**Thuật toán cây quyết định** (Decision Tree) là một phương pháp học máy giám sát, thường được sử dụng trong các bài toán phân loại và hồi quy. Thuật toán này hoạt động bằng cách chia tập dữ liệu thành các nhóm con dựa trên các thuộc tính (features) của dữ liệu, tạo ra một cấu trúc cây với các nhánh tương ứng với các điều kiện kiểm tra trên các thuộc tính và các lá chứa các giá trị dự đoán.

Cây quyết định ([Decision Tree](https://trituenhantao.io/tu-dien-thuat-ngu/decision-tree/)) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Các thuộc tính của đối tượngncó 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.

Tóm lại, cho 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.

**Quá trình tạo cây quyết định:**

1. **Chọn thuộc tính phân chia tốt nhất:** Sử dụng các chỉ số như Entropy, Information Gain (ID3), Gini Index (CART) để xác định thuộc tính nào sẽ giúp phân chia dữ liệu tốt nhất.
2. **Tạo các nhánh dựa trên giá trị của thuộc tính đã chọn:** Mỗi giá trị của thuộc tính sẽ tạo thành một nhánh riêng.
3. **Lặp lại cho từng nhánh con:** Đối với mỗi nhánh con, tiếp tục quá trình tìm kiếm thuộc tính tốt nhất để phân chia dữ liệu, cho đến khi đạt điều kiện dừng (ví dụ: không còn thuộc tính nào để chia, đạt độ sâu tối đa, hoặc đạt độ chính xác mong muốn).

**Ví dụ triển khai cây quyết định với scikit-learn**

|  |
| --- |
| **# Import các thư viện cần thiết from sklearn.datasets import load\_iris from sklearn.tree import DecisionTreeClassifier from sklearn.model\_selection import train\_test\_split from sklearn.metrics import accuracy\_score from sklearn import tree import matplotlib.pyplot as plt  # Bước 1: Tải dữ liệu Iris data = load\_iris() X = data.data y = data.target  # Bước 2: Chia dữ liệu thành tập huấn luyện và tập kiểm tra X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  # Bước 3: Tạo và huấn luyện mô hình cây quyết định clf = DecisionTreeClassifier(criterion="gini", max\_depth=3, random\_state=42) clf.fit(X\_train, y\_train)  # Bước 4: Dự đoán và đánh giá mô hình y\_pred = clf.predict(X\_test) accuracy = accuracy\_score(y\_test, y\_pred) print("Độ chính xác của mô hình:", accuracy)  # Bước 5: Trực quan hóa cây quyết định plt.figure(figsize=(12,8)) tree.plot\_tree(clf, feature\_names=data.feature\_names, class\_names=data.target\_names, filled=True) plt.show()** |

**Giải thích:**

1. **Tải dữ liệu:** Sử dụng load\_iris() để tải bộ dữ liệu Iris.
2. **Chia dữ liệu:** Dùng train\_test\_split() để chia dữ liệu thành tập huấn luyện và tập kiểm tra với tỷ lệ 70:30.
3. **Huấn luyện mô hình:** Sử dụng DecisionTreeClassifier với độ sâu tối đa là 3 và tiêu chuẩn phân chia là "gini".
4. **Đánh giá mô hình:** Dự đoán nhãn của tập kiểm tra và tính độ chính xác bằng accuracy\_score.
5. **Trực quan hóa cây:** Sử dụng tree.plot\_tree để vẽ cây quyết định, hiển thị các nút lá với các điều kiện phân chia và các giá trị dự đoán.

**Ưu điểm và Hạn chế của Cây Quyết Định**

* **Ưu điểm:**
  + Dễ hiểu và giải thích.
  + Không yêu cầu chuẩn hóa hoặc xử lý các giá trị rỗng.
  + Có thể xử lý dữ liệu cả dạng phân loại và dạng số.
* **Hạn chế:**
  + Có thể bị overfitting nếu cây quá phức tạp.
  + Độ nhạy cảm với dữ liệu nhiễu.

**Bài tập trên lớp :**

|  |
| --- |
| **import cv2 import numpy as np import os import random from sklearn.model\_selection import train\_test\_split from sklearn.tree import DecisionTreeClassifier from sklearn.metrics import classification\_report, confusion\_matrix from sklearn.datasets import load\_iris  # Đường dẫn đến thư mục chứa ảnh nha khoa dental\_image\_dir = r"C:\Users\hungh\Downloads\Panoramic radiographs with periapical lesions Dataset\Panoramic radiographs with periapical lesions Dataset\Periapical Dataset\Periapical Lesions\Original JPG Images"  features = [] labels = []  # Lấy danh sách tất cả các ảnh nha khoa và chọn ngẫu nhiên 300 ảnh dental\_images = [img for img in os.listdir(dental\_image\_dir) if img.endswith(('.png', '.jpg', '.jpeg'))] selected\_dental\_images = random.sample(dental\_images, 300)   # Đọc và gán nhãn "0" cho ảnh nha khoa for img\_name in selected\_dental\_images:  img\_path = os.path.join(dental\_image\_dir, img\_name)  img = cv2.imread(img\_path, cv2.IMREAD\_GRAYSCALE)   avg\_gray = np.mean(img)  contrast = np.std(img)   feature\_vector = [avg\_gray, contrast]  features.append(feature\_vector)  labels.append(0) # Nhãn cho ảnh nha khoa là 0  # Tải dữ liệu IRIS và chuyển đổi thành ảnh iris = load\_iris() iris\_data = iris.data[:300] # Giới hạn ở 150 mẫu đầu tiên  # Tạo ảnh từ dữ liệu IRIS và gán nhãn "1" for sample in iris\_data:  img = np.reshape(sample, (2, 2)) # Chuyển đổi dữ liệu thành hình ảnh 2x2 (giả định)  avg\_gray = np.mean(img)  contrast = np.std(img)   feature\_vector = [avg\_gray, contrast]  features.append(feature\_vector)  labels.append(1) # Nhãn cho ảnh hoa là 1  # Chuyển đổi danh sách đặc trưng và nhãn sang mảng NumPy để huấn luyện mô hình features = np.array(features) labels = np.array(labels)  # Chia dữ liệu thành tập huấn luyện và kiểm tra X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, labels, test\_size=0.2, random\_state=42)  # CART classifier (Gini Index) cart\_clf = DecisionTreeClassifier(criterion='gini') cart\_clf.fit(X\_train, y\_train) y\_pred\_cart = cart\_clf.predict(X\_test)  # ID3 classifier (Information Gain) id3\_clf = DecisionTreeClassifier(criterion='entropy') id3\_clf.fit(X\_train, y\_train) y\_pred\_id3 = id3\_clf.predict(X\_test)  # In kết quả print("CART (Gini Index) - Confusion Matrix:") print(confusion\_matrix(y\_test, y\_pred\_cart)) print(classification\_report(y\_test, y\_pred\_cart))  print("\nID3 (Information Gain) - Confusion Matrix:") print(confusion\_matrix(y\_test, y\_pred\_id3)) print(classification\_report(y\_test, y\_pred\_id3))** |