

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN: TRÍ TUỆ NHÂN TẠO**

**Đề tài: Tìm hiểu ID3 và ứng dụng dự đoán nhu cầu mở tài khoản ký quỹ**

**Giảng viên hướng dẫn: ThS. Lê Thị Thủy**

**Lớp học phần: 20221IT6046011**

**Nhóm 15:**

- 1. Kiều Đức Anh - 2020601000**
- 2. Phan Tiến Phương - 2020602782**
- 3. Thân Ngọc Thiện - 2021602775**

**Hà Nội, năm 2022**

# MỤC LỤC

<b>CHƯƠNG 1: TỔNG QUAN</b>	<b>4</b>
1.1. Trí tuệ nhân tạo là gì?	4
1.2. Tìm hiểu về thuật toán ID3	5
1.2.1. Giới thiệu về cây quyết định	5
1.2.2. Thuật toán ID3	6
a, Ý Tưởng	6
b, Hàm số entropy	7
c, Thuật toán ID3	8
d, Ví dụ	10
e, Điều kiện dừng	14
1.2.3 Lập trình Python cho ID3	15
1.3. Một số thuật toán xây dựng cây quyết định khác	16
1.3.1. Thuật toán C4.5	16
1.3.2. Thuật toán CART	18
<b>CHƯƠNG 2. XÂY DỰNG MÔ HÌNH HUẤN LUYỆN</b>	<b>21</b>
2.1 Tổng quan bài toán và thuật toán sử dụng	21
2.1.1 Mô tả bài toán	21
2.1.2 Thuật toán sử dụng (ID3)	21
2.2 Bộ dữ liệu:	26
2.3 Cài đặt	27
2.3.1 Tiền xử lý dữ liệu	27
2.3.2 Thiết lập mô hình	31
a, Thư viện sklearn	31
b, Thực hiện mô hình bài toán	33
2.4 Kết quả và đánh giá	35
2.5 Vẽ cây quyết định	35
<b>TÀI LIỆU THAM KHẢO</b>	<b>37</b>

## LỜI MỞ ĐẦU

Ngày nay, Trí tuệ nhân tạo (AI) dần đi vào cuộc sống của chúng ta một cách mạnh mẽ và được xác định là lĩnh vực mũi nhọn trong ngành kinh tế các nước. Cách mạng công nghiệp 4.0 mở ra một thời kỳ mới với việc ứng dụng trí tuệ nhân tạo trong hầu hết các lĩnh vực trong đời sống, mang lại những thay đổi lớn trong xã hội, đặc biệt là trong kinh tế và khoa học ứng dụng.

Nội dung cuốn báo cáo này cung cấp cho người đọc những kiến thức chung nhất về trí tuệ nhân tạo, có một cái nhìn tổng quát về việc áp dụng thuật toán ID3 và Ứng dụng dự đoán nhu cầu mở tài khoản ký quỹ. Các vấn đề cụ thể trình bày trong cuốn báo cáo được chia thành 2 chương:

Chương 1: Tổng quan

Chương 2: Xây dựng mô hình huấn luyện

Nhóm em xin gửi lời cảm ơn đến cô Lê Thị Thủy - Giảng viên Trường ĐHCNHN, cô đã tận tình hướng dẫn và góp ý để hoàn thành bài báo cáo. Mặc dù đã rất cố gắng xong báo cáo này không tránh khỏi những thiếu sót, nhóm em mong nhận những đóng góp ý kiến của cô và bạn đọc để bài báo cáo được hoàn thiện hơn. Xin chân thành cảm ơn!

Nhóm sinh viên thực hiện

# CHƯƠNG 1: TỔNG QUAN

## 1.1. Trí tuệ nhân tạo là gì?

Trí tuệ nhân tạo (Artificial Intelligence) là một ngành thuộc lĩnh vực khoa học máy tính do con người lập trình tạo nên với mục tiêu giúp máy tính có thể tự động hóa các hành vi thông minh như con người.

Có 2 loại:

- Strong AI: Có thể tạo ra thiết bị có trí thông minh và các chương trình máy tính thông minh hơn người.
- Weak AI: Chương trình máy tính có thể mô phỏng các hành vi thông minh của con người.

### So sánh lập trình hệ thống và lập trình trí tuệ nhân tạo

Lập trình hệ thống	Lập trình AI
Dữ liệu + Thuật toán = Chương trình	Tri thức + Điều khiển = Chương trình
Xử lý dữ liệu	Xử lý dữ liệu định tính
Dữ liệu trong bộ nhớ được đánh địa chỉ số	Xử lý dựa trên tri thức cho phép dùng các thuật giải Heuristic, các cơ chế suy diễn
Xử lý theo các thuật toán	Tri thức được cấu trúc hóa, để trong bộ nhớ và làm việc theo ký hiệu
Định hướng xử lý theo các đại lượng, định lượng số	Định hướng xử lý các đại lượng định tính, các ký hiệu tượng trưng và danh sách
Xử lý tuần tự theo mẻ	Xử lý theo chế độ tương tác
Không giải thích trong quá trình thực hiện	Có giải thích hành vi của hệ thống trong quá trình thực hiện
Kết quả chính xác, không được mắc lỗi	Kết quả tốt, cho phép mắc lỗi

### Một số kỹ thuật Trí tuệ nhân tạo cơ bản :

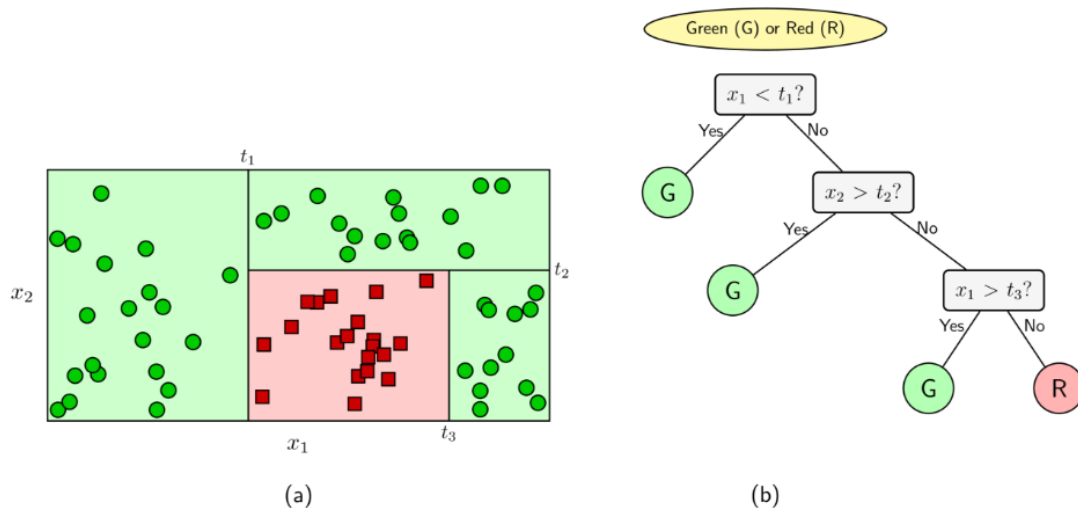
- Lý thuyết giải bài toán và suy diễn thông minh

- Lý thuyết tìm kiếm may rủi
- Các ngôn ngữ về TTNT
- Lý thuyết thể hiện tri thức và hệ chuyên gia
- Lý thuyết nhận dạng và xử lý tiếng nói
- Người máy
- Tâm lý học xử lý thông tin
- Xử lý danh sách, kỹ thuật đệ quy, kỹ thuật quay lui và xử lý cú pháp hình thức

## 1.2. Tìm hiểu về thuật toán ID3

### 1.2.1. Giới thiệu về cây quyết định

Cây quyết định được dùng để đưa ra tập luật if – then nhằm mục đích dự báo, giúp con người nhận biết về tập dữ liệu. Cây quyết định cho phép phân loại đối tượng tùy thuộc vào các điều kiện tại các nút trong cây, bắt đầu từ gốc cây tới các nút sát lá-Nút xác định phân loại đối tượng. Mỗi nút trong của cây xác định điều kiện đối với thuộc tính mô tả của đối tượng. Mỗi nhánh tương ứng với điều kiện: Nút (thuộc tính) bằng giá trị nào đó. Đối tượng được phân loại nhờ tích hợp các điều kiện bắt đầu từ nút gốc của cây và các thuộc tính mô tả với giá trị của thuộc tính đối tượng.



## Hình 2.1 Ví dụ về bài toán phân lớp sử dụng cây quyết định.

Ưu/nhược điểm của thuật toán cây quyết định

### **Ưu điểm**

Cây quyết định là một thuật toán đơn giản và phổ biến. Thuật toán này được sử dụng rộng rãi bởi những lợi ích của nó:

- Mô hình sinh ra các quy tắc dễ hiểu cho người đọc, tạo ra bộ luật với mỗi nhánh lá là một luật của cây.
- Dữ liệu đầu vào có thể là dữ liệu missing, không cần chuẩn hóa hoặc tạo biến giả
- Có thể làm việc với cả dữ liệu số và dữ liệu phân loại
- Có thể xác thực mô hình bằng cách sử dụng các kiểm tra thống kê
- Có khả năng làm việc với dữ liệu lớn

### **Nhược điểm**

Kèm với đó, cây quyết định cũng có những nhược điểm cụ thể:

- Mô hình cây quyết định phụ thuộc rất lớn vào dữ liệu của bạn. Thậm chí, với một sự thay đổi nhỏ trong bộ dữ liệu, cấu trúc mô hình cây quyết định có thể thay đổi hoàn toàn.
- Cây quyết định hay gặp vấn đề overfitting.

## **1.2.2. Thuật toán ID3**

### **a, Ý Tưởng**

ID3 là một thuật toán decision tree được áp dụng cho các bài toán classification mà tất cả các thuộc tính đều ở dạng categorical.

Trong ID3, chúng ta cần xác định thứ tự của thuộc tính cần được xem xét tại mỗi bước. Với các bài toán có nhiều thuộc tính và mỗi thuộc tính có nhiều giá trị khác nhau, việc tìm được nghiệm tối ưu thường là không khả thi. Thay vào đó, một

phương pháp đơn giản thường được sử dụng là tại mỗi bước, một thuộc tính tốt nhất sẽ được chọn ra dựa trên một tiêu chuẩn nào đó (chúng ta sẽ bàn sớm). Với mỗi thuộc tính được chọn, ta chia dữ liệu vào các child node tương ứng với các giá trị của thuộc tính đó rồi tiếp tục áp dụng phương pháp này cho mỗi child node. Việc chọn ra thuộc tính tốt nhất ở mỗi bước như thế này được gọi là cách chọn greedy (tham lam). Cách chọn này có thể không phải là tối ưu, nhưng trực giác cho chúng ta thấy rằng cách làm này sẽ gần với cách làm tối ưu. Ngoài ra, cách làm này khiến cho bài toán cần giải quyết trở nên đơn giản hơn.

Sau mỗi câu hỏi, dữ liệu được phân chia vào từng child node tương ứng với các câu trả lời cho câu hỏi đó. Câu hỏi ở đây chính là một thuộc tính, câu trả lời chính là giá trị của thuộc tính đó. Để đánh giá chất lượng của một cách phân chia, chúng ta cần đi tìm một phép đo.

Trước hết, thế nào là một phép phân chia tốt? Bằng trực giác, một phép phân chia là tốt nhất nếu dữ liệu trong mỗi child node hoàn toàn thuộc vào một class—khi đó child node này có thể được coi là một leaf node, tức ta không cần phân chia thêm nữa. Nếu dữ liệu trong các child node vẫn lẫn vào nhau theo tỉ lệ lớn, ta coi rằng phép phân chia đó chưa thực sự tốt. Từ nhận xét này, ta cần có một hàm số đo độ tinh khiết (purity), hoặc độ vẩn đục (impurity) của một phép phân chia. Hàm số này sẽ cho giá trị thấp nhất nếu dữ liệu trong mỗi child node nằm trong cùng một class (tinh khiết nhất), và cho giá trị cao nếu mỗi child node có chứa dữ liệu thuộc nhiều class khác nhau.

## **b, Hàm số entropy**

Cho một phân phối xác suất của một biến rời rạc  $x$  có thể nhận  $n$  giá trị khác nhau  $x_1, x_2, x_3, \dots, x_n$ . Giả sử rằng xác suất để  $x$  nhận các giá trị này là

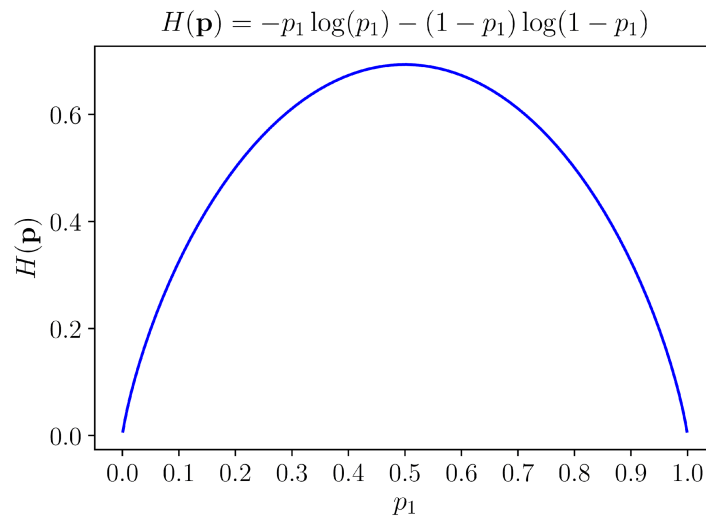
$p_i = p(x = x_i)$  với  $0 \leq p_i \leq 1$ ,  $\sum_{i=1}^n p_i = 1$ . Ký hiệu phân phối này là

$p = (p_1, p_2, \dots, p_n)$ . Entropy của phân phối này được định nghĩa là

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

Trong đó  $\log$  là logarit tự nhiên và quy ước  $0 \log \log(0) = 0$ .

Xét một ví dụ với  $n=2$  được cho trên hình 2.2. Trong trường hợp  $\mathbf{p}$  là tinh khiết nhất, tức một trong hai giá trị của  $p_i$  bằng 1, giá trị kia bằng 0, entropy của phân phối này là  $H(p) = 0$ . Khi  $\mathbf{p}$  là vắn đục nhất, tức cả hai giá trị  $p_i = 0.5$ , Hàm entropy đạt giá trị cao nhất.



Hình 2.2 Đồ thị của hàm entropy với  $n = 2$

Tổng quát lên với  $n > 2$ , hàm entropy đạt giá trị nhỏ nhất nếu có một giá trị  $p_i = 1$ , đạt giá trị lớn nhất nếu tất cả các  $p_i$  bằng nhau. Những tính chất này của hàm entropy khiến nó được sử dụng trong việc đo độ vắn đục của một phép phân chia của ID3. Vì lý do này, ID3 còn được gọi là entropy-based decision tree.

### c, Thuật toán ID3

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. Như đã đề cập, 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,  $N_c, c = 1, 2, 3, \dots, C$  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:

$$H(S) = - \sum_{c=1}^C \frac{N_c}{N} \log \log \left( \frac{N_c}{N} \right) \quad (2)$$

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  $S_1, S_2, \dots, S_K$ , với số điểm trong mỗi child node lần lượt là  $m_1, m_2, \dots, m_K$ . Ta định nghĩa

$$H(x, S) = - \sum_{k=1}^K \frac{m_k}{N} H(S_k) \quad (3)$$

là tổng có trọng số entropy của mỗi child node—được tính tương tự như (2). 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:

$$x^* = \arg \arg H(x, S)$$

tức thuộc tính khiến cho information gain đạt giá trị lớn nhất.

#### d, Ví dụ

Để mọi thứ được rõ ràng hơn, chúng ta cùng xem ví dụ với dữ liệu huấn luyện được cho trong Bảng dưới đây. Bảng dữ liệu này được lấy từ cuốn sách [Data Mining: Practical Machine Learning Tools and Techniques](#), trang 11. 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  $3 \times 3 \times 2 \times 2 = 36$  loại thời tiết khác nhau, trong đó 14 loại được thể hiện trong bảng.)

Đây có thể được coi là một bài toán dự đoán liệu đội bóng có chơi bóng không dựa trên các quan sát thời tiết. Ở đây, các quan sát đều ở dạng categorical. Cách dự đoán dưới đây tương đối đơn giản và khá chính xác, có thể không phải là cách ra quyết định tốt nhất:

- 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 năm giá trị bằng *no* và chín giá trị bằng *yes*. Entropy tại *root node* của bài toán là:

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

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
3	overcast	hot	high	weak	yes
7	overcast	cool	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes

id	outlook	temperature	humidity	wind	play
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
10	rainy	mild	normal	weak	yes
14	rainy	mild	high	strong	no

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*.

Bạn đọc có thể kiểm tra được rằng

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

$$H(S_o) = 0$$

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

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

Xét thuộc tính *temperature*, ta có phân chia như các Bảng dưới đây.

id	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
13	overcast	hot	normal	weak	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

Gọi  $S_h, S_m, S_c$  là ba tập con tương ứng với *temperature* bằng *hot, mild, cool*. Bạn đọc có thể tính được

$$H(S_h) = -\frac{2}{4}\log\log\left(\frac{2}{4}\right) - \frac{2}{4}\log\log\left(\frac{2}{4}\right) \approx 0.693$$

$$H(S_m) = -\frac{4}{6}\log\log\left(\frac{4}{6}\right) - \frac{2}{6}\log\log\left(\frac{2}{6}\right) \approx 0.637$$

$$H(S_c) = -\frac{3}{4}\log\log\left(\frac{3}{4}\right) - \frac{1}{4}\log\log\left(\frac{1}{4}\right) \approx 0.562$$

$$H(\text{temperature}, S) = \frac{4}{14}H(S_h) + \frac{6}{14}H(S_m) + \frac{4}{14}H(S_c) \approx 0.48$$

Việc tính toán với hai thuộc tính còn lại được dành cho bạn đọc. Nếu các kết quả là giống nhau, chúng sẽ bằng:

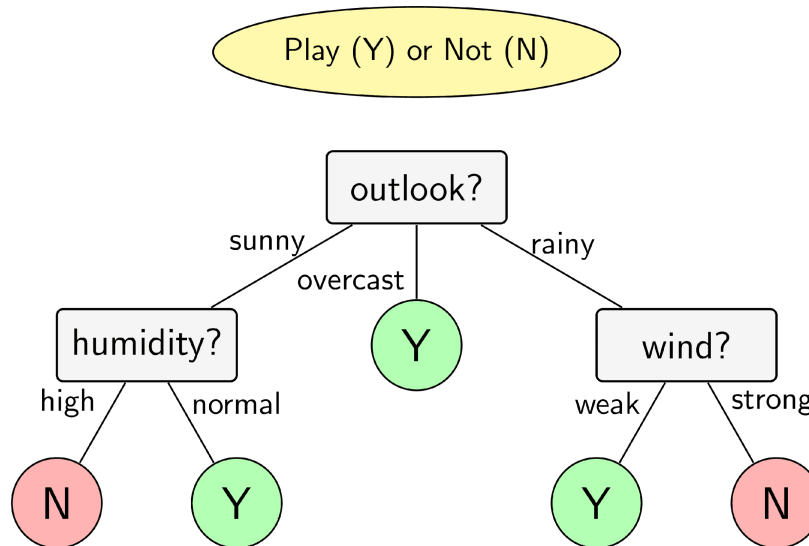
$$H(\text{Humidity}, S) \approx 0.547, H(\text{Wind}, S) \approx 0.618$$

Như vậy, thuộc tính cần chọn ở bước đầu tiên là *outlook* vì  $H(\text{Outlook}, S)$  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 *trùng khớp*. 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 2.3



Hình 2.3 Decision tree cho bài toán ví dụ sử dụng thuật toán ID3.

### e, Điều kiện dừng

Trong các thuật toán decision tree nói chung và ID3 nói riêng, nếu ta tiếp tục phân chia các node *chưa tinh khiết*, ta sẽ thu được một tree mà mọi điểm trong tập huấn luyện đều được dự đoán đúng (giả sử rằng không có hai input giống nhau nào cho output khác nhau). Khi đó, tree có thể sẽ rất phức tạp (nhiều node) với nhiều leaf node chỉ có một vài điểm dữ liệu. Như vậy, nhiều khả năng overfitting sẽ xảy ra.

Để tránh overfitting, một trong số các phương pháp sau có thể được sử dụng. Tại một node, nếu một trong số các điều kiện sau đây xảy ra, ta không tiếp tục phân chia node đó và coi nó là một leaf node:

- Nếu node đó có entropy bằng 0, tức mọi điểm trong node đều thuộc một class.
- Nếu node đó có số phần tử nhỏ hơn một ngưỡng nào đó. Trong trường hợp này, ta chấp nhận có một số điểm bị phân lớp sai để tránh overfitting. Class cho leaf node này có thể được xác định dựa trên class chiếm đa số trong node.
- Nếu khoảng cách từ node đó đến root node đạt tới một giá trị nào đó. Việc hạn chế *chiều sâu của tree* này làm giảm độ phức tạp của tree và phần nào giúp tránh overfitting.
- Nếu tổng số leaf node vượt quá một ngưỡng nào đó.
- Nếu việc phân chia node đó không làm giảm entropy quá nhiều (information gain nhỏ hơn một ngưỡng nào đó).

Ngoài các phương pháp trên, một phương pháp phổ biến khác được sử dụng để tránh overfitting là *pruning*, tạm dịch là *cắt tỉa*.

### 1.2.3 Lập trình Python cho ID3

Module *DecisionTree* trong sklearn không thực hiện thuật toán ID3 mà là một thuật toán khác được đề cập trong bài tiếp theo. Phiên bản hiện tại trong sklearn chưa hỗ trợ các thuộc tính ở dạng categorical. Với dữ liệu có thuộc tính categorical, cách thường dùng là chuyển đổi các thuộc tính đó sang dạng numerical (1, 2, 3 cho mỗi giá trị). Chẳng hạn, các giá trị *hot*, *mild*, *cool* có thể lần lượt được thay bằng 1, 2, 3. Cách làm này có hạn chế vì trong cách chuyển đổi này, *mild* là trung bình cộng của *hot* và *cool*, nhưng nếu thứ tự các giá trị được đặt khác đi, việc chuyển đổi có thể ảnh hưởng lớn tới kết quả. Nhắc lại rằng các thuộc tính categorical, ví dụ màu sắc, thường không có tính thứ tự.

#### Xây dựng class *TreeNode*

```
from __future__ import print_function
import numpy as np
import pandas as pd

class TreeNode(object):
    def __init__(self, ids = None, children = [], entropy = 0, depth = 0):
        self.ids = ids          # index of data in this node
        self.entropy = entropy  # entropy, will fill later
        self.depth = depth      # distance to root node
        self.split_attribute = None # which attribute is chosen, it non-leaf
        self.children = children # list of its child nodes
        self.order = None       # order of values of split_attribute in children
        self.label = None       # label of node if it is a leaf

    def set_properties(self, split_attribute, order):
        self.split_attribute = split_attribute # split at which attribute
        self.order = order # order of this node's children

    def set_label(self, label):
        self.label = label # set label if the node is a leaf
```

#### Hàm tính entropy dựa trên tần suất

Trong hàm này, chúng ta phải chú ý bỏ các tần suất bằng 0 đi vì logarit tại đây không xác định.

```
def entropy(freq):
    # remove prob 0
    freq_0 = freq[np.array(freq).nonzero()[0]]
    prob_0 = freq_0/float(freq_0.sum())
    return -np.sum(prob_0*np.log(prob_0))
```

Dữ liệu trong ví dụ được lưu trong file weather.csv. Việc huấn luyện decision tree dựa trên ID3 cho tập dữ liệu này và đầu ra dự đoán cho training set được cho bởi

```
df = pd.DataFrame.from_csv('weather.csv')
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
tree = DecisionTreeID3(max_depth = 3, min_samples_split = 2)
tree.fit(X, y)
print(tree.predict(X))
```

Kết quả

```
['no', 'no', 'yes', 'yes', 'yes', 'no', 'yes', 'no', 'yes', 'yes', 'yes', 'yes',
```

## 1.3. Một số thuật toán xây dựng cây quyết định khác

### 1.3.1. Thuật toán C4.5

#### a, Khái niệm

C4.5 được xem là phiên bản nâng cấp của ID3 với khả năng xử lý được cả dữ liệu định lượng dạng liên tục (continuous data) và cả dữ liệu định tính, và là thuật toán Decision tree tiêu biểu. C4.5 sử dụng thêm công thức Gain ratio để khắc phục các khuyết điểm của Information Gain của ID3 trong việc lựa chọn cách phân nhánh tối ưu.

#### b, Cách thức hoạt động

Để xây dựng cây quyết định bằng thuật toán C4.5 cần hai bước:



- **Phát triển cây quyết định:** đi từ gốc, đến các nhánh, phát triển quy nạp theo hình thức chia để trị.

**Bước 1.** Chọn thuộc tính “tốt” nhất bằng một độ đo đã định trước

**Bước 2.** Phát triển cây bằng việc thêm các nhánh tương ứng với từng giá trị của thuộc tính đã chọn

**Bước 3.** Sắp xếp, phân chia tập dữ liệu đào tạo tới node con

**bước 4.** Nếu các ví dụ được phân lớp rõ ràng thì dừng.

**Ngược lại:** lặp lại bước 1 tới bước 4 cho từng node con

- **Cắt tỉa cây:** nhằm đơn giản hóa, khái quát hóa cây, tăng độ chính xác

**Thuật toán Hunt sử dụng trong C4.5, CDP,...**

$S = \{S_1, S_2, \dots, S_n\}$  là tập dữ liệu đào tạo

$C = \{C_1, C_2, \dots, C_m\}$  là tập các lớp

**Trường hợp 1:**  $S_i$  ( $i=1 \dots n$ ) thuộc về  $C_j \Rightarrow$  Cây quyết định là 1 lá ứng  $C_j$ .

**Trường hợp 2:**  $S$  thuộc về nhiều lớp trong  $C$ .

- Chọn 1 test trên thuộc tính đơn có nhiều giá trị  $O = \{O_1, \dots, O_k\}$  (k thường bằng 2).
- Test từ gốc của cây, mỗi  $O_i$  tạo thành 1 nhánh, chia  $S$  thành các tập con có giá trị thuộc tính =  $O_i$ . Đệ quy cho từng tập con  $\Rightarrow$  cây quyết định gồm nhiều nhánh, mỗi nhánh tương ứng với  $O_i$

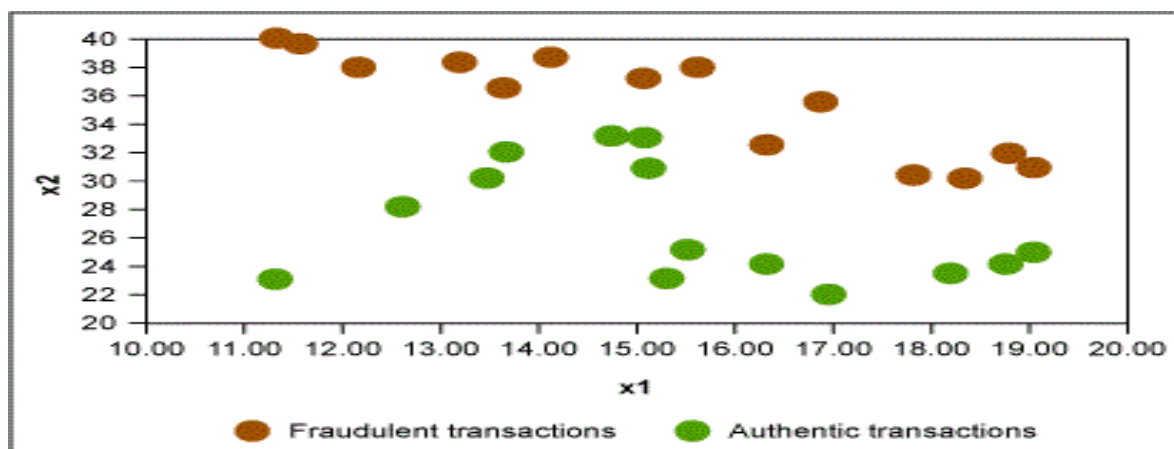
### 1.3.2. Thuật toán CART

#### a, Khái niệm

CART (Classification and Regression Tree) dịch ra là cây phân loại và hồi quy. CART là một kỹ thuật học máy có giám sát phổ biến được áp dụng để dự đoán biến mục tiêu định tính (categorical target variable), tạo cây phân loại hoặc biến mục tiêu liên tục (continuous target variable), tạo ra cây hồi quy. Ưu điểm của thuật toán này là có thể sử dụng cho cả bài toán phân loại và hồi quy.

#### b, Cách thức hoạt động

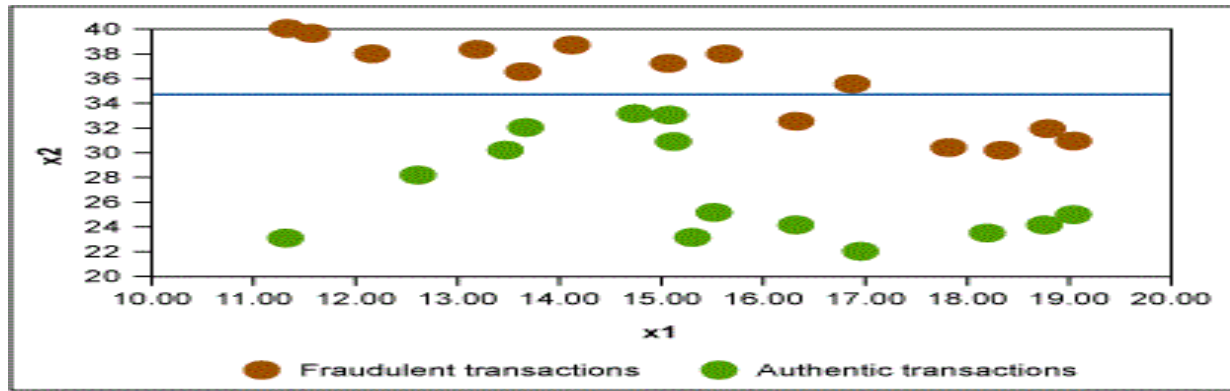
Giả sử có một tập hợp các giao dịch thẻ tín dụng được dán nhãn là gian lận hoặc xác thực. Có hai thuộc tính của mỗi giao dịch: số tiền (giao dịch) và độ tuổi của khách hàng. Hình 1 hiển thị một bản đồ ví dụ về các giao dịch gian lận và xác thực.



Hình 1: Giao dịch gian lận và xác thực

Thuật toán CART hoạt động để tìm biến độc lập tạo ra nhóm đồng nhất tốt nhất khi tách dữ liệu. Đối với một bài toán phân loại trong đó biến phản hồi có tính phân loại, điều này được quyết định bằng cách tính toán thông tin thu được dựa trên entropy do sự phân tách. Đối với phản hồi số, tính đồng nhất được đo lường bằng

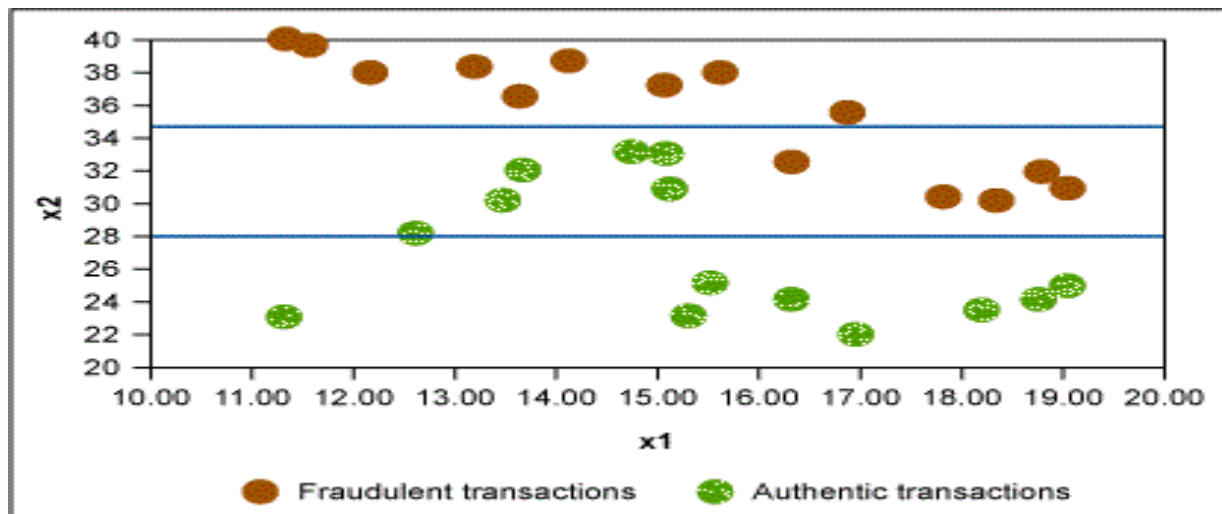
các thống kê như độ lệch chuẩn hoặc phương sai.



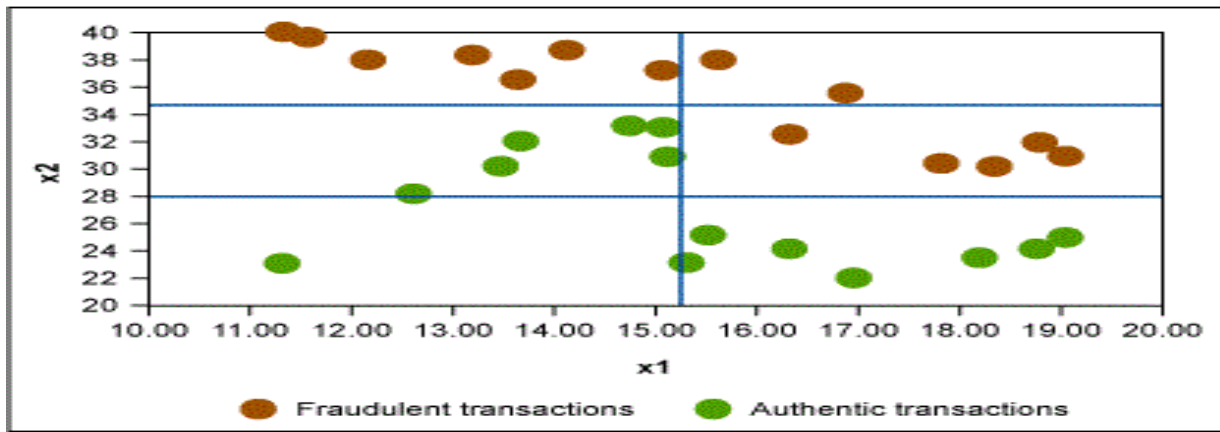
**Hình 2: Tách các giao dịch**

Hai tham số quan trọng của kỹ thuật CART là tiêu chí tách tối thiểu và tham số phức tạp ( $C_p$ ). Tiêu chí phân tách tối thiểu là số lượng bản ghi tối thiểu phải có trong một nút trước khi có thể thử phân tách. Điều này phải được chỉ định ngay từ đầu.  $C_p$  là một tham số phức tạp để tránh chia nhỏ những nút rõ ràng là không đáng giá. Một cách khác để xem xét các tham số này là  $C_p$  giá trị được xác định sau khi “trồng cây” và giá trị tối ưu được sử dụng để “cắt tỉa cây”.

Trong ví dụ này, Hình 2 cho thấy quy tắc đầu tiên được hình thành là  $x2 > 35 \rightarrow$  giao dịch gian lận. Tương tự, các quy tắc khác được hình thành như trong Hình 3 và Hình 4.

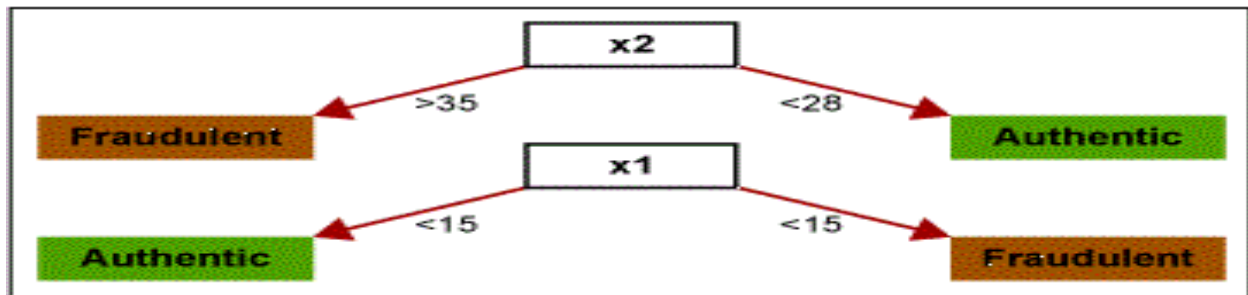


**Hình 3: Hai tách**



**Hình 4: Ba phân tách**

Bằng cách này, thuật toán CART tiếp tục phân chia tập dữ liệu cho đến khi mỗi nút “lá” còn lại với số lượng bản ghi tối thiểu như được chỉ định bởi tiêu chí phân chia tối thiểu. Điều này dẫn đến một cấu trúc giống cây như thể hiện trong Hình 5. Các giá trị sau đó được lập biểu đồ dựa trên các cấp độ khác nhau của cây và giá trị tối ưu được sử dụng để cắt tỉa cây.



**Hình 5: Kết quả phân tích CART**

Để tìm hiểu kỹ hơn về thuật toán này, ta có thể tìm kiếm ở các nguồn sau:

- [Sử dụng Cây phân loại và hồi quy \(CART\) để có thông tin chi tiết về dữ liệu nhanh,](#)
- [Cây phân loại và hồi quy \(Classification and Regression Tree - CART\) là gì?,](#)
- [Thuật toán cây quyết định \(P.2\): CART \(Gini index\)](#)

## CHƯƠNG 2. XÂY DỰNG MÔ HÌNH HUẤN LUYỆN

### 2.1 Tổng quan bài toán và thuật toán sử dụng

#### 2.1.1 Mô tả bài toán

- Tại một ngân hàng ở Bồ Đào Nha, sau khi điều tra, họ phát hiện ra rằng khách hàng của họ có dự kiến mở tài khoản ký quỹ hay không dựa trên một số yếu tố nhất định. Vì vậy, ngân hàng muốn xác định những khách hàng hiện tại có cơ hội đăng ký tài khoản cao hơn và tập trung nỗ lực tiếp thị vào những khách hàng đó.
- Yêu cầu đặt ra là dùng trí tuệ nhân tạo, dựa trên những thông số đã thu thập để xây dựng lên một mô hình dự đoán với tỷ lệ chính xác càng lớn càng tốt. Cụ thể các yếu tố quyết định đến việc mở tài khoản sẽ được trình bày kỹ hơn ở phần sau.

#### 2.1.2 Thuật toán sử dụng (ID3)

Để bài toán thực hiện một cách đơn giản, phù hợp với những kiến thức đã học và tìm hiểu, nhóm chúng em quyết định sử dụng thuật toán ID3 cho bài toán này. Các bước thực hiện bài toán trên bộ dữ liệu nhỏ như sau:

ST T	Từng mở thẻ	Mua nhà	Nợ	Liên hệ	QC trước	Quyết định
1	no	yes	no	cellular	nonexistent	NO
2	unknown	no	no	telephone	nonexistent	NO
3	no	yes	no	cellular	nonexistent	NO
4	no	no	yes	cellular	success	YES
5	yes	no	no	telephone	nonexistent	NO
6	no	no	no	cellular	failure	YES

7	no	no	yes	cellular	nonexistent	YES
8	unknown	yes	yes	cellular	nonexistent	NO
9	no	yes	no	cellular	success	?
10	unknown	yes	no	cellular	nonexistent	?
11	no	no	no	cellular	failure	?

Có 5 thuộc tính về cơ bản ảnh hưởng đến quyết định của khách hàng:

1. “Tùng mở thẻ” nhận 3 giá trị: *yes, no, unknown*
2. “Mua nhà” nhận 2 giá trị : *yes, no*
3. “Nợ” nhận 2 giá trị: *yes, no*
4. “Liên hệ” nhận 2 giá trị: *cellular, telephone*
5. “QC trước” nhận 3 giá trị: *nonexistent, success, failure*

#### Lần lặp 1:

- Tập dữ liệu ban đầu:  $S = [3+, 5-]$

$$H(S) = \frac{-3}{8} * \log_2\left(\frac{3}{8}\right) - \frac{5}{8} * \log_2\left(\frac{5}{8}\right) = 0.954434$$

- Với thuộc tính “Tùng mở thẻ”:

value(Tùng mở thẻ) = {yes, no, unknown}

$$S = 0.954434$$

$$S_{yes} = [0+, 1-] \Rightarrow H(S_{yes}) = 0$$

$$S_{no} = [3+, 2-] \Rightarrow H(S_{no}) =$$

$$\frac{-3}{5} * \log_2\left(\frac{3}{5}\right) - \frac{2}{5} * \log_2\left(\frac{2}{5}\right) = 0.970905$$

$$S_{unknown} = [0+, 2-] = 0$$

$$IG(S, \text{Tùng mở thẻ}) = H(S) -$$

$$\left(\frac{1}{8} * 0 + \frac{2}{8} * 0 + \frac{5}{8} * 0.970905\right) = 0.347618$$

- Với thuộc tính “Mua nhà”:

value(Mua nhà) = {yes, no}

$$S = 0.954434$$

$$S_{yes} = [3 -, 0 +] \Rightarrow H(S_{yes}) = 0$$

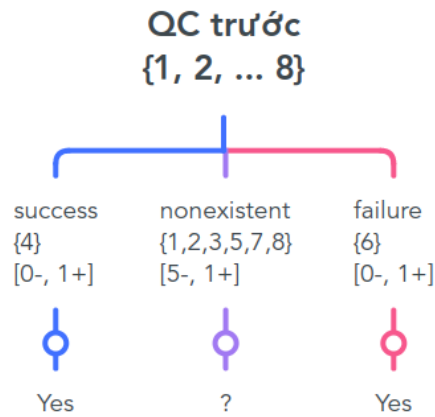
$$S_{no} = [2 -, 3 +] \Rightarrow H(S_{no}) =$$

$$\frac{-2}{5} * \log_2\left(\frac{2}{5}\right) - \frac{3}{5} * \log_2\left(\frac{3}{5}\right) = 0.970905$$

$$IG(S, \text{Mua nhà}) = H(S) - \left(\frac{3}{8} * 0 + \frac{5}{8} * 0.970905\right) = 0.347618$$

- Với thuộc tính “nợ”:  
 $\text{value(Nợ)} = \{\text{yes, no}\}$   
 $S = 0.954434$   
 $S_{\text{yes}} = [1 -, 2 +] \Rightarrow H(S_{\text{yes}}) = 0.918295$   
 $S_{\text{no}} = [4 -, 1 +] \Rightarrow H(S_{\text{no}}) = 0.721928$   
 $\text{IG}(S, \text{Nợ}) = H(S) - (\frac{3}{8} * 0.918295 + \frac{5}{8} * 0.721928) = 0.158868$
- Với thuộc tính “Liên hệ”:  
 $\text{value(Liên hệ)} = \{\text{cellular, telephone}\}$   
 $S = 0.954434$   
 $S_{\text{cellular}} = [3 -, 3 +] \Rightarrow H(S_{\text{cellular}}) = 1$   
 $S_{\text{telephone}} = [2 -, 0 +] \Rightarrow H(S_{\text{telephone}}) = 0$   
 $\text{IG}(S, \text{Liên hệ}) = H(S) - (\frac{6}{8} * 1 + \frac{2}{8} * 0) = 0.204434$
- Với thuộc tính “QC trước”:  
 $\text{value(QC trước)} = \{\text{nonexistent, success, failure}\}$   
 $S = 0.954434$   
 $S_{\text{nonexistent}} = [5 -, 1 +] \Rightarrow H(S_{\text{nonexistent}}) = 0.650022$   
 $S_{\text{success}} = [0 -, 1 +] \Rightarrow H(S_{\text{success}}) = 0$   
 $S_{\text{failure}} = [0 -, 1 +] \Rightarrow H(S_{\text{failure}}) = 0$   
 $\text{IG}(S, \text{QC trước}) = H(S) -$   
 $(\frac{6}{8} * 0.650022 + \frac{1}{8} * 0 + \frac{1}{8} * 0) = 0.466917$

$\text{IG}(S, \text{QC trước})$  là cao nhất nên ta chọn làm nút gốc.



### Lần lặp 2:

- $H(S) = \frac{-5}{6} * \log_2(\frac{5}{6}) - \frac{1}{6} * \log_2(\frac{1}{6}) = 0.650022$
- Với thuộc tính “Tùng mở thẻ”:  
 $\text{value}(\text{Tùng mở thẻ}) = \{\text{yes}, \text{no}, \text{unknown}\}$   
 $S = 0.650022$   
 $S_{\text{yes}} = [0+, 1-] \Rightarrow H(S_{\text{yes}}) = 0$   
 $S_{\text{no}} = [1+, 2-] \Rightarrow H(S_{\text{no}}) =$   
 $\frac{-1}{3} * \log_2(\frac{1}{3}) - \frac{2}{3} * \log_2(\frac{2}{3}) = 0.918295$   
 $S_{\text{unknown}} = [0+, 2-] = 0$   
 $\text{IG}(S, \text{Tùng mở thẻ}) = H(S) -$   
 $(\frac{1}{6} * 0 + \frac{2}{6} * 0 + \frac{3}{6} * 0.918295) = 0.190874$
- Với thuộc tính “Mua nhà”  
 $\text{value}(\text{Mua nhà}) = \{\text{yes}, \text{no}\}$   
 $S = 0.650022$   
 $S_{\text{yes}} = [3 -, 0 +] \Rightarrow H(S_{\text{yes}}) = 0$   
 $S_{\text{no}} = [2 -, 1 +] \Rightarrow H(S_{\text{no}}) =$   
 $\frac{-2}{3} * \log_2(\frac{2}{3}) - \frac{1}{3} * \log_2(\frac{1}{3}) = 0.918295$   
 $\text{IG}(S, \text{Mua nhà}) = H(S) - (\frac{3}{6} * 0 + \frac{3}{6} * 0.918295) = 0.190874$
- Với thuộc tính “nợ”:  
 $\text{value}(\text{Nợ}) = \{\text{yes}, \text{no}\}$   
 $S = 0.650022$



$$S_{yes} = [1 -, 1 +] \Rightarrow H(S_{yes}) = 1$$

$$S_{no} = [4 -, 0 +] \Rightarrow H(S_{no}) = 0$$

$$IG(S, Nợ) = H(S) - \left(\frac{4}{6} * 0.918295 + \frac{2}{6} * 1\right) = 0.316688$$

- Với thuộc tính “Liên hệ”:

value(Liên hệ) = {cellular, telephone}

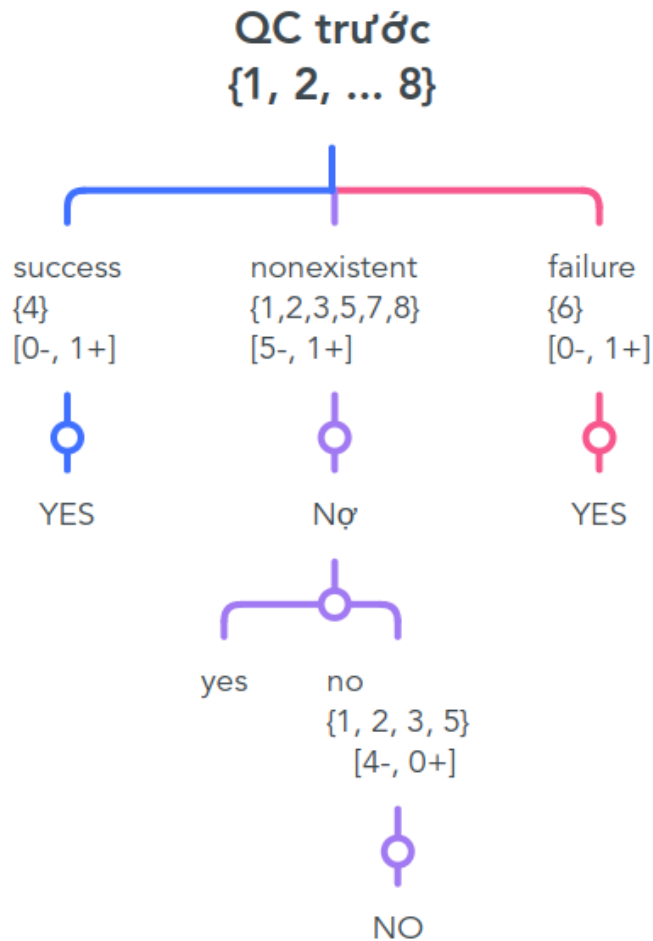
S = 0.650022

$$S_{cellular} = [3 -, 1 +] \Rightarrow H(S_{cellular}) = 0.811278$$

$$S_{telephone} = [2 -, 0 +] \Rightarrow H(S_{telephone}) = 0$$

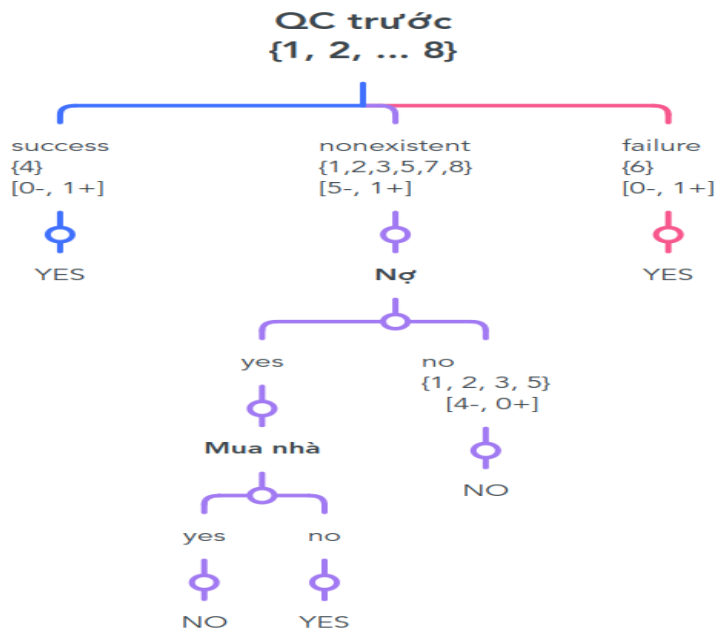
$$IG(S, \text{Liên hệ}) = H(S) - \left(\frac{4}{6} * 0.811278 + \frac{2}{6} * 0\right) = 0.10917$$

IG(S, Nợ) là cao nhất nên ta chọn làm nút gốc tiếp theo.



Lần lặp 3:

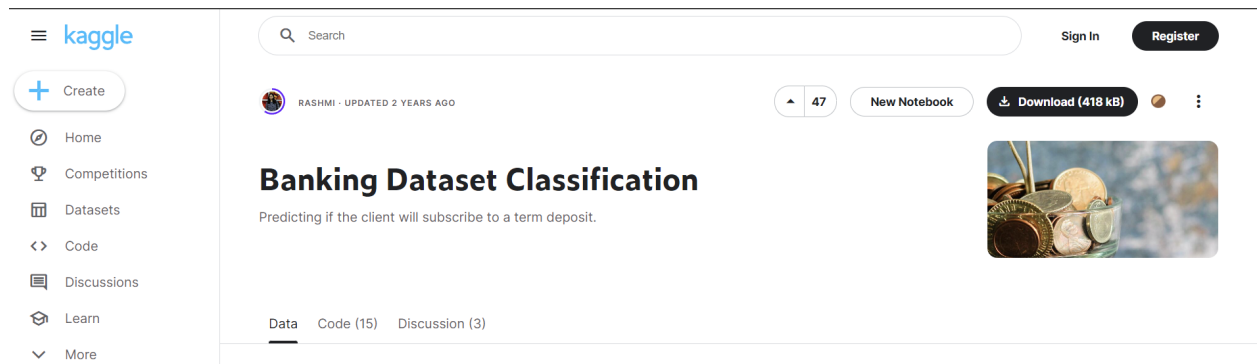
Tương tự ta được:



Như vậy có thể thấy ? ở bộ dữ liệu 9 và 11 là YES  
 Với bộ dữ liệu 10: ? ở đây là NO vì thuộc tính “Nợ” là no.

## 2.2 Bộ dữ liệu:

- Dữ liệu được lấy trên trang kaggle.com - là một trong những trang uy tín, có danh tiếng để tìm dữ liệu phục vụ cho việc luyện tập Trí tuệ nhân tạo.



+ Mô tả dữ liệu:

Feature	Feature_Type	Description
age	numeric	age of a person
job	Categorical,nominal	type of job

		('admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
marital	Categorical,nominal	marital status ( 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
education	Categorical,nominal	('basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
default	Categorical,nominal	has credit in default? ('no','yes','unknown')
housing	Categorical,nominal	has housing loan? ('no','yes','unknown')
loan	Categorical,nominal	has personal loan? ('no','yes','unknown')
contact	Categorical,nominal	contact communication type ('cellular','telephone')
month	Categorical,nominal	last contact month of year ('jan', 'feb', 'mar', ..., 'nov', 'dec')
dayofweek	Categorical,nominal	last contact day of the week ( 'mon','tue','wed','thu','fri')
poutcome	categorical, nominal	'failure','nonexistent','success'

+ Kết quả đầu ra:

y	binary	'yes','no'
---	--------	------------

## 2.3 Cài đặt

### 2.3.1 Tiền xử lý dữ liệu

- Khai báo một số thư viện cần thiết trong quá trình nhập và xử lý dữ liệu:

```
# import các thư viện cần thiết
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

- Kết nối Google Colab với dữ liệu được lưu trong Google Driver. Điều này tránh cho việc trong quá trình training và sử dụng lại code phải tải lên lại dữ liệu

```

✓ [8] #kết nối colab với dữ liệu của Google Driver
2s from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

✓ [9] %cd /content/drive/MyDrive/ColabNotebooks/Datasets
0s /content/drive/MyDrive/ColabNotebooks/Datasets

```

liệu lên Google Colab (do dữ liệu tập huấn tạm thời lưu trong thư mục sample data, sau một khoảng thời gian sẽ tự động bị xóa)

- Nhập dữ liệu và hiển thị thử 5 dòng đầu của dữ liệu đưa vào

```

✓ [10] df = pd.read_csv("new_train1.csv")
0s

```

df.head()

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	previous	poutcome	y
0	49	blue-collar	married	basic.9y	unknown	no	no	cellular	nov	wed	227	0	nonexistent	no
1	37	entrepreneur	married	university.degree	no	no	no	telephone	nov	wed	202	1	failure	no
2	78	retired	married	basic.4y	no	no	no	cellular	jul	mon	1148	0	nonexistent	yes
3	36	admin.	married	university.degree	no	yes	no	telephone	may	mon	120	0	nonexistent	no
4	59	retired	divorced	university.degree	no	no	no	cellular	jun	tue	368	0	nonexistent	no

- Chuẩn hóa các dữ liệu về dạng số:

```

for i in range(0, len(df.age)):
    if 10 < df.age[i] < 20:
        df.age[i] = 0
    elif 20 < df.age[i] < 35:
        df.age[i] = 1
    elif 35 < df.age[i] < 45:
        df.age[i] = 2
    elif 40 < df.age[i] < 55:
        df.age[i] = 3
    elif 55 < df.age[i] < 65:
        df.age[i] = 4
    elif 65 < df.age[i] < 75:
        df.age[i] = 5
    elif 75 < df.age[i] < 85:
        df.age[i] = 6
    elif 75 < df.age[i] < 95:
        df.age[i] = 7
    else:
        df.age[i] = 8

```

```
df.age[i]+=1
print("age",i," is changed to",df.age[i])
```

+ Bắt đầu với thuộc tính tuổi (age). Ở thuộc tính này ta chia dữ liệu thành 9 nhóm như sau:

10 - 20, 20 - 35, 35 - 45, 45 - 55, 55 - 65, 65 - 75, 75 - 85, 85 - 95, và lớn hơn 95 tuổi.

```

↳ age 32892 is changed to 9
   age 32893 is changed to 2
   age 32894 is changed to 4
   age 32895 is changed to 3
   age 32896 is changed to 2
   age 32897 is changed to 4
   age 32898 is changed to 4

```

Trong quá trình chuẩn hóa dữ liệu, ta in ra các bước mà chương trình đã thực hiện được để kiểm soát được chương trình đang làm công việc gì, kết quả ra sao.

+ Tương tự như thuộc tính ‘age’ ta tiếp tục chuẩn hóa các thuộc tính tiếp theo

```
['job', 'marital', 'education', 'default', 'housing', 'loan',
'contact', 'month', 'day_of_week', 'poutcome']
```

```
//job
for i in range(0, len(df.job)):
    if df.job[i] == 'admin.':
        df.job[i] = 0
    elif df.job[i] == 'blue-collar':
        df.job[i] = 1
    elif df.job[i] == 'entrepreneur':
        df.job[i] = 2
    elif df.job[i] == 'housemaid':
        df.job[i] = 3
    elif df.job[i] == 'management':
        df.job[i] = 4
    elif df.job[i] == 'retired':
        df.job[i] = 5
    elif df.job[i] == 'self-employed':
        df.job[i] = 6
    elif df.job[i] == 'services':
        df.job[i] = 7
    elif df.job[i] == 'student':
        df.job[i] = 8
```

```
//default
for i in range(0, len(df.default)):
    if df.default[i] == 'no':
        df.default[i] = 0
    elif df.default[i] == 'yes':
        df.default[i] = 1
    elif df.default[i] == 'unknown':
        df.default[i] = 2
//housing
for i in range(0, len(df.housing)):
    if df.housing[i] == 'no':
        df.housing[i] = 0
    elif df.housing[i] == 'yes':
        df.housing[i] = 1
    elif df.housing[i] == 'unknown':
        df.housing[i] = 2
//loan
for i in range(0, len(df.loan)):
    if df.loan[i] == 'no':
        df.loan[i] = 0
```

```

elif df.job[i] == 'technician':
    df.job[i] = 9
elif df.job[i] == 'unemployed':
    df.job[i] = 10
elif df.job[i] == 'unknown':
    df.job[i] = 11
//marital
for i in range(0, len(df.marital)):
    if df.marital[i] == 'divorced':
        df.marital[i] = 0
    elif df.marital[i] == 'married':
        df.marital[i] = 1
    elif df.marital[i] == 'single':
        df.marital[i] = 2
    elif df.marital[i] == 'unknown':
        df.marital[i] = 3
//education
for i in range(0, len(df.education)):
    if df.education[i] == 'basic.4y':
        df.education[i] = 0
    elif df.education[i] == 'basic.6y':
        df.education[i] = 1
    elif df.education[i] == 'basic.9y':
        df.education[i] = 2
        elif df.education[i] ==
'high.school':
        df.education[i] = 3
        elif df.education[i] ==
'illiterate':
        df.education[i] = 4
        elif df.education[i] ==
'professional.course':
        df.education[i] = 5
        elif df.education[i] ==
'university.degree':
        df.education[i] = 6
    elif df.education[i] == 'unknown':
        df.education[i] = 7
//poutcome
for i in range(0, len(df.poutcome)):
    if df.poutcome[i] == 'failure':
        df.poutcome[i] = 0
        elif df.poutcome[i] ==
'nonexistent':
        df.poutcome[i] = 1

```

```


elif df.loan[i] == 'yes':
    df.loan[i] = 1
elif df.loan[i] == 'unknown':
    df.loan[i] = 2
//contact
for i in range(0, len(df.contact)):
    if df.contact[i] == 'cellular':
        df.contact[i] = 0
    elif df.contact[i] == 'telephone':
        df.contact[i] = 1
for i in range(0, len(df.month)):
    if df.month[i] == 'jan':
        df.month[i] = 0
    elif df.month[i] == 'feb':
        df.month[i] = 1
    elif df.month[i] == 'mar':
        df.month[i] = 2
    elif df.month[i] == 'apr':
        df.month[i] = 3
    elif df.month[i] == 'may':
        df.month[i] = 4
    elif df.month[i] == 'jun':
        df.month[i] = 5
    elif df.month[i] == 'jul':
        df.month[i] = 6
    elif df.month[i] == 'aug':
        df.month[i] = 7
    elif df.month[i] == 'sep':
        df.month[i] = 8
    elif df.month[i] == 'oct':
        df.month[i] = 9
    elif df.month[i] == 'nov':
        df.month[i] = 10
    elif df.month[i] == 'dec':
        df.month[i] = 11
//day_of_week
for i in range(0, len(df.day_of_week)):
    if df.day_of_week[i] == 'mon':
        df.day_of_week[i] = 0
    elif df.day_of_week[i] == 'tue':
        df.day_of_week[i] = 1
    elif df.day_of_week[i] == 'wed':
        df.day_of_week[i] = 2
    elif df.day_of_week[i] == 'thu':

```

```
else:
    df.poutcome[i] = 2
```

```
df.day_of_week[i] = 3
elif df.day_of_week[i] == 'fri':
    df.day_of_week[i] = 4
//duration
for i in range(0, len(df.duration)):
    if df.duration[i] == 0:
        df.duration[i] = 0
    else:
        df.duration[i] = 1
```

- + Sau khi đã chuẩn hóa các thuộc tính của bộ dữ liệu xong ta thu được kết quả như sau:

 df.head()

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	previous	poutcome	y
0	4	1	1	2	2	0	0	0	10	2	1	0	1	0
1	3	2	1	6	0	0	0	1	10	2	1	1	0	0
2	7	5	1	0	0	0	0	0	6	0	1	0	1	1
3	3	0	1	6	0	1	0	1	4	0	1	0	1	0
4	5	5	0	6	0	0	0	0	5	1	1	0	1	0

## 2.3.2 Thiết lập mô hình

### a, Thư viện sklearn

- Để xây dựng mô hình xử lý cho bài toán lần này, nhóm chúng em quyết định sử dụng thư viện Scikit-learn(sklearn) để thực hiện.
- Scikit-learn (Sklearn) là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ Python. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction.
- Scikit-learn hỗ trợ mạnh mẽ trong việc xây dựng các sản phẩm. Nghĩa là thư viện này tập trung sâu trong việc xây dựng các yếu tố: dễ sử dụng, dễ code, dễ tham khảo, dễ làm việc, hiệu quả cao.
- Ví dụ (nguồn: vncoder.vn) **Classification and Regression Trees**

- + Ở ví dụ sau, chúng ta sử dụng cây quyết định Decision tree phân loại để mô hình hóa bộ dữ liệu hoa Iris.
- + Bộ dữ liệu này được cung cấp dưới dạng tập dữ liệu mẫu với thư viện và được tải. Trình phân loại phù hợp với dữ liệu và sau đó dự đoán được thực hiện trên dữ liệu đào tạo.
- + Bộ dữ liệu này được cung cấp dưới dạng tập dữ liệu mẫu ngay trong thư viện sau đó được tải xuống. Thuật toán phân loại bắt đầu huấn luyện mô hình với bộ dữ liệu Iris ban đầu sau đó dự đoán lại các dữ liệu huấn luyện.
- + Cuối cùng, chúng ta đánh giá độ tốt của mô hình bằng quan sát accuracy và confusion matrix của 2 tập nhãn thực tế và nhãn dự đoán của mô hình.

```
# Sample Decision Tree Classifier

from sklearn import datasets

from sklearn import metrics

from sklearn.tree import DecisionTreeClassifier

# load the iris datasets
dataset = datasets.load_iris()

# fit a CART model to the data
model = DecisionTreeClassifier()

model.fit(dataset.data, dataset.target)

print(model)

# make predictions
expected = dataset.target

predicted = model.predict(dataset.data)

# summarize the fit of the model

print(metrics.classification_report(expected, predicted))

print(metrics.confusion_matrix(expected, predicted))
```

Như vậy có thể thấy, khi ta sử dụng thư viện sklearn code sẽ trở nên ngắn gọn và đơn giản hơn rất nhiều. Và vì được xây dựng trước nên cũng tránh được



những sai sót trong quá trình thực hiện nếu tự xây dựng các class, hàm để tạo cây quyết định.

## **b, Thực hiện mô hình bài toán**

- Chọn hệ số tương quan.

+ Code:

```
# chọn bằng hệ số tương quan
cor = df.corr()
```

+ Đôi nét về hàm corr() :

corr() được sử dụng để tìm mối tương quan theo cặp của tất cả các cột trong Python. Mọi giá trị NaN sẽ tự động bị loại trừ. Bất kỳ loại hoặc cột dữ liệu không phải số nào trong Dataframe, nó sẽ bị bỏ qua.

- Chọn các cột có thuộc tính ảnh hưởng đến đầu ra của bài toán

+ Code:

```
cotdt = ['age', 'job', 'marital', 'education', 'default',
'housing', 'loan', 'contact', 'month', 'day_of_week',
'poutcome']

x = df[cotdt]

y = df['y'].astype('int')
```

- Chia tập dữ liệu ra làm hai phần ra train và test. Chia bộ dữ liệu train và test cân bằng nhau, sau đó ta trộn dữ liệu lên random\_state = 26

+ Code:

```
//Thêm một số thành phần sử dụng của thư viện sklearn

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split

//chia bộ dữ liệu làm hai phần

X_train, X_test, y_train, y_test = train_test_split(x, y,
test_size=0.5, random_state=26)
```

```

clf = DecisionTreeClassifier(criterion='entropy', max_depth=5)

clf = clf.fit(X_train, y_train)

predictions = clf.predict(X_test)

```

- Thực hiện hàm hồi quy

- + Code:

```

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC

svc = SVC()

svc.fit(X_train, y_train)

```

- Xây dựng hàm tính điểm chính xác giữa mô hình train và test

- + Code:

```

def compute_accuracy(Y_true, Y_pred):

    correctly_predicted = 0

    # lặp lại nhãn và kiểm tra độ chính xác
    for true_label, predicted in zip(Y_true, Y_pred):

        if true_label == predicted:

            correctly_predicted += 1

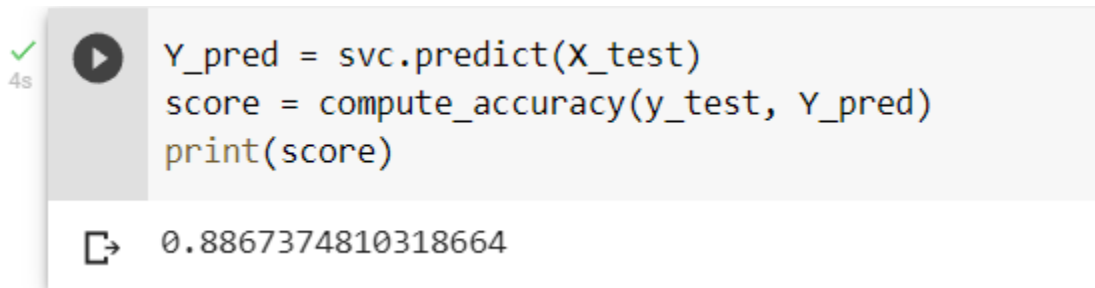
    # Tính toán điểm chính xác của việc so sánh

    accuracy_score = correctly_predicted / len(Y_true)

    return accuracy_score

```

- Hiển thị điểm chính xác của mô hình:

A screenshot of a Jupyter Notebook cell. On the left, there is a green checkmark and the text '4s'. In the center, there is a play button icon. To the right of the play button, the following Python code is displayed: 

```
Y_pred = svc.predict(X_test)
score = compute_accuracy(y_test, Y_pred)
print(score)
```

 Below the code, there is a copy icon and the output value: 

```
0.8867374810318664
```

```
Y_pred = svc.predict(X_test)
score = compute_accuracy(y_test, Y_pred)
print(score)
```

```
0.8867374810318664
```

## 2.4 Kết quả và đánh giá

- Sau khi thực hiện mô hình trên, nhóm nhận thấy điểm chính xác của mô hình là tương đối cao (0.8867374810318664)
- Tuy nhiên, dữ liệu huấn luyện còn hạn chế, nhóm cần tìm hiểu và thêm nhiều dữ liệu chính xác hơn để đảm bảo được tính đúng đắn của thuật toán đưa ra.
- Bên cạnh đó, trong quá trình thực hiện nhóm cũng chưa so sánh được độ chính xác giữa các thuật toán nhằm đưa ra tính hiệu quả của thuật toán ID3 cho bộ dữ liệu đang sử dụng.

## 2.5 Vẽ cây quyết định

- Với bộ dữ liệu chuẩn bị quá lớn, khi dùng toàn bộ để vẽ cây quyết định sẽ dẫn đến một rừng cây và khó để biểu diễn thành hình ảnh nên nhóm quyết định lấy 25 dòng đầu tiên của bộ dữ liệu sử dụng cho phần này.
- Tiến hành vẽ cây:

- + Thêm các thư viện cần thiết:

- Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

- + Liên kết với bộ dataset (train\_mini.csv):

- Code:

```

from google.colab import drive
drive.mount('/content/drive')

%cd /content/drive/MyDrive/ColabNotebooks/Datasets

df = pd.read_csv("train_mini.csv")

```

- + Tương tự như bộ dữ liệu đã dùng, ta chuẩn hóa các thuộc tính về dạng số.
- + Để tăng độ chính xác cũng như rút ngắn thời gian trong quá trình thực hiện, nhóm quyết định sử dụng thư viện `sklearn`, cụ thể là `sklearn.tree` để vẽ cây.

- Một số hàm cần dùng của thư viện:

```

import sklearn.tree as tr

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split

```

- + Ta chọn các thuộc tính ảnh hưởng đến đầu ra và dùng các hàm của `sklearn.tree` để tính toán:

- Code:

```

cotdt = ['age', 'job', 'marital', 'education', 'default',
'housing', 'loan', 'contact', 'month', 'day_of_week',
'poutcome']

x = df[cotdt]

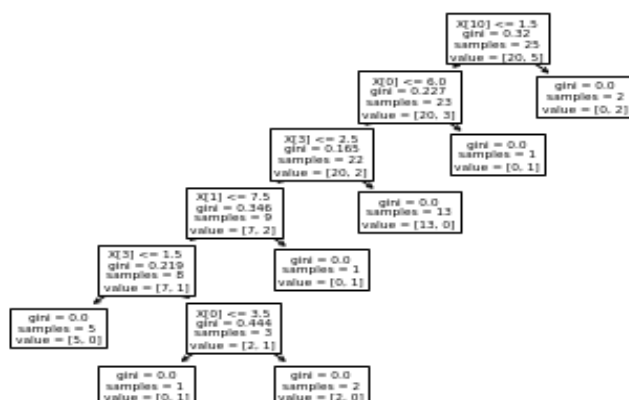
y = df['y'].astype('int')

dtree = tr.DecisionTreeClassifier()

dtree = dtree.fit(x, y)

```

- + Và cuối cùng là hiển thị thực thể ra cây một cách trực quan, ta sử dụng hàm `plot_tree()`, kết quả thu được một ảnh như sau:



## TÀI LIỆU THAM KHẢO

- [1] Tổng quan về trí tuệ nhân tạo: [lib.agu.edu.vn/TTNT](http://lib.agu.edu.vn/TTNT).
- [2] PhD Vũ Hữu Tiệp, Machine Learning cơ bản: [Decision Trees \(1\): Iterative Dichotomiser 3](#).
- [3] Nguyễn Phương Nga, Giáo trình Trí tuệ nhân tạo, Trường Đại học Công nghiệp Hà Nội, [bản 2021](#).
- [4] <http://id3alg.altervista.org/>
- [5] J.R. Quinlan, C4.5 programs for machine learning Morgan Kaufmann Publishers, livres Google.
- [6] Nguyễn Nhật Quang, Tiền xử lý dữ liệu, [Khai phá dữ liệu](#).
- [7] [https://www.javatpoint.com/accuracy\\_score-in-sklearn](https://www.javatpoint.com/accuracy_score-in-sklearn)
- [8] Quách Xuân Nam, Trương Văn Thông, Nguyễn Đình Thanh: “[Decision Tree: ID3](#)”
- [9] Đỗ Mạnh Quang, Trần Thị Thảo Nguyên: [Decision-Tree-ID3](#)

[10] J.R. QUINLAN, Induction of Decision Trees, 1986, Machine Learning 1:81-106

[11] [THUẬT TOÁN CÂY QUYẾT ĐỊNH \(P.1\) : CLASSIFICATION & REGRESSION TREE \(CART\)](#)