**Bài tập 31/10/2024**

**1. Cơ sở lý thuyết**

**1.1 Thuật toán CART**

Thuật toán CART (*Classification and Regression Trees*) là một trong những thuật toán học máy phổ biến nhất để xây dựng cây quyết định. Thuật toán này được sử dụng để tạo ra cả cây phân loại và cây hồi quy nhằm dự đoán kết quả đầu ra dựa trên các thuộc tính đầu vào.

* **Cấu trúc Cơ Bản của CART**

Một cây quyết định bao gồm các nút (nodes) và nhánh (branches):

* Nút gốc (root node): Nút đầu tiên trong cây, chứa toàn bộ dữ liệu ban đầu và là điểm xuất phát cho việc phân chia.
* Nút trong (internal nodes): Là các điểm mà dữ liệu được chia nhỏ dựa trên giá trị của một thuộc tính cụ thể.
* Nút lá (leaf nodes): Là các nút cuối cùng của cây, chứa các nhãn phân loại (cho cây phân loại) hoặc giá trị dự đoán (cho cây hồi quy).

Khi một cây được xây dựng xong, nó có thể được sử dụng để dự đoán giá trị đầu ra cho các dữ liệu mới bằng cách đi từ nút gốc qua các nhánh đến nút lá.

* **Phân Loại và Hồi Quy**
* Phân loại (Classification): Khi đầu ra là một nhãn hoặc một lớp (ví dụ: dự đoán một email là spam hoặc không).
* Hồi quy (Regression): Khi đầu ra là một giá trị liên tục (ví dụ: dự đoán giá nhà).
* **Quy trình của CART**

**CART sử dụng một quá trình gọi là đệ quy phân chia nhị phân. Dưới đây là các bước chi tiết:**

**Chọn điểm chia tốt nhất**: CART chia tập dữ liệu tại mỗi nút dựa trên một thuộc tính để giảm thiểu sai số dự đoán hoặc tối ưu hóa mức độ phân chia. Tiêu chí phổ biến để tìm điểm chia tốt nhất là:

* Gini Impurity cho cây phân loại.
* Mean Squared Error (MSE) cho cây hồi quy.

**Chia dữ liệu thành hai nhóm**: CART luôn chia dữ liệu thành hai nhánh con. Tại mỗi nút, thuật toán chọn thuộc tính và điểm chia (threshold) sao cho việc chia nhỏ này là tốt nhất theo tiêu chí đã chọn.

**Đệ quy tiếp tục chia nhỏ dữ liệu**: Thuật toán tiếp tục chia dữ liệu thành hai nhánh con cho đến khi đạt điều kiện dừng, thường là:

* Độ sâu của cây đạt giới hạn tối đa.
* Số lượng mẫu trong một nhánh con là nhỏ hơn một ngưỡng cho trước.
* Tất cả dữ liệu trong một nhánh con thuộc về cùng một lớp (cho phân loại) hoặc gần như có cùng giá trị (cho hồi quy).
* Cắt tỉa cây (Pruning): Sau khi cây hoàn thành, CART thường thực hiện bước cắt tỉa (pruning) để giảm độ phức tạp của cây và tránh hiện tượng quá khớp (overfitting). Điều này có thể được thực hiện bằng cách loại bỏ bớt một số nhánh có ít giá trị trong việc cải thiện độ chính xác dự đoán.
* **Các Khái Niệm Quan Trọng trong CART**
* Gini Impurity (Độ bất thuần Gini): Được sử dụng trong cây phân loại để đo lường mức độ hỗn hợp của các lớp trong các nhánh con sau mỗi lần chia. Giá trị Gini càng thấp, các nhánh con càng thuần nhất, nghĩa là chúng chứa ít lớp khác nhau hơn.
* Mean Squared Error (MSE): Được sử dụng trong cây hồi quy, MSE đo lường sai số trung bình bình phương giữa giá trị dự đoán và giá trị thực tế trong nhánh con.
* Information Gain: CART không sử dụng Information Gain như một số thuật toán khác (như ID3 hoặc C4.5). CART chỉ dùng Gini Impurity cho cây phân loại và MSE cho cây hồi quy.
* **Ưu Điểm và Nhược Điểm của CART**

**Ưu điểm:**

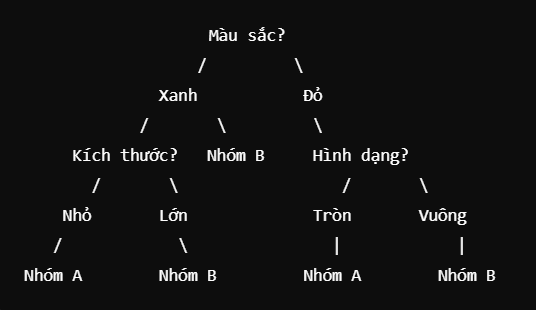
* Dễ hiểu và trực quan.
* Không yêu cầu chuẩn hóa dữ liệu đầu vào.
* Hỗ trợ cả phân loại và hồi quy.

**Nhược điểm:**

* Dễ bị hiện tượng quá khớp (overfitting), đặc biệt là khi cây quá sâu.
* Nhạy cảm với dữ liệu nhiễu.
* Có xu hướng tạo ra các cây không cân đối nếu dữ liệu không được phân bố đồng đều.

**2. Ví dụ code CART trong Python**

|  |
| --- |
| **Cách xây dựng cây quyết định CART để phân loại với Scikit-Learn** |
| from sklearn.tree import DecisionTreeClassifier  from sklearn.datasets import load\_iris  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import accuracy\_score  ***# Bước 1: Tải dữ liệu***  data = load\_iris()  X = data.data # Đặc trưng đầu vào  y = data.target # Nhãn đầu ra  ***# Bước 2: Chia tập 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 mô hình CART***  clf = DecisionTreeClassifier(criterion='gini', random\_state=42)  ***# Bước 4: Huấn luyện mô hình***  clf.fit(X\_train, y\_train)  ***# Bước 5: Dự đoán và đánh giá***  y\_pred = clf.predict(X\_test)  accuracy = accuracy\_score(y\_test, y\_pred)  print("Độ chính xác:", accuracy) |

****

**Gốc của cây** là thuộc tính "Màu sắc". Ở đây, nếu một mục có màu "Xanh", chúng ta tiếp tục đi xuống nhánh bên trái.

* Nếu **Màu sắc là Xanh**:

Cây kiểm tra thuộc tính **Kích thước**:

- Nếu **Kích thước là Nhỏ**, nó được phân vào **Nhóm A**.

- Nếu **Kích thước là Lớn**, nó được phân vào **Nhóm B**.

Nếu **Màu sắc là Đỏ**:

* Dữ liệu sẽ đi theo nhánh bên phải đến **Hình dạng**:

- Nếu **Hình dạng là Tròn**, nó được phân vào **Nhóm A**.

- Nếu **Hình dạng là Vuông**, nó được phân vào **Nhóm B**.

**2. Thuật toán ID3**

Thuật toán ID3 (Iterative Dichotomiser 3) cũng là một thuật toán học máy dùng cho các bài toán phân loại. ID3 xây dựng cây quyết định bằng cách dùng Entropy và Information Gain để chọn thuộc tính tốt nhất tại mỗi nút, giúp tối ưu hóa việc phân chia dữ liệu và làm cho cây ngắn gọn, hiệu quả.

**Cách hoạt động của ID3:**

* **Entropy và Information Gain:**
* Entropy đo lường độ hỗn loạn hay sự không chắc chắn của dữ liệu. Khi tất cả dữ liệu thuộc về cùng một lớp, entropy sẽ là 0; khi dữ liệu được chia đều giữa các lớp, entropy đạt giá trị tối đa.
* Information Gain đo lường lượng thông tin có được sau khi chia dữ liệu theo một thuộc tính. Thuộc tính nào có Information Gain cao nhất sẽ được chọn để làm nút chia.

**Chọn thuộc tính tốt nhất:**

* Với mỗi thuộc tính, ID3 tính toán Information Gain bằng cách so sánh entropy trước và sau khi chia dữ liệu.
* Thuộc tính có Information Gain cao nhất được chọn làm nút để chia dữ liệu tại bước đó.

**Dựng cây:**

ID3 lặp lại quá trình trên cho mỗi nhánh con, chia nhỏ dữ liệu cho đến khi mỗi nhánh chỉ còn một lớp (tức là entropy bằng 0) hoặc khi không còn thuộc tính nào để chia.

**Ví dụ: Quyết định có nên mang ô hay không**

Ví dụ ta có một bộ dữ liệu đơn giản để giúp quyết định xem có nên mang ô khi ra ngoài không, dựa trên các yếu tố thời tiết.

Các thuộc tính (đặc trưng) gồm:

* Trời mưa: Có, Không
* Gió lớn: Có, Không
* Trời nhiều mây: Có, Không

Và kết quả phân loại (nhãn) là Mang ô với hai giá trị: Có và Không.

**Bộ dữ liệu mẫu**

****

**Tính toán Entropy và Information Gain**

1. **Entropy tổng của dữ liệu:**

**-** Tính entropy cho cả bộ dữ liệu với nhãn "Mang ô" là Có và Không.

1. **Chọn thuộc tính tốt nhất:**

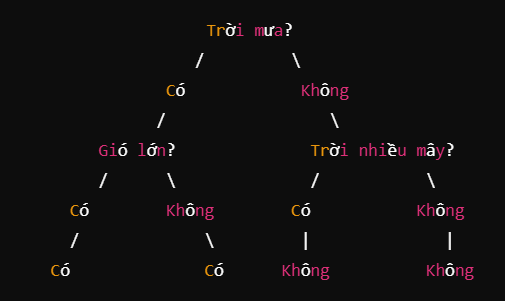
- Tính toán Information Gain cho từng thuộc tính:Trời mưaGió lớnTrời nhiều mây

- Thuộc tính nào có Information Gain cao nhất sẽ được chọn làm thuộc tính gốc (nút gốc) cho cây

**Ví dụ code ID3 trong Python**

|  |
| --- |
| **Một tập dữ liệu về thời tiết nên quyết định có chơi thể thao hay không** |
| import pandas as pd  import numpy as np  # Tính entropy cho một tập dữ liệu  def entropy(data):  labels = data.iloc[:, -1]  label\_counts = labels.value\_counts()  entropy = -np.sum((label\_counts / len(labels)) \* np.log2(label\_counts / len(labels)))  return entropy  # Hàm chọn thuộc tính tốt nhất để chia dữ liệu  def best\_split(data):  base\_entropy = entropy(data)  max\_info\_gain = -1  best\_feature = None    for feature in data.columns[:-1]:  values = data[feature].unique()  feature\_entropy = 0    for value in values:  subset = data[data[feature] == value]  feature\_entropy += len(subset) / len(data) \* entropy(subset)    info\_gain = base\_entropy - feature\_entropy  if info\_gain > max\_info\_gain:  max\_info\_gain = info\_gain  best\_feature = feature    return best\_feature  # Hàm xây dựng cây quyết định ID3  def id3(data, depth=0, max\_depth=5):  labels = data.iloc[:, -1]    # Trường hợp toàn bộ nhãn giống nhau hoặc đạt độ sâu tối đa  if len(labels.unique()) == 1 or depth == max\_depth:  return labels.mode()[0]    # Chọn thuộc tính tốt nhất và chia dữ liệu  best\_feature = best\_split(data)  if best\_feature is None:  return labels.mode()[0]    tree = {best\_feature: {}}  for value in data[best\_feature].unique():  subset = data[data[best\_feature] == value]  subtree = id3(subset.drop(columns=[best\_feature]), depth + 1, max\_depth)  tree[best\_feature][value] = subtree    return tree  # Tập dữ liệu nhỏ về thời tiết  data = pd.DataFrame({  'Outlook': ['Sunny', 'Sunny', 'Overcast', 'Rain', 'Rain', 'Rain', 'Overcast', 'Sunny', 'Sunny', 'Rain', 'Sunny', 'Overcast', 'Overcast', 'Rain'],  'Temperature': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild', 'Cool', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild'],  'Humidity': ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'High'],  'Wind': ['Weak', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak', 'Strong'],  'Play': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']  })  # Xây dựng cây quyết định  tree = id3(data)  print(tree) |

**Cấu trúc cây ID3**

****

Cây bắt đầu với thuộc tính **Trời mưa**:

* Nếu **Trời mưa là Có**, tiếp theo kiểm tra thuộc tính **Gió lớn**:
* Nếu **Gió lớn là Có**: Mang ô.Nếu **Gió lớn là Không**: Mang ô.
* Nếu **Trời mưa là Không**, kiểm tra thuộc tính **Trời nhiều mây**:
* Nếu **Trời nhiều mây là Có**: Không mang ô.
* Nếu **Trời nhiều mây là Không**: Không mang ô