**THỰC HÀNH: CÁC GIẢI THUẬT LOẠI CƠ BẢN**

2.1. GIẢI THUẬT 1: CÂY QUYẾT ĐỊNH VÀ RỪNG CÂY

2.1.1.

a) Quy trình khai phá dữ liệu CRISP – DM (Cross Industry Standard Process for Data Mining) là gì ? Quy trình khai phá dữ liệu SEMMA (Sample, Explore, Modify, Model, Access) là gì?

**Quy trình khai phá dữ liệu CRISP–DM và SEMMA**

Trong lĩnh vực khai phá dữ liệu (Data Mining), việc tổ chức các bước làm việc theo một quy trình chuẩn là vô cùng cần thiết để đảm bảo tính hệ thống, hiệu quả và khả năng tái sử dụng trong các dự án khác nhau. Hai quy trình phổ biến và được sử dụng rộng rãi hiện nay là **CRISP–DM (Cross Industry Standard Process for Data Mining)** và **SEMMA (Sample, Explore, Modify, Model, Assess)**. Mỗi quy trình đều có cấu trúc và mục tiêu riêng, song đều hướng đến việc khai thác tri thức hữu ích từ dữ liệu.

**A. Quy trình CRISP–DM (Cross Industry Standard Process for Data Mining)**

**Định nghĩa:**  
CRISP–DM là một mô hình quy trình chuẩn, mang tính **lặp đi lặp lại (iterative)**, được thiết kế để hướng dẫn toàn bộ quá trình thực hiện một dự án khai phá dữ liệu. Điểm đặc biệt của CRISP–DM là tính **độc lập với ngành nghề và công cụ sử dụng**, nhờ đó có thể áp dụng linh hoạt trong nhiều lĩnh vực khác nhau như tài chính, y tế, thương mại hay giáo dục.

**Sáu giai đoạn của CRISP–DM:**

1. **Hiểu biết về nghiệp vụ (Business Understanding):**  
   Ở giai đoạn này, nhóm dự án cần xác định rõ **mục tiêu kinh doanh**, **bối cảnh dự án** và **tiêu chí đánh giá thành công**. Đây là bước nền tảng để đảm bảo rằng toàn bộ quá trình khai phá dữ liệu sẽ phục vụ đúng nhu cầu thực tế của doanh nghiệp.
2. **Hiểu biết về dữ liệu (Data Understanding):**  
   Sau khi xác định mục tiêu, người thực hiện tiến hành **thu thập dữ liệu ban đầu**, **khám phá dữ liệu** và **đánh giá chất lượng dữ liệu**. Qua đó, ta có thể phát hiện những vấn đề như dữ liệu thiếu, dữ liệu sai hoặc không đồng nhất.
3. **Chuẩn bị dữ liệu (Data Preparation):**  
   Đây là công đoạn **tiền xử lý** để tạo ra tập dữ liệu sẵn sàng cho việc mô hình hóa. Các hoạt động chính gồm **làm sạch dữ liệu**, **chuyển đổi định dạng**, **chuẩn hóa** và **xây dựng các đặc trưng (features)** cần thiết.
4. **Mô hình hóa (Modeling):**  
   Giai đoạn này liên quan trực tiếp đến việc **lựa chọn và áp dụng các thuật toán khai phá dữ liệu** (như Decision Tree, Neural Network, SVM, v.v.) để **xây dựng mô hình dự đoán hoặc phân loại**.
5. **Đánh giá (Evaluation):**  
   Sau khi mô hình được tạo ra, cần **đánh giá mức độ chính xác và tính phù hợp** của mô hình theo tiêu chí kỹ thuật và nghiệp vụ. Bước này giúp đảm bảo mô hình đáp ứng đúng mục tiêu ban đầu đã đề ra.
6. **Triển khai (Deployment):**  
   Đây là bước **đưa mô hình vào ứng dụng thực tế**, chẳng hạn như **tích hợp vào hệ thống ra quyết định, tạo báo cáo phân tích**, hoặc **xây dựng công cụ hỗ trợ doanh nghiệp**.

**B. Quy trình SEMMA (Sample, Explore, Modify, Model, Assess)**

**Định nghĩa:**  
SEMMA là quy trình khai phá dữ liệu được phát triển bởi **SAS Institute**, tập trung nhiều hơn vào **các bước kỹ thuật** trong quá trình xây dựng mô hình. SEMMA hướng đến việc **khai thác tri thức thông qua phân tích thống kê và mô hình hóa dữ liệu**.

**Năm giai đoạn của SEMMA:**

1. **Lấy mẫu (Sample):**  
   Chọn một **tập con dữ liệu đại diện** có kích thước đủ lớn để phản ánh đặc trưng của toàn bộ dữ liệu, nhưng đủ nhỏ để đảm bảo khả năng xử lý hiệu quả.
2. **Khám phá (Explore):**  
   Tiến hành **phân tích dữ liệu bằng các công cụ thống kê và trực quan hóa** nhằm khám phá mối quan hệ, xu hướng và các điểm bất thường trong dữ liệu.
3. **Điều chỉnh (Modify):**  
   Thực hiện **làm sạch dữ liệu**, **xử lý các giá trị ngoại lai**, và **tạo ra các biến mới** giúp cải thiện khả năng dự đoán của mô hình.
4. **Mô hình hóa (Model):**  
   Ứng dụng các **kỹ thuật khai phá dữ liệu** như phân cụm, hồi quy, hoặc phân loại để **tìm ra các mẫu (patterns)** có ý nghĩa.
5. **Đánh giá (Assess):**  
   Đo lường và **đánh giá độ chính xác, độ tin cậy** của mô hình, xác định xem mô hình có phù hợp với mục tiêu ứng dụng hay không.

**C. So sánh hai quy trình CRISP–DM và SEMMA**

Mặc dù CRISP–DM và SEMMA đều là các mô hình khai phá dữ liệu có tính lặp lại, nhưng giữa chúng vẫn tồn tại một số **khác biệt đáng chú ý**:

* **Phạm vi:**  
  CRISP–DM có phạm vi **rộng hơn**, bao gồm cả các bước **hiểu biết nghiệp vụ và triển khai kết quả**. Trong khi đó, SEMMA chủ yếu tập trung vào **các khía cạnh kỹ thuật và thống kê**, từ việc lấy mẫu đến đánh giá mô hình.
* **Trọng tâm:**  
  CRISP–DM nhấn mạnh việc **gắn kết giữa kỹ thuật khai phá dữ liệu với mục tiêu kinh doanh thực tế**, còn SEMMA lại nhấn mạnh **quy trình xử lý và mô hình hóa dữ liệu**.
* **Tính lặp lại:**  
  Cả hai quy trình đều mang tính **lặp đi lặp lại (iterative)**, tuy nhiên CRISP–DM thể hiện rõ hơn tính **chu kỳ** — nghĩa là sau mỗi lần triển khai, người dùng có thể quay lại các bước trước để tinh chỉnh mô hình hoặc cập nhật dữ liệu mới.

**Kết luận** : Tóm lại, **CRISP–DM** và **SEMMA** đều là những khung quy trình quan trọng trong khai phá dữ liệu, mỗi mô hình có thế mạnh riêng tùy theo mục tiêu của dự án. Nếu **CRISP–DM** phù hợp với các dự án cần liên kết chặt chẽ giữa nghiệp vụ và kỹ thuật, thì **SEMMA** lại thích hợp cho các nhà phân tích dữ liệu chú trọng vào quá trình xử lý và mô hình hóa. Việc hiểu rõ và vận dụng linh hoạt hai quy trình này sẽ giúp các nhà khoa học dữ liệu đạt được hiệu quả tối ưu trong quá trình khai phá tri thức từ dữ liệu.

b) Cây quyết định hoạt động như thế nào? Hãy giải thích các thành phần chính (nút gốc, nút lá, nhánh) và cách cây đưa ra dự đoán.

### Cây Quyết định (Decision Tree) Hoạt Động Như Thế Nào?

Trong lĩnh vực học máy (Machine Learning), **Cây quyết định (Decision Tree – DT)** là một trong những mô hình được sử dụng phổ biến nhất nhờ tính dễ hiểu, khả năng diễn giải cao và hiệu quả trong cả bài toán **phân loại (Classification)** lẫn **hồi quy (Regression)**. Để hiểu rõ hơn, cần xem xét cơ chế hoạt động và các thành phần cấu tạo của một cây quyết định.

#### A. Khái niệm và Cơ chế Hoạt động

**Định nghĩa:**  
Cây quyết định (Decision Tree) là một **mô hình học máy có giám sát (Supervised Learning)**, mang tính **phi tham số (Non-parametric)**. Mô hình này không giả định trước dạng phân phối của dữ liệu, mà học trực tiếp từ dữ liệu để đưa ra quy tắc ra quyết định. Về hình thức, cây quyết định có **cấu trúc giống như một lưu đồ (flowchart)**, trong đó mỗi nút đại diện cho một câu hỏi hoặc điều kiện, và mỗi nhánh tương ứng với một kết quả có thể xảy ra của điều kiện đó.

**Cơ chế đưa ra dự đoán:**  
Quá trình dự đoán của cây quyết định diễn ra tuần tự theo các bước sau:

1. **Bắt đầu từ nút gốc (Root Node):**  
   Dữ liệu đầu vào được đưa vào cây tại nút gốc, đại diện cho toàn bộ tập dữ liệu huấn luyện.
2. **Theo các nhánh (Branches) dựa trên điều kiện kiểm tra:**  
   Ở mỗi nút, cây sẽ kiểm tra giá trị của một **thuộc tính (feature)** cụ thể — ví dụ: “Tuổi < 30?”, “Thu nhập > 10 triệu?”.  
   Dựa vào kết quả của điều kiện này, dữ liệu sẽ **rẽ nhánh** sang một hướng tương ứng.
3. **Lặp lại quá trình phân nhánh:**  
   Quá trình trên được lặp lại ở các nút con, với những điều kiện kiểm tra khác nhau, cho đến khi dữ liệu đến **nút lá (Leaf Node)**.
4. **Kết thúc tại nút lá:**  
   Nút lá là nơi chứa **kết quả dự đoán cuối cùng** — có thể là **nhãn lớp (Class Label)** trong bài toán phân loại hoặc **giá trị số (Value)** trong bài toán hồi quy.

Ví dụ, nếu một cây quyết định dùng để dự đoán việc “mua sản phẩm hay không” dựa trên độ tuổi và thu nhập, thì mô hình có thể hoạt động như sau:

* Nếu Tuổi < 30 → kiểm tra tiếp Thu nhập;
* Nếu Thu nhập > 10 triệu → **Dự đoán: Mua sản phẩm**;
* Ngược lại → **Không mua sản phẩm**.

Cơ chế này mô phỏng cách con người thường đưa ra quyết định — bằng cách đặt ra chuỗi câu hỏi “Nếu... thì...”, do đó giúp cây quyết định trở thành một công cụ vừa hiệu quả vừa dễ diễn giải.

#### B. Các Thành phần Chính của Cây Quyết định

Một cây quyết định gồm bốn thành phần cơ bản, mỗi thành phần đảm nhận một vai trò quan trọng trong quá trình học và dự đoán:

1. **Nút gốc (Root Node):**
   * Là **điểm khởi đầu** của cây và **đại diện cho toàn bộ tập dữ liệu** huấn luyện.
   * Tại đây, mô hình chọn ra **thuộc tính phân tách tốt nhất** để chia dữ liệu thành các nhóm nhỏ hơn.
   * Việc chọn thuộc tính này thường dựa vào các tiêu chí như **Information Gain**, **Gain Ratio** hoặc **Gini Index**.
   * Nút gốc **không có nhánh đi vào**, nhưng có thể có nhiều nhánh đi ra.
2. **Nhánh (Branches/Edges):**
   * Mỗi nhánh đại diện cho **một kết quả có thể xảy ra của điều kiện kiểm tra** tại nút cha.
   * Ví dụ, nếu điều kiện là “Tuổi < 30”, thì nhánh “Đúng” dẫn sang một nút con, còn nhánh “Sai” dẫn sang một hướng khác.
   * Các nhánh giúp kết nối **các nút cha và nút con**, tạo thành cấu trúc dạng cây.
3. **Nút bên trong (Internal hoặc Decision Node):**
   * Là các **nút trung gian** thực hiện **kiểm tra điều kiện** trên một thuộc tính cụ thể của dữ liệu.
   * Mỗi nút có **một nhánh đi vào** (từ nút cha) và **hai hoặc nhiều nhánh đi ra** (đến các nút con).
   * Các nút này chịu trách nhiệm **phân tách dữ liệu** sao cho các nhóm con thu được càng “thuần nhất” càng tốt.
4. **Nút lá (Leaf hoặc Terminal Node):**
   * Là **nút cuối cùng** trong cây, **không thể phân tách thêm**.
   * Chứa **giá trị đầu ra** của mô hình — tức là nhãn lớp (trong phân loại) hoặc giá trị dự đoán (trong hồi quy).
   * Mọi đường đi từ nút gốc đến nút lá đều tương ứng với **một quy tắc quyết định cụ thể** (Decision Rule).

### Kết luận

Tóm lại, **cây quyết định** là một mô hình học máy đơn giản nhưng mạnh mẽ, hoạt động bằng cách **phân tách dữ liệu dựa trên các điều kiện logic** cho đến khi đạt được kết quả dự đoán cuối cùng. Mỗi phần tử của cây — từ nút gốc, nhánh, nút bên trong đến nút lá — đều đóng vai trò quan trọng trong việc hình thành quy tắc ra quyết định. Nhờ cơ chế hoạt động dễ hiểu, khả năng mô phỏng cách con người suy nghĩ, và tính linh hoạt trong ứng dụng, cây quyết định đã trở thành một công cụ nền tảng trong khai phá dữ liệu và trí tuệ nhân tạo hiện đại.

c) Các tiêu chí phân tách (splitting criteria) như Gini Index, Entropy, hay Information Gain được sử dụng trong cây quyết định là gì? Chúng khác nhau ra sao?

### Các Tiêu Chí Phân Tách (Splitting Criteria) Trong Cây Quyết Định

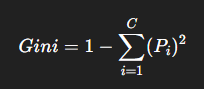
Trong quá trình xây dựng **cây quyết định (Decision Tree)**, việc lựa chọn **thuộc tính (feature)** nào để phân tách dữ liệu ở mỗi nút là yếu tố then chốt quyết định **độ chính xác** và **hiệu quả** của mô hình. Để xác định được thuộc tính “tốt nhất” tại mỗi bước, người ta sử dụng các **tiêu chí phân tách (splitting criteria)** nhằm đo lường mức độ “thuần khiết” hoặc “hỗn loạn” của dữ liệu sau khi phân chia. Ba tiêu chí phổ biến nhất hiện nay là **Gini Index**, **Entropy**, và **Information Gain**.

#### A. Tiêu chí Gini Index (Chỉ số Gini)

**Định nghĩa:**  
Chỉ số **Gini (Gini Index)** đo lường **mức độ tạp chất (impurity)** hoặc **độ không đồng nhất** của các mẫu trong một nút của cây quyết định. Nếu một nút chứa toàn bộ các mẫu thuộc cùng một lớp, thì nút đó được xem là **hoàn toàn thuần khiết**, và khi đó giá trị Gini bằng **0**.

**Mục tiêu:**  
Trong quá trình huấn luyện cây, thuật toán sẽ **chọn thuộc tính phân tách sao cho Gini Index sau khi tách là nhỏ nhất**. Điều này tương đương với việc **làm tăng độ thuần khiết** của các nút con.

**Công thức:**



Trong đó:

* C: số lượng lớp (classes),
* Pi​: tỷ lệ mẫu thuộc lớp i trong nút đang xét.

**Ý nghĩa:**

* Gini=0: tất cả mẫu thuộc cùng một lớp → nút thuần khiết.
* Gini càng cao → dữ liệu trong nút càng lẫn lộn, mức độ hỗn loạn càng lớn.

**Ví dụ:**  
Nếu trong một nút có 50% mẫu thuộc lớp “Có” và 50% thuộc lớp “Không”, thì:

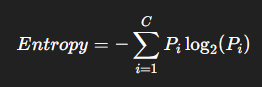
Gini= 1 - (0.5^2 + 0.5^2) = 0.5

→ Đây là một nút có độ hỗn loạn trung bình.

#### B. Tiêu chí Entropy (Độ Hỗn Loạn)

**Định nghĩa:**  
**Entropy** là một khái niệm xuất phát từ **lý thuyết thông tin (Information Theory)**, được dùng để đo lường **mức độ không chắc chắn (uncertainty)** hoặc **độ hỗn loạn (disorder)** của một nút trong cây quyết định. Một nút càng “lẫn lộn” giữa các lớp thì Entropy càng cao.

**Công thức:**



Trong đó:

* Pi​ là xác suất (tỷ lệ) của các mẫu thuộc lớp i trong nút.

**Đặc điểm:**

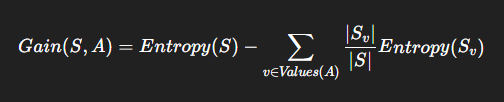
* Entropy=0: nút hoàn toàn thuần khiết (chỉ có một lớp).
* Entropy đạt giá trị tối đa khi các lớp xuất hiện đồng đều.

#### C. Tiêu chí Information Gain (Độ Lợi Thông Tin)

**Định nghĩa:**  
**Information Gain (Độ lợi thông tin)** đo lường **mức giảm Entropy** khi dữ liệu được phân tách theo một thuộc tính cụ thể. Nói cách khác, nó cho biết **thuộc tính đó giúp “giảm sự hỗn loạn” của dữ liệu nhiều đến mức nào**.

**Mục tiêu:**  
Chọn đặc trưng có **Information Gain cao nhất**, tức là đặc trưng mang lại nhiều “thông tin” nhất cho việc phân loại.

**Công thức:**



Trong đó:

* S: tập dữ liệu ban đầu,
* A: thuộc tính dùng để phân tách,
* Values(A): các giá trị có thể có của thuộc tính A,
* Sv​: tập con dữ liệu ứng với giá trị v của A.

**Giải thích:**

* Entropy(S): độ hỗn loạn của toàn bộ dữ liệu trước khi tách.
* ∑ : độ hỗn loạn trung bình sau khi tách.
* Do đó Gain(S,A) càng lớn → việc tách dữ liệu theo A càng hiệu quả.

**Ví dụ:**  
Nếu việc tách dữ liệu theo “Tuổi” giúp giảm Entropy nhiều hơn so với “Thu nhập”, thì “Tuổi” sẽ được chọn làm thuộc tính phân tách tại nút đó.

#### D. Sự Khác Biệt Giữa Các Tiêu Chí

| **Tiêu chí** | **Cơ sở lý thuyết** | **Mức độ tính toán** | **Kết quả phân tách** |
| --- | --- | --- | --- |
| **Gini Index** | Dựa trên xác suất thống kê | Nhanh, không cần logarit | Tương tự Entropy trong đa số trường hợp |
| **Entropy / Information Gain** | Dựa trên lý thuyết thông tin | Tính toán phức tạp hơn (vì có logarit) | Chính xác, nhạy với sự phân bố xác suất |

**Tóm lại:**

* **Gini Index** thường được dùng trong **thuật toán CART (Classification and Regression Tree)** do tốc độ tính toán cao.
* **Information Gain và Entropy** được sử dụng trong **thuật toán ID3 và C4.5**, chú trọng hơn đến khía cạnh lý thuyết thông tin.
* Mặc dù khác nhau về công thức và nền tảng toán học, nhưng **cả hai tiêu chí này thường cho kết quả phân tách tương tự** trong thực tế.

### Kết luận

Các tiêu chí phân tách như **Gini Index**, **Entropy**, và **Information Gain** là những thành phần cốt lõi trong quá trình xây dựng cây quyết định. Chúng giúp mô hình xác định **điểm cắt tối ưu**, đảm bảo mỗi bước phân tách đều hướng tới việc **tăng độ thuần khiết** của dữ liệu và **giảm sự hỗn loạn thông tin**. Việc lựa chọn tiêu chí phù hợp không chỉ ảnh hưởng đến **độ chính xác** của mô hình mà còn tác động đến **tốc độ huấn luyện và khả năng tổng quát hóa** của cây quyết định.

d) Rừng cây (Random Forest) là gì? Nó khác gì so với một cây quyết định đơn lẻ? Tại sao Random Forest thường có hiệu suất tốt hơn cây quyết định trong các bài toán phân loại?

**Mục tiêu:** Định nghĩa, so sánh với Cây Quyết định (Decision Tree) và giải thích lý do tại sao Random Forest thường có hiệu suất tốt hơn.

#### 1. Định nghĩa Random Forest (RF)

**Khái niệm:**  
Random Forest là một **thuật toán học máy kết hợp (Ensemble Learning)**, được phát triển dựa trên **kỹ thuật Bagging (Bootstrap Aggregating)**, sử dụng nhiều **Cây Quyết định (Decision Trees)** độc lập để đưa ra dự đoán chính xác và ổn định hơn.

**Cơ chế hoạt động:**

* **Bootstrap Sampling:** Từ tập dữ liệu huấn luyện ban đầu, tạo ra nhiều tập con bằng cách **lấy mẫu ngẫu nhiên có hoàn lại**. Mỗi tập con này được dùng để huấn luyện một cây quyết định riêng biệt.
* **Feature Randomness:** Khi xây dựng từng cây, chỉ chọn **một tập con ngẫu nhiên các đặc trưng** để tìm kiếm điểm phân tách tối ưu, giúp giảm tương quan giữa các cây.
* **Dự đoán:**
  + **Phân loại:** Kết quả được xác định bằng **bỏ phiếu đa số (Majority Voting)** từ tất cả các cây.
  + **Hồi quy:** Kết quả là **giá trị trung bình** của các dự đoán từ các cây trong rừng.

#### 2. Khác biệt giữa Cây Quyết định đơn lẻ và Random Forest

| **Đặc điểm** | **Cây Quyết định đơn lẻ (Decision Tree)** | **Random Forest (RF)** |
| --- | --- | --- |
| **Mô hình** | Một cây duy nhất, sử dụng toàn bộ dữ liệu | Tập hợp (Ensemble) của nhiều cây quyết định |
| **Phân tách** | Sử dụng tất cả các đặc trưng | Chỉ sử dụng một tập con ngẫu nhiên các đặc trưng |
| **Phương sai (Variance)** | Cao, dễ bị quá khớp (Overfitting) | Thấp hơn nhờ trung bình kết quả nhiều cây |
| **Độ ổn định** | Kém, thay đổi nhỏ trong dữ liệu có thể ảnh hưởng lớn | Ổn định hơn, do kết hợp nhiều mô hình con |

#### 3. Lý do Random Forest có hiệu suất tốt hơn Cây Quyết định

* **Giảm phương sai (Variance Reduction):**  
  Kỹ thuật Bagging và việc trung bình/bỏ phiếu kết quả giúp giảm phương sai, từ đó **giảm hiện tượng quá khớp**.
* **Giảm tương quan giữa các cây (Decorrelation):**  
  Việc chọn ngẫu nhiên các đặc trưng tại mỗi nút giúp các cây không học giống nhau, làm mô hình tổng hợp mạnh mẽ hơn.
* **Chống nhiễu và ngoại lai (Noise Robustness):**  
  Random Forest ít bị ảnh hưởng bởi nhiễu vì **sai lệch của một cây riêng lẻ sẽ được bù trừ** bởi kết quả của các cây khác trong rừng.

e) Những ưu điểm và hạn chế của cây quyết định và Random Forest là gì? Trong trường hợp nào thì cây quyết định có thể hoạt động kém hiệu quả?

**Mục tiêu:** Liệt kê các điểm mạnh, điểm yếu của hai mô hình Cây Quyết định (Decision Tree) và Rừng Cây (Random Forest), đồng thời chỉ ra những trường hợp Cây Quyết định hoạt động kém hiệu quả.

#### 1. Ưu điểm chung

* **Dễ hiểu và dễ giải thích (đặc biệt là Decision Tree):**  
  Cấu trúc dạng lưu đồ giúp người dùng có thể trực quan hóa và diễn giải kết quả một cách rõ ràng, phù hợp cho việc trình bày với các bên liên quan.
* **Không yêu cầu chuẩn hóa dữ liệu:**  
  Cả hai mô hình đều **không cần bước chuẩn hóa hay chuẩn tắc hóa** dữ liệu đầu vào (Scaling/Normalization), giúp tiết kiệm thời gian tiền xử lý.
* **Xử lý linh hoạt nhiều loại dữ liệu:**  
  Có khả năng xử lý cả **biến định tính (categorical)** và **biến định lượng (numerical)**, phù hợp với nhiều dạng bài toán thực tế.

#### 2. Hạn chế và nhược điểm

| **Mô hình** | **Ưu điểm nổi bật** | **Hạn chế nổi bật** |
| --- | --- | --- |
| **Cây Quyết định (Decision Tree)** | - Mô hình đơn giản, trực quan, dễ triển khai.  - Giải thích logic dễ hiểu. | - **Dễ bị quá khớp (Overfitting)** khi cây quá sâu.  - **Không ổn định**: dữ liệu thay đổi nhỏ có thể làm thay đổi cấu trúc cây đáng kể. |
| **Rừng Cây (Random Forest)** | - **Độ chính xác cao**, khả năng **chống quá khớp tốt** nhờ kết hợp nhiều cây.  - Cung cấp **Feature Importance** giúp đánh giá tầm quan trọng của các đặc trưng. | - **Tốn kém tính toán và bộ nhớ** do phải huấn luyện nhiều cây.  - **Khó giải thích hơn Decision Tree**, vì kết quả là tổng hợp của nhiều mô hình. |

#### 3. Các trường hợp Cây Quyết định hoạt động kém hiệu quả

* **Khi dữ liệu có mối quan hệ phức tạp:**  
  Cây Quyết định **không thể hiện tốt các mối quan hệ tuyến tính hoặc phi tuyến tính phức tạp**, vì mỗi nút chỉ dựa trên một điều kiện phân tách đơn giản.
* **Khi dữ liệu nhạy cảm hoặc thay đổi nhỏ:**  
  Do tính **không ổn định cao**, chỉ cần một thay đổi nhỏ trong dữ liệu huấn luyện cũng có thể dẫn đến **cấu trúc cây hoàn toàn khác**, gây ra kết quả không nhất quán.
* **Khi dữ liệu mất cân bằng hoặc có đặc trưng nhiều cấp độ:**  
  Cây Quyết định có xu hướng **thiên vị các đặc trưng có nhiều giá trị phân loại (nhiều cấp độ)** hoặc **thiên vị lớp chiếm ưu thế**, dẫn đến sai lệch trong dự đoán.

f) Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để xây dựng một mô hình cây quyết định không? Hãy mô tả các bước thực hiện

**Mục tiêu:** Trình bày các bước cơ bản để xây dựng, huấn luyện và đánh giá mô hình **Cây Quyết định (Decision Tree)** bằng thư viện **Scikit-learn**, theo hướng dẫn từng giai đoạn mà không cần viết mã chi tiết.

#### 1. Chuẩn bị Dữ liệu

* **Import Thư viện:**  
  Sử dụng các thư viện cần thiết như pandas, numpy để xử lý dữ liệu, và các công cụ trong sklearn.model\_selection và sklearn.tree như train\_test\_split và DecisionTreeClassifier.
* **Tải Dữ liệu:**  
  Nạp tập dữ liệu cần sử dụng, ví dụ tập dữ liệu **Iris**, hoặc một tập dữ liệu mẫu khác phù hợp cho bài toán phân loại.
* **Chia Dữ liệu:**  
  Sử dụng hàm train\_test\_split để chia dữ liệu thành hai phần:
  + **Tập huấn luyện (Training Set):** Dùng để huấn luyện mô hình.
  + **Tập kiểm tra (Testing Set):** Dùng để đánh giá hiệu quả của mô hình.  
    Thông thường, tỷ lệ chia là **70% – 30%** hoặc **80% – 20%**.

#### 2. Xây dựng và Huấn luyện Mô hình

* **Khởi tạo Mô hình:**  
  Tạo một đối tượng của lớp DecisionTreeClassifier với các tham số tùy chọn như:
  + criterion='gini' hoặc criterion='entropy' để xác định tiêu chí phân tách.
  + max\_depth để giới hạn độ sâu của cây, tránh hiện tượng quá khớp.
* **Huấn luyện Mô hình:**  
  Sử dụng phương thức fit(X\_train, y\_train) để tiến hành huấn luyện mô hình trên tập dữ liệu huấn luyện.

#### 3. Đánh giá Mô hình

* **Dự đoán Kết quả:**  
  Áp dụng phương thức predict(X\_test) để dự đoán nhãn (class label) cho tập kiểm tra.
* **Đánh giá Hiệu suất:**
  + Tính toán **độ chính xác (Accuracy)** bằng accuracy\_score.
  + Hiển thị **ma trận nhầm lẫn (Confusion Matrix)** để xem phân bố dự đoán đúng/sai giữa các lớp.
  + Tạo **báo cáo phân loại (Classification Report)** gồm độ chính xác (Precision), độ thu hồi (Recall) và điểm F1 cho từng lớp.

#### 4. Trực quan hóa Cây (Tùy chọn)

**- Hiển thị Cấu trúc Cây:**  
Sử dụng hàm plot\_tree trong sklearn.tree để **vẽ cây quyết định**, giúp quan sát các nhánh, điều kiện phân tách, và nhãn lớp ở các nút lá.  
Việc trực quan hóa này hỗ trợ người dùng **hiểu rõ quá trình ra quyết định** của mô hình.

1. Import thư viện cần thiết

from sklearn.datasets import load\_iris

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, classification\_report

import matplotlib.pyplot as plt

2. Tải và xem qua dữ liệu

# Tải dữ liệu Iris

iris = load\_iris()

# Tạo biến đặc trưng (X) và nhãn (y)

X = iris.data # 4 đặc trưng: chiều dài, chiều rộng cánh, đài hoa...

y = iris.target # 3 lớp: Setosa, Versicolor, Virginica

3. Chia dữ liệu thành tập huấn luyện và kiểm tra

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.3, random\_state=42

)

4. Khởi tạo và huấn luyện mô hình cây quyết định

# Tạo mô hình cây quyết định

model = DecisionTreeClassifier(

criterion="gini", # hoặc "entropy"

max\_depth=3, # giới hạn độ sâu để tránh overfitting

random\_state=42

)

# Huấn luyện mô hình

model.fit(X\_train, y\_train)

5. Dự đoán và đánh giá mô hình

# Dự đoán nhãn cho tập kiểm tra

y\_pred = model.predict(X\_test)

# Độ chính xác

print("Độ chính xác:", accuracy\_score(y\_test, y\_pred))

# Báo cáo chi tiết

print("\nBáo cáo phân loại:\n", classification\_report(y\_test, y\_pred))

6. Trực quan hóa cây quyết định

plt.figure(figsize=(10, 6))

plot\_tree(

model,

feature\_names=iris.feature\_names,

class\_names=iris.target\_names,

filled=True, rounded=True, fontsize=10

)

plt.show()

g) Làm thế nào để triển khai một mô hình Random Forest trong Python? Bạn thường thiết lập các tham số nào (ví dụ: n\_estimators, max\_depth)?

**Mục tiêu:** Trình bày các bước cơ bản để xây dựng, huấn luyện và đánh giá mô hình **Random Forest (RF)** bằng thư viện **Scikit-learn**, đồng thời mô tả các tham số (hyperparameters) quan trọng trong quá trình tối ưu mô hình.

#### 1. Chuẩn bị Dữ liệu

* **Import Thư viện:**  
  Sử dụng lớp RandomForestClassifier trong thư viện sklearn.ensemble, cùng với các công cụ quen thuộc như train\_test\_split, accuracy\_score, confusion\_matrix, và classification\_report.
* **Tải và Chia Dữ liệu:**  
  Thực hiện tương tự như phần Cây Quyết định (Decision Tree):
  + Nạp tập dữ liệu (ví dụ: **Iris dataset**).
  + Chia thành **tập huấn luyện (Training Set)** và **tập kiểm tra (Testing Set)** bằng hàm train\_test\_split.
  + Thông thường, tỷ lệ chia là **80% – 20%** hoặc **70% – 30%**.

#### 2. Xây dựng và Huấn luyện Mô hình Random Forest

* **Khởi tạo Mô hình:**  
  Tạo một đối tượng của lớp RandomForestClassifier với các tham số mong muốn (các tham số này có thể được tinh chỉnh để cải thiện hiệu suất).
* **Huấn luyện Mô hình:**  
  Sử dụng phương thức fit(X\_train, y\_train) để huấn luyện mô hình trên tập dữ liệu huấn luyện.  
  Trong quá trình này, mô hình sẽ tự động xây dựng nhiều cây quyết định độc lập bằng cách lấy mẫu ngẫu nhiên và lựa chọn đặc trưng ngẫu nhiên cho từng cây.

#### 3. Thiết lập Các Tham số Quan trọng (Hyperparameters)

* **n\_estimators (Số lượng cây trong rừng):**
  + **Mô tả:** Xác định số lượng cây quyết định sẽ được tạo ra trong mô hình.
  + **Thiết lập:** Giá trị càng cao thường mang lại kết quả ổn định hơn và chính xác hơn, nhưng cũng làm tăng thời gian huấn luyện.
  + **Gợi ý:** Thông thường chọn trong khoảng **100 – 500** cây.
* **max\_depth (Chiều sâu tối đa của cây):**
  + **Mô tả:** Giới hạn chiều sâu tối đa mà mỗi cây có thể phát triển.
  + **Thiết lập:** Dùng để **kiểm soát hiện tượng quá khớp (Overfitting)**. Nên chọn ở mức vừa phải, ví dụ **10 – 20**.
* **min\_samples\_leaf (Số lượng mẫu tối thiểu tại nút lá):**
  + **Mô tả:** Quy định số lượng mẫu tối thiểu cần có trong mỗi nút lá.
  + **Thiết lập:** Giá trị cao hơn giúp mô hình **tổng quát hóa tốt hơn**, giảm nhiễu và hạn chế quá khớp.

#### 4. Đánh giá Mô hình

* **Dự đoán:**  
  Sử dụng phương thức predict(X\_test) để dự đoán nhãn của tập kiểm tra.
* **Đánh giá Hiệu suất:**
  + Tính toán **độ chính xác (Accuracy)** bằng accuracy\_score.
  + Sử dụng **ma trận nhầm lẫn (Confusion Matrix)** để phân tích kết quả dự đoán.
  + Tạo **báo cáo phân loại (Classification Report)** để xem các chỉ số như **Precision**, **Recall**, và **F1-score**.

h) Làm thế nào để đánh giá tầm quan trọng của các đặc trưng (feature importance) trong RandomForest bằng Python?

#### **Mục tiêu:**

Trình bày phương pháp đánh giá tầm quan trọng của các đặc trưng (Feature Importance) trong mô hình Random Forest và cách triển khai bằng Python.

#### **A. Khái niệm Feature Importance**

**Định nghĩa:**  
Feature Importance là một chỉ số đo lường mức độ đóng góp của từng đặc trưng (feature) đối với độ chính xác của mô hình học máy, cụ thể là mô hình Random Forest. Nói cách khác, chỉ số này cho biết đặc trưng nào có ảnh hưởng lớn nhất đến việc ra quyết định của mô hình.

**Cơ chế tính toán (Gini Importance - Mean Decrease Impurity, MDI):**  
Trong quá trình huấn luyện, mỗi cây trong rừng quyết định (Decision Tree) sẽ chia dữ liệu tại các nút dựa trên các đặc trưng khác nhau. Mức độ giảm “độ hỗn loạn” (ví dụ như **Gini Index** hoặc **Entropy**) khi chia theo một đặc trưng được ghi nhận lại.  
Tổng hợp qua tất cả các cây trong rừng, đặc trưng nào giúp giảm độ hỗn loạn nhiều nhất sẽ được gán **độ quan trọng cao hơn**.

#### **B. Các bước triển khai bằng Python (Scikit-learn)**

1. **Huấn luyện mô hình Random Forest:**  
   Đầu tiên, mô hình RandomForestClassifier được khởi tạo và huấn luyện với tập dữ liệu huấn luyện (X\_train, y\_train) bằng phương thức fit().
2. **Truy cập thuộc tính Feature Importance:**  
   Sau khi huấn luyện, có thể truy cập trực tiếp tầm quan trọng của các đặc trưng thông qua thuộc tính:

model.feature\_importances\_

Trong đó, mỗi giá trị trong mảng tương ứng với mức độ quan trọng của từng đặc trưng.

1. **Xử lý và hiển thị dữ liệu tầm quan trọng:**
   * Tạo một **DataFrame** để ánh xạ giữa tên đặc trưng và giá trị tầm quan trọng.
   * Sắp xếp danh sách theo thứ tự giảm dần để xác định các đặc trưng có ảnh hưởng lớn nhất đến mô hình.  
     Ví dụ:

import pandas as pd

feature\_importance = pd.DataFrame({

'Feature': feature\_names,

'Importance': model.feature\_importances\_

}).sort\_values(by='Importance', ascending=False)

1. **Trực quan hóa kết quả:**

- Có thể sử dụng **matplotlib** hoặc **seaborn** để vẽ biểu đồ thanh thể hiện tầm quan trọng của các đặc trưng.  
Ví dụ:

import seaborn as sns

import matplotlib.pyplot as plt

sns.barplot(x='Importance', y='Feature', data=feature\_importance.head(10))

plt.title('Top 10 Đặc trưng quan trọng nhất trong Random Forest')

plt.show()

#### **C. Kết luận:**

Việc đánh giá tầm quan trọng của các đặc trưng giúp hiểu rõ hơn về cách mô hình đưa ra quyết định, hỗ trợ trong việc **giải thích mô hình** (Model Interpretability) và **giảm chiều dữ liệu** (Feature Selection). Những đặc trưng có tầm quan trọng thấp có thể được loại bỏ để tối ưu hiệu năng mô hình mà không làm giảm đáng kể độ chính xác.

i) Điều chỉnh siêu tham số (hyperparameter tuning) cho cây quyết định hoặc Random Forest chưa? Hãy mô tả cách bạn sử dụng GridSearchCV hoặc RandomizedSearchCV

#### **Mục tiêu:**

Trình bày các phương pháp điều chỉnh siêu tham số (Hyperparameter Tuning) nhằm tối ưu hiệu suất mô hình, bao gồm **GridSearchCV** và **RandomizedSearchCV** trong thư viện Scikit-learn.

#### **A. Khái niệm Điều chỉnh Siêu Tham số (Hyperparameter Tuning)**

**Định nghĩa:**  
Siêu tham số (Hyperparameter) là các tham số được cấu hình **bên ngoài quá trình huấn luyện** của mô hình học máy, ví dụ như max\_depth, n\_estimators, criterion trong Random Forest.  
**Điều chỉnh siêu tham số** là quá trình tìm kiếm tổ hợp các giá trị tối ưu cho các tham số này nhằm đạt **hiệu suất mô hình cao nhất** trên tập kiểm định chéo (cross-validation).

#### **B. Phương pháp GridSearchCV**

**Mô tả:**  
GridSearchCV là phương pháp tìm kiếm toàn diện (Exhaustive Search) trong một “lưới” các giá trị siêu tham số được xác định trước.  
Cụ thể, phương pháp này sẽ huấn luyện mô hình với **tất cả các tổ hợp có thể có** của các tham số trong lưới, đồng thời sử dụng **Cross-Validation** để đánh giá hiệu suất từng tổ hợp.

**Ưu điểm:**

* Đảm bảo tìm thấy **tổ hợp siêu tham số tốt nhất** trong phạm vi đã định nghĩa.
* Dễ hiểu, dễ triển khai.

**Hạn chế:**

* **Tốn kém thời gian và tài nguyên tính toán**, đặc biệt khi số lượng tham số hoặc phạm vi giá trị lớn.

**Ví dụ minh họa:**

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import RandomForestClassifier

param\_grid = {

'n\_estimators': [100, 200, 300],

'max\_depth': [5, 10, 20],

'min\_samples\_leaf': [1, 2, 4]

}

grid\_search = GridSearchCV(

estimator=RandomForestClassifier(),

param\_grid=param\_grid,

cv=5,

n\_jobs=-1

)

grid\_search.fit(X\_train, y\_train)

best\_params = grid\_search.best\_params\_

#### **C. Phương pháp RandomizedSearchCV**

**Mô tả:**  
RandomizedSearchCV là phương pháp **tìm kiếm ngẫu nhiên** (Random Search), trong đó chỉ **chọn ngẫu nhiên một số lượng tổ hợp nhất định** của các siêu tham số trong phạm vi tìm kiếm.  
Tương tự GridSearchCV, phương pháp này cũng sử dụng **Cross-Validation** để đánh giá hiệu suất của từng tổ hợp được chọn.

**Ưu điểm:**

* Hiệu quả hơn nhiều về **thời gian tính toán**, đặc biệt khi số lượng siêu tham số lớn.
* Có khả năng **tìm được kết quả gần tối ưu** nhanh hơn so với GridSearchCV.

**Hạn chế:**

* Không đảm bảo tìm thấy tổ hợp **tốt nhất tuyệt đối**, do chỉ xét một phần nhỏ không gian tham số.

**Ví dụ minh họa:**

from sklearn.model\_selection import RandomizedSearchCV

from sklearn.ensemble import RandomForestClassifier

from scipy.stats import randint

param\_dist = {

'n\_estimators': randint(100, 500),

'max\_depth': randint(5, 30),

'min\_samples\_leaf': randint(1, 5)

}

random\_search = RandomizedSearchCV(

estimator=RandomForestClassifier(),

param\_distributions=param\_dist,

n\_iter=20,

cv=5,

n\_jobs=-1,

random\_state=42

)

random\_search.fit(X\_train, y\_train)

best\_params = random\_search.best\_params\_

#### **D. Các bước chung khi áp dụng (ví dụ với Random Forest)**

1. **Định nghĩa phạm vi tham số:**
   * Tạo param\_grid hoặc param\_distributions để xác định các siêu tham số cần điều chỉnh và phạm vi giá trị tương ứng.
2. **Khởi tạo đối tượng tìm kiếm:**
   * Chọn phương pháp phù hợp: GridSearchCV (tìm toàn bộ) hoặc RandomizedSearchCV (tìm ngẫu nhiên).
   * Truyền mô hình cơ sở (ví dụ: RandomForestClassifier), lưới tham số, số lần kiểm định (cv) và số lượng mẫu thử (n\_iter nếu dùng RandomizedSearchCV).
3. **Thực hiện tìm kiếm:**
   * Gọi phương thức fit(X\_train, y\_train) để tiến hành tìm kiếm và huấn luyện mô hình.
4. **Trích xuất kết quả:**
   * Lấy tổ hợp tham số tối ưu bằng best\_params\_.
   * Lấy mô hình tối ưu bằng best\_estimator\_.

#### **E. Kết luận:**

Điều chỉnh siêu tham số là một bước **quan trọng trong quá trình tối ưu mô hình học máy**, giúp cải thiện độ chính xác và khả năng khái quát của mô hình.  
Tùy theo quy mô dữ liệu và tài nguyên tính toán, có thể chọn **GridSearchCV** để tìm kiếm toàn diện hoặc **RandomizedSearchCV** để đạt kết quả nhanh và hiệu quả hơn.