BÁO CÁO ĐỒ ÁN MÔN HỌC
(Đồ án tìm hiểu CTDL/Giải thuật)
Lớp: IT003.O21.CNTT

SINH VIÊN THỰC HIỆN

Mã sinh viên: 23520673 **Họ và tên: Đoàn Việt Khải**

TÊN ĐỀ TÀI:

XÂY DỰNG MÔ HÌNH PHÂN LOẠI DỮ LIỆU TỪ THUẬT TOÁN DECISION TREE - CLASSIFICATION

MỤC LỤC

1. Giới thiệu đồ án:

Trong cuộc sống, có rất nhiều tình huống chúng ta quan sát, suy nghĩ và đưa ra quyết định bằng cách đặt câu hỏi. Chẳng hạn như bản thân em, mỗi lần đến thời gian dùng bữa, trong thâm tâm em luôn xuất hiện câu hỏi "Hôm nay ăn gì?"

Em sẽ xác định:

- ✓ Nếu là đầu tháng thì đi ăn lẩu với bạn
- ✓ Nếu là cuối tháng thì ăn cơm chay

Dựa theo ngày em sẽ quyết định được ngày hôm đó mình ăn món gì.

Vậy liệu rằng có cấu trúc dữ liệu nào hoạt động bằng cách đặt câu hỏi để đưa ra quyết định giống con người, đó chính là Decision Tree

2. Quá trình thực hiện:

a. Tuần 1: Đánh giá tiến độ 50%/100%

- ✓ Nêu được tổng quan về decision tree (định nghĩa; các khái niệm cơ bản; phân loại)
- ✓ Tìm hiểu được cấu trúc dữ liệu decision tree (điểm nổi bật; các thuộc tính như gini, entropy, log-loss)
- ✓ Tìm hiểu các thuật toán của decision tree như ID3 và CART (các thao tác cơ sở, pseudocode, code decision tree bằng python)
- ✓ Tìm hiểu các thư viện hỗ trợ decision tree trong python (thư viện sklearn)

b. Tuần 2: Đánh giá tiến độ 100%/100%

- ✓ Ứng dụng decision tree vào thực tiễn
- ✓ Xây dựng mô hình phân loại (classification) từ decision tree
- ✓ Thử nghiệm với bộ dữ liệu và đánh giá
- ✓ So sánh giữa mô hình tự code và mô hình decision tree sử dụng thư viện sklearn
- ✓ Rút ra ưu và nhược điểm của decision tree
- ✓ Đề xuất cách cải tiến mô hình để tránh nhược điểm trên

3. Kết quả đạt được:

a. Các định nghĩa/khái niệm cơ bản

Định nghĩa decision tree:

Cây quyết định (decision tree) là một cấu trúc dữ liệu dạng phân cấp được dùng để phân lớp các đối tượng, trong đó mỗi nút bên trong biểu thị một tính năng, các nhánh biểu thị các quy tắc và các nút lá biểu thị kết quả.[1]

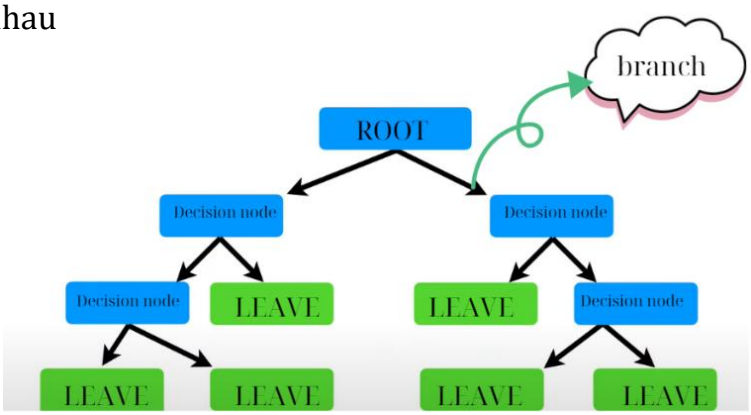
Các thuộc tính của đối tượng có thể thuộc các kiểu dữ liệu khác nhau như Nhị phân (Binary) , Định danh (Nominal), Thứ tự (Ordinal), Số lượng (Quantitative) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal.[2]

Vậy hiểu đơn giản là khi ta có dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các dữ liệu chưa biết. [2]

Các khái niệm cơ bản:

Về cơ bản, cây quyết định gồm các phần sau:

- ✓ Nút gốc: Là nút trên cùng, khởi đầu cho quá trình quyết định
- ✓ Nút quyết định (nút nội bộ): Là nút tượng trưng cho sự lựa chọn, liên quan đến đầu vào (input)
- ✓ Nút lá: Là nút không có nút con, đại diện cho kết quả (label hoặc số)
- ✓ Kết nối giữa các nút là các nhánh, tùy theo cách kết hợp mà cây có độ phức tạp khác nhau



Phân loại

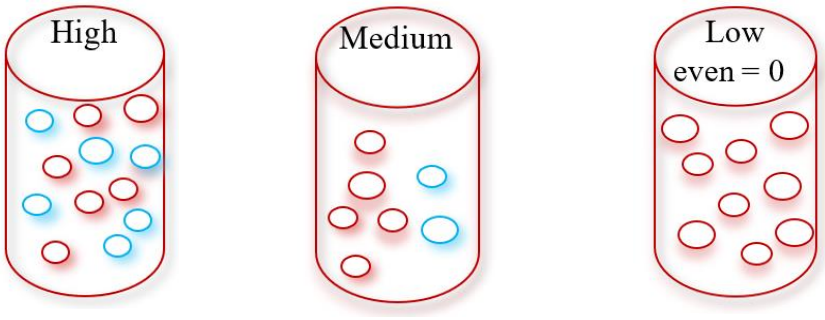
Decision tree là một thuật toán máy học có giám sát (supervised learning) dùng cho cả 2 dạng bài toán hồi quy (regression) và phân loại (classification). Tuy nhiên trong báo cáo đồ án này em chỉ tập trung ở bài toán phân loại (classification)

b. Các đặc trưng nổi bậc

Đặc trưng về thuộc tính: Không đưa ra giả định cụ thể về hình dạng và phân phối dữ liệu: Nó tạo ra các quy tắc quyết định dựa trên tính năng của dữ liệu, dẫn đến kết quả trực quan hơn.

Decision tree đưa ra các quy tắc quyết định dựa vào việc tính chỉ số Gini hoặc entropy hoặc log-loss, và lấy chỉ số nhỏ nhất làm tiêu chí phân tách.

Nguyên nhân của cơ sở này là vì cả ba chỉ số trên đều là thước đo độ "hỗn loạn" của dữ liệu, chính vì vậy chỉ số càng cao tức là dữ liệu càng có nhiều lớp, gây khó khăn cho việc phân tách



Cả ba chỉ số chỉ khác nhau về mặt toán học:

✓ Gini: $I_{Gini}(p) = 1 - \sum_{i=1}^{classes} p_i^2 \longrightarrow Total\ of\ gini = \sum_{i=1}^{classes} P_i . I_{Gini}(p)$ [3]

✓ $Entropy = -\sum_{i=1}^{classes} p_i \log_2(p_i) \longrightarrow \begin{cases} overall\_entropy = \sum_{i=1}^{classes} \left(\frac{T_i}{T}\right).Entropy(T) \\ Information\ Gain(T) = Entropy(T) - \sum_{i=1}^{classes} \left(\frac{T_i}{T}\right).Entropy(T) \end{cases}$ [3]

+) classes: số lớp trong subset

+) $i = \{1,2,3,...classes\}$

+) p_i: xác suất mà lớp i có trong subset

✓ Log-loss: Là hàm được xây dựng sẵn trong thư viện sklearn, tuy nhiên khi triển khai thuật toán này, thường dùng gini và entropy nhiều hơn

Vì cả ba khá giống nhau nên ở đây em lấy một ví dụ đặc trưng về cách decision tree lựa chọn tiêu chí phân tách

Ví dụ: *Dựa vào Gini index, xây dựng decision tree cho bộ dữ liệu sau*

Movie	Eat	Fee	Hang out
YES	YES	70	NO
NO	YES	100	NO
NO	NO	120	YES
YES	NO	200	NO
NO	YES	250	YES
YES	YES	340	YES
YES	NO	500	YES

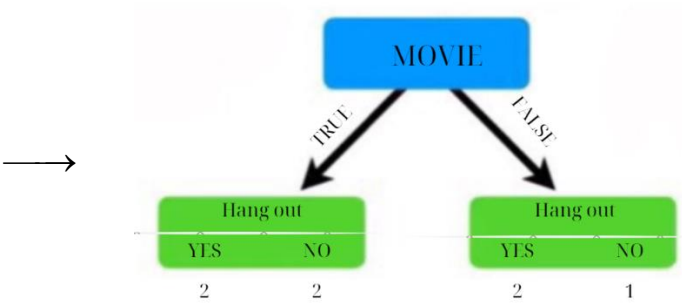
Đầu tiên, ta xem thử liệu "movie", "eat" hay "fee" sẽ là question của root, bằng cách từ từng feature riêng biệt mà dự đoán label "hang out".

Để làm được điều này, ta tạo cây đơn giản chỉ để hỏi xem có những ai thích từng feature, từ đó đưa ra số người "hang out"

Vì "movie" và "eat" là biến phân loại nên ta sẽ tính trước, còn "fee" là biến liên tục nên ta xử lí sau

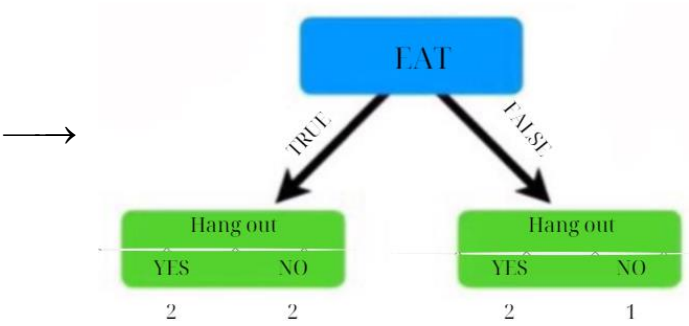
✓ Movie

Movie	Hang out
YES	NO
NO	NO
NO	YES
YES	NO
NO	YES
YES	YES
YES	YES



✓ Eat

Eat	Hang out
YES	NO
YES	NO
NO	YES
NO	NO
YES	YES
YES	YES
NO	YES



Áp dụng cách tính Gini index đã được trình bày ở trên:

+) Gini index of movie:

$$I_{Gini}(movie\_True) = 1 - (Yes\ of\ \ hang\ \ out)^2 - (No\ of\ \ hang\ \ out)^2$$

$$= 1 - \left(\frac{2}{2+2}\right)^2 - \left(\frac{2}{2+2}\right)^2 = 0{,}500$$

$$I_{Gini}(movie\_False) = 1 - (Yes\ of\ \ hang\ \ out)^2 - (No\ of\ \ hang\ \ out)^2$$

$$= 1 - \left(\frac{2}{2+1}\right)^2 - \left(\frac{1}{2+1}\right)^2 \approx 0{,}444$$

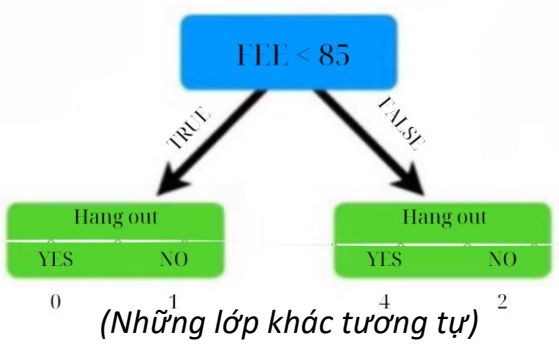
Total Gini index (movie) $= 0{,}5.\left(\frac{4}{4+3}\right) + 0{,}444.\left(\frac{3}{4+3}\right) = 0{,}476$

+) Tương tự: Total Gini index (eat) $\approx 0{,}476$

Đối với "fee", do là biến liên tục nên ta sẽ chọn "ngưỡng" bằng cách tính giá trị trung bình của 2 giá trị kề nhau, ứng với mỗi giá trị trung bình ta cũng tính gini index như trên

✓ Fee

Average value	Fee	Hang out
	70	NO
85		
	100	NO
110		
	120	YES
160		
	200	NO
225		
	250	YES
295		
	340	YES
420		
	500	YES



(Những lớp khác tương tự)

+) Total Gini index (fee < 85) $\approx 0{,}380$ +) Total Gini index (fee < 110) $\approx 0{,}230$

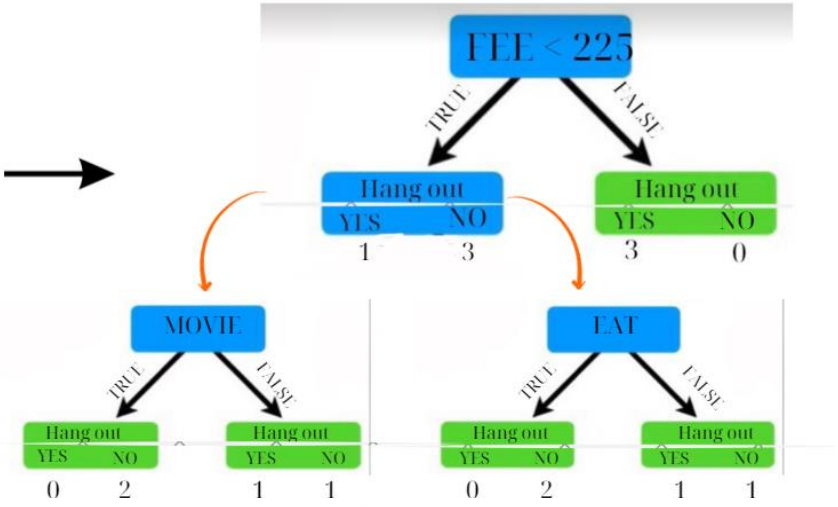
+) Total Gini index (fee < 160) $\approx 0{,}404$ +) Total Gini index (fee < 225) $\approx 0{,}214$

+) Total Gini index (fee < 295) $\approx 0{,}343$ +) Total Gini index (fee < 420) $\approx 0{,}428$

Vì (fee < 225) có gini index nhỏ nhất trong biến liên tục, nên chỉ cần chọn (fee < 225) là được, so sánh với (movie) và (eat) thì (fee < 225) vẫn nhỏ nhất nên root sẽ là (fee < 225)

Ta tiếp tục so sánh gini index của "movie" và "eat" trong trường hợp root là fee < 225, để chọn ra thuộc tính tối ưu

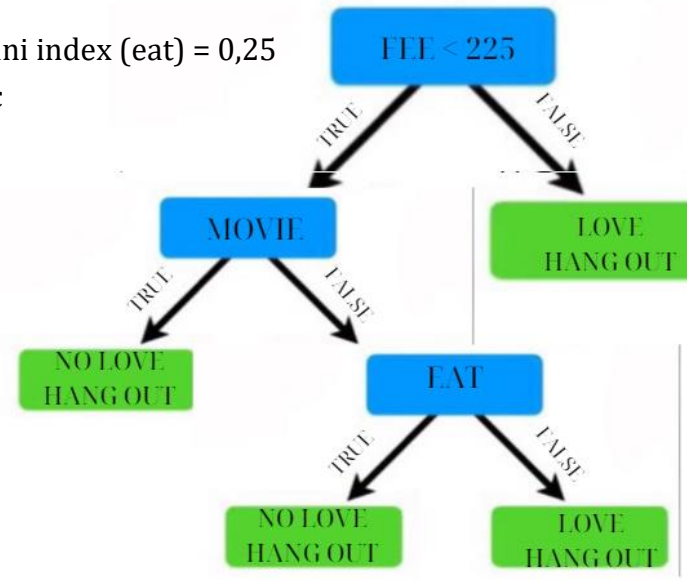
Movie	Eat	Fee	Hang out
YES	YES	70	NO
NO	YES	100	NO
NO	NO	120	YES
YES	NO	200	NO
NO	YES	250	YES
YES	YES	340	YES
YES	NO	500	YES



+) Total Gini index (movie) = Total Gini index (eat) = 0,25

⟶ Chọn thuộc tính nào cũng được

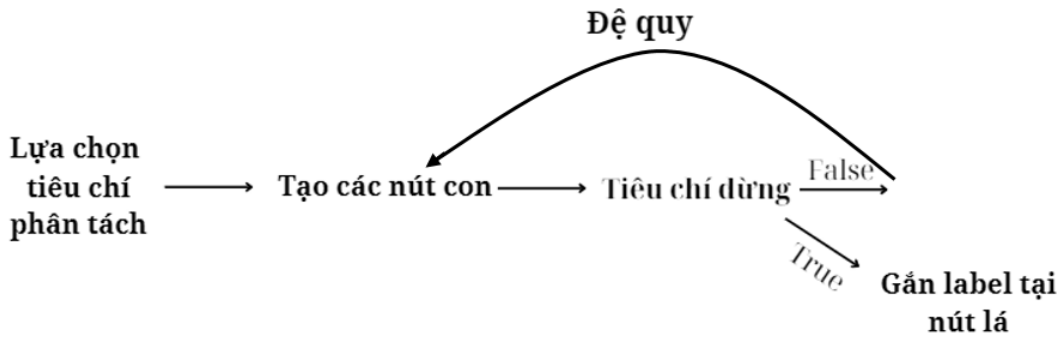
Mô hình decision tree hoàn chỉnh:



Đặc trưng về các làm việc với dữ liệu:

- ✓ Xử lí được biến liên tục và phân loại:
  +) Biến liên tục: chọn "ngưỡng" và chia dữ liệu thành 2 nhóm dựa trên ngưỡng
  Ví dụ: nếu thuộc tính là "tuổi", thì cây sẽ đưa ra quy tắc: "Nếu tuổi lớn hơn (hoặc bé hơn) thì…"
  +) Biến rời rạc: dựa trên giá trị phân loại để chia thành 2 nhóm
  Ví dụ: nếu thuộc tính là "giới tính", thì cây sẽ đưa ra quy tắc "Nếu giới tính là nam (hoặc là nữ) thì …"
- ✓ Xử lí được dữ liệu không cân bằng: bằng cách đánh giá tầm quan trọng của từng nút dựa trên sự phân lớp

c. Các thao tác cơ sở



Xây dựng decision tree

Cây quyết định được xây dựng bằng cách phân vùng đệ quy, bắt đầu từ nút gốc (nút cha đầu tiên), biểu diễn nút con trái và nút con phải, từ hai nút con, tiếp tục phân tách thêm để được cây hoàn chỉnh.

Tuy nhiên, nếu phân tách không điểm dừng thì sẽ dẫn đến cây có độ cao rất sâu và khi áp dụng vào các bài toán phân loại sẽ dẫn đến hiện tượng overfitting. Do đó, cần dựa vào thuộc tính phân lớp của cây để tìm độ cao phù hợp.

Tiến trình kết thúc, khi thỏa một trong các điều kiện sau:

- ✓ Tất cả các mẫu của một nút cho trước đều thuộc cùng một lớp
- ✓ Không còn thuộc tính mà mẫu dựa vào để phân tách xa hơn
- ✓ Không còn mẫu nào cho nhánh test_attribute = a_i (nhánh mà thuộc tính được kiểm tra)[4]

Thuật toán CART

Pseudocode

(tự thiết kế)

Source code

(CART) [5]

```
Function_CART (dataset, feature)
begin
if (all data trong dataset đều thuộc cùng một lớp)
return nút lá được gắn label bởi lớp đó
else if (feature == empty)
return nút lá với label là hợp tất cả các lớp trong dataset
else
begin
    Chọn feature A tốt nhất dựa trên gini index nhỏ nhất
    Gán A cho root
    Đối với mỗi giá trị v của A thực hiện các bước sau
        1. Thêm một nhánh mới từ root cho v
        2. Đặt vào split_data trong dataset có giá trị v tại thuộc tính A
        3. Gọi đệ quy Function_CART(split_data, feature)gán kết quả vào nhánh v
end
END
```

Thuật toán ID3

Pseudocode

(tự thiết kế)

Source code

(ID3) [5]

```
Function_ID3 (dataset, feature)
begin
if (all data trong dataset đều thuộc cùng một lớp)
return nút lá được gắn label bởi lớp đó
else if (feature == empty)
return nút lá với label là hợp tất cả các lớp trong dataset
else
begin
    Chọn feature A tốt nhất dựa trên information gain hoặc overall_entropy
    Gán A cho root
    Đối với mỗi giá trị v của A thực hiện các bước sau
        1. Thêm một nhánh mới từ root cho v
        2. Đặt vào split_data trong dataset có giá trị v tại thuộc tính A
        3. Gọi đệ quy Function_ID3(split_data, feature)gán kết quả vào nhánh v
end
END
```

d. Các thư viện hỗ trợ (C++/Python)

Decision tree đã được thư viện scikit-learn (sklearn) hỗ trợ sẵn trong python

Tuy nhiên em sẽ build lại decision tree từ thuật toán ID3 ở trên sau đó đánh giá thuật toán tự code và thuật toán dùng thư viện được xây dựng sẵn (phần này sẽ được trình bày kĩ ở phần vận dụng thực tế)

e. Vận dụng thực tế:

Decision tree – classification có ứng dụng rộng rãi trên nhiều lĩnh vực khác nhau

- ✓ Y tế: Dựa trên các thông tin về sức khỏe, đưa ra dự đoán về bệnh lý, hỗ trợ bác sĩ đưa ra quyết định chẩn đoán và điều trị

  Ví dụ:

  +) Dựa vào lượng tiểu cầu dự đoán người mắc bệnh sốt xuất huyết

  +) Dựa vào các dấu hiệu phân loại bệnh nhân mắc Covid-19[6]
- ✓ Kinh tế - Xã hội: Dựa trên nhu cầu về thói quen, sở thích để phân tích hành vi người dùng, từ đó ra mắt sản phẩm phù hợp với nhu cầu chung của xã hội, thúc đẩy quảng bá, truyền thông
- ✓ Môi trường: Từ đặc điểm về độ ẩm, nhiệt độ xây dựng hệ thống dự báo thời tiết
- ✓ An toàn thông tin:

  +) Phân loại malware

  +) Dựa vào URL, tên miền của trang web phát hiện trang web lừa đảo[7]

  +) Dựa trên thông tin người gửi, nội dung email phân loại mail spam

Để làm được những điều nên, đòi hỏi decision tree phải "học" những đặc điểm của dữ liệu và tiến hành phân loại dựa trên bộ dữ liệu tương tự

Sau đây, em sẽ xây dựng một mô hình phân loại dữ liệu cơ bản của decision tree từ thuật toán ID3

Chuẩn bị: dataset IRIS

Mục tiêu: Phân loại các loại hoa

Tiền xử lí dữ liệu: Dữ liệu không có NULL, không bị outlier, không bị missing data nên không cần xử lí quá nhiều

Xây dựng mô hình:

i. *Load dataset*
ii. *Kiểm tra thông tin cơ bản của dataset (kích thước, đặc điểm dataset)*
iii. *Chia dataset thành 2 tập: Train_data (mô hình học), test_data (kiểm thử)*
iv. *Xây dựng hàm kiểm tra tiêu chí dừng*
v. *Xây dựng hàm gắn nhãn dữ liệu (lớp phổ biến nhất)*
vi. *Xây dựng hàm tính giá trị tách tốt nhất*
vii. *Xây dựng hàm tạo các nút con (rẽ nhánh data)*
viii. *Xây dựng hàm tính entropy*
ix. *Xây dựng hàm chọn tiêu chí phân tách tốt nhất (feature có entropy min)*
x. *Gọi đệ quy các hàm trên đến khi đáp ứng tiêu chí dừng*
xi. *Xây dựng hàm predict (trả về label tương ứng với feature)*
xii. *Xây dựng hàm tính accuracy*

So sánh và đánh giá:

- ✓ Tùy vào việc chọn tỉ lệ chia train_test mà kết quả dự đoán khác nhau
- ✓ Có thể thay đổi tham số của decision tree như max_depth hay min_sample_split để tối ưu việc phân tách của cây
- ✓ Thuật toán decision tree sử dụng thư viện đem lại kết quả chính xác cao hơn so với thuật toán tự code (nếu chọn đúng max_depth và min_sample_split)

- ✓ Dễ xảy ra hiện tượng overfitting (nguyên nhân là do decision tree phân tách rất sâu và dataset này khá đơn giản nên dễ xảy ra việc dữ liệu được gắn nhãn chính xác 100%)

Nhận xét về ưu và nhược điểm của decision tree:

Ưu điểm

- ✓ Giống hệt quy trình mà con người tuân theo khi đưa ra quyết định nên decison tree khá dễ hiểu
- ✓ Là thuật toán giám sát phi tham số, do đó không có giả định về phân chia bộ nhớ và cấu trúc phân lớp[4]
- ✓ Yêu cầu xử lí dữ liệu ít hơn so với các thuật toán khác
- ✓ Có thể xử lí cả thuộc tính tên và số đầu vào[4]
- ✓ Có khả năng xử lí các bộ dữ liệu gây lỗi và những bộ dữ liệu rỗng (missing data)[4]

Nhược điểm

- ✓ Hầu hết các thuật toán (điển hình như ID3, C4.5) đều yêu cầu các thuộc tính mục tiêu phải là các giá trị rời rạc.[4]
- ✓ Cần dựa vào tiêu chí phân tách để xây dựng cây, do đó nếu không tìm được tiêu chí phân tách phù hợp sẽ dẫn đến cây quá phức tạp, gây ra hiện tượng overfitting, giảm hiệu suất phân loại trên những dữ liệu mới.
- ✓ Nhạy cảm, chỉ cần một thay đổi nhỏ trong bộ dữ liệu cũng có thể dẫn đến thay đổi cấu trúc mô hình.

Cải tiến mô hình tránh overfitting

- ✓ Tỉa cây (Pruning) nhằm loại bỏ những phần không mang lại hiệu quả cao [10]
- ✓ Sử dụng bản cải tiến của decision tree là random forest [11]

4. Tài liệu tham khảo:

1. definition of decision-tree
2. decision tree
3. Gini_inpury và Entropy
4. Decision tree for master
5. ID3_algorithm
6. Decision-Tree-model-for-COVID-19
7. Phishing_Website
8. sklearn.tree.DecisionTreeClassifier
9. guide to buid decision tree
10. pruning-decision-trees
11. random_forest

5. Phụ lục:

1. Tổng hợp source code: Final_project_23520673

   Em dùng file .ipynb nên sẽ không hiển thị code trong phần preview trên github. Thầy chỉ cần tải về là chạy code được

2. Thuật toán ID3: **ID3**

   Input gồm 2 hàng

   Hàng 1: là mảng 2D chứa các mẫu dữ liệu

   Hàng 2: là mảng 1D chứa các label ứng với các mẫu hàng 1

   Output: mẫu được gắn nhãn tương ứng với nó

```
#Testing
X = np.array([[2, 2], [2, 1], [2, 0], [1, 1],[1, 2], [1, 0], [0, 0], [0, 1], [0, 2]])
y = np.array([11, 32, 93, 4, 65, 26, 87, 48, 19])
```

```
Prediction for sample [2 2]: 11
Prediction for sample [2 1]: 32
Prediction for sample [2 0]: 93
Prediction for sample [1 1]: 4
Prediction for sample [1 2]: 65
Prediction for sample [1 0]: 26
Prediction for sample [0 0]: 87
Prediction for sample [0 1]: 48
Prediction for sample [0 2]: 19
```

3. Thuật toán CART: Từ ID3, em build lại **CART**

   Input và output giống với ID3

4. Mô hình phân loại sử dụng module hỗ trợ trong python: build_model

   Em tham khảo các tham số đặc trưng của decision tree[8] rồi sử dụng module sklearn trong python để build

   Input là file csv IRIS

   Thầy chỉ cần tải file IRIS về máy, rồi đổi lại đường dẫn file IRIS trong code là chạy được. Lưu ý: dùng dấu "/" thay vì "\"

```
#load dataset
dataFrame = pd.read_csv("C:/Users/Admin/OneDrive/Documents/UIT\DSA/final_project_IT003/Iris.csv")
print(dataFrame.head(10))   #in ra 10 hàng đầu tiên
```

   0.0s

   Output: trả về accuracy, tức là tỉ lệ chính xác mà mô hình dự đoán

5. Mô hình phân loại tự build: tự_build

Tham khảo ý tưởng từ thuật toán ID3[5] và xem cách build scratch[9], sau đó tự build lại decision tree (không sử dụng thư viện)

Input và output giống với mô hình sử dụng thư viện

6. Ảnh tự thiết kế: Toàn bộ ảnh và ví dụ trong báo cáo đều là em tự thiết kế bằng canva
7. Pseudocode tự thiết kế: Pseudo code tự thiết kế bằng hackmd