TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Báo cáo Bài tập 1

Tìm hiểu Decision Tree

HỌ VÀ TÊN: VÕ DUY NHẤT
MSSV: 20521711

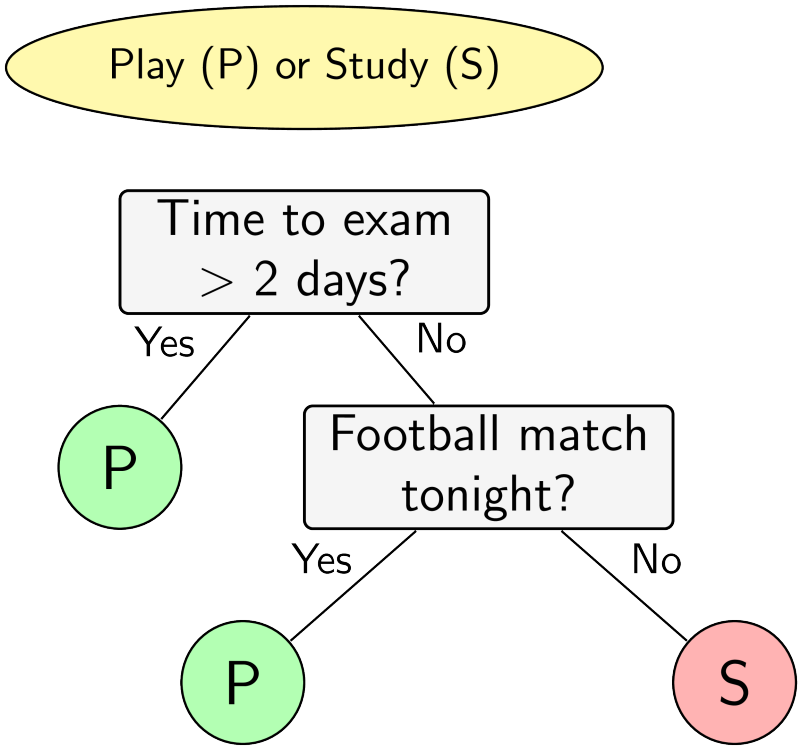
Hồ Chí Minh, 9-2022

Mục lục

1 Khái niệm

Cây quyết định (Decision Trees) là một phương pháp học tập có giám sát không tham số, được áp dụng vào hai bài toán classification và regression . Mục tiêu là tạo ra một mô hình dự đoán giá trị của một biến mục tiêu bằng cách tìm hiểu các quy tắc quyết định đơn giản được suy ra từ các tính năng dữ liệu.

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



Hình 1: Ví dụ về Decision tree

Các node thể hiện đầu ra (màu lục và đỏ) được gọi là node lá (leaf node hoặc terminal node). Các node thể hiện câu hỏi là các non-leaf node. Non-leaf node trên cùng (câu hỏi đầu tiên) được gọi là node gốc (root node). Các non-leaf node thường có hai hoặc nhiều node con (child node). Các child node này có thể là một leaf node hoặc một non-leaf node khác. Các child node có cùng bố mẹ được gọi là sibling node. Nếu tất cả các non-leaf node chỉ có hai child node, ta nói rằng đó là một binary decision tree (cây quyết định nhị phân). Các câu hỏi trong binary decision tree đều có thể đưa được về dạng câu hỏi đúng hay sai. Các decision tree mà một leaf node có nhiều child node cũng có thể được đưa về dạng một binary decision tree.

2 Công dụng và cách sử dụng

2.1 Classification

Cây quyết định thường được áp dụng cho các bài toán classification bằng cách sử dụng DecisionTreeClassifier.

DecisionTreeClassifier là một lớp có khả năng thực hiện phân loại nhiều lớp trên một tập dữ liệu.

Như với các bộ phân loại khác, DecisionTreeClassifier lấy hai mảng đầu vào: một mảng X, có hình dạng (n samples, n features) chứa các mẫu huấn luyện và một mảng Y chứa các giá trị nguyên, hình dạng (n samples), là nhãn của các lớp cho các mẫu huấn luyện:

```
» from sklearn import tree
» X = [[0, 0], [1, 1]]
» Y = [0, 1]
» clf = tree.DecisionTreeClassifier()
» clf = clf.fit(X, Y)
```

Sau khi được huấn luyện, mô hình sau đó có thể được sử dụng để dự đoán loại mẫu:

```
» clf.predict([[2., 2.]])
array([1])
```

Trong trường hợp có nhiều lớp với xác suất giống nhau và cao nhất, bộ phân loại sẽ dự đoán lớp có chỉ số thấp nhất trong số các lớp đó.

Để thay thế cho việc xuất ra một lớp cụ thể, có thể dự đoán xác suất của mỗi lớp, đó là phần nhỏ của các mẫu huấn luyện của lớp trong một lá:

```
» clf.predict_proba([[2., 2.]])
array([[0., 1.]])
```

DecisionTreeClassifier có khả năng phân loại cả binary classification (trong đó nhãn là [-1, 1] hoặc [0, 1]) và multiclass classification (trong đó nhãn là [0,. . . , K-1]).

Ví dụ về Decision Tree Classification: xây dựng cây quyết định đối với tập dữ liệu hoa Iris. Link demo Google Colab: `https://colab.research.google.com/drive/1k7BXMj89srgItafEQlb-vUWiRQ89bA5L?usp=sharing`

2.2 Regression

Cây quyết định cũng có thể được áp dụng cho các bài toán regression như dự đoán giá nhà, dự đoán lượng nước dùng trong tháng, ... bằng cách sử dụng DecisionTreeRegressor.

Giống như trong bài toán classification, phương thức fit sẽ coi là mảng đối số X và y, chỉ có điều trong trường hợp này, y được mong đợi có giá trị dấu phẩy động thay vì giá trị số nguyên:

```
» from sklearn import tree
» X = [[0, 0], [2, 2]]
» y = [0.5, 2.5]
» clf = tree.DecisionTreeRegressor()
» clf = clf.fit(X, y)
» clf.predict([[1, 1]])
array([0.5])
```

Ví dụ về Decision Tree Regression - Link demo Google Colab: `https://colab.research.google.com/drive/1ACUxEl-9mf0DJz6f78Qn2Xb4yHcuIHCM?usp=sharing`

3 Một số tiêu chuẩn split

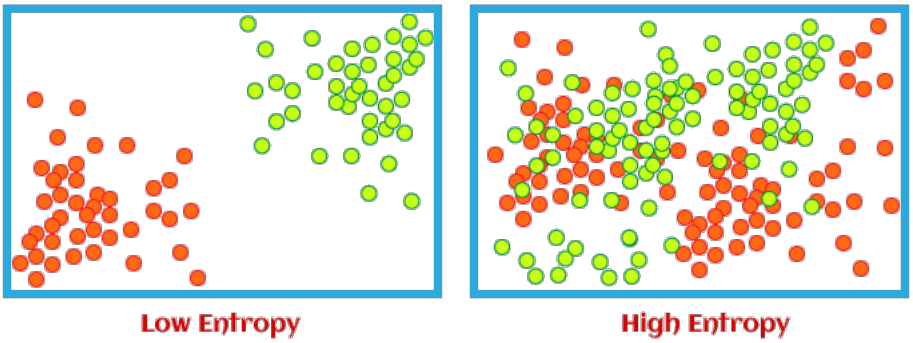
3.1 Entropy

3.1.1 Khái niệm

Entropy là phép đo sự rối loạn hoặc tạp chất trong thông tin được xử lý trong máy học. Nó xác định cách cây quyết định chọn để phân chia dữ liệu.

Chúng ta có thể hiểu thuật ngữ entropy bằng bất kỳ ví dụ đơn giản nào: lật một đồng xu. Khi chúng ta tung một đồng xu, thì có thể có hai kết quả. Tuy nhiên,

rất khó để kết luận đâu sẽ là kết quả chính xác khi tung đồng xu vì không có mối quan hệ trực tiếp nào giữa việc tung đồng xu và kết quả của nó. Có 50% xác suất của cả hai kết quả; khi đó, trong các tình huống như vậy, entropy sẽ cao. Đây là bản chất của entropy trong học máy.



Hình 2: Ví dụ về Entropy

3.1.2 Công thức

Cho một phân phối xác suất của một biến rời rạc x có thể nhận n giá trị khác nhau $x_1, x_2, x_3, ...x_n$. Giả sử rằng xác suất để x nhận các giá trị này là $p_i = p(x = x_i)$ với $0 \leq p_i \leq 1, \sum_{i=1}^{n} p_i = 1$. Ký hiệu phân phối này là $P = (p_1, p_2, ..., p_n)$. Entropy của phân phối này được xác định là

$$E = H(P) = -\sum_{i=1}^{n} p_i log_2(p_i)$$

Chỉ số entropy càng cao thì độ tạp chất càng lớn (Hình 2). Entropy luôn nằm trong khoảng từ 0 đến 1, tuy nhiên tùy thuộc vào số lượng lớp trong tập dữ liệu, nó có thể lớn hơn 1.

Hãy hiểu nó với một ví dụ trong đó chúng ta có một tập dữ liệu có ba màu sắc của trái cây là đỏ, xanh lá cây và vàng. Giả sử chúng ta có 2 quan sát màu đỏ, 2 màu xanh lá cây và 4 màu vàng trong toàn bộ tập dữ liệu. Sau đó, theo phương trình trên:

$$E = -(p_r log_2(p_r) + p_p log_2(p_p) + p_y log_2(p_y))$$

P_r: Xác suất chọn được quả đỏ $=> P_r = \frac{2}{8} = \frac{1}{4}$ (Vì chỉ 2 trong số 8 quả đại diện cho quả màu đỏ)

P_g = Xác suất chọn được quả xanh => $P_g = \frac{2}{8} = \frac{1}{4}$ (Vì chỉ 2 trong số 8 quả đại diện cho quả màu xanh)

P_y = Xác suất chọn được quả vàng => $P_y = \frac{4}{8} = \frac{1}{2}$ (Vì chỉ 4 trong số 8 quả đại diện cho quả màu vàng)

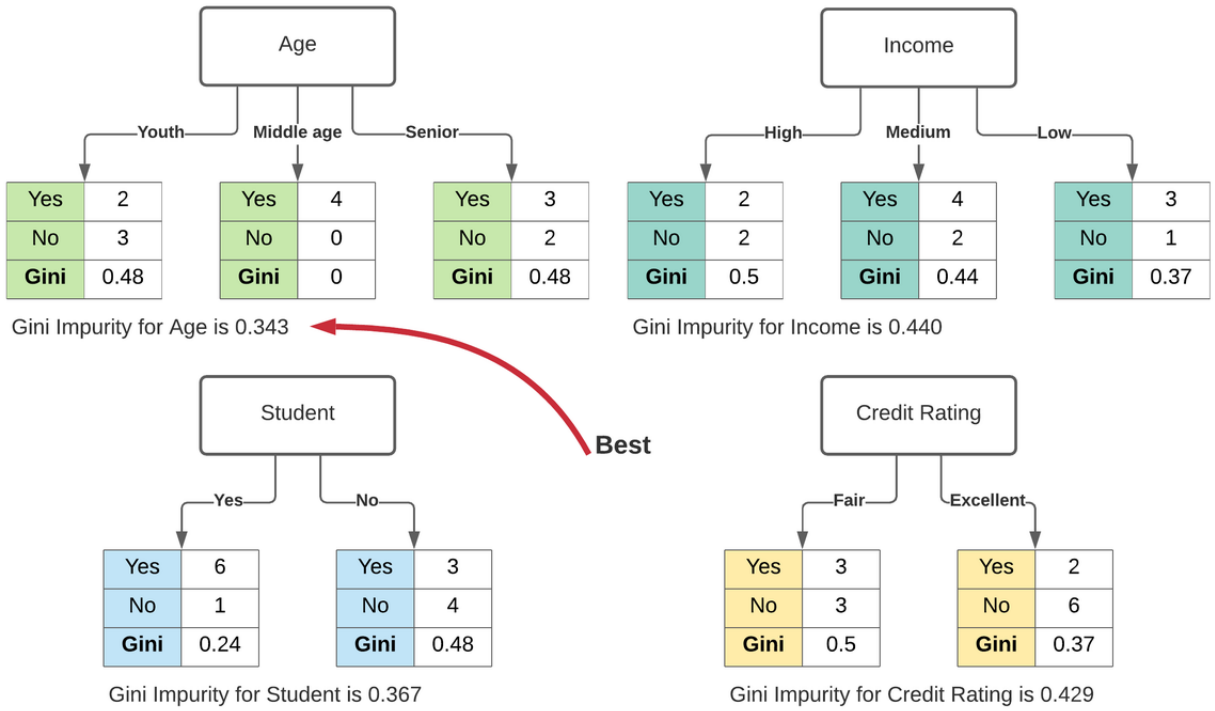
Bây giờ phương trình cuối cùng của chúng ta sẽ là:

$$E = -(\frac{1}{4}log_2(\frac{1}{4}) + \frac{1}{4}log_2(\frac{1}{4}) + \frac{1}{2}log_2(\frac{1}{2})) = -(-\frac{1}{2} - \frac{1}{2} - \frac{1}{2}) = -(-1.5) = 1.5 > 1$$

3.2 Gini Impurity

3.2.1 Khái niệm

Gini Impurity là một phép đo được sử dụng để xây dựng cây quyết định nhằm xác định cách các tính năng của tập dữ liệu sẽ phân chia các node để tạo thành cây. Chính xác hơn, Gini Impurity của tập dữ liệu là một số nằm trong khoảng từ 0-0,5, cho biết khả năng dữ liệu ngẫu nhiên mới bị phân loại sai nếu nó được gán nhãn lớp ngẫu nhiên theo phân phối lớp trong tập dữ liệu.



Hình 3: Ví dụ về Gini Impority

Ví dụ: Giả sử bạn muốn tạo một bộ phân loại để xác định xem ai đó sẽ vỡ nợ trên thẻ tín dụng của họ hay không. Bạn có một số dữ liệu được gắn nhãn với các tính năng, chẳng hạn như các nhóm độ tuổi, thu nhập, xếp hạng tín dụng và liệu mỗi người có phải là sinh viên hay không. Để tìm tính năng tốt nhất cho lần phân chia đầu tiên của cây - node gốc - bạn có thể tính toán mức độ kém của mỗi tính năng chia dữ liệu vào đúng lớp, mặc định ("Yes") hoặc không mặc định ("No"). Tính toán này sẽ đo tạp chất của phần tách và đặc điểm có tạp chất thấp nhất sẽ xác định tính năng tốt nhất để tách node hiện tại. Quá trình này sẽ tiếp tục cho mỗi node tiếp theo bằng cách sử dụng các tính năng còn lại.

Trong hình trên, có minumum Gini Impurity, vì vậy được chọn làm gốc trong cây quyết định.

3.2.2 Công thức

Xem xét một tập dữ liệu D chứa các mẫu từ k các lớp. Xác suất của các mẫu thuộc lớp i tại một node nhất định có thể được ký hiệu là p_i. Sau đó Gini Impurity của tập dữ liệu D được định nghĩa là:

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

node có phân bố lớp đồng đều có tạp chất cao nhất. Tạp chất tối thiểu thu được khi tất cả các bản ghi thuộc cùng một lớp. Một số ví dụ được đưa ra trong bảng sau để chứng minh tính toán Gini Impurity.

	Count		Probability		Gini Impurity
	n_1	n_2	p_1	p_2	$1 - p_1^2 - p_2^2$
Node A	0	10	0	1	$1 - 0^2 - 1^2 = 0$
Node B	3	7	0.3	0.7	$1 - 0.3^2 - 0.7^2 = 0.42$
Node C	5	5	0.5	0.5	$1 - 0.5^2 - 0.5^2 = 0.5$

Hình 4: Tính toán Gini Impority trong 1 số ví dụ

Một thuộc tính có Gini Impurity nhỏ nhất được chọn để tách node.

Nếu một tập dữ liệu D được phân chia trên một thuộc tính A thành hai tập hợp con D_1 và D_2 với các kích cỡ n_1 và n_2 tương ứng, Gini Imputity có thể được định nghĩa là:

$$Gini_A(D) = \frac{n_1}{n}Gini(D_1) + \frac{n_2}{n}Gini(D_2)$$

Khi huấn luyện cây quyết định, thuộc tính cung cấp giá trị nhỏ nhất $Gini_A(D)$ được chọn để tách node.

Để có được thông tin về một thuộc tính, tạp chất có trọng số của các nhánh được trừ đi tạp chất ban đầu. Phần chia tốt nhất cũng có thể được chọn bằng cách tối đa hóa Gini gain . Gini gain được tính như sau:

$$\triangle Gini(A) = Gini(D) - Gini_A(D)$$

3.3 Information Gain

3.3.1 Khái niệm

Information Gain là thước đo lượng thông tin mà một đối tượng cung cấp về một lớp. Information Gain giúp xác định thứ tự của các thuộc tính trong các node của cây quyết định.

node chính được gọi là node cha, trong khi các node phụ được gọi là node con. Chúng ta có thể sử dụng Information Gain để xác định mức độ tốt của việc tách các node trong cây quyết định.

3.3.2 Công thức

Về mặt toán học, thông tin thu được có thể được biểu thị bằng công thức dưới đây:

$$Gain = E_{parent} - E_{children}$$

Gain đại diện cho Information Gain. E_{parent} là entropy của node cha và $E_{children}$ là entropy trung bình của các node con. Hãy sử dụng một ví dụ để hình dung Information Gain và tính toán của nó.

Giả sử chúng ta có một tập dữ liệu với hai lớp. Tập dữ liệu này có 5 ví dụ màu tím và 5 màu vàng. Giá trị ban đầu của entropy sẽ được tính bởi phương trình dưới đây:

$$E_{initial} = -((0.5log_20.5) + (0.5log_20.5)) = 1$$

Giả sử chúng ta chia tập dữ liệu thành hai nhánh. Một nhánh kết thúc có 4 giá trị trong khi nhánh kia có 6 giá trị. Nhánh bên trái có 4 màu tím trong khi nhánh bên phải có 5 màu vàng và 1 màu tím.

Nhánh bên trái thuộc cùng một lớp (màu tím) nên entropy bằng 0 vì tập dữ liệu là thuần túy. Như vậy, entropy của nhánh trái $E_{left} = 0$. Mặt khác, nhánh bên phải có 5 màu vàng và 1 màu tím. Như vậy:

$$E_{right} = -(\frac{5}{6}log_2(\frac{5}{6}) + \frac{1}{6}log_2(\frac{1}{6})) = 0.65 => E_{split} = 0.6 * 0.65 + 0.4 * 0 = 0.39$$

Entropy trước khi phân tách, ta gọi là entropy ban đầu $E_{initial} = 1$. Sau khi tách, giá trị hiện tại là 0.39. Từ đó suy ra:

$$Gain = 1 - 0.39 = 0.61$$

Entropy bị loại bỏ càng nhiều thì Information Gain càng lớn. Information Gain càng cao thì sự phân chia càng tốt.

4 Xây dựng thuật toán cây quyết định: ID3

4.1 Phương thức tính toán

Trong ID3, tổng các trọng số của entropy tại các leaf-node sau khi xây dựng decision tree được coi là hàm mất mát của decision tree đó. Các trọng số ở đây tỉ lệ với số điểm dữ liệu được phân vào mỗi node. Công việc của ID3 là tìm các cách phân chia hợp lý (thứ tự chọn thuộc tính hợp lý) sao cho hàm mất mát cuối cùng đạt giá trị càng nhỏ càng tốt. Việc này đạt được bằng cách chọn ra thuộc tính sao cho nếu dùng thuộc tính đó để phân chia, entropy tại mỗi bước giảm đi một lượng lớn nhất. Bài toán xây dựng một decision tree bằng ID3 có thể chia thành các bài toán nhỏ, trong mỗi bài toán, ta chỉ cần chọn ra thuộc tính giúp cho việc phân chia đạt kết quả tốt nhất. Mỗi bài toán nhỏ này tương ứng với việc phân chia dữ liệu trong một non-leaf node. Chúng ta sẽ xây dựng phương pháp tính toán dựa trên mỗi node này.

Xét một bài toán với C class khác nhau. Giả sử ta đang làm việc với một non-leaf node với các điểm dữ liệu tạo thành một tập S với số phần tử là $|S| = N$. Giả sử thêm rằng trong số N điểm dữ liệu này, $N_c, c = 1, 2, ..., C$ điểm thuộc vào class c. Xác suất để mỗi điểm dữ liệu rơi vào một class c được xấp xỉ bằng $\frac{N_c}{N}$(maximum likelihood estimation). Như vậy, entropy tại node này được tính bởi:

$$H(S) = -\sum_{c=1}^{C} \frac{N_c}{N} log_2(\frac{N_c}{N})$$

Tiếp theo, giả sử thuộc tính được chọn là x. Dựa trên x các điểm dữ liệu trong S được phân ra thành K child node $S1, S2, ..., S_k$ với số điểm trong mỗi child node lần lượt là $m_1, m_2, ..., m_k$. Ta định nghĩa

$$H(x, S) = \sum_{k=1}^{K} \frac{m_k}{N} H(S_k)$$

là tổng có trọng số entroy của mỗi child node–được tính tương tự như H(S). Việc lấy trọng số này là quan trọng vì các node thường có số lượng điểm khác nhau.

Tiếp theo, ta định nghĩa Information Gain dựa trên thuộc tính x:

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

Trong ID3, tại mỗi node, thuộc tính được chọn được xác định dựa trên:

$$x^* = argmax_x G(x, S) = argmin_x H(x, S)$$

tức thuộc tính khiến cho Information Gain đạt giá trị lớn nhất.

4.2 Ví dụ

Đây là một bảng dữ liệu được sử dụng rất nhiều trong các bài giảng về decision tree. Bảng dữ liệu này mô tả mối quan hệ giữa thời tiết trong 14 ngày (bốn cột đầu, không tính cột id) và việc một đội bóng có chơi bóng hay không (cột cuối cùng). Nói cách khác, ta phải dự đoán giá trị ở cột cuối cùng nếu biết giá trị của bốn cột còn lại.

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

Có bốn thuộc tính thời tiết:

1. *Outlook*: nhận một trong ba giá trị: sunny, overcast, rainy.

2. *Temperature*: nhận một trong ba giá trị: hot, cool, mild.

3. *Humidity*: nhận một trong hai giá trị: high, normal.

4. *Wind*: nhận một trong hai giá trị: weak, strong.

(Tổng cộng có 3×3×2×2=36 loại thời tiết khác nhau, trong đó 14 loại được thể hiện trong bảng.)

- Nếu $outlook = sunny$ và $humidity = high$ thì $play = no$.
- Nếu $outlook = rainy$ và $windy = true$ thì $play = no$.
- Nếu $outlook = overcast$ thì $play = yes$.
- Ngoài ra, nếu $humidity = normal$ thì $play = yes$.
- Ngoài ra, $play = yes$. Chúng ta sẽ cùng tìm thứ tự các thuộc tính bằng thuật toán ID3.

Trong 14 giá trị đầu ra ở Bảng trên, có 5 giá trị bằng *no* và 9 giá trị bằng *yes*. Entroy tại root node của bài toán là:

$$H(S) = -\frac{5}{14}log_2(\frac{5}{14}) - \frac{9}{14}log_2(\frac{9}{14}) \approx 0.65$$

Tiếp theo, chúng ta tính tổng có trọng số entropy của các child node nếu chọn một trong các thuộc tính $outlook, temperature, humidity, wind, play$ để phân chia dữ liệu.

Xét thuộc tính *outlook*. Thuộc tính này có thể nhận một trong ba giá trị $sunny, overcast, rainy$. Mỗi một giá trị sẽ tương ứng với một child node. Gọi tập hợp các điểm trong mỗi child node này lần lượt là S_s, S_o, S_r với tương ứng m_s, m_o, m_r phần tử. Sắp xếp lại Bảng ban đầu theo thuộc tính *outlook* ta đạt được ba Bảng nhỏ sau đây.

id	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
11	sunny	mild	normal	strong	yes

id	outlook	temperature	humidity	wind	play
4	rainy	mild	high	weak	yes
8	sunny	mild	high	weak	no
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
14	rainy	mild	high	strong	no

id	outlook	temperature	humidity	wind	play
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
9	sunny	cool	normal	weak	yes

Quan sát nhanh ta thấy rằng child node ứng với $outlook = overcast$ sẽ có entropy bằng 0 vì tất cả $m_o = 4$ output đều là yes. Hai child node còn lại với $m_s = m_r = 5$ có entropy khá cao vì tần suất output bằng yes hoặc no là xấp xỉ nhau. Tuy nhiên, hai child node này có thể được phân chia tiếp dựa trên hai thuộc tính $humidity$ và $wind$.

$$H(S_s) = -\frac{2}{5}log_2(\frac{2}{5}) - \frac{3}{5}log_2(\frac{3}{5}) \approx 0.673$$

$$H(S_o) = 0$$

$$H(S_r) = -\frac{2}{5}log_2(\frac{2}{5}) - \frac{3}{5}log_2(\frac{3}{5}) \approx 0.673$$

$$H(outlook, S) = \frac{5}{14}H(S_s) + \frac{4}{14}H(S_o) + \frac{5}{14}H(S_r) \approx 0.48$$

Các thuộc tính còn lại tính tương tự $outlook$, chúng sẽ bằng:

$$H(temperature, S) \approx 0.631$$

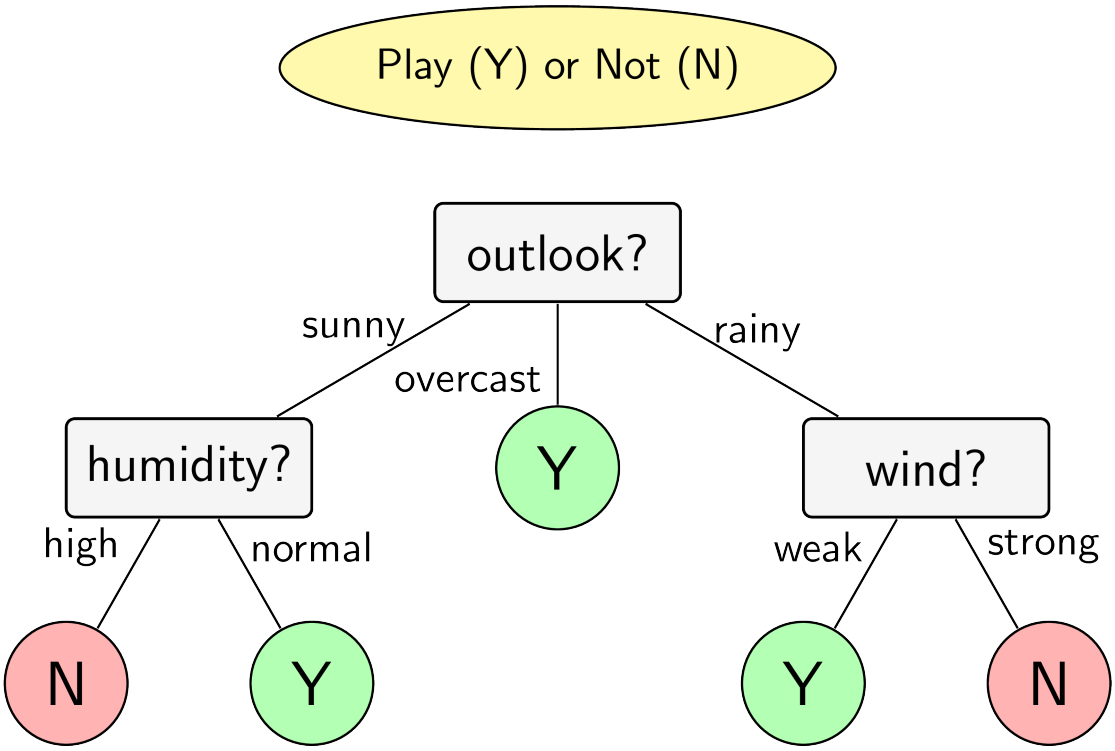
$$H(humidity, S) \approx 0.547$$

$$H(wind, S) \approx 0.618$$

Như vậy, thuộc tính cần chọn ở bước đầu tiên là $outlook$ vì $H(outlook, S)$ đạt giá trị nhỏ nhất (Information Gain là lớn nhất).

Sau bước phân chia đầu tiên này, ta nhận được ba child node với các phần tử như trong ba Bảng phân chia theo $outlook$. Child node thứ hai không cần phân chia tiếp vì nó đã tinh khiết. Với child node thứ nhất, ứng với $outlook = sunny$, kết quả tính được bằng ID3 sẽ cho chúng ta thuộc tính $humidity$ vì tổng trọng số của entropy sau bước này sẽ bằng 0 với output bằng yes khi và chỉ khi $humidity = normal$. Tương tự, child node ứng với $outlook = wind$ sẽ được tiếp tục phân chia bởi thuộc tính $wind$ với output bằng yes khi và chỉ khi $wind = weak$.

Như vậy, cây quyết định cho bài toán này dựa trên ID3 sẽ có dạng như trong Hình 5.



Hình 5: Decision tree cho bài toán ví dụ sử dụng thuật toán ID3

Link demo Google Colab:
`https://colab.research.google.com/drive/1DnvSRXc4CVuEnMfIYbAprGA4XFGxkR6H?usp=sharing`