



# Decision Tree & Random Forest

---

FPT SOFTWARE 2021

# Nội dung

---

- **Mục đích của bài học**
- **Giới thiệu tổng quan về Machine Learning models**
  - Linear Regression
  - Classification
- **Tree based model**
  - Decision Tree
  - Random Forest
- **Exercises**

# Mục đích

---

- **Nắm được cấu trúc tổng quan của một model Machine Learning**
- **Hiểu được các thuật toán cơ bản**
  - Regression vs Classification
  - Decision Tree
  - Random Forest
- **Ứng dụng thuật toán Random Forest vào đánh giá dữ liệu Banking**

# Predictive Modeling

- Mô hình dự đoán - Predictive modeling
  - Đưa ra các phân tích, dự đoán, xu hướng về tương lai
  - Sử dụng các kỹ thuật như: Data mining, Statistics, Modeling, ML, AI nhằm cải thiện khả năng dự đoán
  - Nắm bắt các xu hướng quan trọng của dữ liệu
  - Dự đoán các giá trị cho những dữ liệu khác nhau hoặc dữ liệu mới
- Ví dụ về xây dựng mô hình dự đoán
  - Dự đoán khả năng xảy ra sự cố, rủi ro: xác suất (%) khách hàng vỡ nợ là gì?
  - Dự đoán một giá trị hữu hình: số tiền thực tế khách hàng sẽ vay là bao nhiêu?
  - Dự đoán kết quả dạng định tính: dự báo khách hàng sẽ rời dịch vụ (Có / Không)?

# Predictive Modeling

- Quy trình Predictive modeling :
  - Business understanding
  - Data understanding
  - Data preparation
  - Modeling
    - Xác định bài toán: Regression vs Classification ...
    - Algorithms: Machine Learning, Deep Learning
    - Loss function: MSE, cross entropy, etc.
  - Evaluation
    - Evaluation metrics: Accuracy, Precision, Recall, etc.
    - Trade-off (accuracy vs running time vs memory)
  - Deployment

# Regression vs Classification



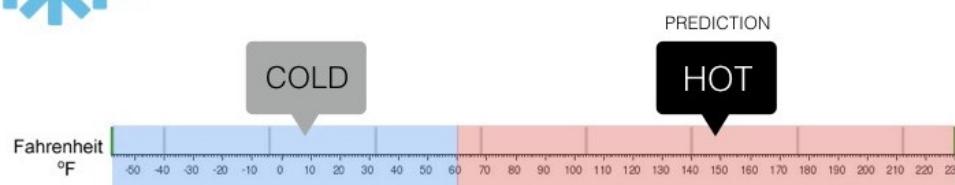
## Regression

What is the temperature going to be tomorrow?



## Classification

Will it be Cold or Hot tomorrow?



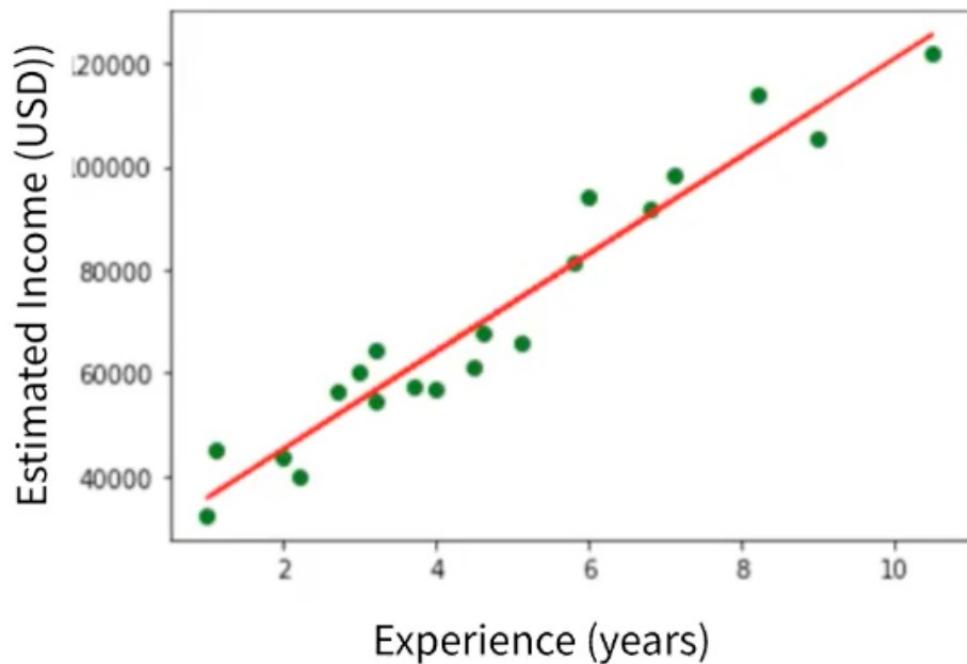
**Regression:** Là bài toán dự đoán một giá trị cụ thể

**Classification:** là bài toán phân loại các lớp (label) của dữ liệu ban đầu

# Linear Regression

$$\text{estimated income} = a_1(\text{age})^2 + a_2(\text{education}) + a_3(\text{experience}) + b$$

Figure 1. Plot of estimated income vs. experience

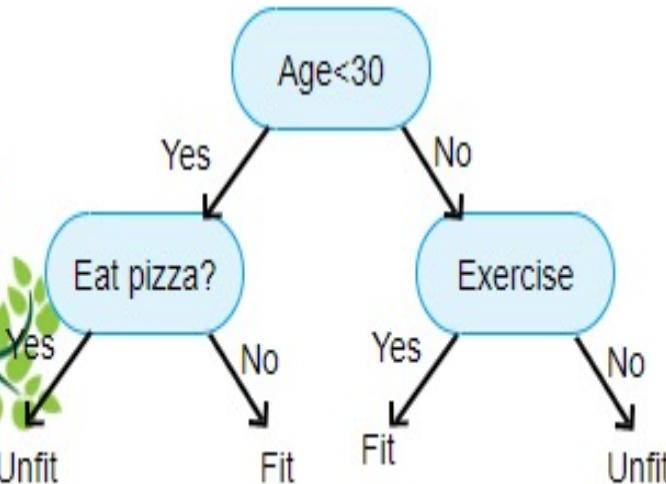


- Giải thích kết quả Y dựa vào các yếu tố X, nếu có linear relationship
- Hiểu rõ quan hệ và tác động của các yếu tố X đến kết quả Y có thể giúp giải thích sự việc, và đưa ra dự đoán.
  - VD.  $a_3 = \sim 10000$  (fig.1) thì có thể suy ra: tăng 1 năm kinh nghiệm  $\Rightarrow$  tăng \$10,000 dự tính lương (estimated income)



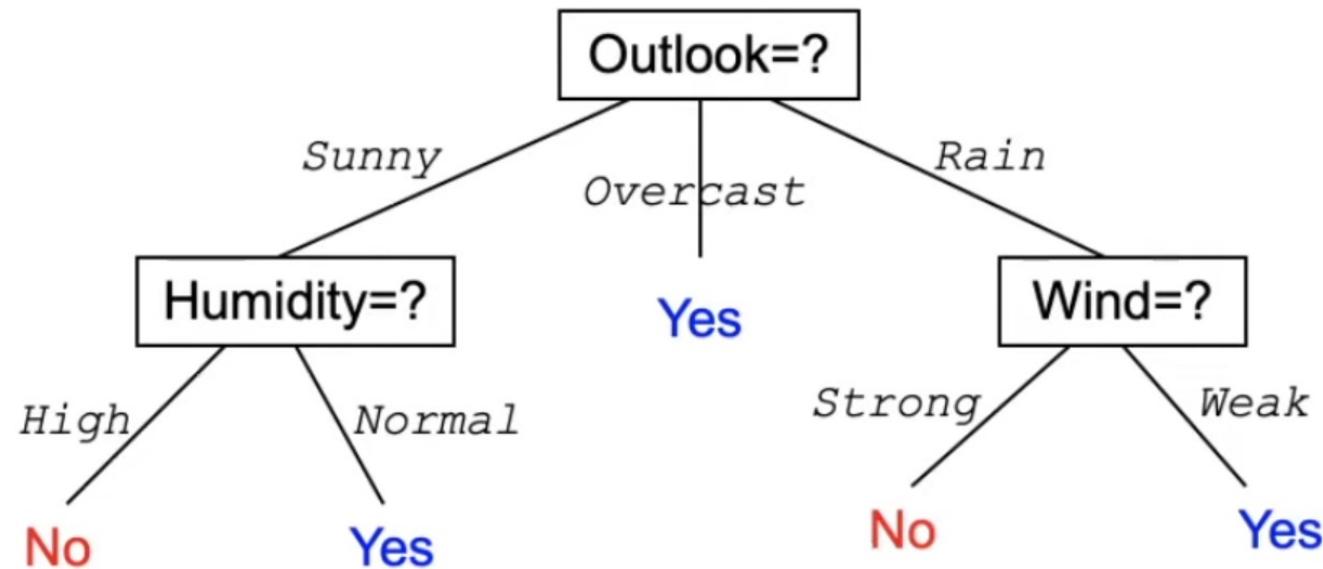
# Tree-based model

# Decision Tree



- **Mục tiêu:**
  - Đưa ra các kết luận dựa vào các nhóm thông tin có sẵn
- **Phương pháp:**
  - Đặt mục tiêu: tối đa Information Gain
  - Với mỗi nhánh (feature), đi qua các giá trị của nhánh trong dữ liệu ban đầu
  - Chọn giá trị cut-off giúp đạt tối đa mục tiêu (maximize Information Gain)

# Example of Decision Tree



- (Outlook=Overcast, Humidity=High, Wind=Weak) → **Yes**
- (Outlook=Rain, Humidity=High, Wind=Strong) → **No**
- (Outlook=Sunny, Humidity=High, Wind=Strong) → **No**

# Classification problem

Data representation:

- Each observation is represented by n attributes/features, e.g.,

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{in})^T.$$

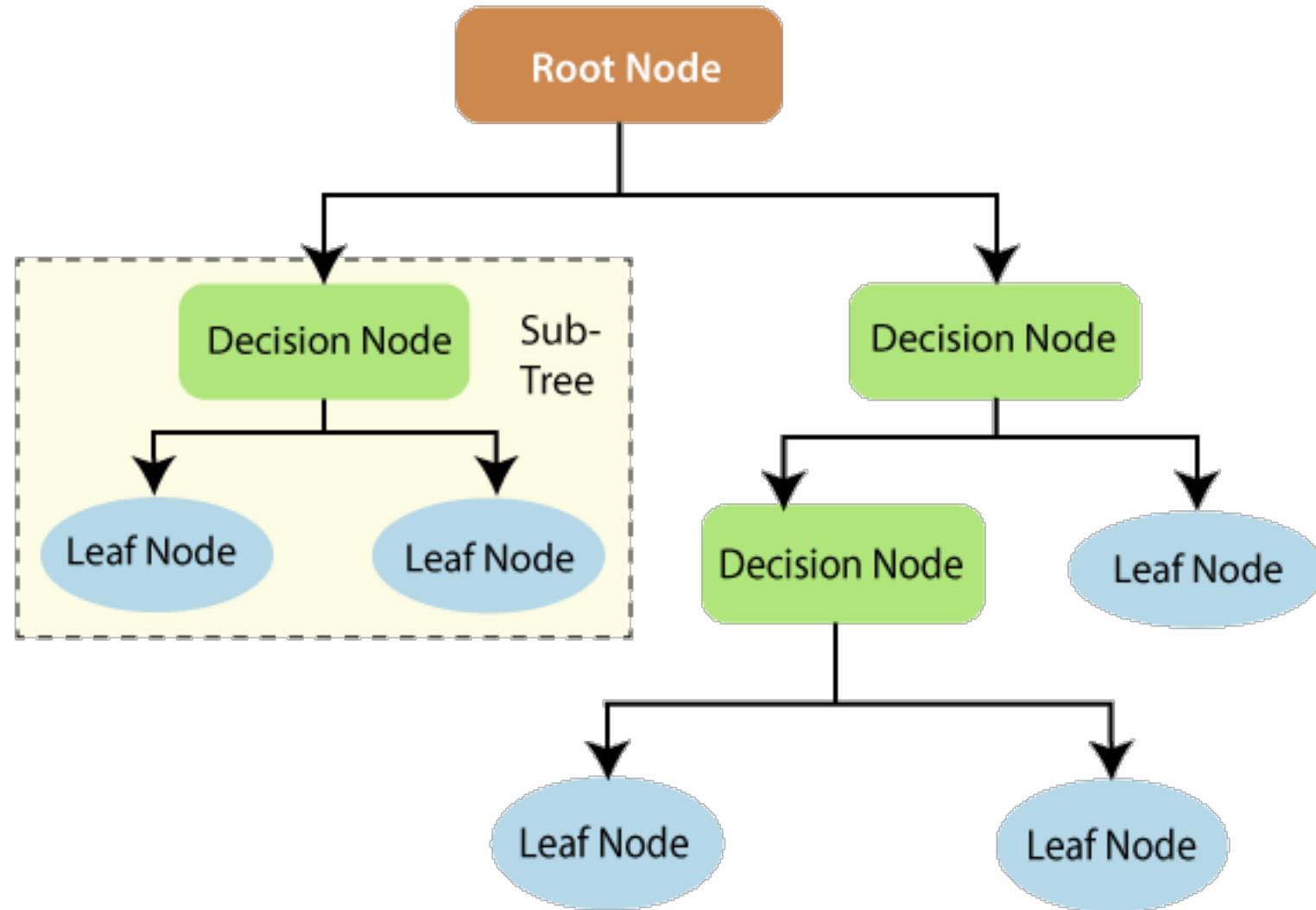
- Each attribute is **nominal/categorical**, i.e., represents names, labels or categories, e.g.,

$$x_{i1} \in \{high, normal\}, x_{i2} \in \{male, female, other\}$$

- There is a set C of predefined labels.
- We have to learn a function from a training dataset:

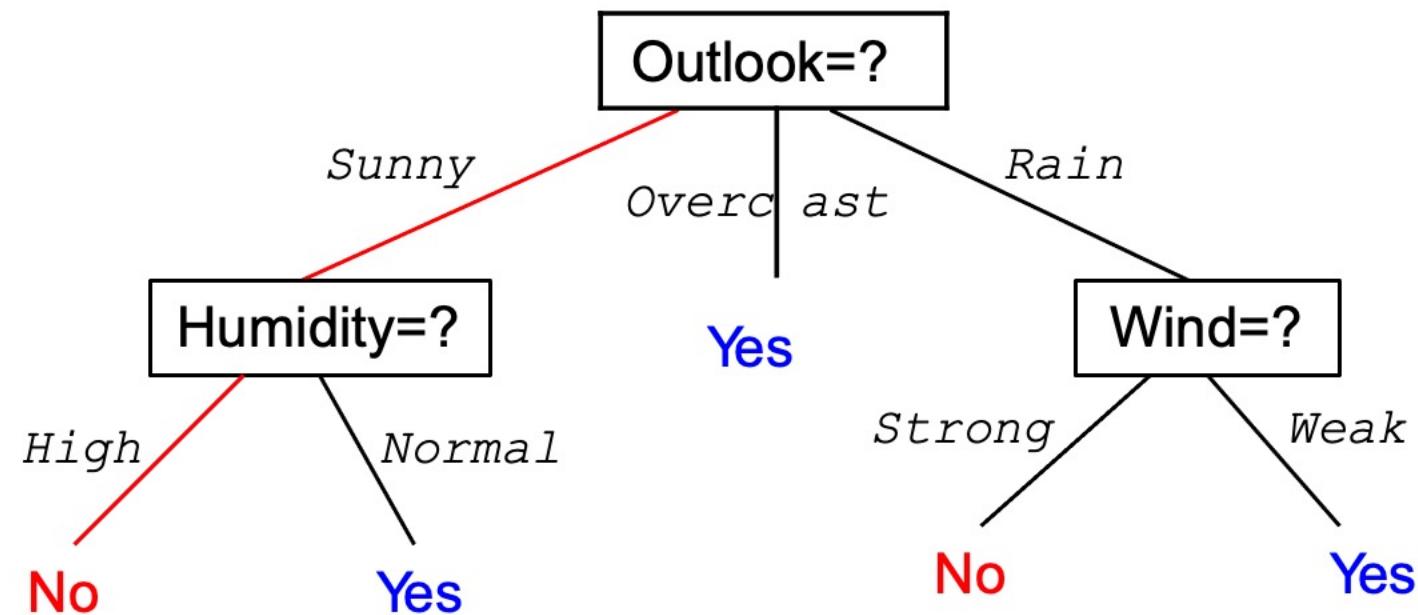
$$\mathbf{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_M, y_M)\}$$

# Decision Tree representation



# Decision Tree representation

- Mỗi nhánh từ gốc đến lá bao gồm kết hợp của nhiều các test attributes
- Decision Tree là kết hợp của nhiều mệnh đề khác nhau

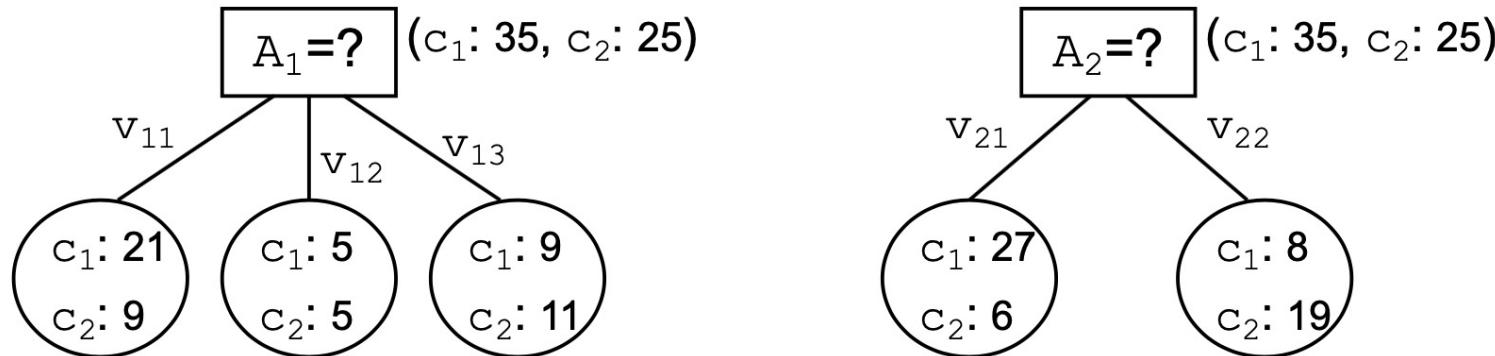


# Learning a decision tree by ID3

- ID3 (Iterative Dichotomiser 3) là một thuật toán được Ross Quinlan đề xuất vào năm 1986
- ID3 sử dụng mô hình top-down
- Phương pháp
  - Ở mỗi node N, chọn ra thuộc tính A giúp phân tách dữ liệu D tốt nhất
  - Tạo ra các nhánh con (branch) cho mỗi thuộc tính A và chia dữ liệu đến các nhánh con tương ứng
- Quy tắc dừng
  - Phân loại chính xác trên dữ liệu training, hoặc
  - Tất cả các thuộc tính đều đã được sử dụng
- Quy tắc
  - Mỗi thuộc tính chỉ được sử dụng một lần duy nhất trên mỗi nhánh của cây.

# How to choose the optimal attributes?

- Ở mỗi node, cần chọn ra một tập các thuộc tính phân loại (**discriminative**)
  - có khả năng phân tách dữ liệu tốt nhất
- Làm sao để tìm thuộc tính phân loại?**
- Example: Giả sử có 2 class trong data. thuộc tính nào sẽ được chọn là **discriminative attribute**

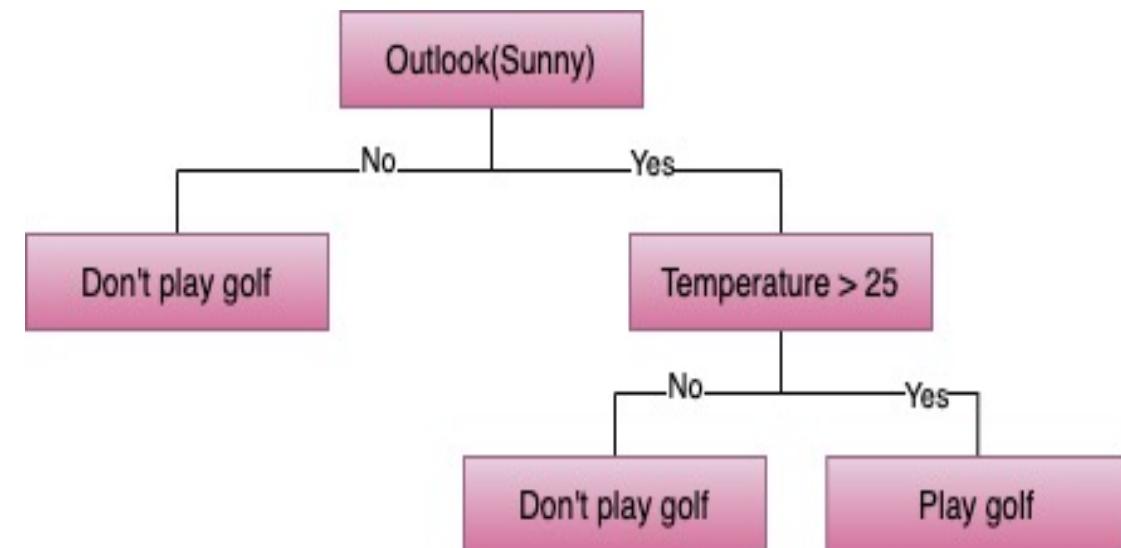


- Thước đo giá trị thông tin nhận được **Information gain**

# Decision Tree

Bài toán: Xem xét có nên đi chơi golf không?

- Đặt mục tiêu: **tối đa Information Gain (IG - giá trị thông tin nhận được).**
- Chọn nhánh (feature) nào mang lại nhiều giá trị thông tin nhất thì sẽ đánh giá feature đấy trước.
- Chọn **cut-off value** (giá trị phân chia) ở mỗi nhánh (feature)
  - Categorical (giá trị phân loại):** Thời tiết(mưa vs nắng)
  - Numerical (giá trị số):** Nhiệt độ (20, 30, 35)  
-> xác định giá trị nào mang lại IG lớn nhất để làm **cutoff value**



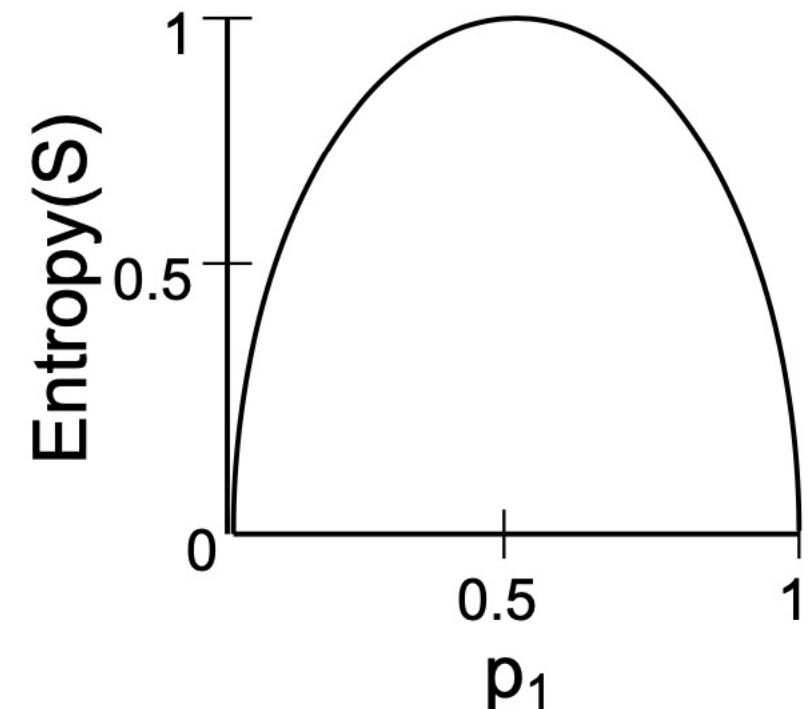
# Decision Tree – Entropy

$$E(S) = \sum_{i=1}^n p_i \log_2 p_i$$

$p_i$  là tỉ lệ các phần tử thuộc class  $i$

VD: For binary classification  $n = 2$

- **E** tinh khiết (purity):  $p_i = 0$  hoặc  $p_i = 1$
- **E** hỗn tạp (impurity):  $p_i = 0.5$ , khi đó hàm Entropy đạt đỉnh cao nhất



# Decision Tree – Entropy

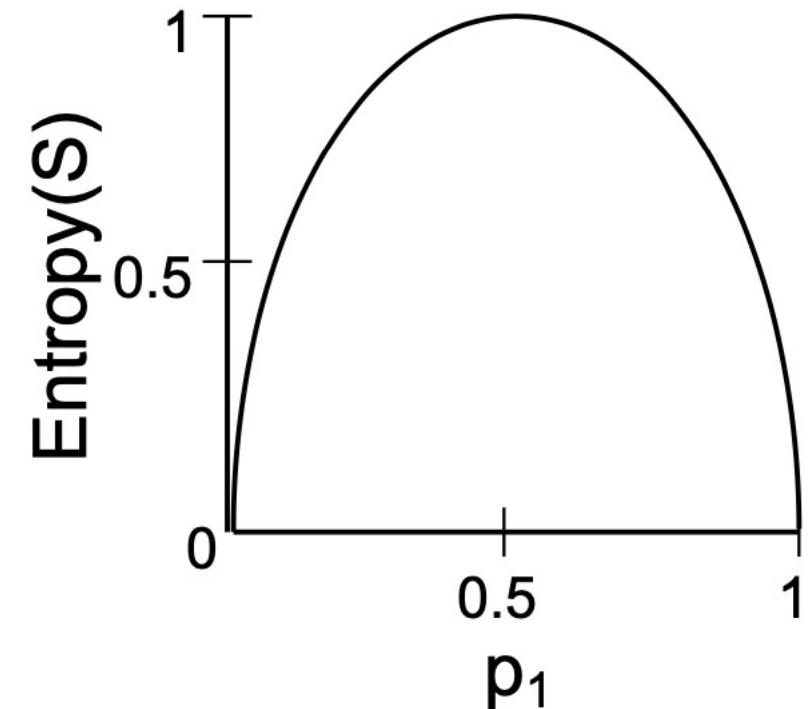
Example: Data  $S$

- $c1: 9$  giá trị
- $c2: 5$  giá trị

$\text{Entropy}(S) =$

$$-(9/14).\log_2(9/14) -(5/14).\log_2(5/14) \approx 0.94$$

- Entropy = 0 nếu tất cả giá trị của  $S$  trong cùng 1 lớp
- Entropy = 1 nếu hai lớp của  $S$  có cùng một size
- Entropy luôn có giá trị từ  $(0, 1)$



# Decision Tree – Information Gain

## Maximizing Information Gain = Minimizing Entropy

- Information gain (IG) của thuộc tính trong **S**
  - Đo lường lượng thông tin nhận được của mỗi đặc trưng
  - Đo mức độ giảm entropy tương ứng với mỗi đặc trưng

$$Gain(S, X) = Entropy(S) - Entropy(S, X)$$

- *Gain(T, X) là lượng thông tin còn lại khi chia **S** theo **X***

# Decision Tree – Example

- Play golf dataset:
  - Mục tiêu: Play golf (Yes), Don't play golf (No)
  - Thuộc tính (attributes):
    - Outlook: thời tiết
    - Temp: Nhiệt độ
    - Humidity: độ ẩm
    - Wind: Sức gió

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

# Decision Tree – Example

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

Play golf	
Yes	No
9	5



$$\begin{aligned}\text{Entropy(PlayGolf)} &= \text{Entropy}(5, 9) = \text{Entropy}(0.36, 0.64) \\ &= - (0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\ &= 0.94\end{aligned}$$

# Decision Tree – Example

		Play golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
		14		

$$\begin{aligned}
 E(\text{PlayGolf}, \text{Outlook}) &= P(\text{Sunny}) * E(3, 2) + P(\text{Overcast}) * E(4, 0) + P(\text{Rainy}) * E(2, 3) \\
 &= (5/14) * 0.971 + (4/14) * 0.0 + (5/14) * 0.971 \\
 &= 0.693
 \end{aligned}$$

-Outlook: thời tiết  
 -Overcast: nhiều mây

# Decision Tree – Example

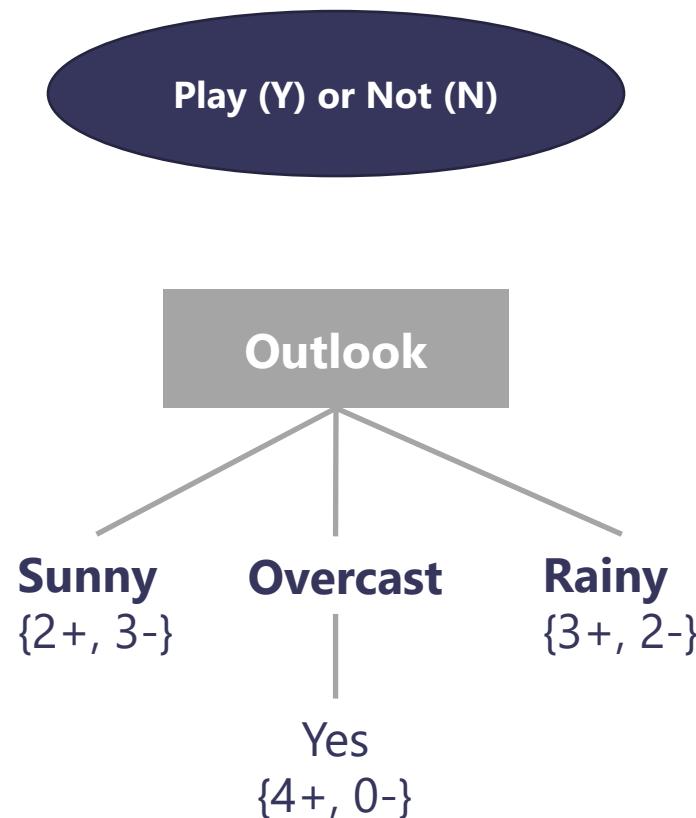
$$\begin{aligned} G(\text{PlayGolf}, \text{Outlook}) &= E(\text{PlayGolf}) - E(\text{PlayGolf}, \text{Outlook}) \\ &= 0.940 - 0.693 = \mathbf{0.247} \end{aligned}$$

$$\begin{aligned} G(\text{PlayGolf}, \text{Temperature}) &= E(\text{PlayGolf}) - E(\text{PlayGolf}, \text{Temperature}) \\ &= 0.940 - 0.911 = 0.029 \end{aligned}$$

$$\begin{aligned} G(\text{PlayGolf}, \text{Humidity}) &= E(\text{PlayGolf}) - E(\text{PlayGolf}, \text{Humidity}) \\ &= 0.940 - 0.788 = 0.152 \end{aligned}$$

$$\begin{aligned} G(\text{PlayGolf}, \text{Windy}) &= E(\text{PlayGolf}) - E(\text{PlayGolf}, \text{Windy}) \\ &= 0.940 - 0.893 = 0.048 \end{aligned}$$

# Decision Tree – Example



<b>id</b>	<b>outlook</b>	<b>temperature</b>	<b>humidity</b>	<b>wind</b>	<b>play</b>
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

<b>id</b>	<b>outlook</b>	<b>temperature</b>	<b>humidity</b>	<b>wind</b>	<b>play</b>
3	overcast	hot	high	weak	yes
7	overcast	cool	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes

<b>id</b>	<b>outlook</b>	<b>temperature</b>	<b>humidity</b>	<b>wind</b>	<b>play</b>
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

# Decision Tree – Example

Day	Outlook	Temp.	Humidity	Wind	Decision
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

$$\begin{aligned} E(\text{Sunny}) &= \left(-\frac{3}{5} \log_2 \left(\frac{3}{5}\right) - \frac{2}{5} \log_2 \left(\frac{2}{5}\right)\right) \\ &= 0.971 \end{aligned}$$

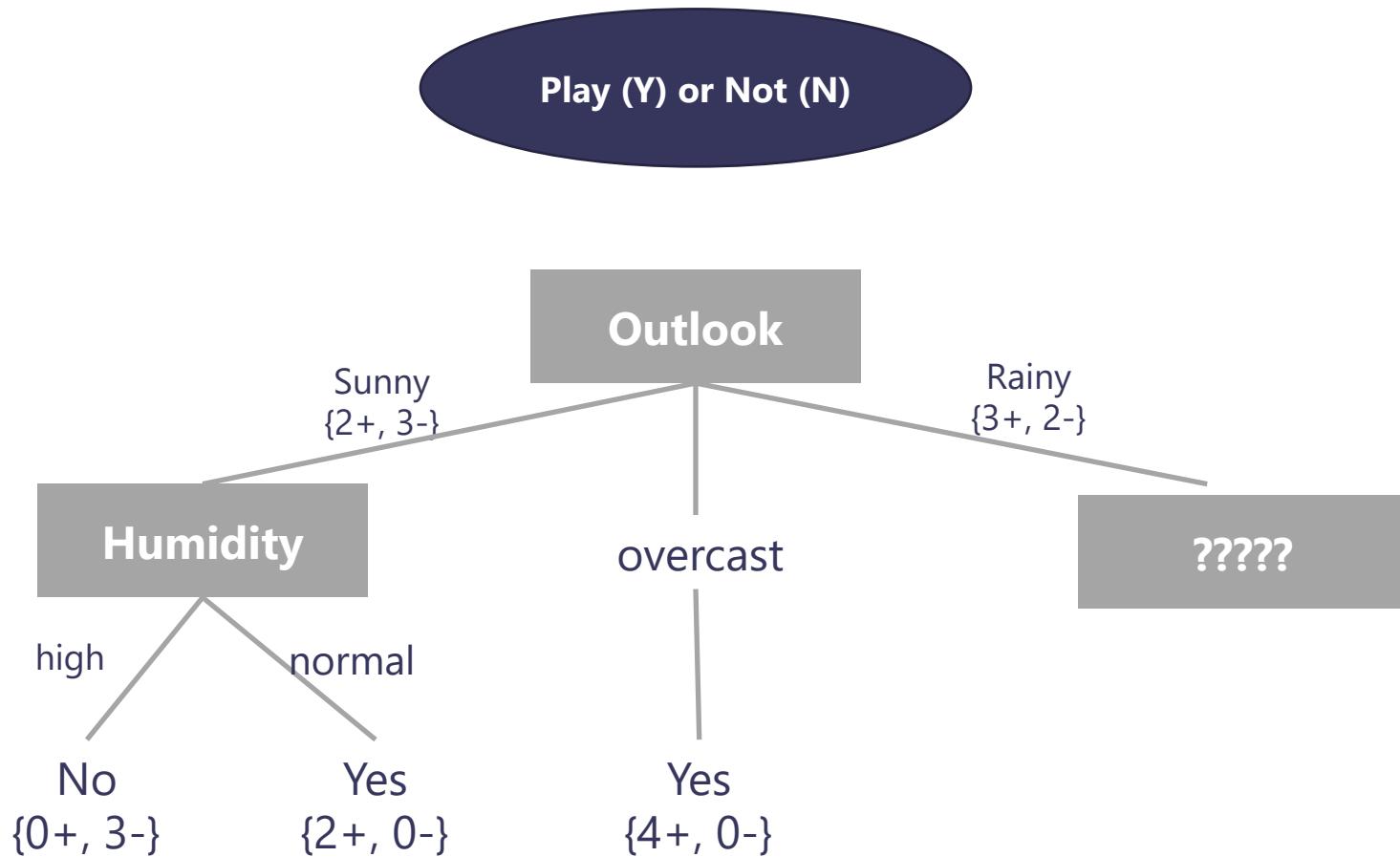
# Decision Tree – Example

$$\begin{aligned}G(\text{Sunny, Temperature}) &= E(\text{Sunny}) - E(\text{Sunny, Temperature}) \\&= 0.971 - 0.4 = 0.571\end{aligned}$$

$$\begin{aligned}G(\text{Sunny, Humidity}) &= E(\text{Sunny}) - E(\text{Sunny, Humidity}) \\&= \mathbf{0.971}\end{aligned}$$

$$\begin{aligned}G(\text{Sunny, Windy}) &= E(\text{Sunny}) - E(\text{Sunny, Windy}) \\&= 0.020\end{aligned}$$

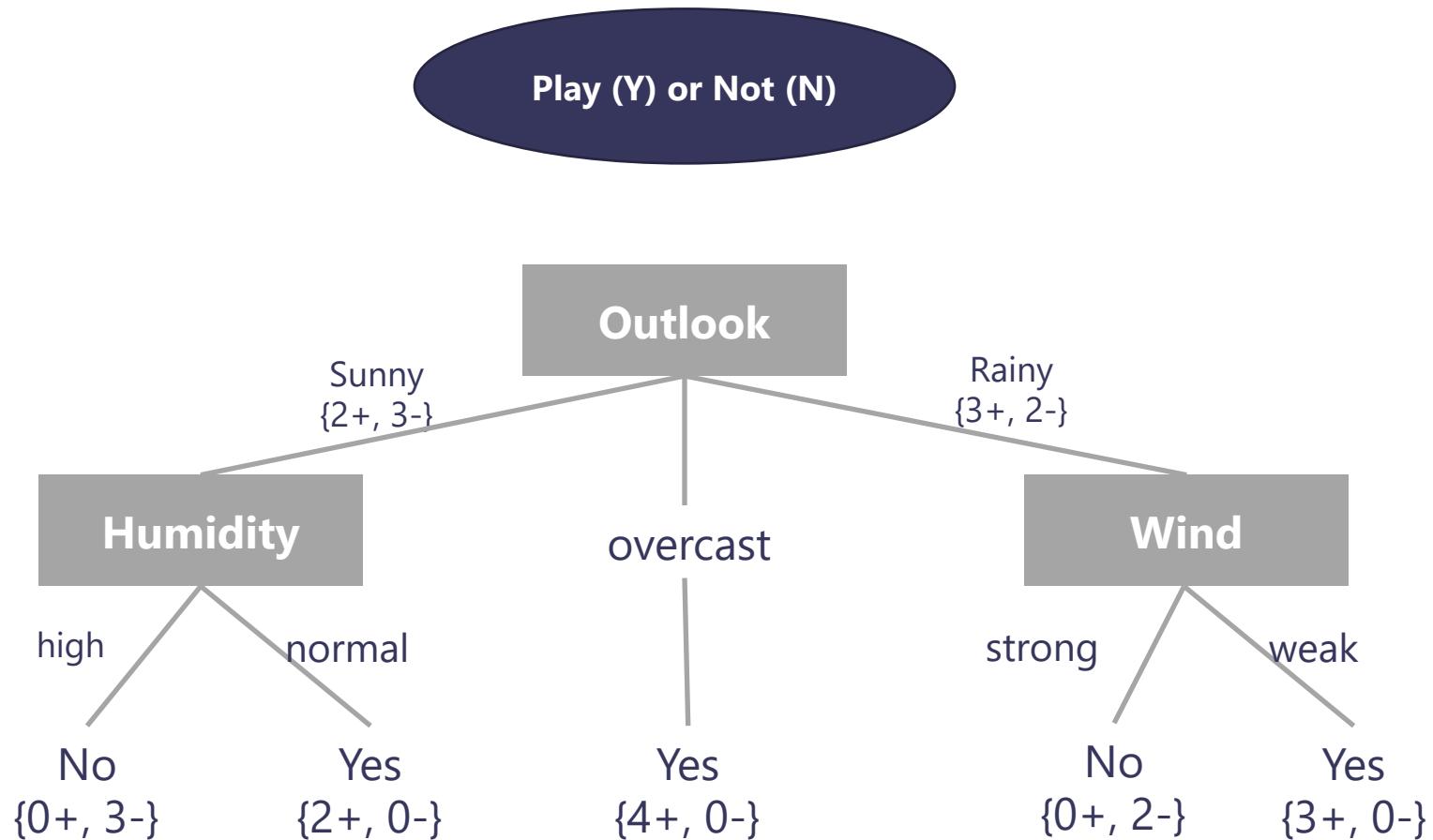
# Decision Tree – Example



# Decision Tree – Example

<b>id</b>	<b>outlook</b>	<b>temperature</b>	<b>humidity</b>	<b>wind</b>	<b>play</b>
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

# Decision Tree – Example



# Decision Tree – ID3 algorithm

- ID3 có khả năng diễn đạt hoàn toàn trên dữ liệu training
  - Phát triển dần dần các nhánh cây cho đến khi thỏa mãn tất cả các thuộc tính của tất cả các quan sát
- Information Gain quyết định hướng phát triển của Decision Tree
- Thuật toán ID3:
  - Xu hướng tạo ra các cây tối ưu cục bộ (**local optimal**)
  - ID3 không xem xét lại một thuộc tính đã lựa chọn trước đó

# Decision Tree – ID3 algorithm

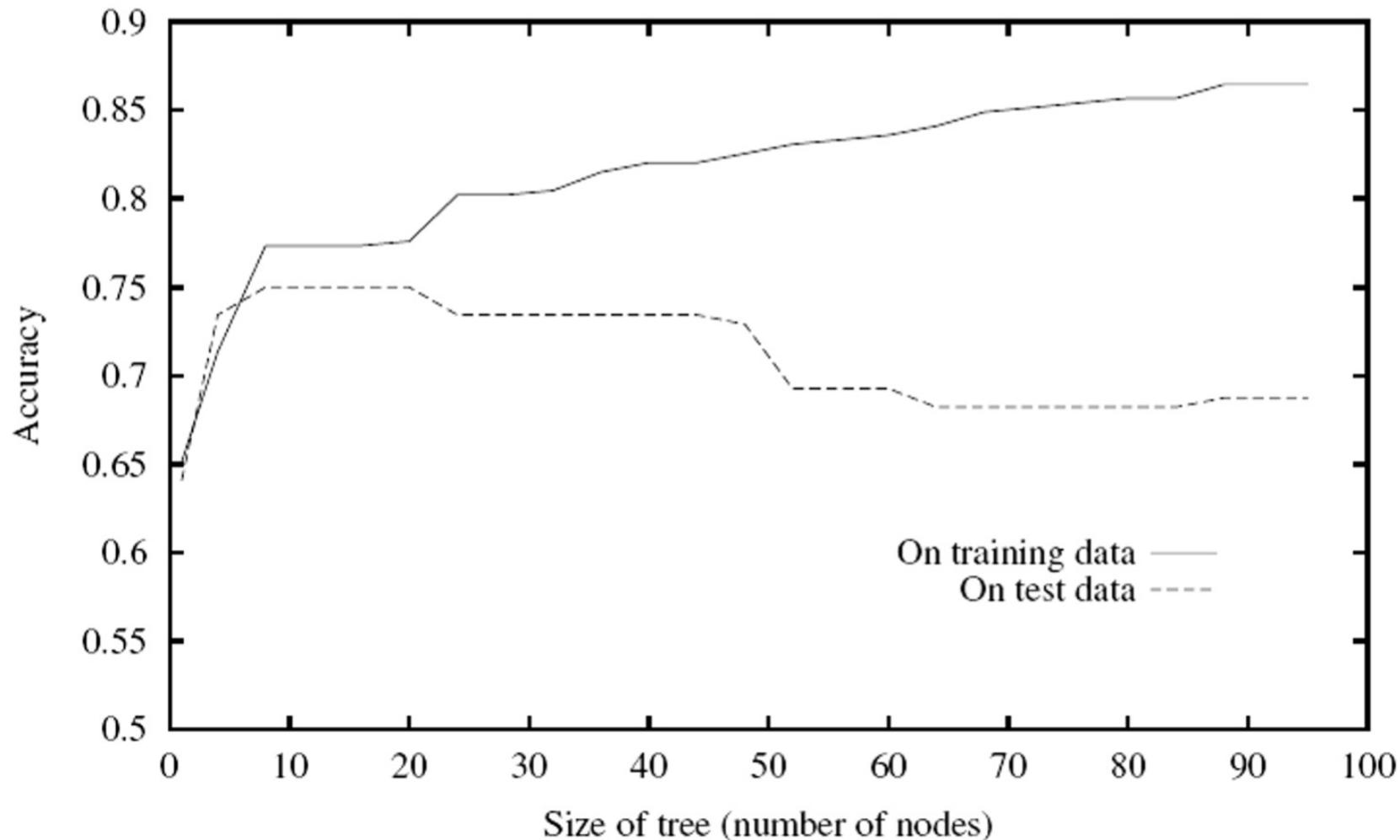
- **ID3 sẽ ưu tiên chọn cây nào ?**
  - ID3 ưu tiên cây đầu tiên mà fit với dữ liệu training
    - vì ID3 không xem xét lại các thuộc tính đã được chọn
- Thuật toán ID3:
  - Tạo ra các cây đơn giản, dễ giải thích
  - Các thuộc tính có IG cao sẽ được đặt gần với gốc của cây

# Decision Tree – ID3 problems in real world

- Decision Tree tạo bởi ID3 có thể bị overfitting trên tập dữ liệu training
- Không thích hợp cho các thuộc tính có giá trị thực (real number)
- Information Gain có phải là metric đo tốt nhất ?
- Missing values problems

# Overfitting in ID3

Example: Nếu tiếp tục phát triển cây có thể cải thiện accuracy trên dữ liệu training, tuy nhiên sẽ giảm trên dữ liệu testing.



# Overfitting in ID3: Solutions

- Stop learning early: dừng quá trình học của cây trước khi dữ liệu training được fit hoàn toàn.
  - > Rất khó để quyết định khi nào nên dừng
    - Tối thiểu số lượng mẫu có trong node phân nhánh.
    - Tối thiểu kích thước mẫu hay số lượng quan sát có trong leaf node
    - Tối đa số lượng thuộc tính dùng để phân nhánh
    - Tối đa chiều sâu của cây quyết định
- Pruning tree: để cây phát triển tối đa và sau đó tiến hành tัด (prune) các nhánh
  - > Khi nào quyết định dừng cắt tỉa? Kích cỡ tối ưu của cây là bao nhiêu?
    - Tất cả các quan sát nằm trong cùng một phân lớp
    - Tất cả các giá trị của biến dữ liệu là như nhau
- Sử dụng dữ liệu validation:
  - reduced-error pruning, and rule-post pruning.

# ID3: Gini index

$$\mathbf{Gini} = 1 - \sum_{i=1}^k [p_i]^2$$

$$\mathbf{Gini} (split) = \sum_{i=1}^n p_i (1 - p_i)$$

- Gini index là chỉ số đo lường mức độ đồng nhất, hay mức độ nhiễu loạn thông tin (purity).
- Gini Index thường được sử dụng phổ biến hơn.
- Hệ số Gini split càng nhỏ tức cách phân nhánh càng tối ưu

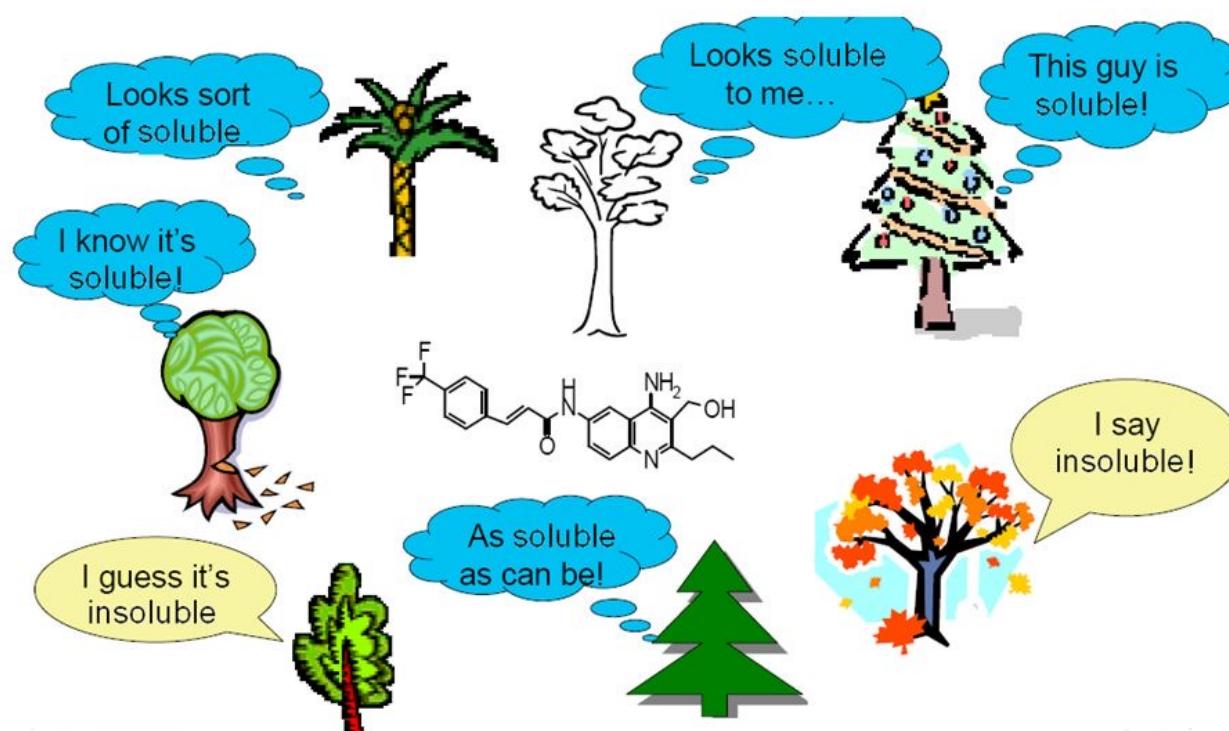
# Decision Tree – Missing or real values

- How to work with real attributes?
  - Dữ liệu, thuộc tính dạng số thực rất phổ biến
  - Transforming dữ liệu dạng số thực dưới dạng discretization (rời rạc hoá)
  - Ex: [0, 1] -> {[0, 0.25); [0.25, 0.5); [0.5, 0.75); [0.75, 1]}
- How to deal with missing values?
  - Missing value phổ biến trong các bài toán thực tế
  - Ex: Dữ liệu quan sát  $\mathbf{x}$  bị mất thông tin về thuộc tính  $\mathbf{x}_A$
  - Solution 1: fill  $\mathbf{x}_A$  với giá trị phổ biến nhất của thuộc tính A trong dữ liệu training
  - Solution 2: fill  $\mathbf{x}_A$  với giá trị phổ biến nhất của thuộc tính A mà có cùng class với  $\mathbf{x}$  trong dữ liệu training

# Random Forest

## Random Forest

Machine Learning Method



Random Forest là phương pháp được đề xuất bởi

Leo Breiman (2001) cho cả bài toán classification và regression.

Trong rừng cây, mỗi cây sẽ đưa ra một quyết định.

Kết quả cuối cùng dựa vào:

- Tính giá trị trung bình
- Quyết định dựa vào số đông (**majority voting**)

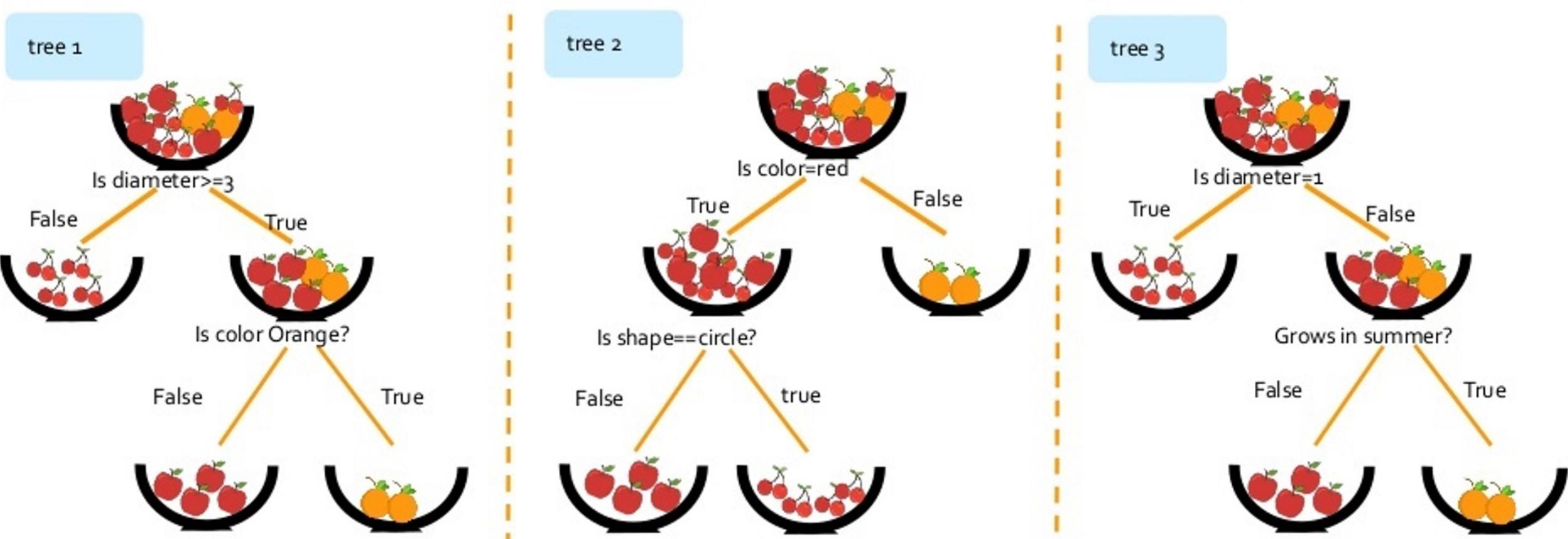
# Random Forest - Example

Bài toán: Làm thế nào để phân loại từng loại hoa quả



Color	Diameter	Label
Red	3	Apple
Yellow	3	Lemon
Purple	1	Grapes
Red	3	Apple
Yellow	3	Lemon
Purple	1	Grapes

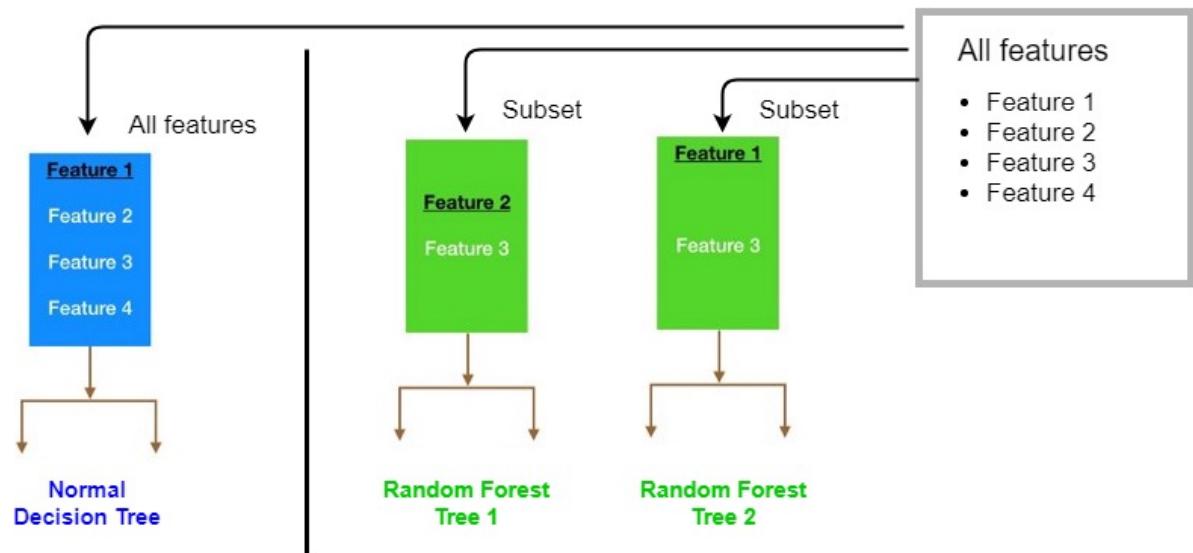
# Random Forest - Example



# Random Forest: 3 basic ingredients

- **Randomization**
  - Mỗi cây sẽ được train với một data ngẫu nhiên
  - Mỗi data sẽ lựa chọn một tập các thuộc tính ngẫu nhiên
  - Mỗi cây sẽ được phát triển tối đa và không cắt tỉa (pruning)
- **Combination**
  - Kết quả dự đoán của RF được tổng hợp trung bình từ mỗi Decision Tree độc lập
- **Bagging**
  - Dữ liệu training của Decision Tree được chọn ngẫu nhiên từ dữ liệu training ban đầu, có trùng lặp (replacement)

# Random Forest - Randomness



- **Random sampling:** mỗi cây trong RF sẽ được huấn luyện (train) với **một nhóm data ngẫu nhiên**.
- **Random feature selection:** mỗi cây trong RF sẽ sử dụng **một nhóm các features (subset of features)** khác nhau để đảm bảo tính khách quan
- Mỗi random tree sẽ được phát triển phụ thuộc vào thuộc tính và dữ liệu được lựa chọn ngẫu nhiên.

# Random Forest: Algorithm

- **Input:** dữ liệu training  $D$
- **Phương pháp học:** phát triển cây  $K$  theo
  - Tạo ra dữ liệu training  $D_i$  (sampling with replacement) từ dữ liệu  $D$
  - Training  $i^{th}$  tree từ dữ liệu  $D_i$ 
    - Lấy ngẫu nhiên  $S$  thuộc tính từ  $D_i$
    - Chia các node thành các nhánh con và lá tương ứng với mỗi thuộc tính  $S$
    - Phát triển cây tối đa và không cắt tỉa (pruning)
- **Prediction**
  - Lấy trung bình dự đoán từ tất cả các cây

# Random Forest

- **Advantages:**

- RF là một trong những phương pháp phổ biến và chính xác nhất
- RF có thể giảm overfitting so với Decision Tree và tăng độ chính xác
- Đơn giản, dễ triển khai và hiệu quả trong các bài toán thực tế
- Có thể ứng dụng trên cả dữ liệu categorical và continuous

- **Limitation:**

- Không phù hợp với dữ liệu mất cân bằng
- Khó giải thích kết quả (black-box model)

# Bagging

- Random Forest sử dụng ý tưởng **Bootstrap aggregation** (bagging):
  - Bootstrap:
    - Chia training data thành M samples và giữ nguyên các thuộc tính
    - Train mỗi decision tree trên mỗi bộ data sample tương ứng
  - Aggregate:
    - Kết quả cuối cùng tổng hợp của tất cả classifier trees
  - Mô hình thực thi song song
- Advantages
  - Ổn định, chính xác, có thể ứng dụng cho cả classification và regression.
  - Giảm variance, tránh overfitting.
  - Không bị phụ thuộc vào missing values

# Difference between Bagging vs Random forest

Bagging	Random Forest
Bagging is one of the oldest and simplest ensemble-based algorithms applied to tree-based algorithms to enhance predictive accuracy.	Random Forest is the enhanced version of bagging which is essentially an ensemble of decision trees trained with a bagging mechanism.
The concept of bagging is to train a bunch of unpruned decision trees on different random subsets of the training data, sampling with replacement.	The concept of random forests is to build multiple decision trees and aggregate them to get an accurate result with as little bias as possible.
It improves the accuracy and stability of machine learning models using ensemble learning and reduces the complexity of overfitting models.	The random forest algorithm is very robust against overfitting and it is good with unbalanced and missing data.



# Exercise

