	yes	fair	yes
10.	senior	medium	yes	fair	yes
1.	youth	high	no	fair	no
2.	youth	high	no	excellent	no
8.	youth	medium	no	fair	no
9.	youth	low	yes	fair	yes

$\Rightarrow$

age	class: buys_computer		Cộng
	yes	no	
youth	2	3	5
middle_aged	4	0	4
senior	3	2	5
CỘNG	***9***	***5***	***14***

$$Info_{age}(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j) = \frac{D_{youth}}{D} \times Info(D_{youth}) + \frac{D_{middle\_aged}}{D} \times Info(D_{middle\_aged}) + \frac{D_{senior}}{D} \times Info(D_{senior})$$

$$Info_{age}(D) = \left(\frac{5}{14} \times \left(-\frac{2}{5}log_2\left(\frac{2}{5}\right) - \frac{3}{5}log_2\left(\frac{3}{5}\right)\right)\right) + \left(\frac{4}{14} \times \left(-\frac{4}{4}log_2\left(\frac{4}{4}\right) - \frac{0}{4}log_2\left(\frac{0}{4}\right)\right)\right) + \left(\frac{5}{14} \times \left(-\frac{3}{5}log_2\left(\frac{3}{5}\right) - \frac{2}{5}log_2\left(\frac{2}{5}\right)\right)\right)$$

$$= \mathbf{0.694\ bits}$$

$\Rightarrow$ *Gain(age) = Info(D)−Info$_{age}$(D)* = 0.940−0.694 = 0.246 bits

2.2.1. Độ lợi thông tin (Information Gain)

- Ví dụ:
  - Tính thông tin mong đợi cho từng thuộc tính.
    ii. Thuộc tính ***income***:

RID	age	income	student	credit-rating	class: buys_computer
1.	youth	high	no	fair	no
2.	youth	high	no	excellent	no
3.	middle_aged	high	no	fair	yes
13.	middle_aged	high	yes	fair	yes
6.	senior	low	yes	excellent	no
7.	middle_aged	low	yes	excellent	yes
5.	senior	low	yes	fair	yes
9.	youth	low	yes	fair	yes
14.	senior	medium	no	excellent	no
8.	youth	medium	no	fair	no
12.	middle_aged	medium	no	excellent	yes
4.	senior	medium	no	fair	yes
10.	senior	medium	yes	fair	yes
11.	youth	medium	yes	excellent	yes

⇒

income	*class: buys_computer*		*Cộng*
	yes	no	
high	2	2	4
low	3	1	4
medium	4	2	6
CỘNG	***9***	***5***	***14***

$\Rightarrow Gain(income) = Info(D) - Info_{income}(D) = 0.940 - 0.911 = 0{,}029$ bits

2.2.1. Độ lợi thông tin (Information Gain)

- Ví dụ:
  - Tính thông tin mong đợi cho từng thuộc tính.
    iii. Thuộc tính ***student***:

RID	age	income	student	credit-rating	class: buys_computer
1	youth	high	no	fair	no
2.	youth	high	no	excellent	no
14.	senior	medium	no	excellent	no
8.	youth	medium	no	fair	no
3.	middle_aged	high	no	fair	yes
12.	middle_aged	medium	no	excellent	yes
4.	senior	medium	no	fair	yes
6.	senior	low	yes	excellent	no
13.	middle_aged	high	yes	fair	yes
7.	middle_aged	low	yes	excellent	yes
5.	senior	low	yes	fair	yes
9.	youth	low	yes	fair	yes
10.	senior	medium	yes	fair	yes
11.	youth	medium	yes	excellent	yes

$\Rightarrow$

student	class: buys_computer		Cộng
	yes	no	
yes	6	1	7
no	3	4	7
CỘNG	***9***	***5***	***14***

$\Rightarrow Gain(student) = Info(D) - Info_{student}(D) = 0.940 - 0.789 = 0{,}151$ bits

2.2.1. Độ lợi thông tin (Information Gain)

- Ví dụ:
  - Tính thông tin mong đợi cho từng thuộc tính.
    iv. Thuộc tính ***credit-rating***:

RID	age	income	student	credit-rating	class: buys_computer
2.	youth	high	no	excellent	no
14.	senior	medium	no	excellent	no
6.	senior	low	yes	excellent	no
12.	middle_aged	medium	no	excellent	yes
7.	middle_aged	low	yes	excellent	yes
11.	youth	medium	yes	excellent	yes
1	youth	high	no	fair	no
8.	youth	medium	no	fair	no
3.	middle_aged	high	no	fair	yes
4.	senior	medium	no	fair	yes
13.	middle_aged	high	yes	fair	yes
5.	senior	low	yes	fair	yes
9.	youth	low	yes	fair	yes
10.	senior	medium	yes	fair	yes

⇒

Credit-rating	class: buys_computer		Cộng
	yes	no	
excellent	3	3	6
fair	6	2	8
CỘNG	***9***	***5***	***14***

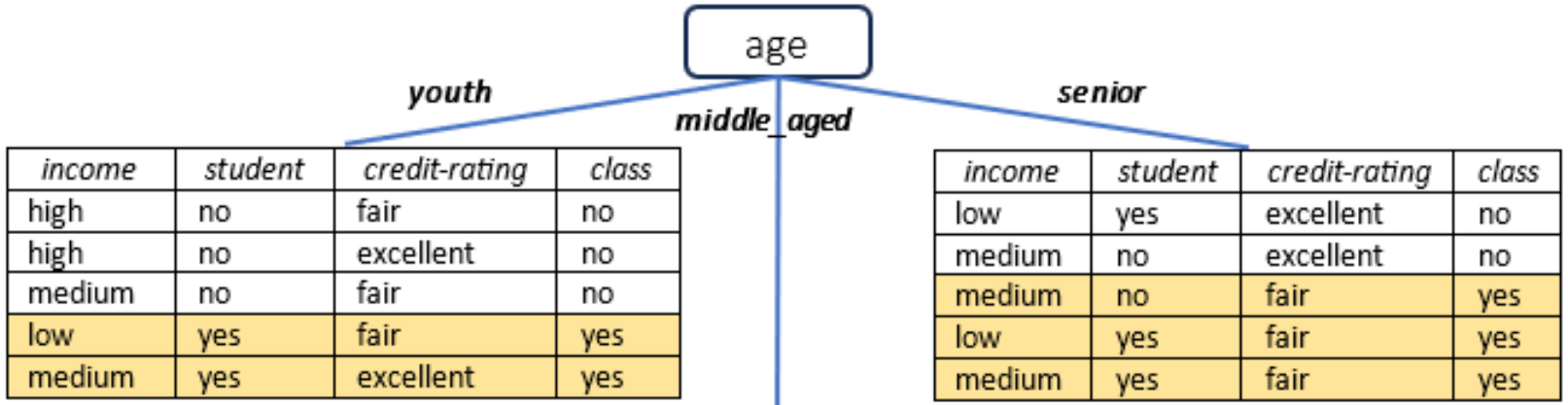
⇒ $Gain(credit\text{-}rating) = Info(D) - Info_{credit\text{-}rating}(D) = 0.940 - 0.892 = 0{,}048$ bits

Tổng hợp Gain của các thuộc tính:

Attribute	Gain
age	**0.246**
income	0.029
student	0.151
credit_rating	0.048

2.2.1. Độ lợi thông tin (Information Gain)

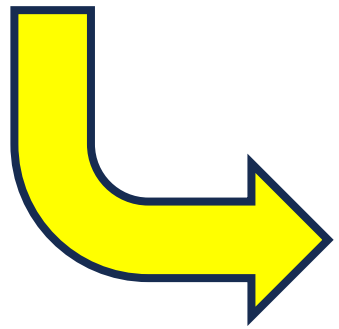
- Ví dụ: vì thuộc tính age có độ lợi thông tin lớn nhất nên được chọn làm tiêu chí phân tách

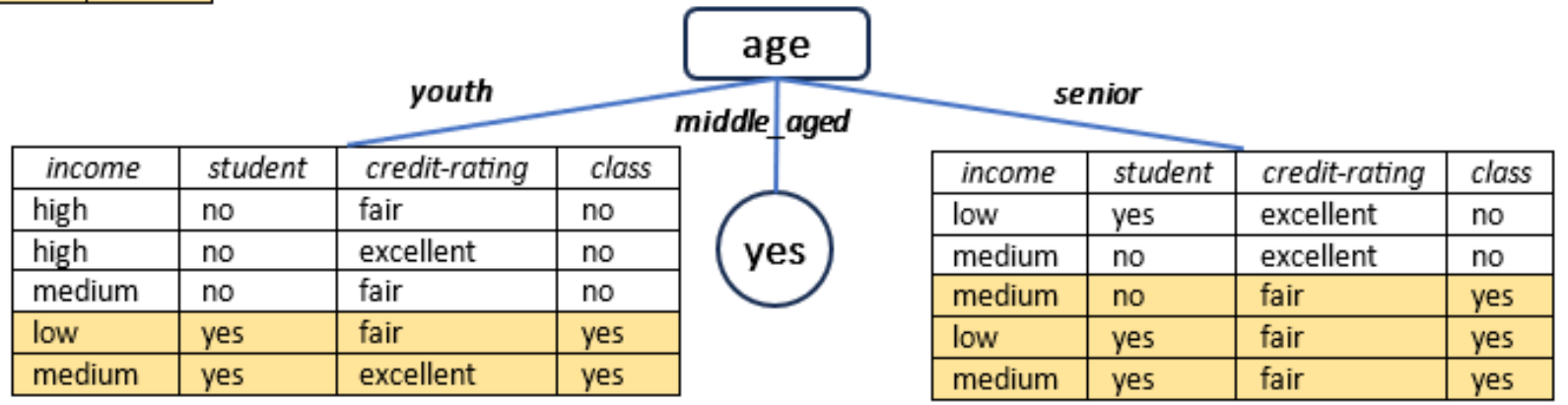


income	*student*	*credit-rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	*student*	*credit-rating*	*class*
low	yes	excellent	no
medium	no	excellent	no
medium	no	fair	yes
low	yes	fair	yes
medium	yes	fair	yes

income	*student*	*credit-rating*	*class*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes



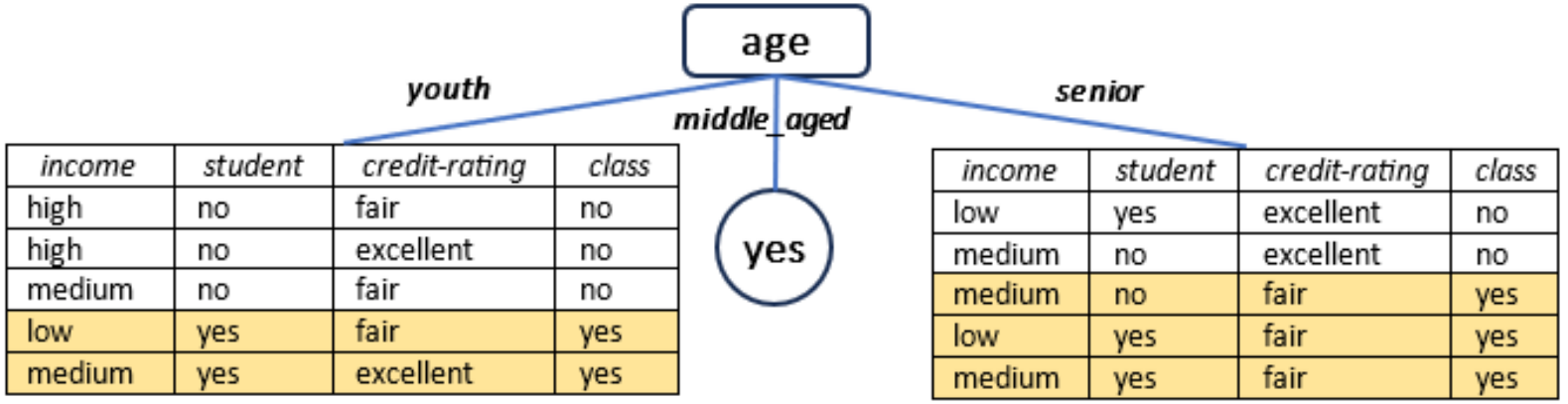


income	*student*	*credit-rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	*student*	*credit-rating*	*class*
low	yes	excellent	no
medium	no	excellent	no
medium	no	fair	yes
low	yes	fair	yes
medium	yes	fair	yes

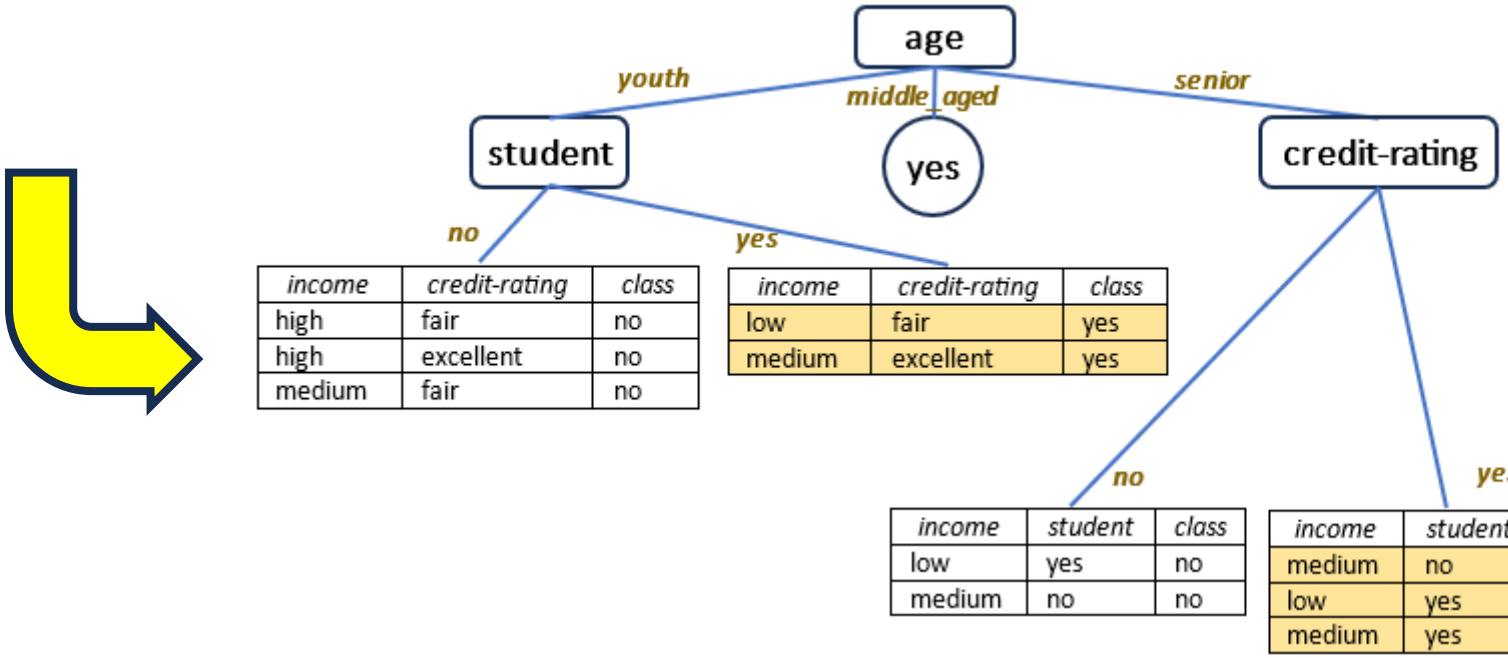
2.2.1. Độ lợi thông tin (Information Gain)

- Ví dụ: tiếp tục tính độ lợi thông tin cho từng nhánh để chọn tiêu chí phân tách



income	student	credit-rating	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	student	credit-rating	class
low	yes	excellent	no
medium	no	excellent	no
medium	no	fair	yes
low	yes	fair	yes
medium	yes	fair	yes

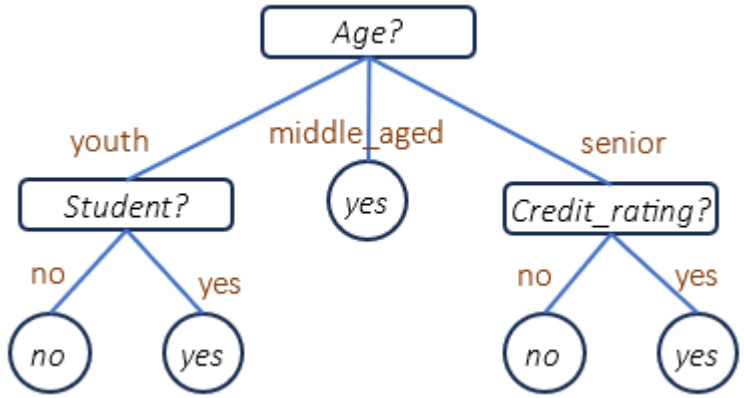


income	credit-rating	class
high	fair	no
high	excellent	no
medium	fair	no

income	credit-rating	class
low	fair	yes
medium	excellent	yes

income	student	class
low	yes	no
medium	no	no

income	student	class
medium	no	yes
low	yes	yes
medium	yes	yes



2.2.2. Tỷ lệ khuyếch đại (Gain Ratio)

- Việc sử dụng Độ lợi thông tin sẽ bất cập khi gặp thuộc tính có số lượng phân vùng lớn (như thuộc tính ID), và số lượng các bộ dữ liệu trên mỗi vùng là ít (có thể chỉ gồm 1 bộ/phân vùng).
- Gain Ratio sử dụng giá trị "tách thông tin" (*split information*) được xác định tương tự với Info(D)

$$SplitInfo_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times log_2\left(\frac{|D_j|}{D}\right)$$

- Công thức tính tỷ lệ khuyếch đại:

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

- Thuộc tính có **tỷ lệ khuếch đại tối đa được chọn làm thuộc tính phân tách**.
- ***Lưu ý***: khi thông tin phân tách tiến tới 0, tỷ lệ này sẽ không ổn định. Một ràng buộc được thêm vào để tránh điều này, theo đó mức tăng thông tin của thử nghiệm được chọn phải lớn hơn hoặc là bằng mức tăng trung bình của tất cả các thử nghiệm được dùng để kiểm tra.

2.2.3. Chỉ số Gini (Gini index)

- Chỉ số Gini được sử dụng trong *Cây phân loại và hồi quy* (*Classification and Regression Trees* – CART). Chỉ số ***Gini*** đo lường mức độ "tạp chất" (*impurity*) của D và được tính dựa trên công thức sau:

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

trong đó:

- p_i là xác suất để một bộ dữ liệu trong D thuộc lớp C_i và được ước tính bởi $|C_{i,D}|/|D|$.
- ***m***: tổng số lớp.

2.2.3. Chỉ số Gini (Gini index)

- Chỉ số ***Gini*** xem xét việc phân chia nhị phân (*binary split*) cho từng thuộc tính. Để xác định phép tách nhị phân tốt nhất trên *A*, ta kiểm tra tất cả các tập con có thể có mà có thể được hình thành bằng cách sử dụng các giá trị đã biết của *A*. Mỗi tập hợp con, S_A, có thể được coi là thử nghiệm nhị phân cho thuộc tính *A* có dạng "A ∈ S_A?" Cho một bộ, thử nghiệm này được thỏa mãn nếu giá trị của *A* của bộ đó nằm trong số các giá trị được liệt kê trong S_A. Nếu *A* có *v* giá trị có thể thì có thể có 2^v tập con.
  - Ví dụ: thuộc tính income có ba giá trị là {*low, medium, high*} thì các tập hợp con có thể là {*low, medium, high*}, {*low, medium*}, {*low, high*}, {*medium, high*}, {*low*}, {*medium*}, {*high*} và {}. Loại trừ tập {*low, medium, high*} và tập trống khỏi việc xem xét vì về mặt khái niệm, chúng không đại diện cho sự phân chia. Do đó, có **$2^v - 2$** cách có thể để **tạo thành hai phân vùng dữ liệu** trên D, dựa trên phép chia nhị phân trên A.
  - Nếu phân chia nhị phân trên *A* phân vùng *D* thành D_1 và D_2, chỉ số Gini của *D* cho phân vùng đó là:

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

2.2.3. Chỉ số Gini (Gini index)

- Đối với mỗi thuộc tính, mỗi phân chia nhị phân có thể được xem xét.
- Tiêu chí phân tách đối với một thuộc tính có giá trị:
  - *Rời rạc*: tập hợp con cung cấp chỉ số Gini tối thiểu cho thuộc tính đó sẽ được chọn làm tập hợp con phân tách của nó.
  - *Liên tục*: mỗi điểm phân chia có thể phải được xem xét (tương tự như chiến lược để thu thập thông tin), trong đó có thể lấy điểm giữa giữa mỗi cặp giá trị liền kề (đã được sắp xếp) làm điểm phân chia. Điểm đưa ra **chỉ số Gini tối thiểu cho một thuộc tính** nhất định (có giá trị liên tục) **được lấy làm điểm phân chia** của thuộc tính đó.
- Việc giảm tạp chất có thể xảy ra do phân chia nhị phân trên thuộc tính A có giá trị rời rạc hoặc liên tục là:

$$\Delta\, Gini(A) = Gini(D) - Gini_A(D)$$

$\Rightarrow$ Thuộc tính làm giảm tối đa tạp chất ($\Delta\, Gini(A)$ lớn nhất hoặc có chỉ số Gini tối thiểu) được chọn làm thuộc tính phân tách

2. Cây quyết định *(Decision Tree)*

2.2. Các thước đo hỗ trợ việc lựa chọn thuộc tính

2.2.3. Chỉ số Gini *(Gini index)*

- Ví dụ: lấy lại dữ liệu từ việc mua máy tính trong đó có 9 bộ dữ liệu thuộc lớp *buys_computer = yes* và 5 bộ dữ liệu còn lại thuộc lớp *buys_computer = no*.

RID	age	income	student	credit-rating	class: buys_computer
1.	youth	high	no	fair	no
2.	youth	high	no	excellent	no
3.	middle_aged	high	no	fair	yes
4.	senior	medium	no	fair	yes
5.	senior	low	yes	fair	yes
6.	senior	low	yes	excellent	no
7.	middle_aged	low	yes	excellent	yes
8.	youth	medium	no	fair	no
9.	youth	low	yes	fair	yes
10.	senior	medium	yes	fair	yes
11.	youth	medium	yes	excellent	yes
12.	middle_aged	medium	no	excellent	yes
13.	middle_aged	high	yes	fair	yes
14.	senior	medium	no	excellent	no

  - ***B1***: tính chỉ số Gini để tính tạp chất của D

$$Gini(D) = 1 - \Sigma_{i=1}^{m} p_i^2 = 1 - \left(\left(\frac{9}{14}\right)^2 + \left(\frac{5}{14}\right)^2\right) = 0.459$$

  - ***B2***: tính chỉ số Gini để tìm tiêu chí phân tách cho các bộ trong D
    - Thuộc tính ***income***:

subsets	D_1				D_2				$Gini_{(subset)}$
	Quantity	class		$Gini(D_1)$	Quantity (=14-D1)	class		$Gini(D_1)$	
		Yes	No			Yes	No		
{ low }	4	3	1		10				0.450
{ medium }	6	4	2		8				0.458
{ high }	4	2	2		10				0.443
{ low, medium }	10	7	3	$\frac{10}{14}\left(1-\left(\frac{7}{10}\right)^2-\left(\frac{3}{10}\right)^2\right)=0.3$	4	2	2	$\frac{4}{14}\left(1-\left(\frac{2}{4}\right)^2-\left(\frac{2}{4}\right)^2\right)=0.143$	0.443
{ low, high }	8	5	3		6				0.458
{ medium, high }	10	6	4		4				0.450

⇒ Tiêu chí phân tách được chọn dựa trên tập con {*low, medium*} do có giá trị nhỏ nhất

2.2.3. Chỉ số Gini (Gini index)

- Ví dụ:
  - B2: tính chỉ số Gini để tìm tiêu chí phân tách cho các bộ trong D
    - Thuộc tính ***age***: chọn subset {*youth*, *senior*} (hoặc {*middle_aged*}) là mức phân chia thuộc tính ***age*** tốt nhất với chỉ số Gini là 0,375.
    - Thuộc tính ***student***: là thuộc tính nhị phân, với giá trị chỉ số Gini là 0,367.
    - Thuộc tính ***credit***: là thuộc tính nhị phân, với giá trị chỉ số Gini 0,429

Thuộc tính	*Tập con*	*Gini(D)*	$Gini_A(D)$	$\Delta Gini(A) = Gini(D) - Gini_A(D)$
income	{low, medium}	0,459	0,443	0,016
age	{youth, senior} (hoặc {middle_aged})		0,375	0,102
student	Thuộc tính nhị phân		0,367	0,092
credit	Thuộc tính nhị phân		0,429	0,030

$\Rightarrow$ Tiêu chí phân tách cho tập dữ liệu D được chọn dựa trên tập con {*low*, *medium*} do có giá trị nhỏ nhất

2.2.4. Một số biện pháp chọn lựa thuộc tính khác (Other Attribute Selection Measures)

- Nhiều biện pháp lựa chọn thuộc tính khác đã được đề xuất như:

i. *CHAID*: một thuật toán cây quyết định phổ biến trong tiếp thị, sử dụng thước đo lựa chọn thuộc tính dựa trên thử nghiệm thống kê χ^2 về tính độc lập.

ii. *C-SEP:* hoạt động tốt hơn *information gain* và *Gini Index* trong một số trường hợp nhất định.

iii. *G-statistic:* một thước đo lý thuyết thông tin gần đúng với phân phối χ^2.

2.2.4. Một số biện pháp chọn lựa thuộc tính khác (Other Attribute Selection Measures)

- Nhiều biện pháp lựa chọn thuộc tính khác đã được đề xuất như:

iv. *MDL (Minimum Description Length)*: lựa chọn thuộc tính dựa trên nguyên tắc Độ dài mô tả tối thiểu (MDL) có ít sai lệch nhất đối với các thuộc tính đa giá trị. Các biện pháp dựa trên MDL (với ý tưởng chính của nó là ưu tiên giải pháp đơn giản nhất) sử dụng các kỹ thuật mã hóa để xác định cây quyết định "tốt nhất" là cây yêu cầu số bit ít nhất để:

1) Mã hóa cây
2) Mã hóa các ngoại lệ đối với cây (tức là các trường hợp không được phân loại chính xác theo cây).

2.2.4. Một số biện pháp chọn lựa thuộc tính khác *(Other Attribute Selection Measures)*

- Nhiều biện pháp lựa chọn thuộc tính khác đã được đề xuất như:

v. Các biện pháp lựa chọn thuộc tính khác xem xét việc phân chia nhiều biến (tức là, trong đó việc phân chia các bộ dữ liệu dựa trên sự kết hợp của các thuộc tính, thay vì trên một thuộc tính duy nhất).

Ví dụ: hệ thống CART có thể tìm thấy các phân chia đa biến dựa trên sự kết hợp tuyến tính của các thuộc tính. Phân chia đa biến là một dạng xây dựng thuộc tính (hoặc tính năng), trong đó các thuộc tính mới được tạo dựa trên các thuộc tính hiện có.

2.2.4. Một số biện pháp chọn lựa thuộc tính khác (Other Attribute Selection Measures)

- Hầu hết các biện pháp đều cho kết quả khá tốt. Nhưng "*Phương pháp nào lựa chọn thuộc tính nào là tốt nhất?*"
- Đặc điểm chung:
  - Tất cả các biện pháp đều có một số sai lệch.
  - Độ phức tạp về thời gian của việc tạo cây quyết định thường tăng theo cấp số nhân theo chiều cao của cây. Do đó, các biện pháp có xu hướng tạo ra các cây nông hơn (ví dụ: phân chia nhiều chiều thay vì phân chia nhị phân và thiên về phân chia cân bằng hơn) có thể được ưu tiên hơn.
  - Tuy nhiên, những cây nông thường có số lượng lá lớn và tỷ lệ lỗi cao hơn.

$\Rightarrow$ Mặc dù có nhiều nghiên cứu so sánh, nhưng **không có biện pháp lựa chọn thuộc tính nào được cho là vượt trội hơn đáng kể so với các biện pháp khác**.

2. CÂY QUYẾT ĐỊNH *(Decision Tree)*

- Cây quyết định *(Decision Tree Induction)*
- Các thước đo hỗ trợ việc lựa chọn thuộc tính *(Attribute Selection Measures)*
- Cắt tỉa cây *(Tree Pruning)*
- Khả năng mở rộng của cây quyết định *(Scalability and Decision Tree Induction)*
- Trực quan hóa quá trình tạo cây quyết định *(Visual Mining for Decision Tree Induction)*

2.3. Cắt tỉa cây (Tree Pruning)

- Khi cây quyết định được xây dựng, nhiều nhánh sẽ phản ánh sự bất thường trong dữ liệu huấn luyện do nhiễu hoặc các ngoại lệ. Các phương pháp cắt tỉa cây giải quyết vấn đề dữ liệu quá khớp (*overfitting*) này.
- Những phương pháp cắt tỉa thường sử dụng các biện pháp thống kê để loại bỏ các nhánh có độ tin cậy kém nhất. Cây được cắt tỉa có xu hướng nhỏ hơn và ít phức tạp hơn, do đó dễ hiểu hơn. Chúng thường nhanh hơn và tốt hơn trong việc phân loại chính xác dữ liệu thử nghiệm độc lập (tức là các bộ dữ liệu chưa được nhìn thấy trước đó) so với các cây không được cắt tỉa.
- Có hai cách tiếp cận phổ biến để cắt tỉa cây:
  - Cắt tỉa trước (prepruning approach)
  - Cắt tỉa sau (postpruning approach)

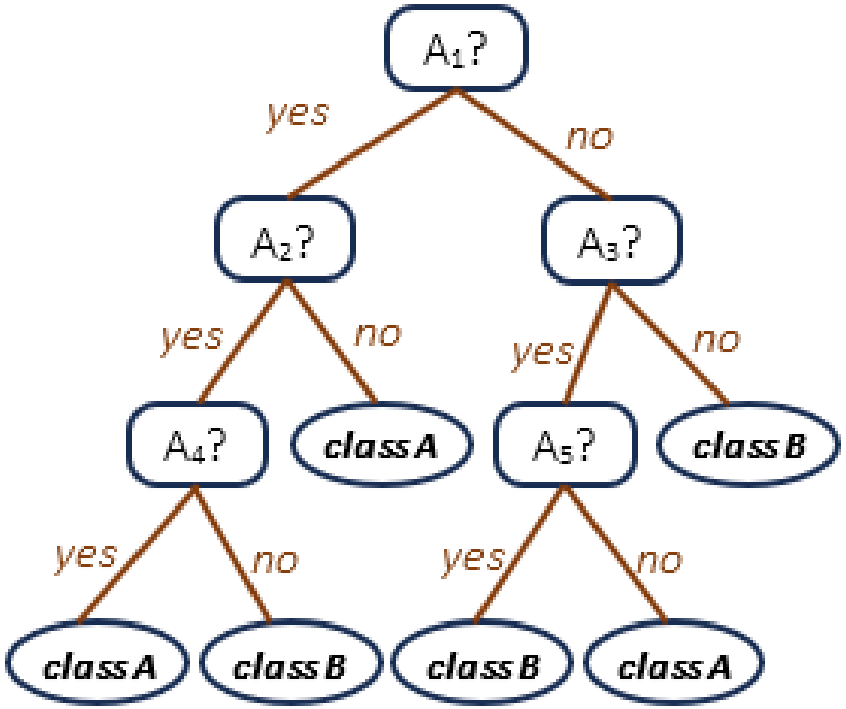
2.3.1. Cắt tỉa trước (prepruning approach)

- Trong phương pháp cắt tỉa trước, một cây được "cắt tỉa" bằng cách **tạm dừng quá trình xây dựng của nó sớm** (ví dụ: bằng cách quyết định không phân chia hoặc phân vùng thêm tập hợp con của các bộ dữ liệu huấn luyện tại một nút nhất định). Khi dừng lại, nút sẽ trở thành một chiếc lá. Lá có thể chứa lớp phổ biến (*frequent class*) nhất trong số các bộ dữ liệu tập con (*subset tuples*) hoặc phân bố xác suất (*probability distribution*) của các bộ dữ liệu đó.
- Khi xây dựng cây, các thước đo như ý nghĩa thống kê (*statistical significance*), độ lợi thông tin (*information gain*), chỉ số Gini (*Gini index*), v.v., có thể được sử dụng để đánh giá mức độ tốt của việc phân chia. Nếu việc phân vùng các bộ dữ liệu tại một nút sẽ dẫn đến **sự phân chia giảm xuống dưới ngưỡng được chỉ định trước** thì việc phân vùng tiếp theo của tập hợp con đã cho **sẽ bị tạm dừng**.

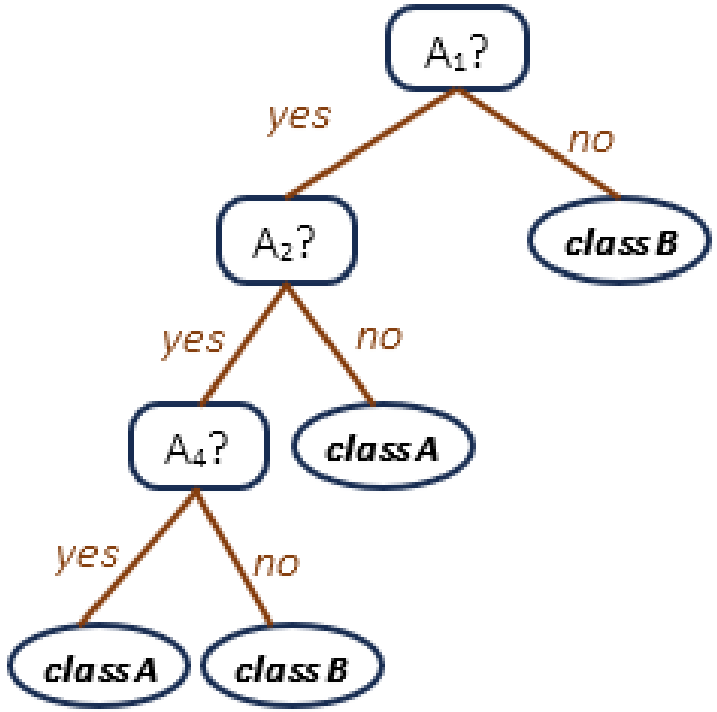
  Tuy nhiên, có những khó khăn trong việc lựa chọn một ngưỡng thích hợp. Ngưỡng cao có thể dẫn đến cây được đơn giản hóa quá mức, trong khi ngưỡng thấp có thể dẫn đến sự đơn giản hóa rất ít.

2.3.2. Cắt tỉa sau (postpruning approach)

- Cách Cắt tỉa sau được sử dụng phổ biến hơn
- Cách này sẽ thực hiện loại bỏ các cây con khỏi cây "đã phát triển đầy đủ".
- Cây con tại một nút nhất định được cắt bớt bằng cách loại bỏ các nhánh của nó và thay thế bằng một lá. Lá được dán nhãn là lớp phổ biến nhất (*the most frequent class*) trong số cây con được thay thế.
- Ví dụ:



TRƯỚC khi cắt tỉa cây



SAU khi cắt tỉa cây

2.3.2. Cắt tỉa sau (postpruning approach)

- **Thuật toán CART**:
  - Sử dụng cắt tỉa độ phức tạp chi phí (***costcomplexity** pruning*). Cách tiếp cận này coi độ phức tạp về chi phí của một cây là một hàm của số lá trong cây và tỷ lệ lỗi của cây (trong đó tỷ lệ lỗi - *error rate*- là tỷ lệ phần trăm các bộ dữ liệu bị phân loại sai bởi cây).
  - Bắt đầu từ dưới cùng của cây. Đối với mỗi nút bên trong, N, nó tính toán độ phức tạp về chi phí của cây con tại N trước và sau khi cắt tỉa (tức là khi nhánh đó được thay thế bằng một nút lá). Hai giá trị được so sánh. Nếu việc cắt tỉa cây con tại nút N sẽ dẫn đến độ phức tạp về chi phí nhỏ hơn thì cây con đó sẽ được cắt tỉa. Nếu không, nó sẽ được giữ lại.
  - Một tập hợp các bộ dữ liệu được gắn nhãn lớp được sử dụng để ước tính độ phức tạp của chi phí. Bộ này độc lập với tập huấn luyện được sử dụng để xây dựng cây không được cắt tỉa và bất kỳ bộ kiểm tra nào được sử dụng để ước tính độ chính xác. Thuật toán tạo ra một tập hợp các cây được cắt tỉa dần dần.

2.3.2. Cắt tỉa sau (postpruning approach)

- **Thuật toán C4.5**:
  - Sử dụng một phương pháp gọi là cắt tỉa bi quan (*pessimistic pruning*), tương tự như phương pháp phức tạp về chi phí (*the cost complexity method*) ở chỗ nó cũng sử dụng ước tính tỷ lệ lỗi để đưa ra quyết định liên quan đến việc cắt tỉa cây con.
  - Tuy nhiên, phương pháp *pessimistic pruning* không yêu cầu sử dụng bộ cắt tỉa. Thay vào đó, nó sử dụng tập huấn luyện để ước tính tỷ lệ lỗi. Hãy nhớ lại rằng ước tính về độ chính xác hoặc lỗi dựa trên tập huấn luyện là quá chủ quan và do đó có độ lệch mạnh. Do đó, phương pháp *pessimistic pruning* sẽ điều chỉnh tỷ lệ lỗi thu được từ tập huấn luyện bằng cách thêm một "hình phạt" (*penalty*) để chống lại độ lệch phát sinh.

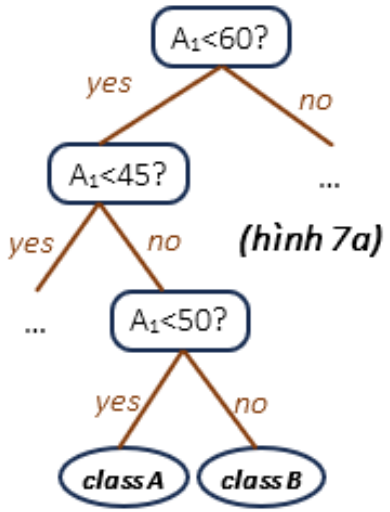
2.3.2. Cắt tỉa sau (postpruning approach)

- **Các thuật toán sử dụng nguyên tắc MDL** *(Minimum Description Length)* :
  - Thay vì tỉa cây dựa trên tỷ lệ lỗi ước tính, ta có thể tỉa cây dựa trên số bit cần thiết để mã hóa chúng. Cây được cắt tỉa "tốt nhất" là cây giảm thiểu số bit mã hóa.
  - Ý tưởng cơ bản là giải pháp đơn giản nhất được ưu tiên. Không giống như việc cắt bớt độ phức tạp về chi phí, nó không yêu cầu một bộ dữ liệu độc lập.

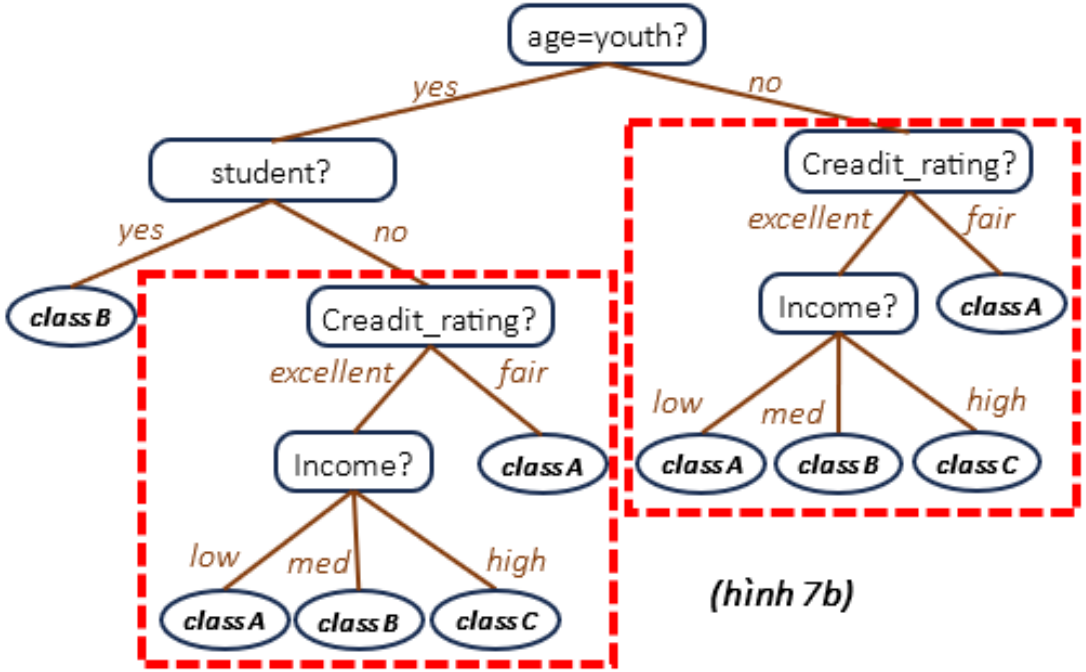
2.3.3. Nhận xét về các phương pháp cắt tỉa

- Không có phương pháp cắt tỉa nào được cho là vượt trội hơn tất cả các phương pháp khác.
- Việc cắt tỉa trước và cắt tỉa sau có thể được xen kẽ cho một phương pháp kết hợp. Việc cắt tỉa sau đòi hỏi nhiều tính toán hơn so với việc cắt tỉa trước nhưng nhìn chung mang lại một cây đáng tin cậy hơn.
- Mặc dù những cây được cắt tỉa có xu hướng nhỏ gọn hơn so với những cây không được cắt tỉa nhưng chúng vẫn có thể khá lớn và phức tạp. Cây quyết định có thể bị lặp lại và sao chép, khiến chúng khó diễn giải.

sự lặp lại của cây con, trong đó một thuộc tính được kiểm tra lặp đi lặp lại dọc theo một nhánh nhất định của cây



(hình 7a)



(hình 7b)

sao chép cây con, trong đó các cây con trùng lặp tồn tại trong một cây

2. CÂY QUYẾT ĐỊNH *(Decision Tree)*

- Cây quyết định *(Decision Tree Induction)*
- Các thước đo hỗ trợ việc lựa chọn thuộc tính *(Attribute Selection Measures)*
- Cắt tỉa cây *(Tree Pruning)*
- Khả năng mở rộng của cây quyết định *(Scalability and Decision Tree Induction)*
- Trực quan hóa quá trình tạo cây quyết định *(Visual Mining for Decision Tree Induction)*

2.4. Khả năng mở rộng của cây quyết định (Scalability and Decision Tree Induction)

- Hiệu quả của các thuật toán cây quyết định được tạo từ các thuật toán như *ID3*, *C4.5* và *CART*, đã được thiết lập tốt cho các **tập dữ liệu tương đối nhỏ** (các bộ dữ liệu huấn luyện phải **nằm trong bộ nhớ**). Hiệu quả trở thành vấn đề đáng quan tâm khi các thuật toán này được áp dụng để khai thác CSDL thế giới thực rất lớn.
- Trong các ứng dụng khai thác dữ liệu, **tập huấn luyện rất lớn** gồm hàng triệu bộ dữ liệu là phổ biến. Thông thường, dữ liệu huấn luyện **sẽ không vừa với bộ nhớ**! Do đó, việc xây dựng cây quyết định trở nên kém hiệu quả do việc hoán đổi các bộ dữ liệu huấn luyện vào và ra khỏi bộ nhớ chính và bộ nhớ đệm. Cần có nhiều cách tiếp cận có khả năng mở rộng hơn, có khả năng xử lý dữ liệu huấn luyện quá lớn để chứa trong bộ nhớ.

2.4. Khả năng mở rộng của cây quyết định (Scalability and Decision Tree Induction)

- Một số phương pháp quy nạp cây quyết định có thể mở rộng đã được giới thiệu trong các nghiên cứu gần đây như:

(i) ***RainForest:*** điều chỉnh theo dung lượng bộ nhớ chính có sẵn và áp dụng cho bất kỳ thuật toán cảm ứng cây quyết định nào. Phương thức này duy trì một *AVC-set* (*"Attribute-Value, Classlabel"*) cho mỗi thuộc tính, tại mỗi nút cây, mô tả các bộ dữ liệu huấn luyện tại nút đó. Tập hợp AVC của thuộc tính A tại nút N cung cấp số lượng nhãn lớp cho từng giá trị của A đối với các bộ dữ liệu tại N.

Ví dụ: Hình bên hiển thị các tập hợp AVC cho dữ liệu bộ dữ liệu của ví dụ trước. Tập hợp tất cả các bộ AVC tại nút N là *AVC-group* của N. Kích thước của *AVC-set* cho thuộc tính A tại nút N chỉ phụ thuộc vào số lượng giá trị phân biệt của A và số lượng lớp trong tập hợp số bộ dữ liệu tại N.

age	*buys_computer*	
	yes	*no*
youth	2	3
middle_aged	4	0
senior	3	2

income	*buys_computer*	
	yes	*no*
low	3	1
medium	4	2
high	2	2

student	*buys_computer*	
	yes	*no*
yes	6	1
no	3	4

credit_rating	*buys_computer*	
	yes	*no*
fair	6	2
excellent	3	3

⇒ phương pháp này có khả năng mở rộng cao để tạo cây quyết định trong các tập dữ liệu rất lớn.

2.4. Khả năng mở rộng của cây quyết định (Scalability and Decision Tree Induction)

- Một số phương pháp quy nạp cây quyết định có thể mở rộng:

ii. BOAT (Bootstrapped Optimistic Algorithm for Tree construction)

- Sử dụng một kỹ thuật thống kê được gọi là "*bootstrapping*" để tạo một số mẫu (hoặc tập hợp con) nhỏ hơn của dữ liệu huấn luyện nhất định, mỗi mẫu phù hợp với bộ nhớ. Mỗi tập hợp con được sử dụng để xây dựng một cây, tạo ra một số cây. Các cây được kiểm tra và sử dụng để xây dựng một cây mới, T_0, hóa ra là "rất gần" (*very close*) với cây lẽ ra đã được tạo ra nếu tất cả dữ liệu huấn luyện ban đầu nằm gọn trong bộ nhớ.

2.4. Khả năng mở rộng của cây quyết định (Scalability and Decision Tree Induction)

- Một số phương pháp quy nạp cây quyết định có thể mở rộng:

ii. ***BOAT*** (*Bootstrapped Optimistic Algorithm for Tree construction*)

- *Ưu điểm:*
  - Có thể sử dụng bất kỳ biện pháp lựa chọn thuộc tính nào để chọn các phân tách nhị phân và dựa trên khái niệm về độ tinh khiết của các phân vùng, chẳng hạn như chỉ số *Gini*. *BOAT* sử dụng giới hạn dưới của thước đo lựa chọn thuộc tính để phát hiện xem cây "rất tốt" (*very good*) $\boldsymbol{T_0}$ này có khác với cây "thực" (*real*) $\boldsymbol{T}$ mà lẽ ra được tạo ra bằng cách sử dụng tất cả dữ liệu hay không. Nó tinh chỉnh $\boldsymbol{T_0}$ để đến được $\boldsymbol{T}$.
  - Chỉ yêu cầu hai lần quét D. Trong khi các thuật toán cây quyết định truyền thống yêu cầu một lần quét cho mỗi cấp độ cây.
  - *BOAT* nhanh hơn *RainForest* từ 2-3 lần trong khi xây dựng chính xác cùng một cây.
  - Có thể được sử dụng để cập nhật dần dần. Nghĩa là, *BOAT* có thể thực hiện các thao tác chèn và xóa mới đối với dữ liệu huấn luyện và cập nhật cây quyết định để phản ánh những thay đổi này mà không cần phải xây dựng lại cây từ đầu.

2. CÂY QUYẾT ĐỊNH *(Decision Tree)*

- Cây quyết định *(Decision Tree Induction)*
- Các thước đo hỗ trợ việc lựa chọn thuộc tính *(Attribute Selection Measures)*
- Cắt tỉa cây *(Tree Pruning)*
- Khả năng mở rộng của cây quyết định *(Scalability and Decision Tree Induction)*
- Trực quan hóa quá trình tạo cây quyết định *(Visual Mining for Decision Tree Induction)*

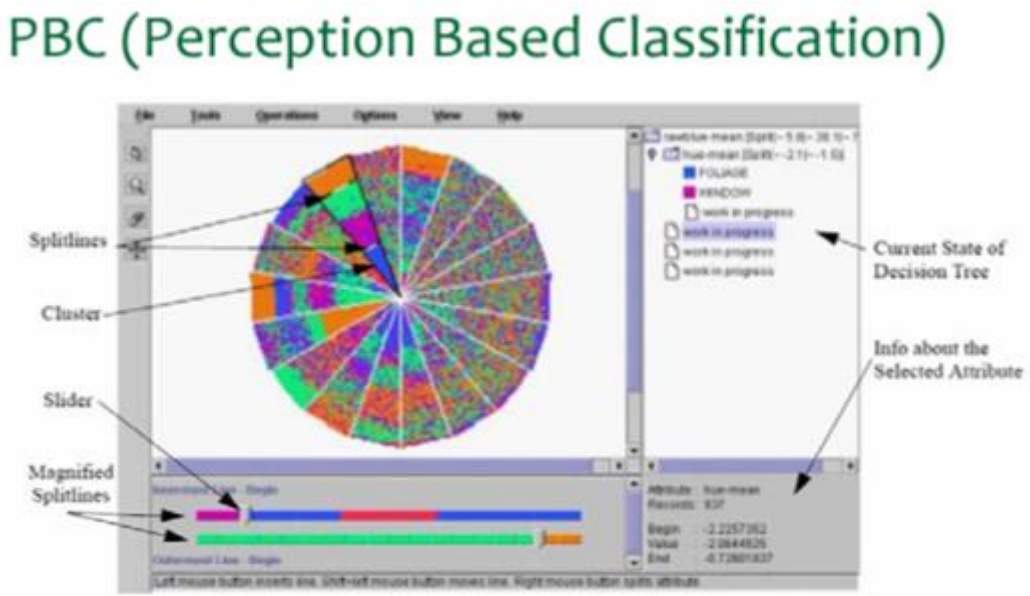
2.5. Trực quan hóa quá trình tạo cây quyết định (Visual Mining for Decision Tree Induction)

- Có cách tiếp cận tương tác nào đối với việc tạo ra cây quyết định cho phép ta trực quan hóa dữ liệu và cây khi nó đang được xây dựng không?
- Phân loại dựa trên nhận thức (*Perception-based classification -PBC*) là một cách tiếp cận tương tác dựa trên kỹ thuật trực quan đa chiều (*multidimensional visualization techniques*) và cho phép người dùng kết hợp kiến thức nền tảng về dữ liệu khi xây dựng cây quyết định.
- Bằng cách tương tác trực quan với dữ liệu, người dùng cũng có thể hiểu sâu hơn về dữ liệu. Cây kết quả có xu hướng nhỏ hơn so với cây được xây dựng bằng phương pháp quy nạp truyền thống và do đó dễ diễn giải hơn trong khi đạt được độ chính xác tương tự.

2.5. Trực quan hóa quá trình tạo cây quyết định (Visual Mining for Decision Tree Induction)

- PBC sử dụng cách tiếp cận hướng pixel để xem dữ liệu đa chiều với thông tin nhãn lớp của nó. Cách tiếp cận các phân đoạn vòng tròn được điều chỉnh, ánh xạ các đối tượng dữ liệu ***d*** chiều vào một vòng tròn được phân chia thành ***d*** phân đoạn, mỗi phân đoạn đại diện cho một thuộc tính. Ở đây, giá trị thuộc tính của đối tượng dữ liệu được ánh xạ tới một pixel màu, phản ánh nhãn lớp của đối tượng. Việc ánh xạ này được thực hiện cho từng cặp thuộc tính-giá trị của từng đối tượng dữ liệu. Việc sắp xếp được thực hiện cho từng thuộc tính để xác định thứ tự sắp xếp trong một phân đoạn.

- Ví dụ: các giá trị thuộc tính trong một phân đoạn nhất định có thể được tổ chức để hiển thị các vùng đồng nhất (đối với nhãn lớp) trong cùng một giá trị thuộc tính. Lượng dữ liệu huấn luyện có thể được hiển thị cùng một lúc được xác định gần đúng bằng tích của số thuộc tính và số lượng đối tượng dữ liệu.



NỘI DUNG CHƯƠNG 5

1. Một số khái niệm cơ bản
2. Cây quyết định (*Decision Tree*)
3. Phương pháp phân loại Bayes (*Bayes Classification Methods*)
4. Phân loại dựa trên quy tắc (*Rule-Based Classification*)
5. Đánh giá và lựa chọn mô hình (*Model Evaluation and Selection*)
6. Kỹ thuật cải thiện độ chính xác của phân loại (*Techniques to Improve Classification Accuracy*)
7. Bài tập

3. PHƯƠNG PHÁP PHÂN LOẠI BAYES (*Bayes Classification Methods*)

Bộ phân loại Bayes là phân loại thống kê. Phân loại Bayes có thể dự đoán xác suất một bộ dữ liệu nhất định thuộc về một lớp cụ thể.

3.1. Định lý Bayes *(Bayes' Theorem)*

- Định lý Bayes được đặt theo tên của Thomas Bayes, một mục sư người Anh, người đã nghiên cứu sớm về lý thuyết xác suất và quyết định (*probability and decision*) trong thế kỷ 18.
- Cho ***X*** là một bộ dữ liệu. Theo thuật ngữ Bayesian, ***X*** được coi là "bằng chứng" (*evidence*). Như thường lệ, nó được mô tả bằng các phép đo được thực hiện trên một tập hợp n thuộc tính. Giả sử ***H*** là một giả thuyết nào đó sao cho bộ dữ liệu ***X*** thuộc về một lớp ***C*** cụ thể. Đối với các bài toán phân loại, ta muốn xác định ***P(H|X)***, xác suất mà giả thuyết ***H*** đúng với "bằng chứng" hoặc bộ dữ liệu được quan sát ***X***. Nói cách khác, ta đang **tìm xác suất để bộ *X* thuộc về lớp *C***, với điều kiện là ta biết mô tả thuộc tính của ***X***.

3.1. Định lý Bayes *(Bayes' Theorem)*

- ***Một số khái niệm:***

  - *P(H|**X**)* là xác suất hậu nghiệm (*posteriori probability*) hoặc xác suất hậu nghiệm của *H* tùy thuộc vào *X*.

    Ví dụ: giả sử bộ dữ liệu được giới hạn cho các khách hàng được mô tả bởi các thuộc tính *age* và *income*, và *X* là một Khách hàng 35 tuổi có thu nhập 40.000 USD. Giả sử *H* là giả thuyết cho rằng khách hàng của ta sẽ mua một chiếc máy tính. Khi đó *P(H|**X**)* phản ánh xác suất khách hàng ***X*** sẽ mua máy tính khi biết độ tuổi và thu nhập của khách hàng.

  - *P(H)* là xác suất tiên nghiệm (*prior probability*) hoặc xác suất tiên nghiệm của *H*.

    Ví dụ: là xác suất mà bất kỳ khách hàng nào sẽ mua máy tính, bất kể tuổi tác, thu nhập hoặc bất kỳ thông tin nào khác. Xác suất *P(H|**X**)*, dựa trên nhiều thông tin hơn (ví dụ: thông tin khách hàng) so với xác suất tiên nghiệm, *P(H)*, độc lập với ***X***.

3.1. Định lý Bayes *(Bayes' Theorem)*

- ***Một số khái niệm:***
  - *P(X|H)* là xác suất hậu nghiệm của ***X*** dựa trên *H*.

    Ví dụ: là xác suất để một khách hàng ***X***, 35 tuổi và kiếm được 40.000 USD, với điều kiện là ta biết khách hàng đó sẽ mua một chiếc máy tính.
  - *P(X)* là xác suất tiên nghiệm của *X*.

    Ví dụ: cho biết xác suất khách hàng có thể mua máy tính khi khách hàng đó 35 tuổi và kiếm được 40.000 USD.
- *P(H), P(**X**|H)* và *P(**X**)* có thể được ước tính từ dữ liệu đã cho. Định lý *Bayes* hữu ích ở chỗ nó cung cấp cách tính xác suất hậu nghiệm *P(H|**X**)*, từ *P(H), P(**X**|H)* và *P(**X**)*.
- Định lý Bayes tính xác suất hậu nghiệm *P(H|**X**):*

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)\,P(H)}{P(\mathbf{X})}$$

3.2. Trình phân loại Naïve Bayesian *(Naïve Bayesian Classification)*

Trình phân loại Naïve Bayesian hoặc trình phân loại Bayesian đơn giản (*simple Bayesian classifier*), hoạt động như sau:

i. Cho D là tập huấn luyện gồm các bộ dữ liệu và nhãn lớp liên quan của chúng. Như thường lệ, mỗi bộ dữ liệu được biểu diễn bằng một vectơ thuộc tính *n* chiều, $X = (x_1, x_2, ..., x_n)$, mô tả *n* phép đo được thực hiện trên bộ dữ liệu từ *n* thuộc tính tương ứng là $A_1, A_2, ..., A_n$.

ii. Giả sử có ***m*** lớp $C_1, C_2, ..., C_m$. Cho một bộ dữ liệu ***X***, bộ phân loại sẽ dự đoán rằng ***X*** thuộc về lớp có xác suất hậu nghiệm cao nhất, dựa vào ***X***. Nghĩa là, bộ phân loại Naïve Bayes dự đoán rằng bộ dữ liệu ***X*** thuộc về lớp C_i nếu và chỉ khi

$$P(C_i|X) > P(C_j|X) \qquad \text{for } 1 \le j \le m,\ j \ne i.$$

Vì vậy, ta tối đa hóa $P(C_i|X)$. Lớp C_i mà $P(Ci|X)$ được tối đa hóa được gọi là giả thuyết hậu nghiệm cực đại (*maximum posteriori hypothesis*). Theo định lý Bayes

$$P(C_i|X) = \frac{P(X|C_i)\,P(C_i)}{P(X)}$$

3.2. Trình phân loại Naïve Bayesian *(Naïve Bayesian Classification)*

iii. Vì $P(X)$ là hằng số cho tất cả các lớp nên chỉ $P(X|C_i)P(C_i)$ cần được tối đa hóa. Nếu xác suất trước của lớp không được biết thì người ta thường giả định rằng các lớp có khả năng như nhau, nghĩa là $P(C_1) = P(C_2) = \cdots = P(C_m)$, và do đó ta sẽ tối đa hóa $P(X|C_i)$. Ngược lại, thực hiện tối đa hóa $P(X|C_i)P(C_i)$. Lưu ý rằng xác suất trước của lớp có thể được ước tính bởi $P(C_i) = |C_i,D|/|D|$, trong đó $|C_i,D|$ là số bộ dữ liệu huấn luyện của lớp C_i trong D.

iv. Với các tập dữ liệu có nhiều thuộc tính, việc tính toán $P(\mathbf{X}|Ci)$ sẽ cực kỳ tốn kém về mặt tính toán. Để giảm tính toán khi đánh giá $P(\mathbf{X}|C_i)$, giả định về tính độc lập có điều kiện của lớp (*class-conditional independence*) được đưa ra. Điều này giả định rằng các giá trị của các thuộc tính là độc lập có điều kiện (*conditionally independent*) với nhau, dựa trên nhãn lớp của bộ dữ liệu (nghĩa là không có mối quan hệ phụ thuộc giữa các thuộc tính). Như vậy,

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \dots P(x_n|C_i)$$

3.2. Trình phân loại Naïve Bayesian *(Naïve Bayesian Classification)*

iv.

Đối với mỗi thuộc tính, cần xem xét liệu thuộc tính đó có giá trị phân loại hay liên tục. Ví dụ, để tính $P(X|C_i)$, cần xem xét các điều sau:

a) *Nếu A_k là phân loại*: $P(x_k|C_i)$ là số bộ dữ liệu thuộc lớp C_i trong D có giá trị x_k cho A_k, chia cho $|C_i,D|$, số bộ dữ liệu thuộc lớp C_i trong D.

b) *Nếu A_k có giá trị liên tục*: Một thuộc tính có giá trị liên tục thường được giả định là có phân bố Gaussian với giá trị trung bình μ và độ lệch chuẩn σ, được xác định bởi

$$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

để có thể

$$P(X|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Ví dụ: đặt X = (35, 40.000 USD), trong đó *A1* và *A2* lần lượt là các thuộc tính age và income. Hãy để thuộc tính nhãn lớp là *buys_computer*. Nhãn lớp liên quan cho ***X*** là ***yes*** (tức là *buys_computer* = ***yes***). Giả sử rằng *age* chưa được rời rạc hóa và do đó tồn tại dưới dạng thuộc tính có giá trị liên tục. Giả sử từ tập huấn luyện, ta thấy rằng khách hàng ở D mua máy tính ở độ tuổi nằm trong độ tuổi *38±12*. Nói cách khác, đối với thuộc tính age và lớp này, ta có $\mu = 38$ và $\sigma = 12$. Có thể thế các đại lượng này, cùng với $x_1 = 35$ cho bộ *X*, vào công thức (13) để ước tính P(age = 35| *buys_computer* = ***yes***).

3.2. Trình phân loại Naïve Bayesian *(Naïve Bayesian Classification)*

v. Để dự đoán nhãn lớp của $\boldsymbol{X}$, $P(\boldsymbol{X}|C_i)P(C_i)$ được đánh giá cho mỗi lớp C_i. Bộ phân loại dự đoán rằng nhãn lớp của bộ $\boldsymbol{X}$ là lớp C_i khi và chỉ khi

$$P(\boldsymbol{X}|C_i)P(C_i) > P(\boldsymbol{X}|C_j)P(C_j) \qquad \text{for } 1 \le j \le m, j \ne i$$

Nói cách khác, nhãn lớp được dự đoán là lớp C_i mà $P(\boldsymbol{X}|C_i)P(C_i)$ là lớn nhất

3.2. Trình phân loại Naïve Bayesian

- Ví dụ: Cho dữ liệu huấn luyện sau:
  - Đặt *C1* tương ứng với lớp *buys_computer* = ***yes*** và *C2* tương ứng với lớp *buys_computer* = ***no***. Bộ dữ liệu muốn phân loại là:

RID	*age*	*income*	*student*	*credit-rating*	*class: buys_computer*
1.	youth	high	no	fair	no
2.	youth	high	no	excellent	no
3.	middle_aged	high	no	fair	yes
4.	senior	medium	no	fair	yes
5.	senior	low	yes	fair	yes
6.	senior	low	yes	excellent	no
7.	middle_aged	low	yes	excellent	yes
8.	youth	medium	no	fair	no
9.	youth	low	yes	fair	yes
10.	senior	medium	yes	fair	yes
11.	youth	medium	yes	excellent	yes
12.	middle_aged	medium	no	excellent	yes
13.	middle_aged	high	yes	fair	yes
14.	senior	medium	no	excellent	no

$$\boldsymbol{X} = (age = youth, income = medium, student = yes, credit\_rating = fair)$$

- Cần tối đa hóa $P(\boldsymbol{X}|C_i) \times P(C_i)$, *với* $i = 1, 2$. $P(C_i)$, xác suất trước đó của mỗi lớp, có thể được tính toán dựa trên các bộ dữ liệu huấn luyện:

$$P(buys\_computer = yes) = 9/14 = 0.643$$

$$P(buys\_computer = no) = 5/14 = 0.357$$

3.2. Trình phân loại Naïve Bayesian

- Ví dụ: Để tính $P(\boldsymbol{X}|Ci)$, với $i = 1, 2$, ta tính các xác suất có điều kiện sau:

P(age = youth	buys_computer = yes)	= 2/9 = 0.222
P(age = youth	buys_computer = no)	= 3/5 = 0.600
P(income = medium	buys_computer = yes)	= 4/9 = 0.444
P(income = medium	buys_computer = no)	= 2/5 = 0.400
P(student = yes	buys_computer = yes)	= 6/9 = 0.667
P(student = yes	buys_computer = no)	= 1/5 = 0.200
P(credit_rating = fair	buys_computer = yes)	= 6/9 = 0.667
P(credit_rating = fair	buys_computer = no)	= 2/5 = 0.400

Sử dụng những xác suất này, ta có được:

$$P(X|buys\_computer = yes) = P(age = youth \mid buys\_computer = yes) \times P(income = medium \mid buys\_computer = yes) \times P(student = yes \mid buys\_computer = yes) \times P(credit\_rating = fair \mid buys\_computer = yes) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044.$$

$$P(X|buys\_computer = no) = P(age = youth \mid buys\_computer = no) \times P(income = medium \mid buys\_computer = no) \times P(student = yes \mid buys\_computer = no) \times P(credit\_rating = fair \mid buys\_computer = no) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019.$$

3.2. Trình phân loại Naïve Bayesian

- Ví dụ:
  - Tính các xác suất có điều kiện sau:

$P(age = youth \mid buys\_computer = yes) = 2/9 = 0.222$
$P(income = medium \mid buys\_computer = yes) = 4/9 = 0.444$
$P(student = yes \mid buys\_computer = yes) = 6/9 = 0.667$
$P(credit\_rating = fair \mid buys\_computer = yes) = 6/9 = 0.667$

$P(age = youth \mid buys\_computer = no) = 3/5 = 0.600$
$P(income = medium \mid buys\_computer = no) = 2/5 = 0.400$
$P(student = yes \mid buys\_computer = no) = 1/5 = 0.200$
$P(credit\_rating = fair \mid buys\_computer = no) = 2/5 = 0.400$

  - Sử dụng những xác suất này, để tính $P(\boldsymbol{X}|Ci)$, với $i = 1, 2$:

$$P(X|buys\_computer = yes) = P(age = youth \mid buys\_computer = yes) \times P(income = medium \mid buys\_computer = yes) \times P(student = yes \mid buys\_computer = yes) \times P(credit\_rating = fair \mid buys\_computer = yes)$$
$$= 0.222 \times 0.444 \times 0.667 \times 0.667 = \mathbf{0.044}$$

$$P(X|buys\_computer = no) = P(age = youth \mid buys\_computer = no) \times P(income = medium \mid buys\_computer = no) \times P(student = yes \mid buys\_computer = no) \times P(credit\_rating = fair \mid buys\_computer = no)$$
$$= 0.600 \times 0.400 \times 0.200 \times 0.400 = \mathbf{0.019}$$

  - Để tìm lớp *Ci* làm cực đại hóa $P(X|Ci) \times P(Ci)$, ta tính:

$$P(\boldsymbol{X}|buys\_computer = yes) \times P(buys\_computer = yes) = 0.044 \times 0.643 = 0.028$$
$$P(\boldsymbol{X}|buys\_computer = no) \times P(buys\_computer = no) = 0.019 \times 0.357 = 0.007$$

⇒ Do đó, trình phân loại naïve Bayesian dự đoán ***buys_computer = yes*** cho bộ ***X***.
(với $\boldsymbol{X}$ = (*age = youth*, *income = medium*, *student = yes*, *credit_rating = fair*))

3.2. Trình phân loại Naïve Bayesian

- ***Lưu ý: khi xác suất có giá trị = 0***
  - Có thể giả sử rằng CSDL huấn luyện của ta, D, lớn đến mức việc thêm 1 vào mỗi số đếm sẽ chỉ tạo ra sự khác biệt không đáng kể trong giá trị xác suất ước tính (*estimated probability value*), nhưng sẽ tránh được trường hợp giá trị xác suất bằng 0.
  - Kỹ thuật ước tính xác suất này được gọi là phép hiệu chỉnh *Laplacian* (*Laplacian correction*) hay công cụ ước tính *Laplace* (*Laplace estimator*), được đặt theo tên của nhà toán học người Pháp *Pierre Laplace (1749-1827)*.

3.2. Trình phân loại Naïve Bayesian

- ***Lưu ý: khi xác suất có giá trị = 0***
  - Ví dụ: Giả sử rằng đối với lớp *buys_computer = yes* trong CSDL đào tạo D_X nào đó, chứa 1000 bộ giá trị, ta có 0 bộ giá trị có *income=low*, 990 bộ giá trị có *income=medium* và 10 bộ giá trị có *income=high*.
    - Xác suất của những sự kiện này, không tính hiệu chỉnh Laplacian, lần lượt là 0 (= 0/1000), 0.990 (= 990/1000) và 0.010 (= 10/1000).

P(income = low | buys_computer = yes) = 0/1000 = 0.000
P(income = medium | buys_computer = yes) = 990/1000 = 0.990
P(income = high | buys_computer = yes) = 10/1000 = 0.010

    - Sử dụng hiệu chỉnh *Laplacian* cho ba đại lượng, ta có thêm 1 bộ cho mỗi cặp giá trị thu nhập. Bằng cách này, thay vào đó, ta thu được các xác suất sau:

P(income = low | buys_computer = yes) = 1/1003 = 0.001
P(income = medium | buys_computer = yes) = 991/1003 = 0.988
P(income = high | buys_computer = yes) = 11/1003 = 0.011

NỘI DUNG CHƯƠNG 5

1. Một số khái niệm cơ bản
2. Cây quyết định (*Decision Tree*)
3. Phương pháp phân loại Bayes (*Bayes Classification Methods*)
4. Phân loại dựa trên quy tắc (*Rule-Based Classification*)
5. Đánh giá và lựa chọn mô hình (*Model Evaluation and Selection*)
6. Kỹ thuật cải thiện độ chính xác của phân loại (*Techniques to Improve Classification Accuracy*)
7. Bài tập

4. PHÂN LOẠI DỰA TRÊN QUY TẮC *(Rule-Based Classification)*

4.1. Sử dụng các quy tắc IF-THEN cho việc phân loại *(Using IF-THEN Rules for Classification)*

- ***Trình phân loại dựa trên quy tắc*** (*rule-based classifier*) sử dụng một bộ quy tắc *IF-THEN* để phân loại.
  - Quy tắc IF-THEN là một biểu thức có dạng: ***IF*** *condition expression* ***THEN*** *conclusion*

    Trong đó:
    - *IF conditional expression*: của quy tắc được gọi là tiền đề hoặc điều kiện tiên quyết của quy tắc. Trong quy tắc tiền đề, điều kiện có thể bao gồm một hoặc nhiều điều kiện để kiểm tra các thuộc tính được nối với nhau qua các toán tử AND/OR/NOT hoặc sự kết hợp của các toán tử này.
    - *THEN conclusion*: là hệ quả của quy tắc.
  - Ví dụ quy tắc R_1 về dự đoán liệu khách hàng có mua máy tính hay không có thể được viết bằng 1 trong 2 cách sau:
    - *Cách 1:* R_1: *IF age = youth AND student = yes THEN buys_computer = yes*
    - *Cách 2:* R_1: *IF age = youth* $\wedge$ *student = yes* $\Rightarrow$ *buys_computer = yes.*

4.1. Sử dụng các quy tắc IF-THEN cho việc phân loại *(Using IF-THEN Rules for Classification)*

- ***Trình phân loại dựa trên quy tắc***
  - Khi điều kiện (*conditional expression*) trong một tiền đề của quy tắc đúng với một bộ dữ liệu nhất định thì ta nói rằng tiền đề của quy tắc đó được thỏa mãn và quy tắc đó bao trùm bộ dữ liệu đó.
  - Một quy tắc R có thể được đánh giá bằng mức độ bao phủ (*coverage*) và độ chính xác (*accuracy*) của nó. Cho một bộ giá trị X từ tập dữ liệu được dán nhãn lớp D. Có thể định nghĩa phạm vi bao phủ -*coverage (R)*- và độ chính xác - *accuracy (R)*- của R là:

$$coverage\ (R) = \frac{n_{covers}}{|D|}$$

$$accuracy\ (R) = \frac{n_{correct}}{n_{covers}}$$

trong đó:

- n_{covers} là số bộ giá trị được bao phủ bởi *R*;
- $n_{correct}$ là số lượng bộ dữ liệu được phân loại chính xác bởi R;
- |D| là số lượng bộ dữ liệu trong D.

4.1. Sử dụng các quy tắc IF-THEN cho việc phân loại *(Using IF-THEN Rules for Classification)*

- ***Trình phân loại dựa trên quy tắc***

  - Ví dụ: Trở lại dữ liệu trong Bảng 1. Với *R_1: IF age = youth AND student = yes THEN buys_computer = yes.* Nhiệm vụ của ta là dựa trên R_1 để dự đoán liệu khách hàng có mua máy tính hay không.

Chỉ có 2 dòng trong 14 d
dữ liệu thỏa điều kiện của *IF*. Tính *coverage (R1)* và accuracy *(R1)*

$$coverage\ (R) = \frac{n_{covers}}{|D|} = \frac{2}{14} = 14.28\%$$

$$accuracy\ (R) = \frac{n_{correct}}{n_{covers}} = \frac{2}{2} = 100\%$$

RID	age	income	student	credit-rating	class: buys_computer
1.	youth	high	no	fair	no
2.	youth	high	no	excellent	no
3.	middle_aged	high	no	fair	yes
4.	senior	medium	no	fair	yes
5.	senior	low	yes	fair	yes
6.	senior	low	yes	excellent	no
7.	middle_aged	low	yes	excellent	yes
8.	youth	medium	no	fair	no
9.	youth	low	yes	fair	yes
10.	senior	medium	yes	fair	yes
11.	youth	medium	yes	excellent	yes
12.	middle_aged	medium	no	excellent	yes
13.	middle_aged	high	yes	fair	yes
14.	senior	medium	no	excellent	no

Ta muốn phân loại ***X*** theo *buys_computer*. Do X thỏa mãn R_1, điều này kích hoạt quy tắc.

4.1. Sử dụng các quy tắc IF-THEN cho việc phân loại *(Using IF-THEN Rules for Classification)*

- ***Các trường hợp đặc biệt***:
  - Có nhiều quy tắc được kích hoạt?
  - Không có quy tắc nào được X thỏa mãn?

i. Trường hợp (i): Nếu có nhiều quy tắc được kích hoạt, cần một chiến lược giải quyết xung đột (*a conflict resolution strategy*) để tìm ra quy tắc nào sẽ kích hoạt và chỉ định dự đoán lớp của nó cho X. Có nhiều chiến lược khả thi. Ở đây, ta xem xét hai chiến lược là theo *Kích thước của điều kiện tiền đề* (*size ordering*) và theo *Độ ưu tiên của quy tắc* (*rule ordering*).

- ***Size ordering*** chỉ định mức ưu tiên cao nhất cho quy tắc kích hoạt có yêu cầu "khó khăn nhất" (*toughest requirements*), trong đó "độ khó" (*toughness*) được đo bằng kích thước tiền đề của quy tắc. Nghĩa là, quy tắc kích hoạt có nhiều thử nghiệm thuộc tính nhất sẽ được kích hoạt.

4.1. Sử dụng các quy tắc IF-THEN cho việc phân loại *(Using IF-THEN Rules for Classification)*

- ***Các trường hợp đặc biệt***:

  i. ***Trường hợp (i)***: Có nhiều quy tắc được kích hoạt

  - ***Rule ordering*** ưu tiên các quy tắc trước. Thứ tự có thể dựa trên lớp (*class-based ordering*) hoặc dựa trên quy tắc (*rule-based ordering*).
    - *class-based ordering:* các lớp được sắp xếp theo thứ tự giảm dần về "*tầm quan trọng*" (*importance*) chẳng hạn như:
      - Sắp xếp theo thứ tự mức độ phổ biến giảm dần (*order of decreasing importance*): Nghĩa là, tất cả các quy tắc cho loại phổ biến nhất (hoặc thường xuyên nhất) sẽ được ưu tiên trước, các quy tắc cho loại phổ biến tiếp theo sẽ xuất hiện tiếp theo, v.v…
      - Sắp xếp dựa trên chi phí phân loại sai cho mỗi lớp (*sorted based on the misclassification cost per class*). Trong mỗi lớp, các quy tắc không được sắp xếp theo thứ tự—chúng không nhất thiết phải như vậy vì tất cả chúng đều dự đoán cùng một lớp (và do đó không thể có xung đột lớp!).

4.1. Sử dụng các quy tắc IF-THEN cho việc phân loại *(Using IF-THEN Rules for Classification)*

- ***Các trường hợp đặc biệt***:
  - ***i. Trường hợp (i)***: Có nhiều quy tắc được kích hoạt
    - ***Rule ordering*** ưu tiên các quy tắc trước. Thứ tự có thể dựa trên lớp (*class-based ordering*) hoặc dựa trên quy tắc (*rule-based ordering*).
      - *rule-based ordering:* các quy tắc được tổ chức thành một danh sách ưu tiên dài, theo một số thước đo về chất lượng quy tắc, chẳng hạn như độ chính xác, phạm vi hoặc kích thước (số lượng kiểm tra thuộc tính trong quy tắc trước đó) hoặc dựa trên lời khuyên từ các chuyên gia về lĩnh vực.
        - Khi sử dụng thứ tự quy tắc, bộ quy tắc được gọi là danh sách quyết định (*decisionlist*).
        - Với thứ tự quy tắc, quy tắc kích hoạt xuất hiện sớm nhất trong danh sách có mức độ ưu tiên cao nhất và do đó nó sẽ đưa ra dự đoán lớp của mình.
        - Bất kỳ quy tắc nào khác thỏa mãn X đều bị bỏ qua.

        Hầu hết các hệ thống phân loại dựa trên quy tắc đều sử dụng chiến lược sắp xếp thứ tự quy tắc dựa trên lớp.

4.1. Sử dụng các quy tắc IF-THEN cho việc phân loại *(Using IF-THEN Rules for Classification)*

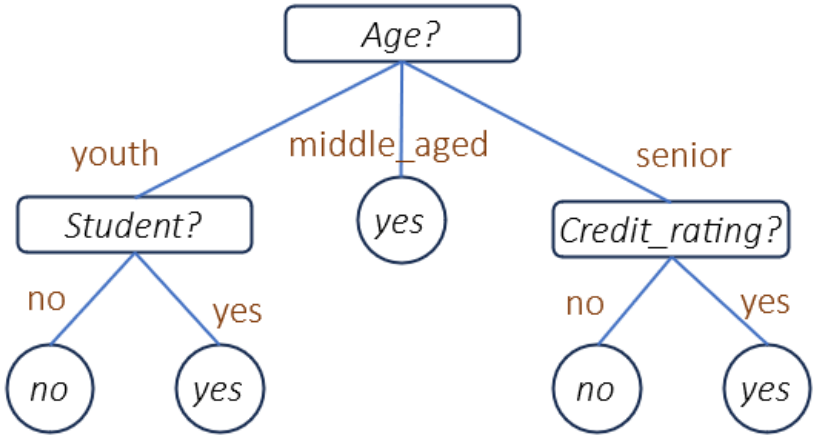
- ***Các trường hợp đặc biệt***:

  ii. Trường hợp (ii): Không có quy tắc nào được X thỏa mãn

  - Vậy làm thế nào ta có thể xác định nhãn lớp của X?
  - Trong trường hợp này, quy tắc dự phòng (*fallback rule*) hoặc quy tắc mặc định (*default rule*) có thể được thiết lập để chỉ định lớp mặc định, dựa trên tập huấn luyện. Đây có thể là lớp đa số hoặc lớp đa số của các bộ dữ liệu không nằm trong bất kỳ quy tắc nào. Quy tắc mặc định được đánh giá ở cuối, khi và chỉ khi không có quy tắc nào khác bao trùm ***X***.

4.2. Trích xuất quy tắc từ cây quyết định *(Rule Extraction from a Decision Tree)*

- Để trích xuất các quy tắc từ cây quyết định, một quy tắc được tạo cho mỗi đường dẫn từ nút gốc đến nút lá. Mỗi tiêu chí phân tách dọc theo một đường dẫn nhất định được kết nối với nhau bởi toán tử AND để tạo thành phần tiền đề của quy tắc ("*IF*"). Nút lá giữ dự đoán lớp, hình thành nên hệ quả của quy tắc ("phần *THEN*").
- Ví dụ: Trích xuất các quy tắc phân loại từ cây quyết định.



Các quy tắc được trích từ cây như sau:

R1: IF *age = youth*	AND *student = no*	THEN *buys_computer = no*
R2: IF *age = youth*	AND *student = yes*	THEN *buys_computer = yes*
R3: IF *age = middle_aged*		THEN *buys_computer = yes*
R4: IF *age = senior*	AND *credit_rating = excellent*	THEN *buys_computer = yes*
R5: IF *age = senior*	AND *creditrating = fair*	THEN *buys_computer = no*

Các quy tắc trên được ngụ ý sẽ nối với nhau qua toán tử **OR**. Bởi vì các quy tắc được trích xuất trực tiếp từ cây nên chúng loại trừ lẫn nhau (*mutually exclusive*) và đầy đủ (*exhaustive*).

4.3. Quy nạp quy tắc bằng cách sử dụng thuật toán bao phủ tuần tự

4.3.1. Giải thuật Learn_One_Rule

- Các quy tắc *IF-THEN* có thể được trích xuất trực tiếp từ dữ liệu huấn luyện (tức là không cần phải tạo cây quyết định trước) bằng thuật toán bao phủ tuần tự (*sequential covering algorithm*).
- Có nhiều thuật toán bao phủ tuần tự như *AQ, CN2, RIPPER, …*
- Chiến lược chung của các thuật toán này như sau: Các quy tắc được học lần lượt. Mỗi lần học một quy tắc, các bộ chứa quy tắc đó sẽ bị xóa và quá trình lặp lại trên các bộ còn lại. Quá trình tiếp tục cho đến khi đáp ứng điều kiện kết thúc, chẳng hạn như khi không còn bộ dữ liệu huấn luyện nào nữa hoặc chất lượng của quy tắc được trả về thấp hơn ngưỡng do người dùng chỉ định.
- Việc học các quy tắc tuần tự này trái ngược với việc quy nạp cây quyết định. Bởi vì đường dẫn đến mỗi lá trong cây quyết định tương ứng với một quy tắc, nên ta có thể coi việc quy nạp cây quyết định là việc học một tập quy tắc cùng một lúc.

4.3. Quy nạp quy tắc bằng cách sử dụng thuật toán bao phủ tuần tự

4.3.1. Giải thuật Learn_One_Rule

- Học các quy tắc: Thông thường, các quy tắc được phát triển theo cách từ tổng quát đến cụ thể (*general-to-specific*). Có thể coi đây là một cách tìm kiếm chùm (*beam search*), trong đó bắt đầu với một quy tắc trống và sau đó dần dần tiếp tục thêm các thử nghiệm thuộc tính vào đó. Ta nối thêm bằng cách thêm thuộc tính test dưới dạng liên kết logic với điều kiện hiện có của quy tắc tiền đề.
- Ví dụ: cho tập huấn luyện D là dữ liệu về ứng dụng cho vay.
  - Các thuộc tính liên quan đến mỗi người nộp đơn gồm:
    - Tuổi tác (*age*)
    - Thu nhập (*income*)
    - Trình độ học vấn (*education level*)
    - Nơi cư trú (*residence*)
    - Xếp hạng tín dụng (*credit rating*)
    - Thời hạn của khoản vay (*term of the loan*).
  - Thuộc tính phân loại là quyết định cho vay (*loan_decision*), cho biết liệu khoản vay được chấp nhận (được coi là an toàn - *safe*) hay bị từ chối (được coi là rủi ro - *risky*).

4.3. Quy nạp quy tắc bằng cách sử dụng thuật toán bao phủ tuần tự

4.3.1. Giải thuật Learn_One_Rule

- Ví dụ: (tiếp theo)
  - Để tìm hiểu một quy tắc cho lớp "chấp nhận" (*accept*), ta bắt đầu với quy tắc chung nhất có thể, đó là điều kiện của quy tắc tiền đề là trống. Quy tắc là

IF ___ THEN loan_decision = accept

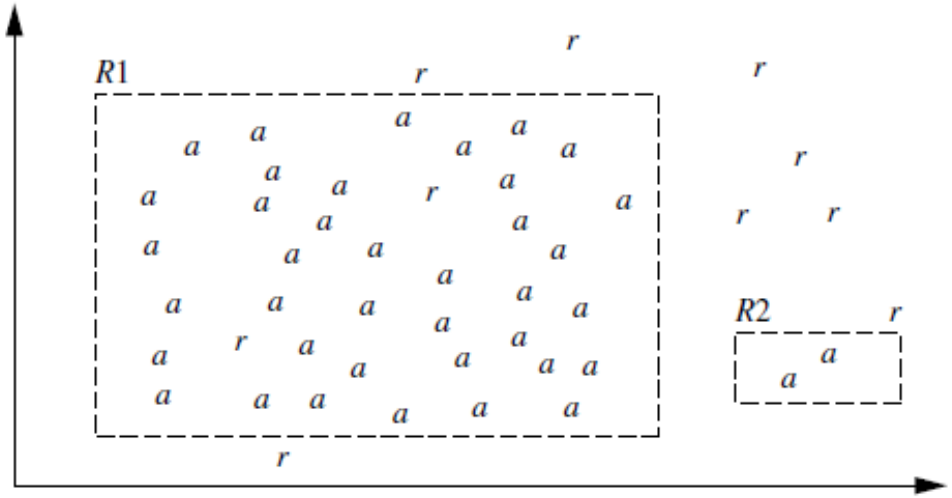
  - Giải thuật *Learn_One_Rule* áp dụng chiến lược tham lam theo chiều sâu (*greedy depth-first*). Mỗi lần sẽ thêm một thử nghiệm thuộc tính mới (kết hợp) vào quy tắc hiện tại, nó sẽ chọn một thử nghiệm thuộc tính cải thiện nhất chất lượng quy tắc, dựa trên các mẫu đào tạo. Giả sử ta sử dụng độ chính xác của quy tắc làm thước đo chất lượng. Quay lại ví dụ, giả sử *Learn_One_Rule* nhận thấy rằng thuộc tính kiểm tra *income = high* cải thiện tốt nhất độ chính xác của quy tắc (đang trống) hiện tại. Giải thuật này sẽ thêm nó vào điều kiện để quy tắc hiện tại trở thành

IF income = high THEN loan_decision = accept

4.3.2. Thước đo chất lượng của quy tắc trong giải thuật Learn_One_Rule

- Giải thuật *Learn_One_Rule* cần thước đo chất lượng của quy tắc. Mỗi lần giải thuật này xem xét một thử nghiệm thuộc tính, giải thuật phải kiểm tra xem liệu việc thêm một thử nghiệm như vậy vào điều kiện của quy tắc hiện tại có dẫn đến một quy tắc được cải thiện hay không?
- Ví dụ về *Lựa chọn giữa hai quy tắc dựa trên độ chính xác*. Xem xét hai quy tắc như được minh họa trong hình sau.

Cả hai đều để cho lớp quyết định *loan_decision = accept*. Sử dụng "*a*" để biểu thị các bộ dữ liệu của lớp "*accept*" và "*r*" cho các bộ dữ liệu của lớp "*reject*". Quy tắc R_1 phân loại chính xác 38 trong số 40 bộ dữ liệu có trong nó. Quy tắc R_2 chỉ bao gồm hai bộ dữ liệu được phân loại chính xác. Độ chính xác tương ứng của chúng là 95% và 100%. Do đó, *R2* có độ chính xác cao hơn *R1*, nhưng đó không phải là quy tắc tốt hơn vì phạm vi bao phủ nhỏ.



4.3.2. Thước đo chất lượng của quy tắc trong giải thuật Learn_One_Rule

- ***Entropy***: qua ví dụ, ta thấy:
  - Bản thân độ chính xác không phải là ước tính đáng tin cậy về chất lượng của quy tắc.
  - Bản thân phạm vi bao phủ cũng không chắc sẽ hữu ích đối với một lớp nhất định, khi có thể có một quy tắc bao gồm nhiều bộ dữ liệu, hầu hết trong số đó thuộc về các lớp khác!

$\Rightarrow$Vì vậy, ta cần tìm kiếm các biện pháp khác để đánh giá chất lượng quy tắc, có thể tích hợp các khía cạnh về độ chính xác và phạm vi bao phủ.

Ở đây sẽ xem xét về *entropy*, một loại khác dựa trên thông tin thu được và một thử nghiệm thống kê xem xét phạm vi bao phủ. Giả sử đang học các quy tắc cho lớp *c*. Quy tắc hiện tại là *R: IF condition THEN class = c*. Ta muốn xem liệu việc thực hiện phép *AND* một thuộc tính với *condition* hiện tại có dẫn đến một quy tắc tốt hơn hay không. Gọi điều kiện mới là $condition_0$, trong đó R_0: *IF* $condition_0$ *THEN class = c* là quy tắc mới tiềm năng. Nói cách khác, ta muốn xem liệu R_0 có tốt hơn *R* hay không.

4.3.2. Thước đo chất lượng của quy tắc trong giải thuật Learn_One_Rule

- ***Entropy***:
  - Đã được giới thiệu trong phần về thước đo độ lợi thông tin (*information gain*) được sử dụng để lựa chọn thuộc tính trong quy nạp cây quyết định.
  - *Entropy* còn được gọi là thông tin dự kiến cần thiết (*expected information needed*) để phân loại một bộ dữ liệu trong tập dữ liệu *D*. Ở đây, *D* là tập hợp các bộ dữ liệu được bao phủ bởi $condition_0$ và p_i là xác suất của lớp C_i trong *D*. *Entropy* càng thấp thì $condition_0$ càng tốt. *Entropy* ưu tiên các điều kiện bao gồm một số lượng lớn các bộ dữ liệu của một lớp và một số bộ dữ liệu của các lớp khác.

4.3.2. Thước đo chất lượng của quy tắc trong giải thuật Learn_One_Rule

- ***Thuật toán bao phủ tuần tự học các quy tắc logic bậc nhất***
  - Một biện pháp khác dựa trên việc thu được thông tin và đã được đề xuất trong *FOIL* (*First Order Inductive Learner*), một thuật toán bao phủ tuần tự (*sequential covering algorithm*) học các quy tắc logic bậc nhất (*learn first-order logic rules*).
  - Việc học các quy tắc bậc nhất phức tạp hơn vì các quy tắc như vậy chứa các biến (*variables*), trong khi các quy tắc ta quan tâm trong phần này là mệnh đề (*propositional* - tức là không có biến).

4.3.2. Thước đo chất lượng của quy tắc trong giải thuật Learn_One_Rule

- ***Thuật toán bao phủ tuần tự học các quy tắc logic bậc nhất***

  - Trong machine learning, các bộ dữ liệu của lớp mà ta đang học các quy tắc được gọi là bộ dữ liệu dương (*positive tuples*), trong khi các bộ dữ liệu còn lại là bộ dữ liệu âm (*negative tuples*). Đặt *pos(neg)* là số bộ dữ liệu dương (âm) được bao phủ bởi R. Đặt pos_0 (neg_0) là số bộ dữ liệu dương (âm) được bao phủ bởi R_0. *FOIL* đánh giá thông tin thu được bằng cách mở rộng $condition_0$ như sau:

$$FOIL\_Gain = pos' \times \left(log_2 \frac{pos'}{pos' + neg'} - log_2 \frac{pos}{pos + neg} \right)$$

4.3.2. Thước đo chất lượng của quy tắc trong giải thuật Learn_One_Rule

- ***Thuật toán bao phủ tuần tự học các quy tắc logic bậc nhất***

  - Cũng có thể sử dụng kiểm tra thống kê có ý nghĩa (*statistical test of significance*) để xác định xem tác động rõ ràng của một quy tắc không phải là do ngẫu nhiên mà thay vào đó chỉ ra mối tương quan thực sự giữa các giá trị thuộc tính và các lớp. Có thể sử dụng thống kê tỷ lệ khả năng xảy ra (*likelihood ratio statistic*) như sau:

$$Likelihood\_Ratio = 2\sum_{i=1}^{m} f_i log\left(\frac{f_i}{e_i}\right)$$

trong đó:

- **m** là số lớp.
- Đối với bộ dữ liệu thỏa mãn quy tắc, f_i là tần suất quan sát được của từng lớp ***i*** trong số các bộ.
- e_i là tần suất mà ta mong đợi ở mỗi lớp nếu quy tắc đưa ra dự đoán ngẫu nhiên.

Thống kê có phân bố χ^2 với $m-1$ bậc tự do. Tỷ lệ khả năng xảy ra càng cao thì càng có nhiều khả năng có sự khác biệt đáng kể về số lượng dự đoán đúng do quy tắc của ta đưa ra với "*công cụ dự đoán ngẫu nhiên*" (*random guessor*). Nghĩa là, việc thực hiện quy tắc của ta không phải do ngẫu nhiên. Tỷ lệ này giúp xác định các quy tắc có phạm vi bao phủ không đáng kể.

4.3.3. Quy tắc cắt tỉa (Rule Pruning)

- *Learn_One_Rule*
  - Không sử dụng bộ kiểm tra khi đánh giá các quy tắc.
  - Việc đánh giá chất lượng quy tắc như được mô tả trước đây được thực hiện bằng các bộ dữ liệu từ dữ liệu huấn luyện ban đầu. Những đánh giá này mang tính chủ quan vì các quy tắc có thể sẽ phù hợp quá mức với dữ liệu. Nghĩa là, các quy tắc có thể hoạt động tốt trên dữ liệu huấn luyện nhưng kém hơn trên dữ liệu tiếp theo.
  - Để bù đắp cho điều này, ta có thể cắt bớt các quy tắc. Một quy tắc được cắt bớt bằng cách loại bỏ một liên từ (loại bỏ 1 kiểm tra thuộc tính). Ta chọn lược bớt một quy tắc R, nếu phiên bản được lược bớt của R có chất lượng cao hơn, như được đánh giá trên một tập các bộ dữ liệu độc lập.
  - Giống như việc cắt tỉa cây quyết định, ta gọi bộ này là bộ cắt tỉa (*pruning set*).
  - Có thể sử dụng nhiều chiến lược cắt tỉa khác nhau, chẳng hạn như phương pháp cắt tỉa bi quan (*pessimistic pruning approach*) được mô tả ở phần trước.

4.3.3. Quy tắc cắt tỉa (Rule Pruning)

- *FOIL:*
  - Sử dụng một phương pháp đơn giản nhưng hiệu quả. Cho một quy tắc R,

$$\text{FOIL\_Prune}(R) = \frac{pos - neg}{pos + neg}$$

trong đó pos và neg lần lượt là số bộ dữ liệu dương và âm được bao phủ bởi R. Giá trị này sẽ tăng theo độ chính xác của R trên bộ cắt tỉa. Do đó, nếu giá trị FOIL_Prune cao hơn đối với phiên bản được cắt bớt của R thì ta sẽ tỉa R.

NỘI DUNG CHƯƠNG 5

1. Một số khái niệm cơ bản
2. Cây quyết định (*Decision Tree*)
3. Phương pháp phân loại Bayes (*Bayes Classification Methods*)
4. Phân loại dựa trên quy tắc (Rule-Based Classification)
5. Đánh giá và lựa chọn mô hình (*Model Evaluation and Selection*)
6. Kỹ thuật cải thiện độ chính xác của phân loại (*Techniques to Improve Classification Accuracy*)
7. Bài tập

5. ĐÁNH GIÁ VÀ LỰA CHỌN MÔ HÌNH (*Model evaluation and selection*)

- Số liệu để đánh giá hiệu suất phân loại
  (Metrics for Evaluating Classifier Performance)
- Phương pháp Holdout và lấy mẫu con ngẫu nhiên
  (Holdout Method and Random Subsampling)
- Xác thực chéo
  (Cross-Validation)
- Phương pháp Bootstrap
- So sánh các phân loại dựa trên đường cong Chi phí-Lợi ích và ROC
  (Comparing Classifiers Based on Cost–Benefit and ROC Curves)

5.1. Số liệu để đánh giá hiệu suất phân loại *(Metrics for Evaluating Classifier Performance)*

- Các biện pháp đánh giá phân loại được giới thiệu trong phần này

Với:

- P: Positive
- N: Negative
- TN: True Negative
- FN: False Negative
- FP: False Positive
- TN: True Negative

Measure	*Formula*
Độ chính xác *(accuracy)*, tỷ lệ nhận dạng *(recognition rate)*	$\frac{TP + TN}{P+N}$
Tỷ lệ lỗi *(error rate)*, tỷ lệ phân loại sai *(misclassification rate)*	$\frac{FP + FN}{P+N}$
Độ nhạy *(sensitivity)*, tỷ lệ âm thực sự *(true positive rate)*, thu hồi *(recall)*	$\frac{TP}{P}$
Độ đặc hiệu *(specificity)*, tỷ lệ âm thực sự *(true negative rate)*	$\frac{TN}{N}$
Độ chính xác *(precision)*	$\frac{TP}{TP + FP}$
F, F1, F-score, trung bình hài hòa của độ chính xác và thu hồi *(harmonic mean of precision and recall)*	$\frac{2 X precision x recall}{precision + recall}$
F_β, trong đó β là số thực không âm *(non-negative real number)*	$\frac{(1 + \beta^2) x precision x recall}{\beta^2 x precision + recall}$

5.1. Số liệu để đánh giá hiệu suất phân loại *(Metrics for Evaluating Classifier Performance)*

- ***Một số thuật ngữ***

  - Bộ dữ liệu dương (hoặc âm):
    - Các bộ dữ liệu dương (*positive tuples* - các bộ dữ liệu của lớp quan tâm chính).
    - Các bộ dữ liệu âm (*negative tuples* – bao gồm tất cả các bộ dữ liệu khác).

      Ví dụ, với hai lớp, các bộ dữ liệu dương (*P - positive*) là *buys_computer = yes* trong khi các bộ dữ liệu âm N-*Negative*) *buys_computer = no*.

  - *Số lượng dương tính thực sự* (*True Positives - TP*): là các bộ giá trị dương đã được bộ phân loại gắn nhãn chính xác.
  - *Số âm tính thật* (*True Negatives - TN*): là các bộ dữ liệu phủ định được bộ phân loại gắn nhãn chính xác.
  - *Số kết quả dương tính giả* (*False Positives - FP*): là các bộ dữ liệu âm được gắn nhãn không chính xác là dương (ví dụ: các bộ dữ liệu của lớp *buys computer = no* mà bộ phân loại dự đoán *buys computer = yes*).
  - *Số kết quả âm tính giả* (*False Negatives - FN*): là các bộ dữ liệu dương được gắn nhãn sai là âm (ví dụ: các bộ dữ liệu của lớp *buys computer = yes* mà bộ phân loại dự đoán *buys computer = no*).

5.1. Số liệu để đánh giá hiệu suất phân loại *(Metrics for Evaluating Classifier Performance)*

- ***Một số thuật ngữ***
  - *Confusion matrix* :
    - Cho ***m*** lớp (trong đó $m \geq 2$), *confusion matrix* là một bảng có kích thước tối thiểu là ***m***
    - Mục nhập $CM_{i,j}$ trong ***m*** hàng và ***m*** cột đầu tiên cho biết số lượng bộ dữ liệu của lớp ***i*** đã được bộ phân loại gắn nhãn là lớp ***j***.

		Predicted class		
		yes	no	Cộng
Actual class	yes	TP	FN	P
	no	FP	TN	N
	Total	P'	N'	P + N

    - Để bộ phân loại có độ chính xác cao, lý tưởng nhất là hầu hết các bộ dữ liệu sẽ được biểu diễn dọc theo đường chéo của confusion matrix, từ mục $CM_{1,1}$ đến mục $CM_{m,m}$, với các mục c khoảng *0*.
    - Ví dụ:

		Predicted class (buys_computer)			Accurate Recognition (%)
		yes	no	Total	
Actual Class (buys_computer)	yes	**6954**	**46**	7.000	99.34 (=6954/7000)
	no	**412**	**2.588**	3.000	86.27 (=2588/3000)
	Total	7.366	2.634	10.000	95.42 (=(6954+2588)/10000)

5. Đánh giá và lựa chọn mô hình (*Model evaluation and selection*)

5.1. Số liệu để đánh giá hiệu suất phân loại *(Metrics for Evaluating Classifier Performance)*

- ***Một số thuật ngữ***
  - *Confusion matrix* :
    - *Độ chính xác* (*accuracy*) của bộ phân loại trên một tập kiểm tra nhất định là tỷ lệ phần trăm các bộ dữ liệu của tập kiểm tra được bộ phân loại phân loại chính xác. Công thức tính độ chính xác:

$$accuracy = \frac{TP + TN}{P + N}$$

    - *Tỷ lệ lỗi* (*error rate,* hoặc tỷ lệ phân loại sai - *misclassification rate*) của một bộ phân loại có thể được tính bằng 1 trong 2 cách:

$$error\ rate = 1 - accuracy$$

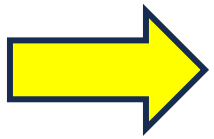
$$error\ rate = \frac{FP + FN}{P + N}$$

    - *Độ nhạy* (*sensitivity*, còn được gọi là tỷ lệ dương tính thực sự -*true positive* hay tỷ lệ nhận dạng - *recognition*): là tỷ lệ các bộ dữ liệu dương được xác định chính xác

$$sensitivity = \frac{TP}{P}$$

    - *Độ đặc hiệu (specificity)* là tỷ lệ âm tính thực sự (tức là tỷ lệ các bộ dữ liệu âm được xác định chính xác).

$$specificity = \frac{TN}{N}$$



$$accuracy = sensitivity \frac{P}{(P+N)} + specificity \frac{N}{(P+N)}$$

5.1. Số liệu để đánh giá hiệu suất phân loại *(Metrics for Evaluating Classifier Performance)*

- ***Một số thuật ngữ***
  - *Confusion matrix* :

Classes	yes	no	Total
yes	**90**	**210**	300
no	**140**	**9.560**	9.700
Total	230	9.770	10.000

Ví dụ: Cho một *confusion matrix* về dữ liệu y tế trong đó các giá trị lớp là *yes* và *no* đối với thuộc tính nhãn lớp là

Các độ

$$sensitivity = \frac{TP}{P} = \frac{90}{300} = 30{,}00\%$$

$$specificity = \frac{TN}{N} = \frac{9560}{9700} = 98{,}56\%$$

$$accuracy = \frac{TP + TN}{P + N} = \frac{90+9560}{300+9700} = 96{,}50\%$$

Do đó, cần lưu ý rằng mặc dù bộ phân loại có độ chính xác cao (98,56%) nhưng khả năng gắn nhãn chính xác cho lớp dương tính lại kém do độ nhạy thấp (30%). Nó có độ đặc hiệu cao, nghĩa là nó có thể nhận dạng chính xác các bộ dữ liệu âm.

5.1. Số liệu để đánh giá hiệu suất phân loại *(Metrics for Evaluating Classifier Performance)*

- ***Một số thuật ngữ***
  - *Độ chính xác (precision) và độ đo thu hồi (recall)* :
    - Độ chính xác có thể được coi là thước đo sự chính xác (*exactness*), nghĩa là bao nhiêu phần trăm bộ dữ liệu được gắn nhãn là dương tính thực sự.

$$precision = \frac{TP}{TP + FP}$$

    - Độ đo thu hồi là thước đo mức độ hoàn chỉnh (*completeness*) cho biết bao nhiêu phần trăm bộ dữ liệu dương được gắn nhãn như vậy.

      Độ đo thu hồi (recall) giống với độ nhạy (*sensitivity* - hoặc tỷ lệ dương tính thực sự - *positive rate*). Những thước đó này có thể được tính là TP.

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P}$$

5.1. Số liệu để đánh giá hiệu suất phân loại *(Metrics for Evaluating Classifier Performance)*

- ***Một số thuật ngữ***
  - *Độ chính xác (precision) và độ đo thu hồi (recall)* :
    - Kết hợp độ chính xác và thu hồi thành một thước đo duy nhất.
      - Độ đo *F* (còn được gọi là điểm F_1 - F_1 *score* - hoặc điểm *F - F score*) là giá trị trung bình hài hòa (*harmonic mean*) của độ chính xác và độ thu hồi. Độ đo này mang lại trọng số như nhau cho độ chính xác và thu hồi.

$$F = \frac{2 \text{ x precision x recall}}{\text{precision + recall}}$$

      - Độ đo F_β là thước đo có trọng số về độ chính xác và khả năng thu hồi. Nó ấn định trọng số gấp β lần độ chính xác. Các thước đo F_β thường được sử dụng là F_2 (có trọng số thu hồi gấp đôi độ chính xác) và $F_{0.5}$ (có trọng số chính xác cao gấp đôi so với độ thu hồi):

$$F_\beta = \frac{(1+\beta^2) \text{ x precision x recall}}{\beta^2 \text{ x precision + recall}}$$

trong đó, β là số thực không âm.

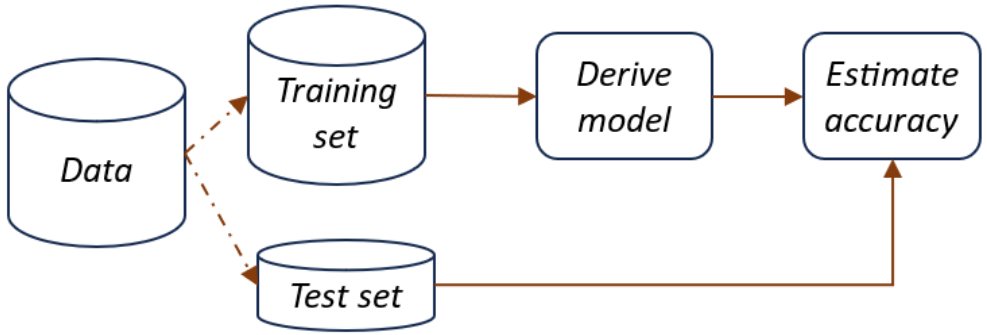
5.1. Số liệu để đánh giá hiệu suất phân loại *(Metrics for Evaluating Classifier Performance)*

- Các khía cạnh khác có thể dùng để đánh giá các bộ phân loại

Ngoài các biện pháp dựa trên độ chính xác, các bộ phân loại cũng có thể được so sánh theo các khía cạnh bổ sung sau:

- ***Speed*** (tốc độ): đề cập đến chi phí tính toán liên quan đến việc tạo và sử dụng bộ phân loại nhất định.
- ***Robustness*** (tính mạnh mẽ): là khả năng của bộ phân loại đưa ra dự đoán chính xác cho dữ liệu nhiễu (*noisy data*) hoặc dữ liệu thiếu giá trị (*missing values*). Độ bền thường được đánh giá bằng một loạt tập dữ liệu tổng hợp biểu thị mức độ nhiễu ngày càng tăng (*increasing degrees of noise*) và các giá trị bị thiếu.
- ***Scalability*** (khả năng mở rộng): đề cập đến khả năng xây dựng bộ phân loại hiệu quả với lượng dữ liệu lớn. Khả năng mở rộng thường được đánh giá bằng một loạt tập dữ liệu có kích thước tăng dần.
- ***Interpretability*** (khả năng diễn giải): đề cập đến mức độ hiểu biết sâu sắc được cung cấp bởi bộ phân loại hoặc bộ dự đoán. Khả năng diễn giải là chủ quan và do đó khó đánh giá hơn.

5.2. Phương pháp Holdout và lấy mẫu con ngẫu nhiên *(Holdout Method and Random Subsampling)*



5.2.1. Holdout method (phương pháp loại bỏ)

- Trong phương pháp này, dữ liệu đã cho được phân chia ngẫu nhiên thành hai tập độc lập, tập huấn luyện và tập kiểm tra.
- Thông thường, hai phần ba dữ liệu được phân bổ cho tập huấn luyện và một phần ba còn lại được phân bổ cho tập kiểm tra. Tập huấn luyện được sử dụng để lấy mô hình. Độ chính xác của mô hình sau đó được ước tính bằng bộ kiểm tra.
- Ước tính này bị cho là bi quan (*pessimistic*) vì chỉ một phần dữ liệu ban đầu được sử dụng để rút ra mô hình.

5.2.2. Random subsampling

Lấy mẫu con ngẫu nhiên là một biến thể của phương pháp loại trừ trong đó phương pháp Holdout được lặp lại k lần. Ước tính độ chính xác tổng thể được lấy bằng mức trung bình của độ chính xác thu được từ mỗi lần lặp.

5.3. Xác thực chéo *(Cross-Validation)*

5.3.1. k-fold cross-validation

- Trong phương pháp này, dữ liệu ban đầu được phân chia ngẫu nhiên thành k tập con loại trừ lẫn nhau hoặc "các tập con", *D1*, D_2,..., *Dk*, mỗi tập có kích thước xấp xỉ bằng nhau.
- Việc huấn luyện và kiểm tra được thực hiện ***k*** lần. Trong lần lặp thứ ***i***, phân vùng $\boldsymbol{D_i}$ được dành riêng làm tập kiểm tra và các phân vùng còn lại được sử dụng chung để huấn luyện mô hình. Nghĩa là, trong lần lặp đầu tiên, các tập con D_2, ..., D_k đóng vai trò chung là tập huấn luyện để thu được mô hình đầu tiên, được thử nghiệm trên D_1; lần lặp thứ hai được huấn luyện trên các tập con D_1, D_3, ..., D_k và thử nghiệm trên D_2; tương tự cho các tập còn lại.
- Không giống như các phương pháp *holdout* và lấy mẫu con ngẫu nhiên, ở đây mỗi mẫu được sử dụng cùng số lần để huấn luyện và một lần để kiểm tra.
- Đối với việc phân loại, ước tính độ chính xác là tổng số phân loại chính xác từ ***k*** lần lặp, chia cho tổng số bộ dữ liệu trong dữ liệu ban đầu.

5.3.2. Leave-one-out *(loại bỏ một lần)*

- Là trường hợp đặc biệt của xác thực chéo k-fold trong đó ***k*** được đặt thành số bộ dữ liệu ban đầu. Nghĩa là, tại một thời điểm chỉ có một mẫu bị "bỏ sót" (*left-out*) trong tập kiểm tra.
- Trong xác thực chéo phân tầng (*stratified cross-validation*), các nếp gấp (fold) được phân tầng sao cho sự phân bố lớp của các bộ dữ liệu trong mỗi nếp gấp gần giống như phân bố trong dữ liệu ban đầu.
- Nói chung, nên xác thực chéo 10 lần phân tầng để ước tính độ chính xác (ngay cả khi khả năng tính toán cho phép sử dụng nhiều lần hơn) do độ lệch (bias) và phương sai (*variance*) tương đối thấp.

5.4. Phương pháp Bootstrap

- Không giống như các phương pháp ước tính độ chính xác vừa được đề cập, phương pháp *bootstrap* lấy mẫu các bộ dữ liệu huấn luyện đã cho một cách đồng nhất và thay thế (*uniformly with replacement*). Nghĩa là, mỗi khi một bộ dữ liệu được chọn, nó có khả năng được chọn lại và thêm lại vào tập huấn luyện như nhau. Ví dụ, hãy tưởng tượng một chiếc máy chọn ngẫu nhiên các bộ dữ liệu cho tập huấn luyện của ta. Khi lấy mẫu có thay thế, máy được phép chọn cùng một bộ nhiều lần.
- Có một số phương pháp bootstrap. Phương pháp thường được sử dụng là *bootstrap .632*, hoạt động như sau. Giả sử ta được cung cấp một tập dữ liệu gồm ***d*** bộ dữ liệu. Tập dữ liệu được lấy mẫu *d* lần, có thay thế, tạo ra mẫu bootstrap hoặc tập huấn luyện gồm *d* mẫu. Rất có khả năng một số bộ dữ liệu gốc sẽ xuất hiện nhiều lần trong mẫu này. Các bộ dữ liệu không được đưa vào tập huấn luyện sẽ tạo thành tập kiểm tra. Giả sử ta thử điều này nhiều lần. Hóa ra, trung bình, 63,2% bộ dữ liệu gốc sẽ xuất hiện trong mẫu *bootstrap* và 36,8% còn lại sẽ tạo thành tập kiểm tra (do đó có tên là ***.632 bootstrap***).

5.5. So sánh các phân loại dựa trên đường cong Chi phí-Lợi ích và ROC *(Comparing Classifiers Based on Cost–Benefit and ROC Curves)*

- Kết quả dương tính thực (*true positives*), âm tính thực (*true negatives*), dương tính giả (*false positives*) và âm tính giả (*false negatives*) cũng hữu ích trong việc đánh giá chi phí và lợi ích (*costs and benefits*) hay còn gọi là rủi ro và lợi ích (*risks and benefits*) liên quan đến mô hình phân loại.
- Một số ví dụ về công dụng trong việc đánh giá chi phí và lợi ích:
  - Chi phí liên quan đến kết quả âm tính giả (chẳng hạn như dự đoán sai rằng bệnh nhân ung thư không phải là ung thư) lớn hơn nhiều so với chi phí của kết quả dương tính giả (gắn nhãn sai một cách thận trọng cho bệnh nhân không ung thư là ung thư).
  - Chi phí cho một người không vỡ nợ vay sẽ vượt xa chi phí kinh doanh bị mất do từ chối cho một người không vỡ nợ vay.
  - Ứng dụng cố gắng xác định các hộ gia đình có khả năng phản hồi việc gửi thư của một số tài liệu quảng cáo nhất định, chi phí gửi thư đến nhiều hộ gia đình không phản hồi có thể lớn hơn chi phí kinh doanh bị mất do không gửi thư đến các hộ gia đình lẽ ra đã phản hồi.

5.5. So sánh các phân loại dựa trên đường cong Chi phí-Lợi ích và ROC

- Ví dụ: Cho bộ dữ liệu sau, trong đó:
  - *Tuple#*: đơn thuần là một số nhận dạng bộ, hỗ trợ cho việc giải thích sau này.
  - *Class*: nhãn lớp thực tế của bộ dữ liệu.
  - *TP, FP, TN, FN, TPR*, và *FPR* lần lượt là: True Positive, False Positive, True Negative, False Negative, True Positive Ra False Positive Rate.

Tuple #	Class	Probability	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	0	1	1.0	0.8
10	N	0.40	5	5	0	0	1.0	1.0

- Bắt đầu với bộ 1, có điểm xác suất cao nhất *(=0.9)* và lấy điểm đó làm ngưỡng. Do đó, bộ phân loại coi bộ 1 là dương và tất cả các bộ khác được coi là âm. Vì nhãn lớp thực tế của bộ 1 là dương nên ta có giá trị dương thực sự, do đó *TP = 1* và *FP = 0*. Trong số chín bộ dữ liệu còn lại, tất cả đều được phân loại là âm, thì có năm bộ thực sự là âm (do đó *TN = 5*) . Bốn bộ còn lại (*2, 4, 5, 9*) đều thực sự dương, do đó, *FN = 4*. Do đó, *TPR=TP/P=1/5= 0,2*; trong khi *FPR = 0*. Do đó, ta có điểm (*0,2; 0*) cho đường cong *ROC*.

Xét bộ thứ 1		*Predicted class*		
		yes	*no*	*Cộng*
Actual class	*yes*	*TP=1*	*FN=4*	*P=5*
	no	*FP=0*	*TN=5*	*N=5*
	Total	*P'=1*	*N'=9*	*10*

=> TPR=TP/P=1/5= 0,2

=> FPR = FP/N = 0/5 = 0

5.5. So sánh các phân loại dựa trên đường cong Chi phí-Lợi ích và ROC

- Ví dụ:
  - Tiếp theo, chọn bộ thứ 2, ngưỡng t được đặt thành *0,8*, giá trị xác suất cho bộ dữ liệu 2, do đó bộ dữ liệu này hiện cũng được coi là dương, trong khi các bộ dữ liệu từ 3 đến 10 được coi là âm. Nhãn lớp thực tế của bộ 2 là dương, do đó bây giờ *TP = 2*. Phần còn lại của hàng có thể dễ dàng được tính toán, dẫn đến điểm (0,4;0).

Tuple #	Class	Probability	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	0	1	1.0	0.8
10	N	0.40	5	5	0	0	1.0	1.0

Xét bộ thứ 2		Predicted class		
		yes	no	Cộng
Actual class	yes	TP=2	FN=3	P=5
	no	FP=0	TN=5	N=5
	Total	P'=2	N'=8	10

=> TPR=TP/P=2/5= 0,4

=> FPR = FP/N = 0/5 = 0

Xét bộ thứ 3		Predicted class		
		yes	no	Cộng
Actual class	yes	TP=2	FN=3	P=5
	no	FP=1	TN=4	N=5
	Total	P'=1	N'=9	10

=> TPR=TP/P=2/5= 0,4

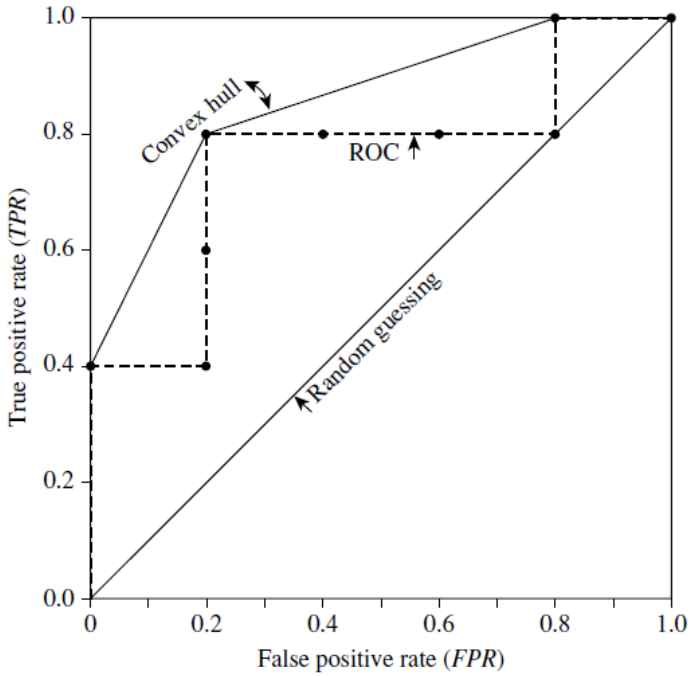
=> FPR = FP/N = 1/5 = 0,2

  - Tiếp theo, kiểm tra nhãn lớp của bộ dữ liệu 3 và đặt ***t*** là *0,7*, giá trị xác suất được bộ phân loại trả về cho bộ dữ liệu đó. Do đó, bộ 3 được coi là dương, tuy nhiên nhãn thực tế của nó là âm và do đó nó là dương tính giả (*false positive*). Do đó, *TP* giữ nguyên và *FP* tăng sao cho *FP = 1*.

5.5. So sánh các phân loại dựa trên đường cong Chi phí-Lợi ích và ROC

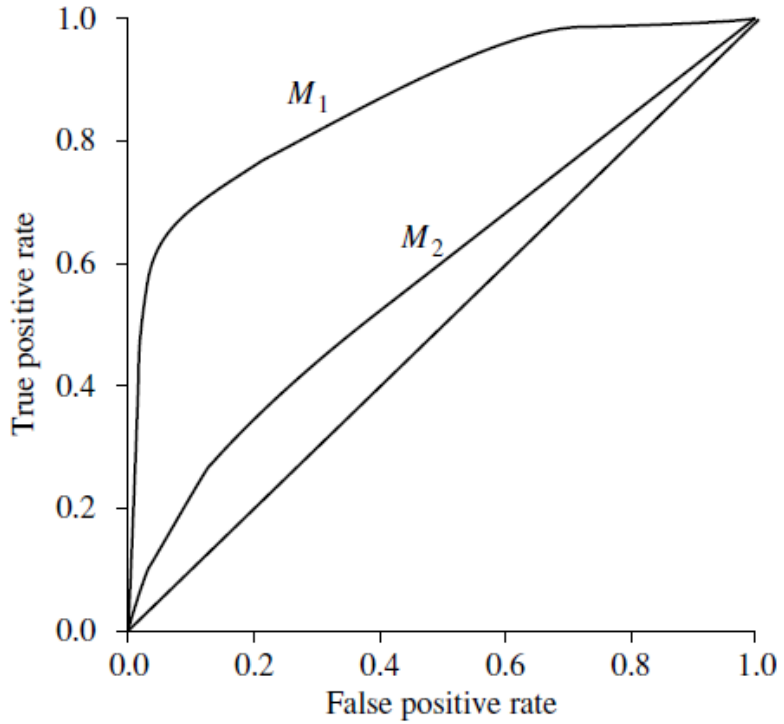
- Ví dụ:
  - Phần còn lại của các giá trị trong hàng cũng có thể được tính toán dễ dàng, thu được điểm (*0,4; 0,2*). Biểu đồ *ROC* thu được, từ việc kiểm tra từng bộ dữ liệu, là đường răng cưa được hiển thị trong hình

Có nhiều phương pháp để thu được đường cong từ những điểm này, trong đó phương pháp phổ biến nhất là sử dụng bao lồi (*convex hull*). Đồ thị cũng hiển thị một đường chéo đại diện cho việc đoán ngẫu nhiên (*random guessing*).



5.5. So sánh các phân loại dựa trên đường cong Chi phí-Lợi ích và ROC

- Hình bên thể hiện đường cong *ROC* của hai mô hình phân loại M1 và M2. Đường chéo thể hiện việc đoán ngẫu nhiên cũng được hiển thị.



  - Đường cong *ROC* của mô hình càng gần đường chéo thì mô hình càng kém chính xác (*less accurate*). đường cong di chuyển dốc lên từ tọa đọ *0;0*. Sau đó, khi ta bắt đầu gặp ngày càng ít kết quả dương tính thực (*true positives*) và ngày càng nhiều kết quả dương tính giả (*false positives*), đường cong sẽ giãn ra và trở nên nằm ngang hơn.
  - Để đánh giá độ chính xác của mô hình, ta có thể đo diện tích dưới đường cong. Hiện có một số gói phần mềm có thể thực hiện tính toán như vậy. Diện tích càng gần *0,5* thì mô hình tương ứng càng kém chính xác. Một mô hình có độ chính xác hoàn hảo sẽ có diện tích là *1,0*.

NỘI DUNG CHƯƠNG 5

1. Một số khái niệm cơ bản
2. Cây quyết định (*Decision Tree*)
3. Phương pháp phân loại Bayes (*Bayes Classification Methods*)
4. Phân loại dựa trên quy tắc (*Rule-Based Classification*)
5. Đánh giá và lựa chọn mô hình (*Model Evaluation and Selection*)
6. Kỹ thuật cải thiện độ chính xác của phân loại (*Techniques to Improve Classification Accuracy*)
7. Bài tập

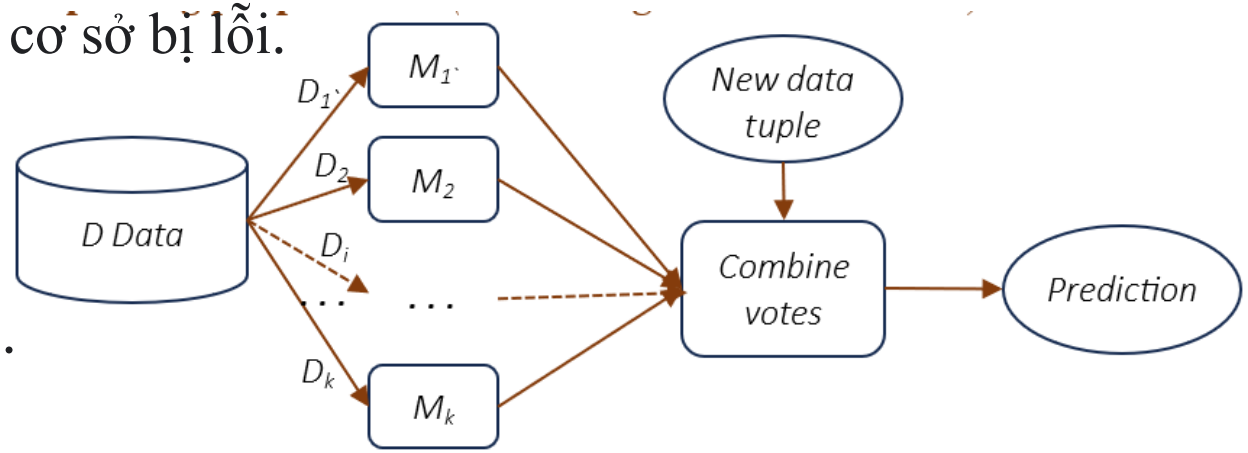
6. KỸ THUẬT CẢI THIỆN ĐỘ CHÍNH XÁC CỦA PHÂN LOẠI
(Techniques to Improve Classification Accuracy)

Các mô hình học tập truyền thống giả định rằng các lớp dữ liệu được phân phối tốt. Tuy nhiên, trong nhiều miền dữ liệu trong thế giới thực, dữ liệu không cân bằng về lớp (*class-imbalanced*), trong đó lớp quan tâm chính chỉ được biểu thị bằng một vài bộ dữ liệu. Điều này được gọi là vấn đề mất cân bằng lớp (*class imbalance problem*). Phần này nghiên cứu các kỹ thuật để cải thiện độ chính xác phân loại của dữ liệu mất cân bằng lớp.

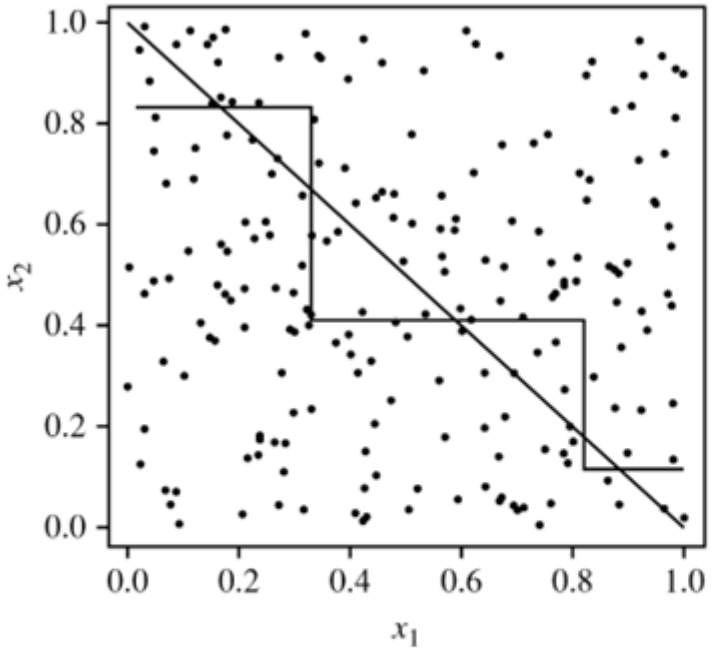
6.1. Các phương pháp nhất trí *(Introducing Ensemble Methods)*

- Ví dụ, hãy xem xét một nhóm thực hiện biểu quyết theo đa số. Nghĩa là, cho một bộ ***X*** để phân loại, nó thu thập các dự đoán nhãn lớp được trả về từ các bộ phân loại cơ sở và đưa ra kết quả dựa trên "*số đông*" (*majority*) của lớp đó. Các bộ phân loại cơ sở có thể mắc lỗi, nhưng tập hợp sẽ chỉ phân loại sai ***X*** nếu hơn một nửa số bộ phân loại cơ sở bị lỗi.

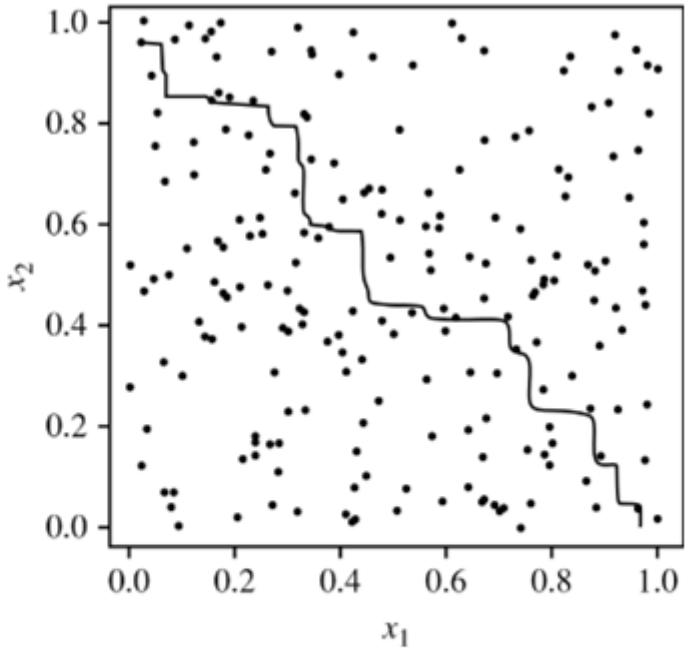
⇒ "*sự nhất trí*" có xu hướng cho kết quả chính xác hơn các bộ phân loại cơ sở của nó.



6.1. Các phương pháp nhất trí *(Introducing Ensemble Methods)*



Kết quả của sự nhất trì từ
1 cây quyết định đơn



Kết quả từ sự nhất trì
của nhiều cây quyết định

6.2. Bagging *(Bootstrap AGGregation)*

- Giả sử bạn là một bệnh nhân và muốn được chẩn đoán dựa trên các triệu chứng của mình. Thay vì hỏi một bác sỹ, bạn có thể chọn hỏi nhiều bác sỹ. Nếu một chẩn đoán (*diagnosis*) nào đó xảy ra nhiều hơn bất kỳ chẩn đoán nào khác, bạn có thể chọn đây là chẩn đoán cuối cùng hoặc tốt nhất. Theo trực giác, một cuộc bỏ phiếu đa số của một nhóm lớn các bác sỹ có thể đáng tin cậy hơn một cuộc bỏ phiếu đa số của một nhóm nhỏ.
- Cho một tập ***D*** gồm ***d*** bộ, phương pháp *bagging* hoạt động như sau:
  - Đối với phép lặp *i(i = 1, 2,..., k)*, một tập huấn luyện $\boldsymbol{D_i}$ của các bộ dữ liệu *d* được lấy mẫu thay thế từ tập các bộ dữ liệu ***D*** ban đầu. Mỗi tập huấn luyện là một mẫu *bootstrap*. Bởi vì việc lấy mẫu có thay thế được sử dụng nên một số bộ dữ liệu ban đầu của ***D*** có thể không được đưa vào D_i, trong khi những bộ dữ liệu khác có thể xuất hiện nhiều lần. Một mô hình phân loại M_i được học cho mỗi tập huấn luyện D_i.
  - Để phân loại một bộ dữ liệu *X* chưa biết, mỗi bộ phân loại M_i trả về dự đoán lớp của nó, và dự đoan đó được tính là một phiếu bầu. Bộ phân loại được đóng gói ($M*$) sẽ thực hiện việc đếm số phiếu bầu và gán lớp có nhiều phiếu bầu nhất cho *X*.

6.3. Boosting and AdaBoost

6.3.1. Boosting: phương pháp tăng cướng "sự nhất trí" thông qua trọng số

- Tương tự như phương pháp Bagging, cùng tham khảo ý kiến của nhiều bác sỹ, nhưng lúc này mỗi bác sỹ sẽ được gán trọng số, với trọng số của mỗi bác sỹ dựa trên độ chính xác của các chẩn đoán trước đó mà họ đã thực hiện.

  Chẩn đoán cuối cùng sau đó là sự kết hợp của các chẩn đoán có trọng số.

- Trong quá trình tăng cường, trọng số cũng được gán cho mỗi bộ dữ liệu huấn luyện. Một loạt các bộ phân loại k được học lặp đi lặp lại. Sau khi học được bộ phân loại Mi, các trọng số sẽ được cập nhật để cho phép bộ phân loại tiếp theo, Mi+1, "chú ý nhiều hơn" đến các bộ dữ liệu huấn luyện đã bị Mi phân loại sai.

  Trình phân loại được tăng cường cuối cùng, M∗, kết hợp phiếu bầu của từng trình phân loại riêng lẻ, trong đó trọng số phiếu bầu của mỗi trình phân loại là một hàm về độ chính xác của nó.

6.3. Boosting and AdaBoost

6.3.2. AdaBoost *(viết tắt của Adaptive Boosting)*

- Cho một tập dữ liệu D gồm các bộ dữ liệu có nhãn lớp d, (X_1, y_1), (X_2, y_2), ..., (X_d, y_d), trong đó y_i là nhãn lớp của bộ dữ liệu X_i.
  - Ban đầu, *AdaBoost* gán cho mỗi bộ dữ liệu huấn luyện một trọng số bằng nhau là ***1/d***.
  - Do mỗi lần thực hiện thuật toán chỉ tạo được 1 bộ phân loại, nên để tạo ***k*** bộ phân loại cho tập dữ liệu ***D***, cần thực hiện lặp lại thuật toán ***k*** lần.
  - Ở vòng ***i***, các bộ dữ liệu từ ***D*** được lấy mẫu để tạo thành tập huấn luyện D_i, có kích thước d.
  - Việc lấy mẫu có thay thế được sử dụng - cùng một bộ dữ liệu có thể được chọn nhiều lần. Cơ hội được chọn của mỗi bộ dữ liệu dựa trên trọng số của nó.

  Mô hình phân loại M_i được bắt nguồn từ các bộ dữ liệu huấn luyện của D_i. Sai số của nó sau đó được tính toán bằng cách sử dụng D_i làm tập kiểm tra. Trọng số của các bộ dữ liệu huấn luyện sau đó được điều chỉnh theo cách chúng được phân loại.

6.3. Boosting and AdaBoost

6.3.1. AdaBoost *(viết tắt của Adaptive Boosting)*

- Nếu một bộ được phân loại không chính xác thì trọng số của nó sẽ tăng lên. Ngược lại, nếu một bộ được phân loại chính xác thì trọng số của nó sẽ giảm.
- Trọng số của một bộ dữ liệu phản ánh mức độ khó phân loại - trọng số càng cao thì nó càng thường xuyên bị phân loại sai.
- Các trọng số này sẽ được sử dụng để tạo các mẫu huấn luyện cho bộ phân loại của vòng tiếp theo.
- Ý tưởng cơ bản là khi xây dựng một bộ phân loại, ta muốn nó tập trung nhiều hơn vào các bộ dữ liệu bị phân loại sai ở vòng trước. Một số bộ phân loại có thể phân loại một số bộ dữ liệu "khó" tốt hơn những bộ khác. Bằng cách này, ta xây dựng một loạt các bộ phân loại bổ sung cho nhau.

6.4. Phương pháp rừng ngẫu nhiên *(Random Forests)*

- Hãy tưởng tượng rằng mỗi bộ phân loại trong quần thể là một bộ phân loại cây quyết định sao cho tập hợp các bộ phân loại là một "rừng".
- Các cây quyết định riêng lẻ được tạo bằng cách sử dụng lựa chọn ngẫu nhiên các thuộc tính tại mỗi nút để xác định sự phân chia. Chính thức hơn, mỗi cây phụ thuộc vào các giá trị của một vectơ ngẫu nhiên được lấy mẫu độc lập và có cùng phân bố cho tất cả các cây trong rừng. Trong quá trình phân loại, mỗi cây sẽ bình chọn và lớp phổ biến nhất sẽ được trả về.

6.4. Phương pháp rừng ngẫu nhiên *(Random Forests)*

- Rừng ngẫu nhiên có thể được xây dựng bằng cách sử dụng phương pháp bagging song song với lựa chọn thuộc tính ngẫu nhiên (*random attribute selection*).
  - Một tập huấn luyện D gồm d bộ dữ liệu được đưa ra. Quy trình chung để tạo k cây quyết định cho tập hợp như sau. Đối với mỗi lần lặp, *i(i = 1, 2,..., k)*, một tập huấn luyện D_i của các bộ dữ liệu ***d*** được lấy mẫu với sự thay thế từ ***D***. Nghĩa là, mỗi D_i là một mẫu khởi động của ***D***, do đó một số bộ có thể xuất hiện nhiều lần trong D_i, trong khi những bộ khác có thể bị loại trừ. Gọi *F* là số lượng thuộc tính được sử dụng để xác định sự phân chia tại mỗi nút, trong đó *F* nhỏ hơn nhiều so với số lượng thuộc tính có sẵn. Để xây dựng bộ phân loại cây quyết định, M_i, chọn ngẫu nhiên, tại mỗi nút, các thuộc tính *F* làm ứng cử viên cho việc phân chia tại nút.
  - Phương pháp *CART* được sử dụng để tăng trưởng cây. Cây được tăng trưởng ở kích thước tối đa và không bị cắt tỉa. Rừng ngẫu nhiên được hình thành theo cách này, với lựa chọn đầu vào ngẫu nhiên (*random input*), được gọi là *Forest-RI*.

6.4. Phương pháp rừng ngẫu nhiên *(Random Forests)*

- Một dạng rừng ngẫu nhiên khác, được gọi là *Forest-RC*, sử dụng kết hợp tuyến tính ngẫu nhiên của các thuộc tính đầu vào.
  - Thay vì chọn ngẫu nhiên một tập hợp con các thuộc tính, nó tạo ra các thuộc tính (hoặc tính năng) mới là sự kết hợp tuyến tính của các thuộc tính hiện có. Nghĩa là, một thuộc tính được tạo ra bằng cách chỉ định ***L***, số lượng thuộc tính ban đầu được kết hợp. Tại một nút nhất định, các thuộc tính *L* được chọn ngẫu nhiên và được cộng cùng với các hệ số (*coefficients*) là số ngẫu nhiên thống nhất trên miền giá trị *[−1,1]*. *F* kết hợp tuyến tính được tạo ra và việc tìm kiếm được thực hiện trên những kết hợp này để có được sự phân chia tốt nhất.
  - Hình thức rừng ngẫu nhiên này rất hữu ích khi chỉ có sẵn một vài thuộc tính, nhằm làm giảm mối tương quan giữa các phân loại riêng lẻ.

6.5. Cải thiện độ chính xác phân loại của dữ liệu mất cân bằng lớp

(Improving Classification Accuracy of Class-Imbalanced Data)

- Mất cân bằng:
  - Với dữ liệu hai lớp (*two-class data*), dữ liệu sẽ mất cân bằng lớp nếu lớp quan tâm chính (lớp dương – *positive class*) chỉ được biểu thị bằng một vài bộ dữ liệu, trong khi phần lớn các bộ dữ liệu đại diện cho lớp âm (*negative class*).
  - Đối với dữ liệu mất cân bằng nhiều lớp, việc phân bổ dữ liệu của mỗi lớp khác nhau là đáng kể, lớp chính hoặc các lớp quan tâm là rất hiếm. Vấn đề mất cân bằng lớp có liên quan chặt chẽ đến chi phí, trong đó chi phí cho sai sót ở mỗi lớp là không bằng nhau.
  - Ví dụ:
    - *Trong chẩn đoán y khoa:* việc chẩn đoán sai một bệnh nhân ung thư là khỏe mạnh (âm tính giả) sẽ tốn kém hơn nhiều so với chẩn đoán sai một bệnh nhân khỏe mạnh là mắc bệnh ung thư (dương tính giả). Một lỗi âm tính giả có thể dẫn đến tử vong và do đó đắt hơn nhiều so với lỗi dương tính giả.
    - *Trong kinh doanh*: phát hiện gian lận
    - *Trong mội trường*: phát hiện sự cố tràn dầu từ hình ảnh radar vệ tinh.
    - …

6.5. Cải thiện độ chính xác phân loại của dữ liệu mất cân bằng lớp

- Các thuật toán phân loại truyền thống:
  - Nhằm mục đích giảm thiểu số lỗi mắc phải trong quá trình phân loại.
  - Cho rằng chi phí của lỗi dương tính giả và lỗi âm tính giả là như nhau.

  $\Rightarrow$ Do đó, bằng cách giả định sự phân bố cân bằng của các lớp và chi phí lỗi bằng nhau, chúng không phù hợp với dữ liệu mất cân bằng lớp.

6.5. Cải thiện độ chính xác phân loại của dữ liệu mất cân bằng lớp

- Các phương pháp chung để cải thiện độ chính xác phân loại của dữ liệu mất cân bằng lớp:
  i. Lấy mẫu quá mức (*oversampling*).
  ii. Lấy mẫu dưới mức (*undersampling*)
  iii. Di chuyển ngưỡng (*threshold moving*)
  iv. Kỹ thuật "nhất trí" (*ensemble techniques*).

- Ví dụ Giả sử tập huấn luyện ban đầu chứa 100 bộ giá trị dương và 1000 bộ giá trị âm.
  - *Oversampling*: sao chép các bộ dữ liệu thuộc lớp hiếm hơn để tạo thành một tập huấn luyện mới chứa 1000 bộ dữ liệu dương và 1000 bộ dữ liệu âm
  - *Undersampling*: loại bỏ ngẫu nhiên các bộ dữ liệu âm để tập huấn luyện mới chứa 100 bộ dữ liệu dương và 100 bộ dữ liệu âm.

- Ba phương pháp đầu tiên không liên quan đến bất kỳ thay đổi nào trong việc xây dựng mô hình phân loại:
  - *Oversampling* và *Undersampling* làm thay đổi sự phân bố của các bộ dữ liệu trong tập huấn luyện
  - Di chuyển ngưỡng ảnh hưởng đến cách mô hình đưa ra quyết định khi phân loại dữ liệu mới.
  - Các phương pháp "*nhất trí*" tuân theo các kỹ thuật đã được mô tả trong các phần trước.

- Thay đổi cách phân phối dữ liệu huấn luyện sao cho lớp hiếm (dương) được thể hiện tốt:
  - *Oversampling* hoạt động bằng cách lấy mẫu lại các bộ dữ liệu dương sao cho tập huấn luyện kết quả chứa số lượng bộ dữ liệu dương và âm bằng nhau.
  - *Undersampling* hoạt động bằng cách giảm số lượng bộ dữ liệu âm. Nó loại bỏ ngẫu nhiên các bộ dữ liệu khỏi lớp đa số (âm) cho đến khi có số lượng bộ dữ liệu dương và âm bằng nhau.

NỘI DUNG CHƯƠNG 5

1. Một số khái niệm cơ bản
2. Cây quyết định (*Decision Tree*)
3. Phương pháp phân loại Bayes (*Bayes Classification Methods*)
4. Phân loại dựa trên quy tắc (*Rule-Based Classification*)
5. Đánh giá và lựa chọn mô hình (*Model Evaluation and Selection*)
6. Kỹ thuật cải thiện độ chính xác của phân loại (Techniques to Improve Classification Accuracy)
7. Bài tập

7. BÀI TẬP

department	*status*	*age*	*salary*	*count*
sales	senior	31...35	46K...50K	30
sales	junior	26...30	26K...30K	40
sales	junior	31...35	31K...35K	40
systems	junior	21...25	46K...50K	20
systems	senior	31...35	66K...70K	5
systems	junior	26...30	46K...50K	3
systems	senior	41...45	66K...70K	3
marketing	senior	36...40	46K...50K	10
marketing	junior	31...35	41K...45K	4
secretary	senior	46...50	36K...40K	4
secretary	junior	26...30	26K...30K	6

1) Bảng sau bao gồm dữ liệu đào tạo từ cơ sở dữ liệu nhân viên. Dữ liệu đã được khái quát hóa. Ví dụ: "31 ... 35" cho độ tuổi đại diện cho độ tuổi từ 31 đến 35. Đối với mỗi mục nhập hàng nhất định, thông tin cần lưu trữ gồm department, status, age, salary, count.

   Đặt status là thuộc tính nhãn lớp.

   Cho một bộ dữ liệu có các giá trị "systems", "26...30" và "46–50K" tương ứng cho các thuộc tính *department, age*, and *salary*,

   a. Lần lượt thực hiện xây dựng cây quyết định từ dữ liệu đã cho bằng 3 thước đo sau:
      i. Độ lợi thông tin (Information Gain)
      ii. Tỷ lệ khuyếch đại (Gain Ratio)
      iii. Chỉ số Gini (Gini index)
   b. Xác định kết quả của bộ dữ liệu đã cho dựa trên cây quyết định vừa có.
   c. Cho biết kết quả theo cách phân loại naïve Bayes của bộ dữ liệu đã cho sẽ như thế nào dựa trên việc phân lớp theo status?

7. BÀI TẬP

2) Cho bộ dữ liêu huấn luyện sau:

TID	Refund	Martial Status	Taxable income	Cheat
1	Yes	SIngle	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

a) Lần lượt thực hiện xây dựng cây quyết định từ dữ liệu đã cho bằng 3 thước đo sau:

i. Độ lợi thông tin (Information Gain)

ii. Tỷ lệ khuyếch đại (Gain Ratio)

iii. Chỉ số Gini (Gini index)

b) Cho biết kết quả của các bộ dữ liệu sau với các giá trị tương ứng cho các thuộc tính Refund, Martial Status, và Cheat như sau:

i. Yes, Divorced, 120K

ii. No, Married, 130K

iii. No, Single, 60K

7. BÀI TẬP

3) Cho bộ dữ liêu huấn luyện sau:

a) Lần lượt thực hiện xây dựng cây quyết định từ dữ liệu đã cho bằng 3 thước đo sau:

i. Độ lợi thông tin (Information Gain)

ii. Tỷ lệ khuyếch đại (Gain Ratio)

iii. Chỉ số Gini (Gini index)

age	*income*	*student*	*credit_rating*	*buys_computer*
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

b) Cho biết kết quả của các bộ dữ liệu sau với các giá trị tương ứng cho các thuộc tính age, income, student, credit_rating, buys_computer như sau:

i. 35, low, no, excellent

ii. 42, high, no, fair

iii. 27, medium, yes, fair

7. BÀI TẬP

4) Các bộ dữ liệu trong bảng sau được sắp xếp theo giá trị xác suất giảm dần, được trả về bởi bộ phân loại. Đối với mỗi bộ dữ liệu, hãy tính các giá trị cho số lượng dương tính thật (TP), dương tính giả (FP), âm tính thực (TN) và âm tính giả (FN). Tính tỷ lệ dương tính thật (TPR) và tỷ lệ dương tính giả (FPR). Vẽ đường cong ROC cho dữ liệu.

Tuple	*Class*	*probability*
1	P	0.95
2	N	0.85
3	P	0.78
4	P	0.66
5	N	0.60
6	P	0.55
7	N	0.53
8	N	0.52
9	N	0.51
10	P	0.40


Q&A
How
Why
When
Where
What
Who