**DECISION TREE CHO CLASSIFICATION**

**I/ CÁCH CHỌN YẾU TỐ PHÂN TÁCH**

**1. Dùng Entropy và Information Gain**

**Entropy** đo mức độ hỗn loạn của dữ liệu, công thức:

Entropy(S)=−∑pilog2pi

* S: Tập dữ liệu đang xét (ví dụ: danh sách khách hàng mua sản phẩm)
* pi​ : Xác suất của từng lớp (Yes/No, 0/1,...) trong tập dữ liệu
* ∑: Tổng hợp lại cho tất cả các lớp
* log2pi: Logarit cơ số 2 của xác suất, giúp chuẩn hóa thông tin

**Ý nghĩa của Entropy**

* Entropy = 0 → Tập dữ liệu hoàn toàn thuần nhất (chỉ có một nhãn)
* Entropy cao → Dữ liệu hỗn loạn, chứa nhiều loại nhãn khác nhau

**Ví dụ:**

* Tập A: {Yes, Yes, Yes, Yes} → Entropy = 0 (Dữ liệu thuần nhất)
* Tập B: {Yes, No, Yes, No} → Entropy = 1 (Dữ liệu lẫn lộn)

**Information Gain (IG)** đo mức độ giảm entropy sau khi chia:

IG=Entropy(parent)−∑(Entropy(Sv))

* Entropy(parent): Entropy trước khi chia tập dữ liệu
* Sv: Nhánh con sau khi chia theo một thuộc tính nào đó
* ∣Sv∣/∣S∣: Tỷ lệ kích thước của tập con so với tổng dữ liệu

**🡪 Chọn thuộc tính có IG cao nhất để phân tách trước**

**📌 2. Dùng Gini Index**

**Gini Index** đo mức độ tạp nhiễu của dữ liệu, công thức:

Gini(S)=1−∑pi2

 pi​: Xác suất của từng lớp (Yes/No, 0/1,...)

* ∑pi2 ​: Tổng bình phương xác suất của các lớp

**Gini = 0** → Nhóm thuần nhất

**Gini càng lớn** → Dữ liệu lẫn lộn nhiều

**🡪 Chọn thuộc tính có Gini thấp nhất để phân tách trước**

**DỰ ĐOÁN KHÁCH HÀNG CÓ MUA BẢO HIỂM KHÔNG?**

A screenshot of a credit score

AI-generated content may be incorrect.

**Bước 1: Tính Entropy của tập dữ liệu gốc**

Tính Entropy của tập dữ liệu ban đầu

* Có 3 Yes, 2 No
* Xác suất: PYes=3/5, PNo​=2/5​
* Áp dụng công thức:

Entropy(parent)=−(log2+log2) =−(0.6×(−0.737)+0.4×(−1.322)) =0.971

**Bước 2: Tính Entropy sau khi chia dữ liệu**

* **Chia theo thuộc tính "Student"**

| **Student = Yes** | * **Buy Insurance?** | | |
| --- | --- | --- | --- |
| Yes | No | | |
| Yes | Yes | | |
| **Student = No** | * **Buy Insurance?** | |
| No | No | |
| No | | Yes |
| No | | Yes |

**Tính Entropy cho mỗi nhóm**

**Nhóm "Student = Yes"** (2 mẫu, 1 Yes, 1 No)

Entropy=−(log2+log2)=1.0

**Nhóm "Student = No"** (3 mẫu, 2 Yes, 1 No)

Entropy=−(log2+log2) =0.918

* **Entropy sau khi chia theo "Student"**

Entropynew=(×1.0)+(×0.918)

* **Information Gain (IG)**

IG=0.971−0.950=0.021

**Bước 3: Chia theo "Age" và tìm threshold tối ưu**

**Chọn Threshold (Tuổi)**

Sắp xếp theo tuổi:

22, 25, 28, 35, 40

1. **Tạo danh sách các điểm cắt trung bình**

=23.5, =26.5, =31.5, =37.5

→ Các ngưỡng thử nghiệm: **23.5, 26.5, 31.5, 37.5**

1. **Tính Entropy và Information Gain cho từng threshold**
   * Chia dữ liệu theo mỗi threshold
   * Tính Entropy từng nhóm
   * Tính Information Gain
2. **Chọn threshold có Information Gain cao nhất** để chia dữ liệu.

Các điểm cắt thử nghiệm:

23.5, 26.5, 31.5, 37.5

**Tính Entropy cho mỗi threshold**

**Threshold = 26.5**

| **Age ≤ 26.5** | **Buy Insurance?** |
| --- | --- |
| 22 | No |
| 25 | No |
| **Age > 26.5** | **Buy Insurance?** |
| 28 | Yes |
| 35 | Yes |
| 40 | Yes |

* **Entropy (Age ≤ 26.5) (2 mẫu, 0 Yes, 2 No)**

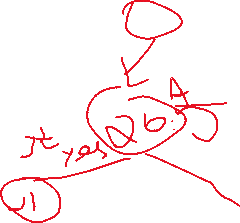
Entropy=−(1×log21+0×log20)=0

* **Entropy (Age > 26.5) (3 mẫu, 3 Yes, 0 No)**

Entropy=−(1×log21+0×log20)=0

* **Entropy sau chia theo Age (26.5)**

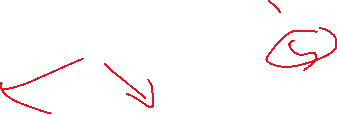
Entropynew=(×0)+(×0)=0



* **Information Gain (IG) của Age = 26.5**

IG=0.971−0=0.971

**🡪 Kết luận:**



* Vì Information Gain cao nhất ở Age ≤ 26.5, nên ta chọn Age là thuộc tính chia trước.
* "Age ≤ 26.5" là tiêu chí đầu tiên để chia nhánh
* Tiếp theo, xét thuộc tính "Student" hoặc "Credit Score"

**II/ Dùng Gini Index hay Information Gain để chọn yếu tố chia nhánh**

CART hoạt động bằng cách giảm Gini Index dần dần khi tạo cây quyết định. Trong khi đó, ID3 và C4.5 sử dụng Information Gain (hoặc Gain Ratio) để chọn thuộc tính tốt nhất.

**Gini Index (Dùng trong CART)**

Gini(A)=1−∑pi2

* CART chọn thuộc tính làm giảm Gini Index nhiều nhất.
* Khi chia nhánh, nếu một nhóm có toàn bộ cùng một lớp (ví dụ: 100% "Yes"), Gini Index = 0 (node thuần nhất).
* Nếu có sự pha trộn giữa các lớp (ví dụ: 50% "Yes" và 50% "No"), Gini Index cao hơn.

Mục tiêu của CART: Tạo cây nhị phân làm giảm Gini dần dần cho đến khi các node trở nên thuần nhất.

**Information Gain (Dùng trong ID3, C4.5)**

**Công thức Entropy**:

Entropy(A)=−∑pilog2pi



**Công thức Information Gain**:

IG(A)=Entropy(S)−∑\*Entropy(Sv)

* ID3 và C4.5 chọn thuộc tính có Information Gain cao nhất.
* Khi dữ liệu có một nhóm thuần nhất, Entropy = 0.
* Nếu dữ liệu có nhiều giá trị khác nhau, Entropy cao hơn.
* C4.5 cải tiến bằng cách sử dụng Gain Ratio để điều chỉnh bias do số lượng nhánh.

🡪Mục tiêu của ID3, C4.5: Chọn thuộc tính giúp giảm Entropy nhiều nhất.

**III/ Vì sao log trong công thức entropy, gini index có dạng như vậy**

**Entropy: Được xây dựng dựa trên lý thuyết thông tin của Shannon**

Claude Shannon, cha đẻ của lý thuyết thông tin, đã đưa ra khái niệm Entropy vào năm 1948 trong bài báo "A Mathematical Theory of Communication"

Entropy đo lường **mức độ không chắc chắn** hoặc **sự rối loạn** trong một tập dữ liệu. Công thức được xây dựng dựa trên nguyên tắc sau:

H(S)=−∑pi\*log2pi

**Log giúp đo lượng thông tin theo trực giác**

Trong lý thuyết thông tin, nếu một sự kiện hiếm khi xảy ra, nó mang nhiều thông tin hơn khi nó thực sự xảy ra. Ngược lại, nếu một sự kiện xảy ra thường xuyên, nó mang ít thông tin hơn.

Công thức Shannon định nghĩa lượng thông tin của một sự kiện có xác suất p là:

I(p)=−log2p

Nhờ log, khi xác suất nhỏ, lượng thông tin tăng mạnh. Khi xác suất lớn, lượng thông tin giảm nhẹ.

**Logarithm giúp biểu diễn thông tin một cách tuyến tính**

**Vấn đề nếu không dùng log**  
Giả sử chúng ta có hai sự kiện độc lập:

* A có xác suất 1/4 (25%)
* B có xác suất 1/8 (12.5%)

Tổng xác suất của cả hai sự kiện là:

P(A)×P(B)=1/4×1/8=1/32​

Nếu ta không dùng log, lượng thông tin sẽ tính bằng phép nhân, gây phức tạp:

I(A,B)=I(A)×I(B)

Nhưng nếu dùng logarithm, phép nhân sẽ trở thành phép cộng, giúp tính toán dễ dàng hơn:

I(A,B)=I(A)+I(B)

**Logarithm giúp chúng ta cộng lượng thông tin thay vì nhân xác suất, đơn giản hóa việc đo lường thông tin.**

**Log giúp đảm bảo tính chất thông tin hợp lý**

Shannon muốn một hàm đo thông tin phải thỏa mãn các điều kiện sau:

1. Nếu một sự kiện chắc chắn xảy ra (p=1), thì lượng thông tin bằng 0.
   * I(1)=−log2(1)=0
   * Vì nếu chúng ta đã biết chắc chắn, thì nó không mang thêm thông tin gì.
2. Nếu một sự kiện càng hiếm gặp (p→0), thì lượng thông tin càng lớn.
   * I(0.001)=−log2(0.001)=9.97
   * Vì nếu một sự kiện rất hiếm nhưng xảy ra, nó mang lại nhiều thông tin mới.
3. Nếu hai sự kiện độc lập xảy ra, tổng thông tin của chúng bằng tổng thông tin từng sự kiện. Dùng log giúp chúng ta cộng lượng thông tin thay vì nhân xác suất.

**Tại sao Shannon chọn log cơ số 2?**

* Vì log cơ số 2 giúp đo lường thông tin theo bit (có hoặc không, đúng hoặc sai, 0 hoặc 1).
* Một câu trả lời Yes/No tương đương với 1 bit thông tin.
* Nếu ta dùng log cơ số 10 hoặc log tự nhiên e, ta sẽ đo thông tin bằng digit hoặc nat thay vì bit.

Nếu xác suất p=1/8:

I(p)=−log2(1/8)=3 (bit)

I(p)=−log10(1/8)=0.9

**Shannon chọn log cơ số 2 để phù hợp với hệ nhị phân trong máy tính.**

Shannon dùng log trong Entropy. Vì:

Logarithm giúp đo lường thông tin theo trực giác – Sự kiện hiếm chứa nhiều thông tin hơn.  
Log chuyển phép nhân xác suất thành phép cộng thông tin – Đơn giản hóa tính toán.  
Log thỏa mãn tính chất hợp lý của thông tin – Giá trị bằng 0 khi xác suất = 1 và tăng khi xác suất giảm.  
Log cơ số 2 giúp đo lường thông tin theo bit, phù hợp với hệ nhị phân.

**Gini Index: Được xây dựng dựa trên nguyên lý đo sự hỗn loạn**

Leo Breiman, trong nghiên cứu về CART (Classification and Regression Trees) vào năm 1984.

**Gini Index đo lường xác suất để hai điểm dữ liệu được chọn ngẫu nhiên có nhãn khác nhau.**

**Gini(S)=1−∑pi2**

Nguyên tắc thiết kế công thức:

1. Nếu tất cả điểm dữ liệu thuộc cùng một lớp, thì Gini = 0 (hoàn hảo).
2. Nếu có nhiều lớp với xác suất gần bằng nhau, Gini cao hơn (tức là dữ liệu hỗn loạn hơn).
3. Không dùng log như Entropy vì Gini chỉ đo sự khác biệt giữa các xác suất chứ không đo lượng thông tin.
4. Nhanh hơn Entropy trong tính toán, vì không cần tính logarit.

**DECISION TREE CHO REGRESSION**

Khác với Decision Tree Classification, Decision Tree Regression không phân loại dữ liệu mà dự đoán giá trị số (continuous values).

Mục tiêu: Chia dữ liệu thành các nhóm nhỏ sao cho tổng sai số giảm dần.

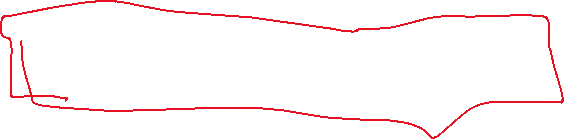
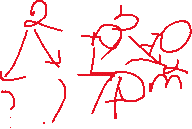
Tiêu chí chia nhánh: Sử dụng Mean Squared Error (MSE) hoặc Mean Absolute Error (MAE) để tìm điểm chia tốt nhất.

Khi nào dừng chia nhánh?

* Khi đạt đến số lượng điểm dữ liệu tối thiểu trong một node.
* Khi sai số giảm không đáng kể.

**A screenshot of a computer

AI-generated content may be incorrect.**



Xây dựng một mô hình Decision Tree Regression để dự đoán giá nhà dựa trên 3 đặc trưng:

* Diện tích (m²)
* Số phòng ngủ
* Khoảng cách đến trung tâm (km)

Trước khi chia, giá trung bình:

yˉ==1480

MSEparent==403680

**📌 Chia theo "Diện tích" (Threshold = 70 m²)**

* Nhánh trái (Diện tích < 70 m²): [800, 1000] (Mean = 900)
* Nhánh phải (Diện tích ≥ 70 m²): [1300, 1800, 2500] (Mean = 1866.67)

MSEleft==10000

MSEright=3=162222.2 MSEsplit=×10000+×162222.2=102444.4

Giảm MSE = 403680 - 102444.4 = 301235.6

**📌 Chia theo "Số phòng ngủ" (Threshold = 3)**

* Nhánh trái (≤3 phòng ngủ): [800, 1000, 1300] (Mean = 1033.33)
* Nhánh phải (>3 phòng ngủ): [1800, 2500] (Mean = 2150)

MSEleft=[(800−1033.33)2+(1000−1033.33)2+(1300−1033.33)2]/3=51777.78

MSEright=[(1800−2150)2+(2500−2150)2]/2=202500

MSEsplit=3/5×51777.78+2/5×202500=101033.33

Giảm MSE = 403680 - 101033.33 = 302646.67

**📌 Chia theo "Khoảng cách đến trung tâm" (Threshold = 5 km)**

* Nhánh trái (≥5 km): [800, 1000, 1300] (Mean = 1033.33)
* Nhánh phải (<5 km): [1800, 2500] (Mean = 2150)

Tương tự như trên, ta tính được:

MSEsplit=101033.33

Giảm MSE = 302646.67

**Chọn thuộc tính nào chia trước? Chọn thuộc tính có MSE giảm nhiều nhất!**



| **Thuộc tính** | **MSE sau chia** | **Giảm MSE** |
| --- | --- | --- |
| **Diện tích** (70 m²) | 102444.4 | 301235.6 |
| **Số phòng ngủ** (3) | 101033.33 | 302646.67 |
| **Khoảng cách** (5 km) | 101033.33 | 302646.67 |

* **Kết quả:**
* "Số phòng ngủ" và "Khoảng cách đến trung tâm" có MSE giảm nhiều nhất.
* Nếu hai giá trị có mức giảm MSE gần bằng nhau, ta có thể chọn dựa trên yếu tố quan trọng hơn về mặt kinh doanh.
* Nếu giá nhà bị ảnh hưởng mạnh hơn bởi khoảng cách đến trung tâm (theo kinh nghiệm thực tế), ta sẽ chọn Khoảng cách đến trung tâm làm nhánh đầu tiên.

**Chọn ngưỡng (Threshold) chia nhánh thế nào?**

Threshold tốt nhất là giá trị tạo ra MSE nhỏ nhất sau khi chia.  
Cách tìm threshold:

🔹 Chạy thử tất cả các giá trị trung bình giữa các điểm dữ liệu  
🔹 Chọn giá trị tạo ra MSE giảm nhiều nhất

Ví dụ với "Diện tích":  
Các threshold tiềm năng: (50+60)/2 = 55, (60+70)/2 = 65, (70+80)/2 = 75, (80+100)/2 = 90  
Tính MSE cho từng giá trị và chọn giá trị có MSE giảm nhiều nhất.

**Xây dựng cây:**

*Bước 1: Chia theo "Khoảng cách đến trung tâm" (Threshold = 5 km)*

Thuộc tính này có mức giảm MSE lớn nhất (302646.67), chia nhánh theo nó trước.

| Nhánh | Dữ liệu | Giá trị trung bình |
| --- | --- | --- |
| Nhánh trái (≥5 km) | [800, 1000, 1300] | 1033.33 |
| Nhánh phải (<5 km) | [1800, 2500] | 2150 |

*Bước 2: Chia tiếp nhánh trái (≥5 km) theo "Số phòng ngủ" (Threshold = 3)*

Vì "Số phòng ngủ" có mức giảm MSE nhiều nhất trong tập này (302646.67), chia theo thuộc tính này.

| Nhánh | Dữ liệu | Giá trị trung bình |
| --- | --- | --- |
| Nhánh con trái (≤3 phòng ngủ) | [800, 1000] | 900 |
| Nhánh con phải (>3 phòng ngủ) | [1300] | 1300 |

Nhánh "≤3 phòng ngủ" ([800, 1000]) đạt điều kiện dừng vì chỉ còn 2 phần tử. Nhánh ">3 phòng ngủ" ([1300]) đạt điều kiện dừng vì chỉ còn 1 phần tử.

*Bước 3: Chia tiếp nhánh phải (<5 km) theo "Diện tích" (Threshold = 70m²)*

Vì "Diện tích" có mức giảm MSE thấp nhất trong tập này (301235.6), chia theo nó.

| Nhánh | Dữ liệu | Giá trị trung bình |
| --- | --- | --- |
| Nhánh con trái (<70m²) | [1800] | 1800 |
| Nhánh con phải (≥70m²) | [2500] | 2500 |

Cả hai nhánh con đã đạt điều kiện dừng vì mỗi nhánh chỉ còn 1 phần tử.

**IV/ Lưu Ý Với Decision Tree**

Decision Tree xây dựng mô hình theo cách tham lam (greedy algorithm), nghĩa là tại mỗi bước, chỉ chọn thuộc tính tốt nhất để chia nhánh tại thời điểm đó, thay vì lập kế hoạch dài hạn xét tất cả các thuộc tính đồng thời. Điều này dẫn đến một số nguyên tắc chung về cách chọn thuộc tính và cách quyết định khi nào cần tiếp tục phân nhánh.

Không chia tất cả các thuộc tính ở mọi nhánh?

1. Mỗi bước chỉ chọn thuộc tính làm giảm độ lỗi (impurity) nhiều nhất
   * Sau khi chia theo một thuộc tính, phân bố dữ liệu con thay đổi, khiến mức độ quan trọng của các thuộc tính khác có thể thay đổi theo.
   * Việc chọn thuộc tính tiếp theo phải dựa trên tập dữ liệu con tại nhánh đó, không phải toàn bộ dữ liệu ban đầu.
2. Không phải thuộc tính nào cũng quan trọng trong mọi nhánh
   * Nếu một thuộc tính đã phân tách dữ liệu đủ tốt, các thuộc tính khác có thể trở nên không cần thiết hoặc ít ảnh hưởng.
   * Mỗi nhánh có thể có các thuộc tính quan trọng khác nhau, tùy theo cách dữ liệu được chia.
3. Tránh làm cây quá phức tạp và giảm overfitting
   * Nếu xét tất cả thuộc tính ở mọi nhánh, cây sẽ trở nên quá sâu và phức tạp, dễ dẫn đến overfitting.
   * Decision Tree dừng lại khi một nhánh đạt độ thuần khiết cao hoặc khi việc chia tiếp không mang lại lợi ích đáng kể.

Khái quát về cách chọn thuộc tính và thứ tự chia nhánh

* Ưu tiên thuộc tính giúp giảm độ lỗi nhiều nhất (giảm entropy🡪 tang IG hoặc giam Gini impurity trong classification, giảm MSE trong regression).
* Sau mỗi lần chia, chỉ xét tập dữ liệu con ở nhánh đó để quyết định thuộc tính tiếp theo.
* Không cần áp dụng tất cả thuộc tính trong mọi nhánh, mà chỉ chọn những thuộc tính có ý nghĩa trong từng bước phân tách.
* Dừng lại khi đạt độ thuần khiết cao hoặc khi không còn thuộc tính nào giúp phân tách dữ liệu hiệu quả hơn.

**V/ Ứng Dụng Trong Y Tế**

**Classification: Chẩn đoán bệnh Tiểu Đường**

**Bài toán: Xây dựng mô hình Decision Tree để phân loại bệnh nhân có mắc tiểu đường hay không dựa trên một số đặc điểm y tế.**

Dữ liệu đầu vào: Glucose (mg/dL), Huyết áp (mmHg), BMI (kg/m²), Tuổi (năm)

Tiền sử gia đình có người mắc bệnh (Có/Không)

Dữ liệu đầu ra: Mắc bệnh Tiểu Đường (Yes/No)

A black screen with many colorful text

AI-generated content may be incorrect.

**Regression: Dự đoán Chi phí điều trị Ung thư**

**Bài toán: Xây dựng mô hình Decision Tree để dự đoán chi phí điều trị ung thư dựa trên đặc điểm bệnh nhân.**  
Dữ liệu đầu vào: Tuổi (năm), Giai đoạn ung thư (I, II, III, IV), Số lần nhập viện trong năm qua, Chỉ số sức khỏe tổng quát (thang điểm 1-100)  
Dữ liệu đầu ra: Chi phí điều trị (USD)

A screenshot of a computer

AI-generated content may be incorrect.

**VI/ Giải quyết vấn đề Overfitting và Imbalance trong Decision Tree**

1. Giải quyết Overfitting trong Decision Tree

Overfitting xảy ra khi cây quá sâu, học quá kỹ dữ liệu huấn luyện và kém hiệu quả trên dữ liệu mới. Các phương pháp giải quyết bao gồm:

a. Pruning (Cắt tỉa cây quyết định)

* Pre-Pruning (Cắt sớm)
  + Giới hạn độ sâu: Hạn chế số mức (depth) tối đa của cây.
  + Giới hạn số lượng mẫu tối thiểu tại mỗi nút: Không chia tiếp nếu số mẫu trong một nút nhỏ hơn một ngưỡng nhất định.
  + Ngưỡng impurity: Ngừng phân tách nếu mức độ thuần khiết (Gini, entropy, MSE) không cải thiện đáng kể.
* Post-Pruning (Cắt sau)
  + Cắt bỏ nhánh không cần thiết sau khi xây cây hoàn chỉnh, giữ lại những nhánh giúp tăng độ tổng quát.
  + Thường dùng cost complexity pruning để loại bỏ nhánh có lợi ích nhỏ nhưng làm tăng độ phức tạp của mô hình.

b. Sử dụng Ensemble Learning (Random Forest, Gradient Boosting)

Random Forest: Kết hợp nhiều Decision Tree nhỏ với tập dữ liệu con khác nhau → Giảm overfitting bằng cách lấy trung bình nhiều mô hình.

Gradient Boosting (XGBoost, LightGBM, CatBoost): Xây dựng nhiều cây, mỗi cây học từ lỗi của cây trước đó, giúp cải thiện tổng thể mô hình mà không quá khớp dữ liệu.

c. Regularization (Giảm độ phức tạp)

Giới hạn số lượng thuộc tính được chọn tại mỗi lần chia (max\_features)

Tăng min\_samples\_split: Yêu cầu nhiều mẫu hơn để chia nhánh, tránh chia quá chi tiết.

Tăng min\_samples\_leaf: Đảm bảo mỗi lá có đủ số lượng mẫu, tránh trường hợp lá chỉ chứa một vài điểm dữ liệu.

2. Giải quyết Imbalanced Data trong Decision Tree

Dữ liệu mất cân bằng xảy ra khi một lớp xuất hiện nhiều hơn các lớp khác, khiến Decision Tree có xu hướng ưu tiên nhóm phổ biến.

a. Điều chỉnh trọng số lớp (Class Weighting)

* Sử dụng balanced weight trong các thư viện như sklearn:

from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(class\_weight="balanced")

* Hoặc tự thiết lập trọng số:

from collections import Counter

class\_weights = {0: 1, 1: sum(Counter(y\_train).values()) / Counter(y\_train)[1]}

model = DecisionTreeClassifier(class\_weight=class\_weights)

b. Oversampling hoặc Undersampling

* Oversampling (SMOTE - Synthetic Minority Over-sampling Technique): Tạo thêm dữ liệu tổng hợp cho lớp thiểu số.
* Undersampling: Giảm số lượng mẫu của lớp đa số để cân bằng dữ liệu.
* Kết hợp cả hai (SMOTE + Undersampling) để tránh tạo quá nhiều dữ liệu giả.

c. Sử dụng Ensemble Methods

* Balanced Random Forest: Chọn mẫu ngẫu nhiên từ mỗi lớp khi xây cây.
* Boosting Methods (XGBoost, LightGBM): Cân bằng trọng số lớp để tập trung học tốt hơn với lớp hiếm

**VI. Decision tree cho bài toàn dữ liệu là ảnh**

**Chi tiết dữ liệu đầu vào và đầu ra cho bài toán X-quang phổi bằng Decision Tree**

**Dữ liệu đầu vào**

* **Dataset**: Một tập hợp **ảnh X-quang phổi** đã được gán nhãn:
  + **0 (Bình thường)**: Phổi không có dấu hiệu bệnh lý.
  + **1 (Viêm phổi)**: Có dấu hiệu nhiễm trùng hoặc tổn thương phổi.
* **Kích thước ảnh**:
  + Ảnh X-quang thường có độ phân giải lớn (1024×1024 hoặc cao hơn).
  + Để xử lý hiệu quả, ta **resize về 224×224 pixels**. = 2048

**Tiền xử lý dữ liệu**

Chuyển ảnh thành dạng số: Vì Decision Tree không thể xử lý trực tiếp ảnh, ta cần chuyển ảnh thành các đặc trưng (feature vector).

Trích xuất đặc trưng bằng CNN (Phổ biến nhất)

* Sử dụng một mô hình CNN đã được huấn luyện trước (pre-trained), chẳng hạn VGG16, ResNet, MobileNet.
* Loại bỏ lớp Fully Connected cuối cùng, chỉ lấy đầu ra của tầng feature map.
* Kết quả: Một vector số biểu diễn đặc trưng của ảnh.
* Dữ liệu đầu vào cho Decision Tree: Mỗi ảnh sẽ được biểu diễn bằng một vector 2048 chiều

**Cấu trúc dữ liệu đầu vào**

Sau khi trích xuất đặc trưng, dữ liệu đầu vào sẽ là:

| **ID** | **Feature 1** | **Feature 2** | **...** | **Feature 2048** | **Label** |
| --- | --- | --- | --- | --- | --- |
| 1 | 0.12 | 0.35 | ... | 0.67 | 0 (Bình thường) |
| 2 | 0.45 | 0.23 | ... | 0.89 | 1 (Viêm phổi) |
| 3 | 0.08 | 0.75 | ... | 0.42 | 1 (Viêm phổi) |
| ... | ... | ... | ... | ... | ... |

**Dữ liệu đầu ra**

**🔹 Output của mô hình**

* Decision Tree sẽ dự đoán nhãn 0 (bình thường) hoặc 1 (viêm phổi).
* Có thể tính toán độ chính xác bằng accuracy, precision, recall, F1-score.

**Lựa chọn thuộc tính để chia nhánh**

Cây quyết định chọn thuộc tính quan trọng nhất để chia trước dựa trên:

* Entropy / Information Gain (ID3, C4.5)
* Gini Impurity (CART)

Ví dụ:

Nếu đặc trưng số 35 (tương ứng với một pattern trong ảnh) giúp phân loại tốt nhất, Decision Tree sẽ chọn nó làm nút gốc. Sau đó tiếp tục chia nhỏ dựa trên các đặc trưng khác.