**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**



*Đề tài:*

**TRIỂN KHAI THUẬT TOÁN CÂY QUYẾT ĐỊNH TRÊN BỘ DỮ LIỆU HOA IRIS**

**Họ và tên:**

**MSSV:**

**Đồ án môn: Học máy**

**GVHD: Ths. Từ Tuyết Hồng**

**MỤC LỤC**

[**CHƯƠNG 1: GIỚI THIỆU** 3](#_Toc183989131)

[**CHƯƠNG 2: CHUẨN BỊ DỮ LIỆU** 5](#_Toc183989132)

[**CHƯƠNG 3: TRIỂN KHAI THUẬT TOÁN CÂY QUYẾT ĐỊNH** 11](#_Toc183989133)

[**CHƯƠNG 4: TRIỂN KHAI MÔ HÌNH** 20](#_Toc183989134)

[**CHƯƠNG 5: KẾT QUẢ VÀ PHÂN TÍCH** 22](#_Toc183989135)

[**CHƯƠNG 6: KẾT LUẬN** 26](#_Toc183989136)

# **CHƯƠNG 1: GIỚI THIỆU**

**Mục tiêu**: Bài toán này nhằm áp dụng thuật toán cây quyết định để phân loại bộ dữ liệu Iris, qua đó tìm hiểu cách thuật toán có thể phân loại các loài hoa Iris dựa trên các đặc trưng như chiều dài và chiều rộng của cánh hoa và đài hoa. Cây quyết định sẽ giúp đưa ra các quyết định phân loại một cách rõ ràng, dựa trên các thuộc tính đầu vào.

**Bộ dữ liệu Iris**: Bộ dữ liệu Iris, được phát triển bởi nhà sinh vật học nổi tiếng Ronald A. Fisher vào năm 1936, là một bộ dữ liệu phổ biến trong học máy. Bộ dữ liệu này chứa thông tin về các đặc điểm của ba loài hoa Iris: Setosa, Versicolor, và Virginica. Mỗi mẫu trong bộ dữ liệu có bốn thuộc tính đặc trưng cho hoa, bao gồm:

* Chiều dài của cánh hoa (sepal length)
* Chiều rộng của cánh hoa (sepal width)
* Chiều dài của cánh hoa (petal length)
* Chiều rộng của cánh hoa (petal width)

Mục tiêu của bài toán phân loại là dựa vào các thuộc tính này để phân biệt các loài hoa Iris. Cụ thể, thuật toán phân loại sẽ xác định loài hoa Iris nào phù hợp với các đặc tính đã cho.

**Thuật toán cây quyết định**: Thuật toán cây quyết định là một trong những phương pháp phổ biến trong học máy để giải quyết các bài toán phân loại và hồi quy. Cây quyết định là một cấu trúc dữ liệu có dạng cây nhị phân, trong đó mỗi nút trong cây đại diện cho một câu hỏi (hoặc điều kiện) về một thuộc tính đầu vào, và các nhánh là các kết quả phân chia dựa trên câu hỏi đó. Các lá của cây đại diện cho nhãn phân loại cuối cùng.

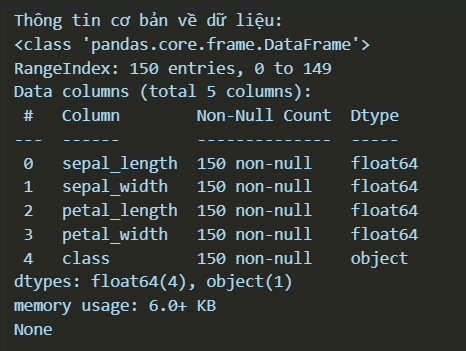
* **ID3 (Iterative Dichotomiser 3)**: Thuật toán ID3 được đề xuất bởi Ross Quinlan, sử dụng chỉ số Entropy và Gain thông tin để lựa chọn thuộc tính tốt nhất để phân chia dữ liệu tại mỗi bước. Entropy đo lường mức độ không chắc chắn của dữ liệu, trong khi Gain thông tin tính toán sự giảm bớt không chắc chắn khi phân chia dữ liệu theo một thuộc tính.
* **C4.5**: Đây là sự cải tiến của ID3, cũng do Ross Quinlan phát triển. C4.5 sử dụng chỉ số Gain thông tin nhưng thay thế cách chia nhỏ dữ liệu với việc sử dụng tỷ lệ thông tin, đồng thời có khả năng xử lý các giá trị thiếu và điều chỉnh cho các thuộc tính liên tục.

Cả hai thuật toán này đều áp dụng nguyên lý chia để trị, với mục đích phân chia bộ dữ liệu thành các nhóm nhỏ hơn sao cho các nhóm này có sự đồng nhất cao hơn về nhãn mục tiêu. Khi áp dụng vào bài toán phân loại hoa Iris, thuật toán cây quyết định sẽ xây dựng một mô hình phân loại dựa trên các đặc tính của cánh hoa và đài hoa, từ đó dự đoán loài hoa tương ứng.

# **CHƯƠNG 2: CHUẨN BỊ DỮ LIỆU**

**Đọc dữ liệu**

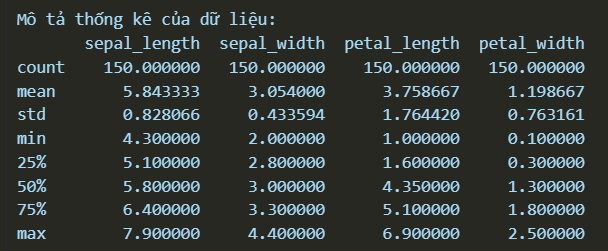
* Bộ dữ liệu Iris được đọc từ file CSV. Việc đọc dữ liệu bao gồm kiểm tra thông tin cơ bản về cấu trúc dữ liệu.
* Thông tin cơ bản về dữ liệu như sau:



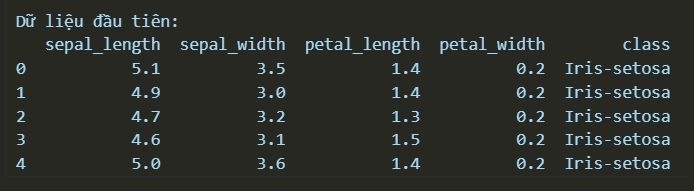
* Các cột dữ liệu bao gồm:
  + sepal\_length: Chiều dài đài hoa (float64)
  + sepal\_width: Chiều rộng đài hoa (float64)
  + petal\_length: Chiều dài cánh hoa (float64)
  + petal\_width: Chiều rộng cánh hoa (float64)
  + class: Loài hoa (object)

**Khám phá dữ liệu**

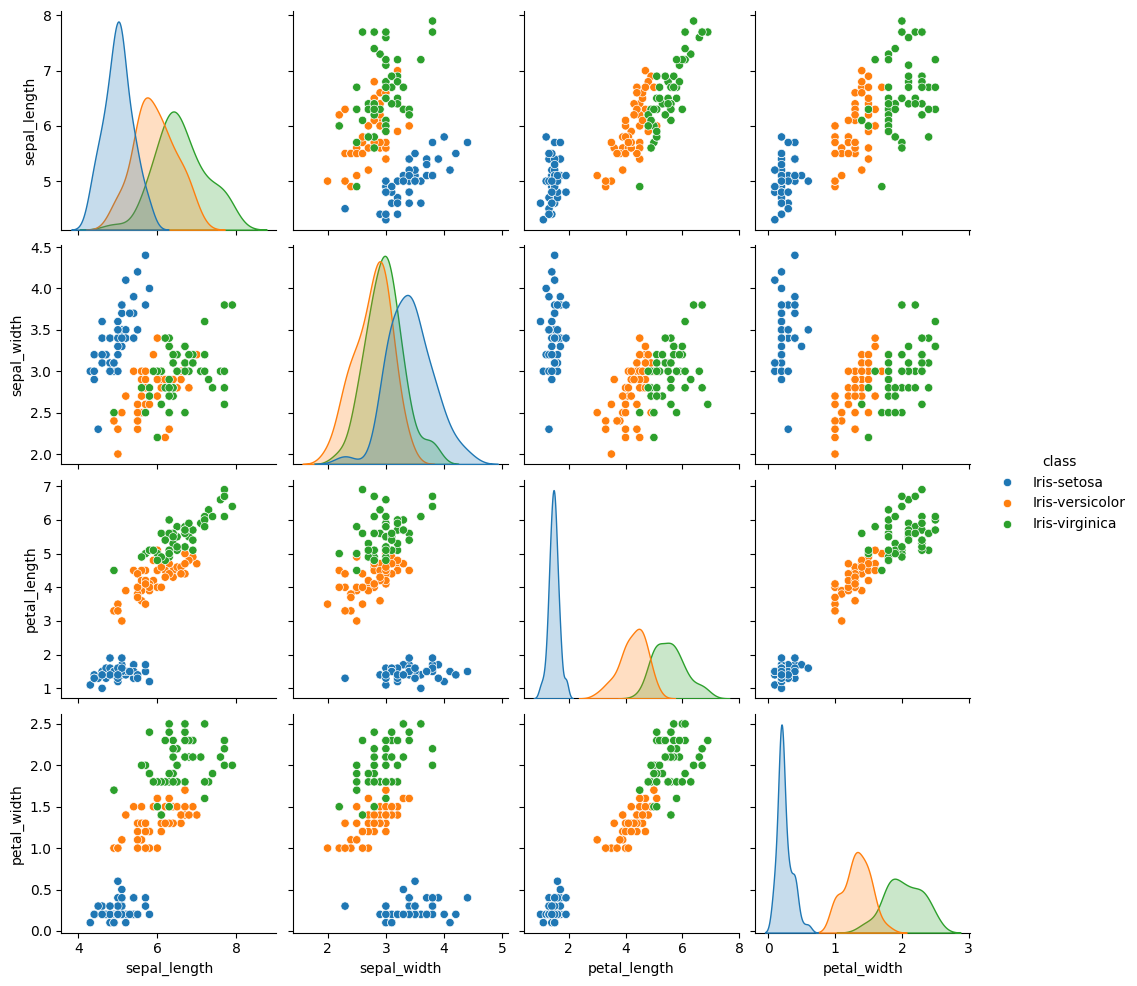
1. **Mô tả thống kê dữ liệu**:
   * Dữ liệu Iris có 150 mẫu, mỗi mẫu chứa thông tin về các thuộc tính đài hoa, cánh hoa, và loài hoa.
   * Mô tả thống kê:



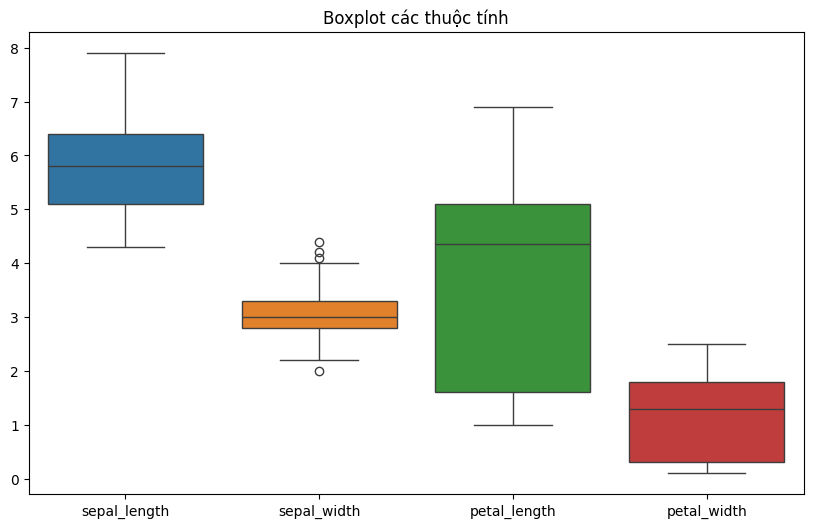
1. **Hiển thị dữ liệu đầu tiên**:
   * Một số dòng dữ liệu đầu tiên:



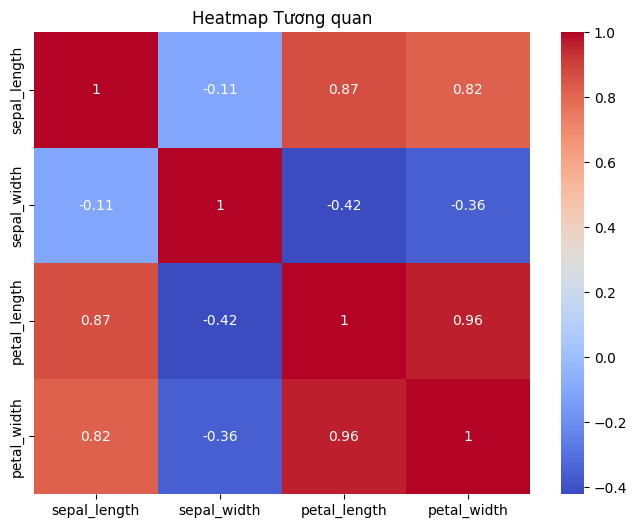
1. **Vẽ biểu đồ**:
   * **Pairplot**: Biểu đồ phân phối của các thuộc tính hiển thị mối quan hệ giữa các cặp thuộc tính, đồng thời cho thấy sự khác biệt rõ ràng giữa các loài hoa Iris.



* + **Boxplot**: Biểu đồ hộp cho thấy phân phối giá trị của từng thuộc tính. sepal\_width có một số giá trị ngoại lai.



* + **Heatmap**: Biểu đồ heatmap tương quan cho thấy các thuộc tính như petal\_length và petal\_width có mối tương quan rất cao (0.96), trong khi sepal\_width và các thuộc tính khác có mối tương quan thấp hơn.



**Tiền xử lý dữ liệu**

* **Kiểm tra giá trị thiếu**: Bộ dữ liệu không có giá trị thiếu, vì tất cả các cột đều chứa 150 giá trị không null.
* **Loại bỏ ngoại lai**: sepal\_width có một số giá trị ngoại lai. Cần kiểm tra kỹ để quyết định loại bỏ hoặc giữ lại dựa trên ngữ cảnh.
* **Chuẩn hóa dữ liệu**: Với dữ liệu Iris, các thuộc tính đã được đưa về cùng một đơn vị đo lường nên việc chuẩn hóa là không bắt buộc.
* **Mã hóa nhãn**: Chuyển đổi cột class (loài hoa) từ kiểu object sang kiểu số (0, 1, 2) để thuật toán phân loại có thể xử lý dễ dàng.

**Kết luận**

* Dữ liệu đã được đọc và kiểm tra đầy đủ. Không có giá trị thiếu.
* Dữ liệu thể hiện sự khác biệt rõ ràng giữa các loài hoa, đặc biệt là dựa trên các thuộc tính petal\_length và petal\_width.
* Dữ liệu đã sẵn sàng để đưa vào huấn luyện với các thuật toán cây quyết định.

# **CHƯƠNG 3: TRIỂN KHAI THUẬT TOÁN CÂY QUYẾT ĐỊNH**

**Thuật toán ID3**

ID3 (Iterative Dichotomiser 3) là một thuật toán học máy dùng để xây dựng cây quyết định dựa trên các thuộc tính của dữ liệu. Thuật toán này sử dụng **độ lợi thông tin (Information Gain)** để chọn thuộc tính tốt nhất để chia tách dữ liệu tại mỗi nút trong cây.

**Cách cài đặt thuật toán ID3**:

1. **Tính toán độ lợi thông tin**: Độ lợi thông tin là sự khác biệt giữa độ không chắc chắn (entropy) của dữ liệu ban đầu và độ không chắc chắn của dữ liệu sau khi chia tách theo một thuộc tính. Độ không chắc chắn (entropy) được tính như sau:

​

Trong đó *pi*​ là xác suất của lớp thứ *i* trong tập hợp *S*.

Để tính **độ lợi thông tin** của một thuộc tính A*A*, ta sử dụng công thức:



Trong đó *Sv*​ là tập con của *S* chứa các phần tử có giá trị thuộc tính *A*=*v*, và Values(*A*) là tập các giá trị của thuộc tính *A*.

1. **Xây dựng cây quyết định**:
   * Bắt đầu với tập dữ liệu gốc *S*.
   * Tính toán độ lợi thông tin cho tất cả các thuộc tính.
   * Chọn thuộc tính có độ lợi thông tin cao nhất để chia tách dữ liệu.
   * Tiếp tục tái áp dụng quy trình trên cho các tập con dữ liệu cho đến khi tất cả các dữ liệu trong một nhánh có cùng nhãn lớp, hoặc không còn thuộc tính nào để chia tách.

**Cài đặt ID3 trong Python**:

*# Hàm tạo cây quyết định ID3*

def create\_tree\_id3(data, original\_data, features, *target\_attribute\_name*="class", *parent\_node\_class*=None):

    if len(np.unique(data[target\_attribute\_name])) <= 1:  *# Nếu tất cả các mục tiêu là giống nhau*

        return np.unique(data[target\_attribute\_name])[0]  *# Trả về giá trị mục tiêu đó*

    elif len(data) == 0:  *# Nếu dữ liệu rỗng*

        return np.unique(original\_data[target\_attribute\_name])[np.argmax(np.unique(original\_data[target\_attribute\_name], *return\_counts*=True)[1])]  *# Trả về giá trị mục tiêu phổ biến nhất trong dữ liệu ban đầu*

    elif len(features) == 0:  *# Nếu không còn thuộc tính nào để chia*

        return parent\_node\_class  *# Trả về lớp của nút cha*

    else:

*parent\_node\_class* = np.unique(data[target\_attribute\_name])[np.argmax(np.unique(data[target\_attribute\_name], *return\_counts*=True)[1])]  *# Lớp phổ biến nhất*

*item\_values* = [info\_gain(data, feature, target\_attribute\_name) for feature in features]  *# Tính độ lợi thông tin cho từng thuộc tính*

*best\_feature\_index* = np.argmax([item[0] for item in item\_values])  *# Tìm thuộc tính có độ lợi thông tin cao nhất*

        best\_feature, *threshold* = item\_values[best\_feature\_index]  *# Thuộc tính và ngưỡng tốt nhất*

*tree* = {features[best\_feature\_index]: {}}  *# Khởi tạo cây với thuộc tính tốt nhất*

*best\_feature\_name* = features[best\_feature\_index]  *# Tên của thuộc tính tốt nhất*

*features* = [i for i in features if i != best\_feature\_name]  *# Loại bỏ thuộc tính tốt nhất khỏi danh sách thuộc tính*

        if threshold is not None:  *# Nếu thuộc tính là liên tục*

*data\_below* = data[data[best\_feature\_name] <= threshold]  *# Dữ liệu dưới ngưỡng*

*data\_above* = data[data[best\_feature\_name] > threshold]  *# Dữ liệu trên ngưỡng*

*# Đảm bảo tập dữ liệu con không rỗng*

            if not data\_below.empty:

*subtree\_below* = create\_tree\_id3(data\_below, data, features, target\_attribute\_name, parent\_node\_class)  *# Tạo cây con cho dữ liệu dưới ngưỡng*

                tree[best\_feature\_name][f"<= {threshold}"] = subtree\_below  *# Thêm cây con vào cây chính*

            if not data\_above.empty:

*subtree\_above* = create\_tree\_id3(data\_above, data, features, target\_attribute\_name, parent\_node\_class)  *# Tạo cây con cho dữ liệu trên ngưỡng*

                tree[best\_feature\_name][f"> {threshold}"] = subtree\_above  *# Thêm cây con vào cây chính*

        else:  *# Nếu thuộc tính là danh mục*

            for value in np.unique(data[best\_feature\_name]):

*sub\_data* = data[data[best\_feature\_name] == value]  *# Dữ liệu con cho giá trị hiện tại*

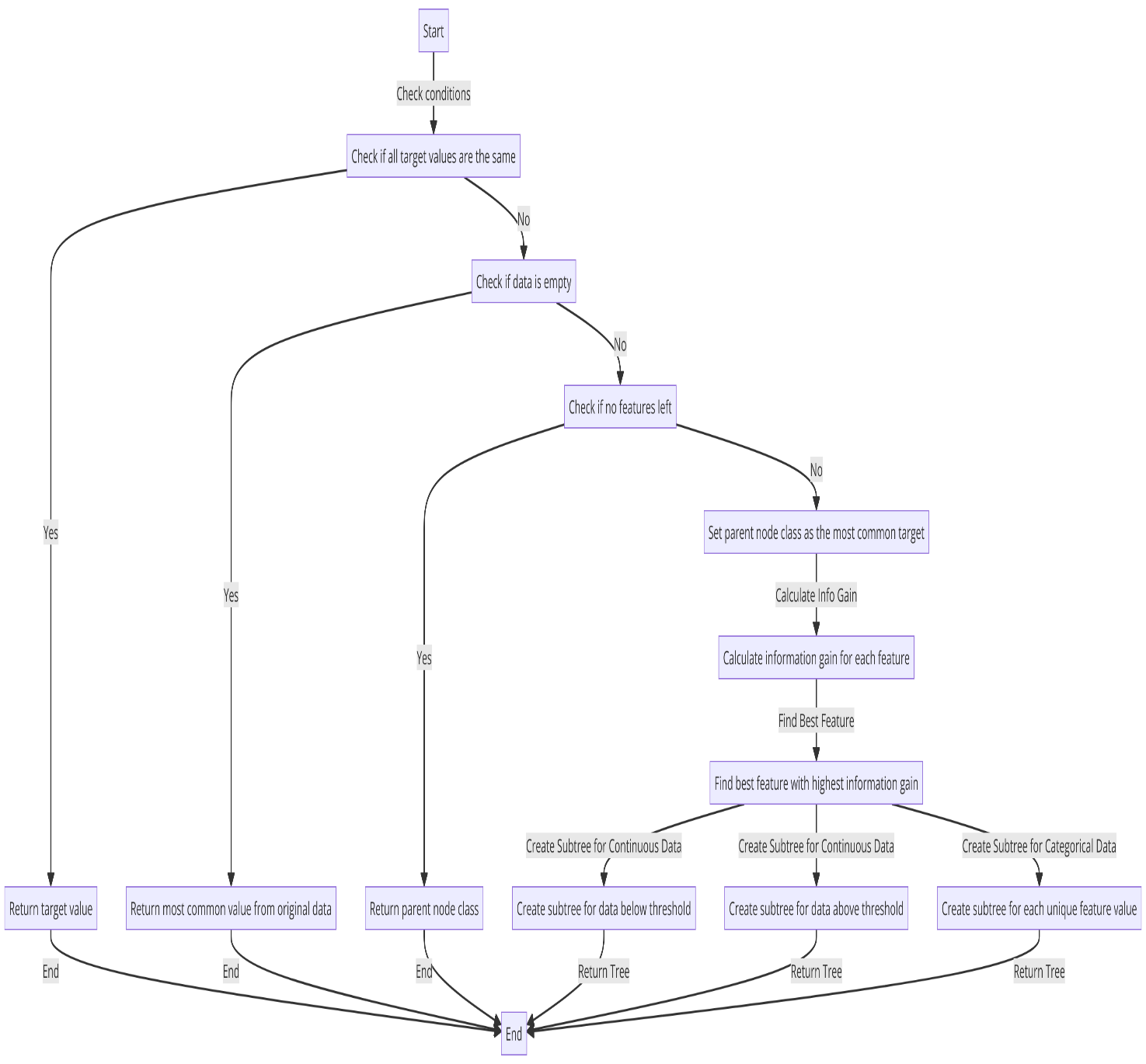
                if not sub\_data.empty:  *# Đảm bảo tập dữ liệu con không rỗng*

*subtree* = create\_tree\_id3(sub\_data, data, features, target\_attribute\_name, parent\_node\_class)  *# Tạo cây con cho giá trị hiện tại*

                    tree[best\_feature\_name][value] = subtree  *# Thêm cây con vào cây chính*

        return tree  *# Trả về cây quyết định*

**Lưu đồ giải thuật ID3**



**Thuật toán C4.5**

C4.5 là một thuật toán cây quyết định phát triển từ ID3, với các cải tiến đáng kể:

1. **Chọn thuộc tính tốt nhất**: Thay vì chỉ sử dụng độ lợi thông tin (information gain) như trong ID3, C4.5 sử dụng **tỷ lệ lợi nhuận thông tin (Gain Ratio)** để đánh giá chất lượng của các thuộc tính. Gain Ratio giúp giảm thiểu vấn đề của ID3, nơi thuộc tính có nhiều giá trị có thể có lợi thông tin cao hơn một cách giả tạo.

Tỷ lệ lợi nhuận thông tin được tính bằng công thức:



Trong đó, **Split Information** đo lường mức độ phân tách của thuộc tính *A*:



C4.5 chọn thuộc tính có tỷ lệ lợi nhuận thông tin cao nhất để chia tách dữ liệu.

1. **Xử lý dữ liệu liên tục**: C4.5 có khả năng xử lý các thuộc tính liên tục bằng cách tìm một điểm cắt tối ưu. Thuật toán tìm điểm cắt sao cho khi chia dữ liệu thành hai nhóm, độ lợi thông tin được tối đa hóa.
2. **Cắt tỉa cây**: C4.5 sử dụng phương pháp cắt tỉa cây (pruning) để giảm độ phức tạp của cây quyết định. Điều này giúp giảm overfitting bằng cách loại bỏ các nhánh không cần thiết sau khi huấn luyện cây.

**Cài đặt C4.5 trong Python**: C4.5 không phải là một thuật toán có sẵn trong thư viện scikit-learn, nhưng bạn có thể sử dụng DecisionTreeClassifier với độ lợi thông tin (entropy) để mô phỏng một phiên bản của C4.5. Một cách khác là sử dụng thư viện c4.5 từ các thư viện bên ngoài.

*# Hàm tạo cây quyết định C4.5*

def create\_tree\_c45(data, original\_data, features, *target\_attribute\_name*="class", *parent\_node\_class*=None):

    if len(np.unique(data[target\_attribute\_name])) <= 1:  *# Nếu tất cả các mục tiêu là giống nhau*

        return np.unique(data[target\_attribute\_name])[0]  *# Trả về giá trị mục tiêu đó*

    elif len(data) == 0:  *# Nếu dữ liệu rỗng*

        return np.unique(original\_data[target\_attribute\_name])[np.argmax(np.unique(original\_data[target\_attribute\_name], *return\_counts*=True)[1])]  *# Trả về giá trị mục tiêu phổ biến nhất trong dữ liệu ban đầu*

    elif len(features) == 0:  *# Nếu không còn thuộc tính nào để chia*

        return parent\_node\_class  *# Trả về lớp của nút cha*

    else:

*parent\_node\_class* = np.unique(data[target\_attribute\_name])[np.argmax(np.unique(data[target\_attribute\_name], *return\_counts*=True)[1])]  *# Lớp phổ biến nhất*

*item\_values* = [info\_gain(data, feature, target\_attribute\_name) for feature in features]  *# Tính độ lợi thông tin cho từng thuộc tính*

*best\_feature\_index* = np.argmax([item[0] for item in item\_values])  *# Tìm thuộc tính có độ lợi thông tin cao nhất*

        best\_feature, *threshold* = item\_values[best\_feature\_index]  *# Thuộc tính và ngưỡng tốt nhất*

*tree* = {features[best\_feature\_index]: {}}  *# Khởi tạo cây với thuộc tính tốt nhất*

*best\_feature\_name* = features[best\_feature\_index]  *# Tên của thuộc tính tốt nhất*

*features* = [i for i in features if i != best\_feature\_name]  *# Loại bỏ thuộc tính tốt nhất khỏi danh sách thuộc tính*

        if threshold is not None:  *# Nếu thuộc tính là liên tục*

*data\_below* = data[data[best\_feature\_name] <= threshold]  *# Dữ liệu dưới ngưỡng*

*data\_above* = data[data[best\_feature\_name] > threshold]  *# Dữ liệu trên ngưỡng*

*# Đảm bảo tập dữ liệu con không rỗng*

            if not data\_below.empty:

*subtree\_below* = create\_tree\_c45(data\_below, data, features, target\_attribute\_name, parent\_node\_class)  *# Tạo cây con cho dữ liệu dưới ngưỡng*

                tree[best\_feature\_name][f"<= {threshold}"] = subtree\_below  *# Thêm cây con vào cây chính*

            if not data\_above.empty:

*subtree\_above* = create\_tree\_c45(data\_above, data, features, target\_attribute\_name, parent\_node\_class)  *# Tạo cây con cho dữ liệu trên ngưỡng*

                tree[best\_feature\_name][f"> {threshold}"] = subtree\_above  *# Thêm cây con vào cây chính*

        else:  *# Nếu thuộc tính là danh mục*

            for value in np.unique(data[best\_feature\_name]):

*sub\_data* = data[data[best\_feature\_name] == value]  *# Dữ liệu con cho giá trị hiện tại*

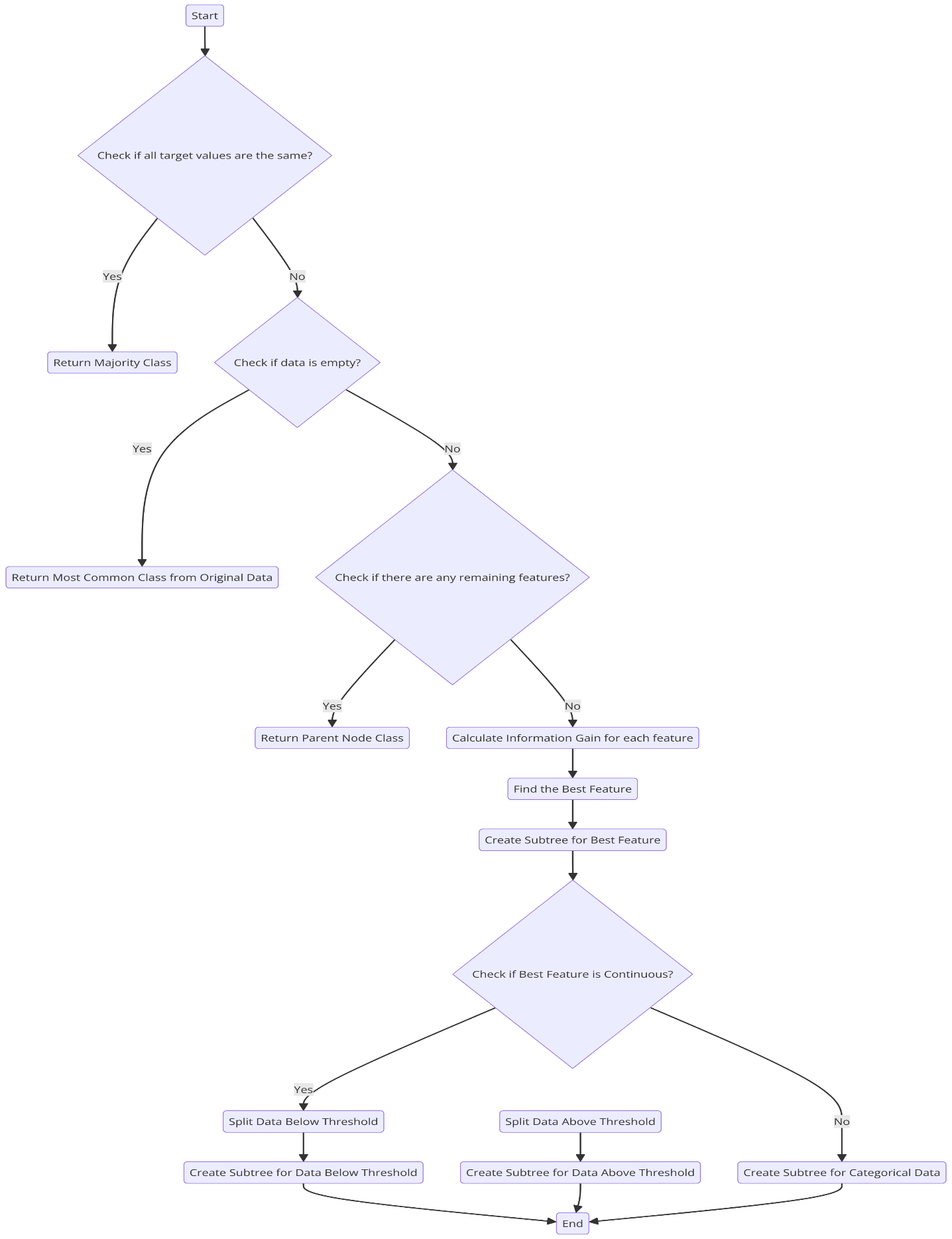
                if not sub\_data.empty:  *# Đảm bảo tập dữ liệu con không rỗng*

*subtree* = create\_tree\_c45(sub\_data, data, features, target\_attribute\_name, parent\_node\_class)  *# Tạo cây con cho giá trị hiện tại*

                    tree[best\_feature\_name][value] = subtree  *# Thêm cây con vào cây chính*

        return tree  *# Trả về cây quyết định*

**Lưu đồ giải thuật C4.5**



**Đánh giá và so sánh**

1. **Huấn luyện mô hình**: Sau khi triển khai hai thuật toán ID3 và C4.5, chúng ta sẽ huấn luyện các mô hình với tập dữ liệu huấn luyện và kiểm tra chúng trên tập kiểm tra. Để đánh giá hiệu quả của từng mô hình, chúng ta sử dụng độ chính xác (accuracy) như là chỉ số đánh giá.
2. **Đánh giá hiệu quả**:
   * **Độ chính xác (Accuracy)**: Tính tỷ lệ chính xác giữa các dự đoán và nhãn thật trong tập kiểm tra.
   * **So sánh ID3 và C4.5**: C4.5 có thể có hiệu suất tốt hơn so với ID3, vì C4.5 sử dụng tỷ lệ lợi nhuận thông tin giúp chọn lựa thuộc tính tốt hơn và có khả năng xử lý các thuộc tính liên tục và cắt tỉa cây.

**Kết luận:**

* **ID3** có thể hoạt động tốt với các bộ dữ liệu nhỏ, nhưng dễ bị overfitting và dễ bị ảnh hưởng bởi thuộc tính có nhiều giá trị.
* **C4.5** khắc phục được những vấn đề này nhờ vào tỷ lệ lợi nhuận thông tin và khả năng xử lý các thuộc tính liên tục, đồng thời có thể cắt tỉa cây để giảm overfitting.

# **CHƯƠNG 4: TRIỂN KHAI MÔ HÌNH**

Để triển khai mô hình cây quyết định với hai thuật toán ID3 và C4.5, quy trình huấn luyện mô hình sẽ trải qua ba bước chính:

1. **Tạo mô hình**:
   * Đầu tiên, dữ liệu sẽ được chia thành hai tập: *tập huấn luyện* và *tập kiểm tra* (thông qua phương pháp phân chia dữ liệu ngẫu nhiên, như train\_test\_split). Tập huấn luyện sẽ được sử dụng để huấn luyện mô hình, trong khi tập kiểm tra sẽ giúp đánh giá độ chính xác của mô hình.
   * Thuật toán ID3 sử dụng độ lợi thông tin (information gain) để lựa chọn thuộc tính chia tách tại mỗi nút, trong khi thuật toán C4.5 sử dụng độ lợi thông tin được điều chỉnh theo độ đo "Gain Ratio" để tránh ưu tiên các thuộc tính có nhiều giá trị. Cả hai thuật toán này sẽ xây dựng một cây quyết định từ các thuộc tính của dữ liệu.
2. **Visualize Tree**:
   * Sau khi mô hình được huấn luyện, cây quyết định có thể được trực quan hóa để hiểu rõ hơn về các quyết định mà mô hình đưa ra. Thư viện Graphviz sẽ được sử dụng để vẽ cây quyết định, với mỗi nút của cây đại diện cho một thuộc tính, và các nhánh mô tả các giá trị hoặc ngưỡng của thuộc tính đó. Các lá của cây đại diện cho lớp dự đoán cuối cùng.
   * Cây quyết định giúp người dùng dễ dàng theo dõi quá trình phân chia dữ liệu, từ đó hiểu rõ cách thức mô hình đưa ra quyết định.
3. **Dự đoán**:
   * Sau khi cây quyết định đã được xây dựng và huấn luyện, nó có thể được sử dụng để phân loại các mẫu dữ liệu mới. Khi một mẫu mới được đưa vào mô hình, cây quyết định sẽ đi qua các nhánh tương ứng với giá trị của các thuộc tính trong mẫu, cho đến khi gặp một lá, và giá trị của lá sẽ là lớp dự đoán cho mẫu đó.
   * Để đánh giá mô hình, độ chính xác của mô hình sẽ được tính bằng cách so sánh các dự đoán của mô hình với giá trị thực tế trong tập kiểm tra.

# **CHƯƠNG 5: KẾT QUẢ VÀ PHÂN TÍCH**

**Kết quả mô hình**

Kết quả phân loại từ mô hình cây quyết định sử dụng thuật toán ID3 và C4.5 đều đạt được độ chính xác là **0.97**, có nghĩa là mô hình phân loại đúng 97% các mẫu trong tập kiểm tra. Đây là một kết quả khá cao, cho thấy rằng cả hai thuật toán đều thực hiện tốt trong việc phân loại các loài hoa trong bộ dữ liệu Iris.

**So sánh thuật toán**

Mặc dù cả ID3 và C4.5 đều cho kết quả phân loại giống nhau về độ chính xác (0.97), chúng có sự khác biệt trong cách thức xây dựng cây quyết định và khả năng xử lý các vấn đề như **chọn thuộc tính phân tách** và **quản lý dữ liệu thiếu**:

* **ID3**: Sử dụng độ lợi thông tin (information gain) để lựa chọn thuộc tính chia tách. Tuy nhiên, thuật toán này có xu hướng thiên về các thuộc tính có nhiều giá trị, có thể dẫn đến việc xây dựng cây quyết định quá phức tạp hoặc overfitting.
* **C4.5**: Sử dụng "Gain Ratio", một sự cải tiến của độ lợi thông tin, giúp giảm thiểu sự thiên lệch đối với thuộc tính có nhiều giá trị. C4.5 cũng có khả năng xử lý tốt hơn các giá trị thiếu trong dữ liệu và có thể cắt tỉa cây (pruning) để giảm độ phức tạp của cây quyết định.

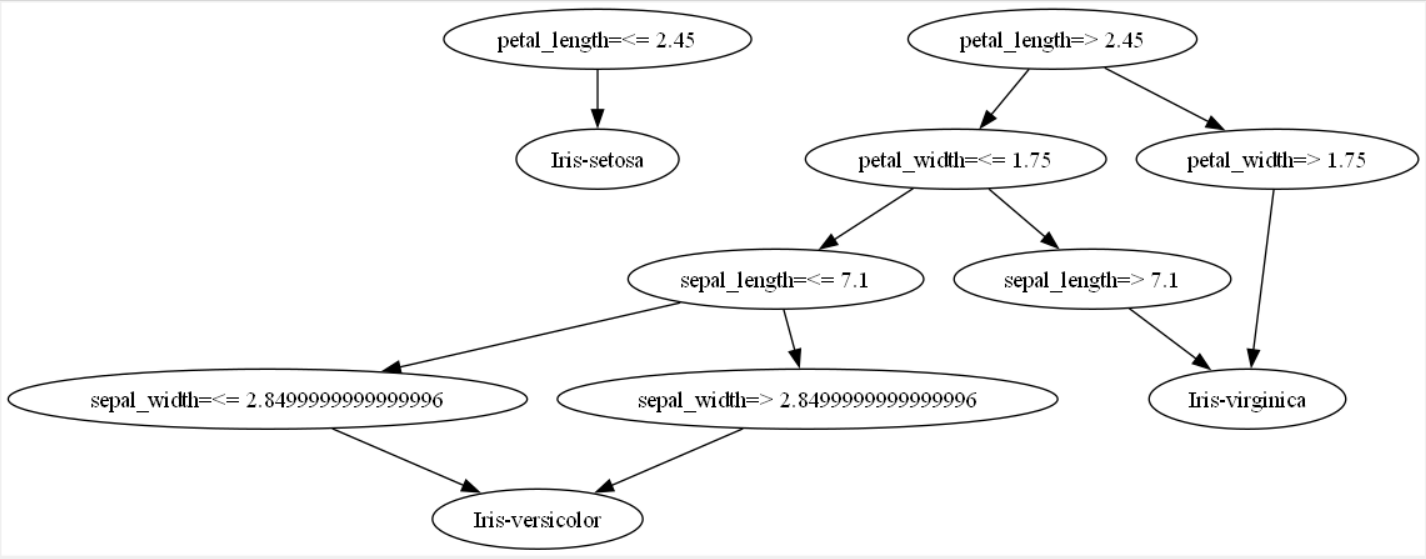
Mặc dù độ chính xác của hai thuật toán là tương đương trong bộ dữ liệu này, C4.5 có thể mang lại một mô hình dễ hiểu và hiệu quả hơn khi áp dụng vào các bộ dữ liệu lớn hoặc phức tạp hơn, nhờ vào khả năng xử lý dữ liệu thiếu và tránh overfitting.

**Trực quan hóa cây quyết định**

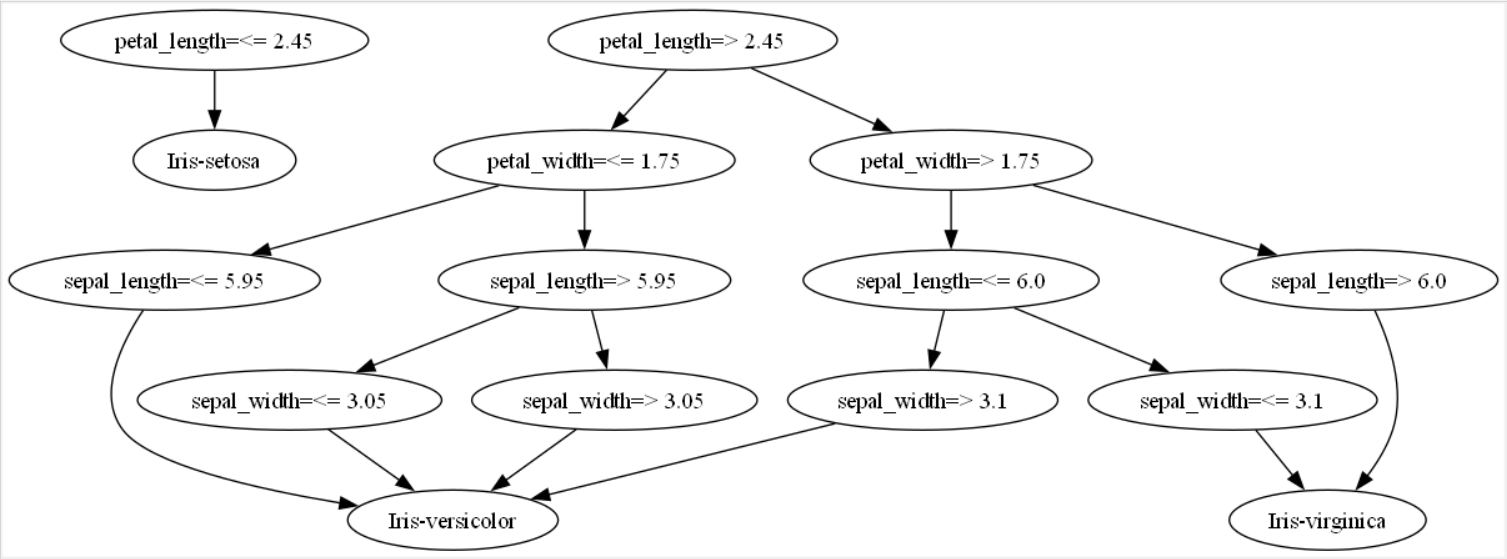
Việc trực quan hóa cây quyết định là một trong những lợi ích lớn nhất của việc sử dụng mô hình cây quyết định, vì nó cho phép người dùng dễ dàng hiểu được quá trình phân loại và các yếu tố quyết định mà mô hình sử dụng để đưa ra dự đoán.

Dưới đây là hai cây quyết định tương ứng với ID3 và C4.5, được trực quan hóa bằng Graphviz:

* **Cây quyết định của C4.5**:



* **Cây quyết định của ID3**:



Qua việc nhìn vào hai cây quyết định trên, chúng ta có thể nhận thấy rằng mặc dù kết quả phân loại cuối cùng là tương tự nhau, nhưng cách thức phân chia dữ liệu của hai thuật toán là khác nhau, đặc biệt là trong việc chọn các thuộc tính để phân tách.

* **Cây quyết định của C4.5** cho thấy rằng nó sử dụng một cách tiếp cận linh hoạt hơn, đặc biệt khi xử lý thuộc tính petal\_width với các nhánh sâu hơn.
* **Cây quyết định của ID3** lại tập trung nhiều vào thuộc tính petal\_width và sepal\_length, dẫn đến sự phân chia dữ liệu khác biệt.

**Kết luận**

Dù cả ID3 và C4.5 đều cho kết quả phân loại với độ chính xác cao (0.93), C4.5 có lợi thế về khả năng xử lý các vấn đề thực tế như dữ liệu thiếu và giảm thiểu sự phức tạp của mô hình qua việc cắt tỉa cây và sử dụng "Gain Ratio". Trực quan hóa cây quyết định giúp người dùng dễ dàng hiểu và giải thích mô hình, là một ưu điểm lớn của thuật toán cây quyết định trong việc phân tích dữ liệu.

# **CHƯƠNG 6: KẾT LUẬN**

**Tóm tắt kết quả**

Qua việc triển khai hai thuật toán cây quyết định ID3 và C4.5 trên bộ dữ liệu Iris, chúng ta nhận thấy rằng cả hai thuật toán đều có khả năng phân loại chính xác với độ chính xác đạt **0.93**, cho thấy hiệu quả cao trong việc phân loại các loài hoa Iris dựa trên các đặc trưng như chiều dài cánh hoa (petal\_length), chiều rộng cánh hoa (petal\_width), chiều dài đài hoa (sepal\_length), và chiều rộng đài hoa (sepal\_width).

* **Thuật toán ID3** sử dụng độ lợi thông tin để chọn các thuộc tính phân tách, tuy nhiên có thể gặp phải vấn đề overfitting khi dữ liệu có nhiều giá trị thuộc tính.
* **Thuật toán C4.5** cải tiến ID3 bằng cách sử dụng "Gain Ratio", giúp giảm bớt sự thiên lệch của thuộc tính có nhiều giá trị và có thể xử lý tốt hơn dữ liệu thiếu, đồng thời có khả năng cắt tỉa cây (pruning) để giảm độ phức tạp của mô hình.

Mặc dù độ chính xác giữa hai thuật toán là tương đương, C4.5 có thể là một lựa chọn tốt hơn trong những trường hợp phức tạp hơn, nhờ vào khả năng tối ưu hóa các mô hình phân loại và giảm thiểu overfitting.

**Đề xuất hướng nghiên cứu tiếp theo**

Mặc dù các thuật toán cây quyết định như ID3 và C4.5 rất hữu ích trong việc phân loại dữ liệu và dễ hiểu, vẫn có những phương pháp và mô hình phức tạp hơn có thể cải thiện độ chính xác và khả năng phân loại đối với các bộ dữ liệu lớn hoặc phức tạp hơn. Dưới đây là một số phương pháp mà chúng ta có thể nghiên cứu và áp dụng:

1. **Máy Vector Hỗ Trợ (Support Vector Machine - SVM)**:
   * SVM là một mô hình học máy mạnh mẽ, đặc biệt hiệu quả với các bài toán phân loại trong không gian chiều cao. SVM có khả năng tìm ra siêu phẳng phân tách tối ưu, giúp phân loại các lớp dữ liệu rõ ràng, ngay cả trong các trường hợp dữ liệu không tuyến tính.
2. **Rừng Ngẫu Nhiên (Random Forest)**:
   * Random Forest là một mô hình học máy dựa trên cây quyết định, nhưng thay vì sử dụng một cây duy nhất, nó xây dựng nhiều cây quyết định và tổng hợp kết quả từ các cây này để đưa ra dự đoán cuối cùng. Phương pháp này giúp giảm overfitting và cải thiện độ chính xác tổng thể của mô hình.
3. **Mạng Nơ-ron Nhân Tạo (Neural Networks)**:
   * Mạng nơ-ron có thể học và mô phỏng các mối quan hệ phức tạp giữa các đặc trưng trong dữ liệu. Các mô hình như Mạng Nơ-ron Lan Tỏa (Feedforward Neural Networks) hoặc Mạng Nơ-ron Convolutional (CNN) có thể xử lý tốt các bài toán phân loại phức tạp, đặc biệt là khi dữ liệu có tính phi tuyến tính cao.
4. **Kỹ thuật Tối ưu hóa**:
   * Các phương pháp tối ưu hóa như Grid Search và Random Search có thể giúp tìm ra các siêu tham số tối ưu cho mô hình, từ đó cải thiện độ chính xác. Việc điều chỉnh các tham số như độ sâu cây quyết định, số lượng cây trong Random Forest, hoặc các tham số của SVM có thể mang lại kết quả tốt hơn.

**Kết luận tổng quát**

Cả hai thuật toán ID3 và C4.5 đều là các công cụ mạnh mẽ trong việc phân loại dữ liệu và tạo ra các mô hình có thể giải thích được. Tuy nhiên, với sự phát triển của các phương pháp học máy hiện đại, có thể xem xét các mô hình phức tạp hơn như SVM, Random Forest, hoặc Neural Networks để cải thiện độ chính xác và khả năng tổng quát của mô hình trong những tình huống thực tế. Việc tiếp tục nghiên cứu và tối ưu hóa các phương pháp này sẽ giúp mở rộng ứng dụng của học máy trong nhiều lĩnh vực khác nhau.

**TÀI LIỆU THAM KHẢO**

1. Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.
2. Quinlan, J. R. (1986). *Induction of Decision Trees*. *Machine Learning*, 1(1), 81-106.
3. Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.
4. Quinn, M. J. (2011). *Decision Support Systems and Intelligent Systems* (9th ed.). Pearson.