BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KỲ

ĐỀ TÀI: ỨNG DỤNG THUẬT TOÁN CÂY QUYẾT ĐỊNH TRONG DỰ ĐOÁN VÀ PHÒNG NGỪA BỆNH

Giảng viên hương dẫn:	Th.s Từ Tuyết Hồng
Sinh viên thực hiện:	Huỳnh Thanh Duy
MSSV:	22110118
Lớp:	22110CL1B
Khóa:	2022
Mã học phần:	241MALE431085
Năm học:	2024 - 2025

Thành phố Hồ Chí Minh, tháng 12 năm 2024

DANH SÁCH THAM GIA ĐỀ TÀI

HỌC KỲ I, NĂM HỌC 2024 – 2025

Tên đề tài: Ứng Dụng Thuật Toán Cây Quyết Định Trong Dự Đoán Và Phòng Ngừa Bệnh

Họ và tên sinh viên	Mã số sinh viên
Huỳnh Thanh Duy	22110118

Nhận xét của giảng viên

.....

.....

.....

.....

.....

.....

.....

.....

Ngày 04, tháng 12, năm 2024

Giảng viên chấm điểm

MỤC LỤC

PHẦN MỞ ĐẦU 1
1. LÝ DO CHỌN ĐỀ TÀI: 1
2. MỤC TIÊU NGHIÊN CỨU: 2
3. PHẠM VI ĐỀ TÀI: 3
4. KẾT CẤU ĐỀ TÀI: 4
PHẦN NỘI DUNG 5
Chương 1: Cơ sở lý thuyết: 5
1. Tổng quan về Cây quyết định – Decision tree và thuật toán CART (Classification and Regression Trees): 5
1.1. Cấu Trúc Của Cây Quyết Định CART: 5
1.2. Quy Trình Xây Dựng Cây Quyết Định CART: 6
1.3. Tiêu Chí Đánh Giá Độ Không Thuần Nhất: 7
2. Ứng dụng thuật toán Cây quyết định CART vào bài toán Dự đoán bệnh: 17
2.1. Giới thiệu về thuật toán CART trong bài toán dự đoán bệnh 17
2.2. Vai trò của thuật toán CART trong bài toán: 17
2.3. Quy trình ứng dụng CART: 23
2.4. Kết quả đạt được từ ứng dụng CART: 24
2.5. Ví dụ minh họa: 25
3. Dữ liệu sử dụng: 25
3.1. Tập Dữ Liệu Triệu Chứng và Bệnh (DiseaseAndSymptoms.csv): 25
3.2. Tập Dữ Liệu Biện Pháp Phòng Ngừa (Disease precaution.csv): 26
3.3. Quy trình xử lý dữ liệu: 27
3.4. Đánh giá chất lượng dữ liệu: 29
3.5. Mối liên hệ giữa dữ liệu và thuật toán CART: 30
Chương 2: Phương pháp nghiên cứu: 32
1. Xử Lý và Chuẩn Hóa Dữ Liệu: 32
2. Xây Dựng và Huấn Luyện Mô Hình: 33
2.1. Chuẩn hóa dữ liệu (Mã hóa triệu chứng và bệnh): 33
2.2. Áp dụng thuật toán CART (Cây quyết định): 34
2.3. Huấn Luyện Mô Hình: 38
3. Dự đoán bệnh và phòng ngừa: 39

3.1. Dự đoán bệnh từ các triệu chứng: 39
3.2. Cung cấp biện pháp phòng ngừa: 41
Chương 3: Kết quả thực nghiệm và phân tích đánh giá: 43
1. Phân tích kết quả từ Quá trình huấn luyện Mô hình (80% Dữ liệu): 43
1.1. Hiểu biết về mô hình: 43
1.2. Hiệu suất huấn luyện: 44
2. Phân tích kết quả từ Quá trình kiểm tra Mô hình (20% Dữ liệu): 56
2.1. Tỷ lệ dự đoán đúng theo từng bệnh: 56
2.2. Ma trận nhầm lẫn đa lớp: 58
2.3. Phân tích từ báo cáo (detailed_analysis_report.txt): 63
3. Đánh Giá Mô Hình: 65
3.1. Ưu điểm: 65
3.2. Nhược điểm: 65
3.3. Đề xuất cải thiện: 66
PHẦN KẾT LUẬN 67
TÀI LIỆU THAM KHẢO 69

PHẦN MỞ ĐẦU

1. LÝ DO CHỌN ĐỀ TÀI:

Trong bối cảnh y tế hiện đại, việc dự đoán sớm các bệnh lý dựa trên triệu chứng đã trở thành một nhu cầu cấp thiết. Điều này không chỉ giúp cải thiện hiệu quả điều trị mà còn giảm thiểu các chi phí không cần thiết và áp lực lên hệ thống y tế hiện đại. Với sự phát triển của trí tuệ nhân tạo và học máy, các thuật toán như **cây quyết định** đã chứng minh tính hiệu quả trong việc phân tích dữ liệu y tế, đặc biệt là trong các bài toán phân loại và dự đoán. Vì lý do đó, đề tài **"Ứng Dụng Thuật Toán Cây Quyết Định Trong Dự Đoán Và Phòng Ngừa Bệnh"** được em nghiên cứu và triển khai làm đồ án kết thúc môn học "**Học máy"**.

Thuật toán cây quyết định được đánh giá là phù hợp cho bài toán này nhờ khả năng trực quan hóa rõ ràng và quy trình ra quyết định dễ hiểu. Các nút trong cây thể hiện cách mà mô hình sử dụng từng triệu chứng để dần thu hẹp phạm vi và đưa ra dự đoán về bệnh. Bên cạnh đó, thuật toán này còn có ưu điểm trong việc xử lý dữ liệu không yêu cầu chuẩn hóa, đồng thời phù hợp với bài toán có cấu trúc như danh sách triệu chứng và loại bệnh.

Sử dụng tập dữ liệu **DiseaseAndSymptoms.csv**, chứa thông tin về các triệu chứng liên quan đến từng bệnh, và **Disease precaution.csv**, cung cấp biện pháp phòng ngừa tương ứng, bài toán đặt ra mục tiêu không chỉ dự đoán chính xác loại bệnh mà còn cung cấp các gợi ý phòng tránh hiệu quả. Đây là một cách tiếp cận toàn diện, kết hợp giữa lý thuyết thuật toán và nhu cầu thực tế trong y tế. Đề tài này mang lại nhiều giá trị cả về mặt học thuật lẫn ứng dụng. Nó cung cấp cơ hội để thử nghiệm các bước từ xử lý dữ liệu, huấn luyện mô hình đến đánh giá hiệu suất thông qua các chỉ số như độ chính xác và ma trận nhầm lẫn. Đồng thời, bài toán còn giúp nhận diện và giải quyết các thách thức thực tế như xử lý dữ liệu không cân bằng hoặc thiếu thông tin.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

2. MỤC TIÊU NGHIÊN CỨU:

Mục tiêu chính:

Xây dựng một hệ thống dự đoán bệnh từ các triệu chứng đầu vào sử dụng thuật toán cây quyết định (Decision Tree). Hệ thống không chỉ cung cấp dự đoán bệnh mà còn đưa ra các biện pháp phòng ngừa tương ứng, giúp hỗ trợ chăm sóc sức khỏe hiệu quả hơn.

Mục tiêu cụ thể:

- **Phân tích dữ liệu triệu chứng và bệnh:**
  - ⇨ Khám phá tập dữ liệu **DiseaseAndSymptoms.csv**, trong đó mỗi bệnh được mô tả bằng tối đa 17 triệu chứng.
  - ⇨ Đánh giá tần suất xuất hiện của từng triệu chứng và phân phối số lượng mẫu cho từng bệnh để xác định mức độ cân bằng của dữ liệu.
- **Áp dụng thuật toán cây quyết định:**
  - ⇨ Sử dụng thuật toán Decision Tree để phân loại bệnh dựa trên triệu chứng.
  - ⇨ Thiết lập cấu hình mô hình với các tham số tối ưu như *max_depth* để đạt được sự cân bằng giữa độ chính xác và khả năng tổng quát hóa.
- **Xây dựng mô hình dự đoán chính xác**:
  - ⇨ Sử dụng thuật toán cây quyết định để phân loại các bệnh dựa trên bộ dữ liệu triệu chứng (DiseaseAndSymptoms.csv).
  - ⇨ Đánh giá hiệu suất của mô hình qua các chỉ số như độ chính xác, ma trận nhầm lẫn, và báo cáo phân loại.
- **Xử lý dữ liệu và cân bằng tập dữ liệu**:
  - ⇨ Xử lý các giá trị thiếu, mã hóa các triệu chứng và bệnh dưới dạng số để mô hình có thể hiểu và xử lý.
  - ⇨ Áp dụng kỹ thuật SMOTE để giải quyết vấn đề dữ liệu mất cân bằng nhằm cải thiện độ chính xác cho các lớp hiếm.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Dự đoán và cung cấp biện pháp phòng ngừa:**

⇨ Dự đoán bệnh từ triệu chứng đầu vào.

⇨ Sử dụng tập dữ liệu **Disease precaution.csv** để tra cứu và đưa ra biện pháp phòng ngừa tương ứng với bệnh dự đoán.

❖ **Phân tích kết quả và tối ưu hóa mô hình**:

⇨ Phân tích các trường hợp dự đoán sai để xác định nguyên nhân, từ đó cải thiện quy trình xử lý dữ liệu hoặc mô hình.

⇨ Kiểm tra tầm quan trọng của các triệu chứng trong việc dự đoán bệnh thông qua phân tích trọng số đặc trưng của mô hình.

❖ **Đánh giá hiệu suất của mô hình:**

⇨ Sử dụng các chỉ số như độ chính xác *(Accuracy)*, ma trận nhầm lẫn *(Confusion Matrix)*, và báo cáo phân loại *(Classification Report)* để đánh giá chất lượng mô hình.

⇨ Trực quan hóa hiệu suất bằng biểu đồ để dễ dàng phân tích kết quả.

3. PHẠM VI ĐỀ TÀI:

❖ **Dữ liệu sử dụng:**

⇨ Tập dữ liệu **DiseaseAndSymptoms.csv**:

- Chứa danh sách 41 bệnh, mỗi bệnh được mô tả bởi tối đa 17 triệu chứng.
- Dữ liệu bao gồm các triệu chứng thường gặp như: itching, skin_rash, fever,...

⇨ Tập dữ liệu **Disease precaution.csv**:

- Chứa thông tin về các biện pháp phòng ngừa của từng bệnh, tối đa 4 biện pháp phòng ngừa cho mỗi bệnh.

Ví dụ:

Bệnh: *Urinary tract infection*.

Phòng ngừa: *drink plenty of water, increase vitamin C intake*.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

- **Thuật toán sử dụng:** *Cây quyết định - CART (Classification and Regression Trees)* được cấu hình với *max_depth = 15 min_samples_split=5 và min_samples_leaf = 2* để tránh cây quá sâu gây hiện tượng *overfitting*. Đồng thời, sử dụng *Gini Index* để đo độ hỗn loạn trong dữ liệu tại mỗi nút, từ đó giúp chọn thuộc tính phân chia sao cho các nhánh con được tách biệt tốt nhất.
- **Phạm vi bài toán:**

  Dự đoán bệnh: Chỉ dự đoán được các bệnh có trong tập dữ liệu **DiseaseAndSymptoms.csv**.

  Triệu chứng: Chỉ xử lý được các triệu chứng đã được liệt kê trong tập dữ liệu.

  Gợi ý phòng ngừa: Chỉ đưa ra các biện pháp phòng ngừa dựa trên dữ liệu sẵn có là **Disease precaution.csv**, không bao gồm các chỉ dẫn điều trị chuyên sâu.
- **Hạn chế:**

  Dữ liệu:

  Tập dữ liệu không bao gồm tất cả các bệnh, nên khả năng ứng dụng chỉ giới hạn trong phạm vi dữ liệu cung cấp. Một số bệnh có thể bị thiếu triệu chứng đặc trưng hoặc dữ liệu không cân bằng (số lượng mẫu bệnh không đồng đều).

  Mô hình:

  Mô hình cây quyết định đơn lẻ có thể không đạt hiệu quả tối ưu trên dữ liệu lớn hoặc phức tạp. Trong tương lai, có thể thử nghiệm các mô hình khác như Random Forest hoặc Gradient Boosting.

4. KẾT CẤU ĐỀ TÀI:

Chương 1: Tổng quan kiến thức.

Chương 2: Phương pháp nghiên cứu.

Chương 3: Kết quả thực nghiệm và phân tích đánh giá.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

PHẦN NỘI DUNG

Chương 1: Cơ sở lý thuyết:

1. Tổng quan về Cây quyết định – Decision tree và thuật toán CART (Classification and Regression Trees):

Cây quyết định là một phương pháp học máy giám sát, được sử dụng để giải quyết cả bài toán phân loại và hồi quy. Phương pháp này hoạt động dựa trên việc phân chia tập dữ liệu thành các nhóm con ngày càng nhỏ dựa trên các thuộc tính hoặc đặc điểm của dữ liệu. Quá trình này tiếp diễn một cách đệ quy cho đến khi dữ liệu được phân chia đủ "thuần nhất", tức là tại một nút chỉ có một lớp hoặc rất ít sai số.

Thuật toán CART (Classification and Regression Trees) là một phương pháp học máy mạnh mẽ và được sử dụng rộng rãi trong cả phân loại (classification) và hồi quy (regression). CART xây dựng cây quyết định nhị phân, trong đó mỗi nút quyết định phân chia dữ liệu dựa trên một đặc trưng cụ thể, và mỗi nhánh con tương ứng với kết quả của phân chia đó. Đây là một trong những thuật toán phổ biến nhất trong học máy và có khả năng xử lý tốt các bài toán với dữ liệu phức tạp.

- ❖ Là cây nhị phân: Mỗi nút chỉ phân chia thành hai nhánh con.
- ❖ Phân loại (Classification): Dự đoán nhãn hoặc lớp của dữ liệu.
- ❖ Hồi quy (Regression): Dự đoán giá trị liên tục.

1.1. Cấu Trúc Của Cây Quyết Định CART:

- ❖ Nút Gốc (Root Node): Nút gốc là điểm bắt đầu của cây quyết định, đại diện cho toàn bộ tập dữ liệu. Tại nút này, thuật toán quyết định phân chia dữ liệu dựa trên một đặc trưng nào đó.
- ❖ Nút Quyết Định (Decision Node): Các nút quyết định trong cây là nơi dữ liệu được phân chia thành hai nhánh con, tùy thuộc vào giá trị của một đặc trưng. Mỗi phân chia tạo ra một nhánh con mới.
- ❖ Nút Lá (Leaf Node): Nút lá là điểm kết thúc của cây, nơi không có phân chia nữa. Mỗi nút lá chứa kết quả phân loại hoặc giá trị dự đoán cuối

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

cùng. Ví dụ, trong bài toán phân loại bệnh, nút lá sẽ chứa tên của bệnh được dự đoán.

1.2. Quy Trình Xây Dựng Cây Quyết Định CART:

❖ Chọn đặc trưng và giá trị phân chia:

Tại mỗi nút quyết định, thuật toán sẽ chọn một đặc trưng và giá trị phân chia sao cho các nhánh con sau phân chia có độ không thuần nhất thấp nhất. Các chỉ số như **Gini Index** và **Entropy** sẽ được sử dụng để đánh giá mức độ đồng nhất của dữ liệu trong mỗi nhánh.

❖ Phân chia dữ liệu:

Dữ liệu sẽ được phân chia thành hai nhóm con dựa trên đặc trưng và giá trị phân chia đã chọn. Sau khi phân chia, các nhóm con sẽ được kiểm tra và tiếp tục phân chia cho đến khi một trong các điều kiện dừng sau đây được thỏa mãn:

- Tất cả các điểm dữ liệu trong nhóm con đều thuộc về một lớp (trong phân loại) hoặc có giá trị mục tiêu giống nhau (trong hồi quy).
- Độ không thuần nhất không thể giảm thêm.
- Độ sâu của cây đạt đến giới hạn đã định.

❖ Lặp lại quy trình:

Quá trình phân chia sẽ lặp lại cho đến khi cây đạt được điều kiện dừng. Kết quả là một cây quyết định nhị phân, với mỗi nút quyết định sẽ chia dữ liệu thành hai nhóm con.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

1.3. Tiêu Chí Đánh Giá Độ Không Thuần Nhất:

Cây quyết định CART sử dụng **Gini Index** để đánh giá độ *"không thuần nhất"* của các nhóm con tại mỗi bước phân chia dữ liệu. Điều này giúp thuật toán chọn đặc trưng phân chia sao cho tối thiểu hóa độ hỗn loạn (**impurity**) trong mỗi nhóm con.

❖ **Gini Index trong CART:**

Gini Index được sử dụng trong thuật toán **CART** để đo độ không thuần nhất của một nhóm con sau khi phân chia. Mục tiêu của thuật toán là giảm thiểu **Gini Index** để tạo ra các nhóm con càng đồng nhất càng tốt. **Gini Index** được sử dụng trong thuật toán **CART** để đo độ không thuần nhất của một nhóm con sau khi phân chia. Mục tiêu của thuật toán là giảm thiểu **Gini Index** để tạo ra các nhóm con càng đồng nhất càng tốt.

Công thức tính Gini Index:

$$Gini(S) = 1 - \sum_{i=1}^{k} p_i^2$$

Trong đó:

S là tập dữ liệu tại 1 nút.

p_i là xác suất của lớp i trong tập dữ liệu S.

k là số lớp trong dữ liệu.

Ý nghĩa Gini Index trong CART:

⇨ **Gini = 0 (Đúng):**

- Khi **Gini = 0**, tất cả các phần tử trong tập con đều thuộc cùng một lớp.
- Điều này biểu thị sự **đồng nhất hoàn toàn**, tức là tập con không có bất kỳ sự pha trộn nào giữa các lớp khác nhau. Ví dụ: Tất cả các mẫu thuộc lớp **"Fungal infection"**.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

- ⇨ **Gini = 1 (Sai ở định nghĩ cụ thể):**
  - Gini = 1 không thể xảy ra trong thực tế với bài toán phân loại vì tổng của các xác suất ($p_i{}^2$) phải nằm trong khoảng [0,1]. Tuy nhiên, Gini đạt giá trị cao nhất khi các lớp được phân phối đồng đều (ví dụ: hai lớp có tỷ lệ 50%-50% hoặc nhiều lớp được chia đều nhau).
  - Điều này thể hiện sự không đồng nhất hoàn toàn, vì mỗi lớp có tỷ lệ như nhau, làm tăng độ bất thuần..
- ⇨ **Gini thấp:**
  - Gini thấp chỉ ra rằng tập con có ít sự phân tán giữa các lớp.
  - Điều này có nghĩa là một hoặc vài lớp chiếm ưu thế trong tập con, giúp mô hình dễ dàng đưa ra dự đoán chính xác hơn.

❖ **Entropy và So Sánh với Gini Index:**

Entropy là một chỉ số đo lường độ không chắc chắn trong lý thuyết thông tin và được sử dụng trong các thuật toán cây quyết định khác như **ID3** và **C4.5**. Tuy nhiên, **CART** chủ yếu sử dụng **Gini Index** để phân chia dữ liệu, và **Entropy** chỉ là một chỉ số được sử dụng để so sánh với Gini trong các nghiên cứu hoặc trường hợp cụ thể.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Công thức tính Entropy:

$$Entropy(S) = -\sum_{i=1}^{k} p_i \log_2 p_i$$

Trong đó:

S là tập dữ liệu tại 1 nút.

p_i là xác suất của lớp i trong tập dữ liệu S.

k là số lớp trong dữ liệu.

Ý nghĩa Entropy:

- ⇨ **Entropy = 0:**
  - Khi Entropy = 0, tất cả các phần tử trong tập con thuộc về một lớp duy nhất.
  - Điều này biểu thị rằng tập con là hoàn toàn đồng nhất, không có sự pha trộn giữa các lớp.
  - Ví dụ: Nếu tất cả các mẫu trong tập con đều thuộc lớp **"Fungal infection",** thì $Entropy = 0$.
- ⇨ **Entropy cao:**
  - Khi **Entropy cao**, dữ liệu trong tập con được phân bổ **đồng đều giữa các lớp**.
  - Entropy cao biểu thị mức **không chắc chắn lớn**, tức là khả năng phân chia dữ liệu trở nên khó khăn vì các lớp không có sự khác biệt rõ ràng.
  - Ví dụ: Nếu một tập con có hai lớp và mỗi lớp chiếm 50% ($p_1 = 0.5, p_{2=0.5}$), thì$\boldsymbol{Entropy = 1}$, biểu thị **sự bất thuần cao nhất**.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Lựa chọn thuộc tính và điểm chia:**

Impurity Reduction (IR) là một thước đo quan trọng trong thuật toán CART (Classification and Regression Tree), dùng để đánh giá hiệu quả của việc chia dữ liệu tại một nút. Nó phản ánh mức độ mà một phép chia giúp làm giảm độ bất thuần (Impurity) của dữ liệu.

Impurity Reduction (IR):

$IR = Impurity_{before\ split} - \sum_{i=1}^{k}(\frac{n_i}{N} \times Impurity_i)$

Trong đó:

$Impurity_{before\ split}$ là độ thuần của nút cha.

$Impurity_i$ là độ bất thuần của nhanh i sau khi chia.

$\frac{n_i}{N}$ là tỷ lệ mẫu thuộc nhánh i.

k là tổng số nhánh con.

Ý nghĩa của IR:

⇨ $\boldsymbol{IR = 0}$:

- Phép chia không làm giảm độ bất thuần.
- Ví dụ: Sau khi chia, các nhóm con vẫn giữ nguyên tỷ lệ phân bố lớp như nhóm cha.

⇨ $\boldsymbol{IR > 0}$: Phép chia làm giảm độ bất thuần, tức là tập con sau khi chia đồng nhất hơn nhóm cha.

⇨ ***IR* lớn nhất:** Đó là phép chia tối ưu, giúp giảm bất thuần nhiều nhất.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **So Sánh Giữa Gini Index và Entropy:**

Tiêu chí	**Gini Index**	**Entropy**
Định nghĩa	Đo lường xác suất mà một mẫu được chọn ngẫu nhiên thuộc về một lớp sai.	Đo lường mức độ "rối loạn" (hoặc không chắc chắn) trong tập dữ liệu tại một nút.
Ý nghĩa	Gini đo lường **xác suất** mà hai mẫu được chọn ngẫu nhiên từ tập dữ liệu thuộc về các lớp khác nhau.	Entropy đo lường mức độ "không chắc chắn" của thông tin trong tập dữ liệu.
	Chỉ số này tập trung vào việc giảm lỗi khi chia tách, giúp tối ưu hóa quá trình phân loại.	Phù hợp hơn trong việc hiểu bản chất của sự không đồng nhất trong dữ liệu.
Ưu Điểm	Tính toán đơn giản hơn vì không yêu cầu tính logarit.	Cung cấp sự phân tích chi tiết hơn về độ bất thuần trong dữ liệu.
	Thường nhanh hơn, phù hợp với các bài toán lớn hoặc khi cần tối ưu hiệu suất.	Phù hợp trong các trường hợp khi cần hiểu rõ mức độ "không chắc chắn" hoặc "mất mát thông tin".
Nhược Điểm	Không cung cấp thông tin về mức độ không chắc chắn như Entropy.	Tính toán phức tạp hơn do yêu cầu logarit, dẫn đến thời gian chạy lâu hơn với tập dữ liệu lớn.
	Ít phù hợp hơn khi xử lý các tập dữ liệu có sự phân bố bất thường hoặc không đồng đều giữa các lớp.	Có thể gây quá tải trong các hệ thống không yêu cầu phân tích chi tiết mức độ bất thuần.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Khác Biệt Cơ Bản	Gini tập trung vào việc đo **tính thuần khiết** (purity) của các nhóm con khi chia dữ liệu.	Entropy đo lường mức độ **không chắc chắn** trong thông tin, dựa trên lý thuyết thông tin.
	Tốc độ xử lý nhanh hơn, thường được sử dụng trong thuật toán CART.	Cung cấp nhiều thông tin hơn, thường được sử dụng trong thuật toán ID3 hoặc C4.5.

❖ **Ví dụ minh họa:**

Chúng ta sẽ minh họa cách tính Entropy và Gini Index bằng một ví dụ cụ thể dựa trên dữ liệu từ file **DiseaseAndSymptoms.csv**.

Dữ liệu giả định:

Giả sử chúng ta có một tập dữ liệu gồm các bệnh và triệu chứng, trong đó *Disease* là nhãn phân loại:

Symptom	**Disease**
fatigue	Typhoid
fatigue	Typhoid
fatigue	Common Cold
headache	Migraine
headache	Typhoid
headache	Common Cold

Tập ban đầu:

Số lượng mẫu là 6

Số lớp:

Typhoid: 3

Common Cold: 2

Migraine: 1

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Tính Entropy:

$$Entropy(S) = -\sum_{i=1}^{k} p_i \log_2 p_i$$

Xác suất các lớp:

$p_{Typhoid} = \frac{3}{6} = 0.5$

$p_{CommonCold} = \frac{2}{6} \approx 0.333$

$p_{Migraine} = \frac{1}{6} \approx 0.167$

⇨ $Entropy = -(0.5 \times \log_2 0.5 + 0.333 \times \log_2 0.333 + 0.167 \times \log_2 0.167)$

⇨ $Entropy = -(-0.5 - 0.528 - 0.432) \approx 1.46$

Tính Gini Index:

$$Gini(S) = 1 - \sum_{i=1}^{k} p_i^2$$

⇨ $Gini = 1 - (0.5^2 + 0.333^2 + 0.167^2) \approx 0.611$

Độ bất thuần dữ liệu ban đầu (trước khi chia nhánh):

$$Entropy_{before} \approx 1.46$$

$$Gini_{before} \approx 0.611$$

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Phân chia dựa trên triệu chứng *fatigue*:

Nhánh 1 (fatigue):

Số mẫu: 3

Phân bố lớp:

Typhoid: 2

Common Cold:

Entropy nhánh 1:

⇨ $Entropy_1 = -(0.667 \times \log_2 0.667 + 0.333 \times \log_2 0.333$

⇨ $Entropy_1 \approx 0.918$

Gini Index nhánh 1:

⇨ $Gini_1 = 1 - (0.667^2 + 0.333^2)$

⇨ $Gini_1 \approx 0.444$

Nhánh 2 (không fatigue):

Số mẫu: 3

Phân bố lớp:

Typhoid: 1

Common Cold: 1

Migraine: 1

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Entropy nhánh 2:

⇨ $Entropy_2 = -(0.333 \times \log_2 0.333 + 0.333 \times \log_2 0.333 + 0.333 \times \log_2 0.333$

⇨ $Entropy_2 \approx 1.585$

Gini Index nhánh 2:

⇨ $Gini_2 = 1 - (0.333^2 + 0.333^2 + 0.333^2)$

⇨ $Gini_1 \approx 0.667$

Độ bất thuần sau chia nhánh:

Entropy sau chia nhánh:

⇨ $Entropy_{after} = \left(\frac{3}{6} \times 0.918\right) + \left(\frac{3}{6} \times 1.585\right)$

⇨ $Entropy_{after} \approx 1.251$

Gini Index sau khi chia nhánh:

⇨ $Gini_{after} = \left(\frac{3}{6} \times 0.444\right) + \left(\frac{3}{6} \times 0.667\right)$

⇨ $Gini_{after} \approx 0.556$

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Tính Impurity Reduction (IR):

IR với Entropy:

$$IR_{entropy} = Entropy_{before} - Entropy_{after} = 1.46 - 1.251 \approx 0.209$$

⇨ **IR (Entropy)**: Việc chia dữ liệu theo fatigue giúp giảm 20.9% độ bất thuần dựa trên Entropy.

IR với Gini:

$$IR_{Gini} = Gini_{before} - Gini_{after} = 0.611 - 0.556 \approx 0.055$$

⇨ **IR (Gini Index)**: Việc chia giảm 5.5% độ bất thuần dựa trên Gini Index.

Việc so sánh **IR của Entropy** và **Gini Index** giúp đánh giá chất lượng của việc chia nhánh để tìm ra phương pháp hiệu quả nhất.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

2. Ứng dụng thuật toán Cây quyết định CART vào bài toán Dự đoán bệnh:

2.1. Giới thiệu về thuật toán CART trong bài toán dự đoán bệnh

CART (Classification and Regression Tree) là một trong những thuật toán học máy được ứng dụng rộng rãi trong các bài toán phân loại và hồi quy. Thuật toán hoạt động dựa trên việc phân chia dữ liệu thành các nhánh dựa vào tiêu chí giảm thiểu độ bất thuần, chẳng hạn như Gini Index hoặc Entropy. Trong lĩnh vực y tế, CART cho phép phân loại chính xác các bệnh dựa trên tập hợp triệu chứng được cung cấp.

Với bài toán dự đoán bệnh từ tập dữ liệu **DiseaseAndSymptoms.csv**, chứa danh sách các triệu chứng liên quan đến từng bệnh, và tập dữ liệu **Disease precaution.csv**, cung cấp biện pháp phòng ngừa, thuật toán CART không chỉ hỗ trợ dự đoán bệnh mà còn đưa ra gợi ý phòng tránh, góp phần nâng cao hiệu quả chẩn đoán và điều trị.

Ví dụ: Nếu triệu chứng *"itching"* xuất hiện, cây có thể phân chia tiếp theo dựa trên triệu chứng *"skin_rash"*. Nếu cả hai triệu chứng đều xuất hiện, bệnh *"Fungal infection"* sẽ được dự đoán với xác suất cao.

2.2. Vai trò của thuật toán CART trong bài toán:

- **Phân loại bệnh dựa trên triệu chứng:**

  Thuật toán CART sử dụng danh sách các triệu chứng để phân loại bệnh. Cụ thể:

  - ⇨ Mỗi triệu chứng được coi là một đặc trưng (feature) đầu vào của thuật toán, còn nhãn (label) đầu ra là các loại bệnh tương ứng.
  - ⇨ CART dựa vào chỉ số Gini Index hoặc Entropy để quyết định cách chia nhánh tại mỗi nút trong cây, chọn ra triệu chứng làm giảm bất thuần lớn nhất.
  - ⇨ Kết quả là một cây quyết định có cấu trúc rõ ràng, trong đó mỗi nút thể hiện cách một triệu chứng cụ thể phân chia dữ liệu thành các

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

nhóm nhỏ hơn để dễ dàng đưa ra dự đoán chính xác.

Ví dụ:

Từ tập dữ liệu **DiseaseAndSymptoms.csv**, nếu triệu chứng **fatigue** xuất hiện, cây có thể xác định các khả năng cao dẫn đến bệnh **Anemia** hoặc **Typhoid**.

❖ **Tính trực quan và dễ diễn giải:**

CART tạo ra một cây quyết định, nơi mà người dùng có thể dễ dàng theo dõi từng bước:

⇨ Mỗi nút đại diện cho một triệu chứng, và các nhánh thể hiện kết quả có hoặc không dựa trên triệu chứng đó.

⇨ Với tính trực quan này, các bác sĩ hoặc người dùng không cần hiểu sâu về thuật toán vẫn có thể biết được cách hệ thống đưa ra dự đoán.

Vai trò này đặc biệt quan trọng trong bối cảnh y tế:

⇨ Người dùng không chuyên môn vẫn có thể nhận biết mối liên hệ giữa triệu chứng và bệnh thông qua cấu trúc cây.

⇨ Các bác sĩ có thể sử dụng cây quyết định như một công cụ hỗ trợ để xác nhận các chẩn đoán của mình.

❖ **Đánh giá tầm quan trọng của triệu chứng:**

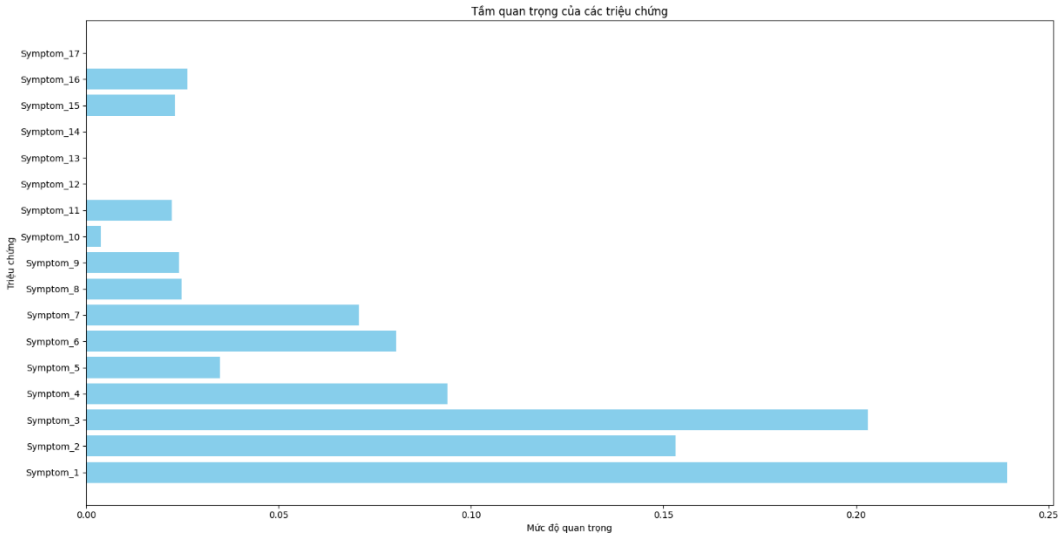
Một ưu điểm lớn của CART là khả năng đánh giá mức độ quan trọng của từng triệu chứng thông qua chỉ số Feature Importance:

⇨ Tầm quan trọng được xác định dựa trên mức giảm bất thuần mà triệu chứng đó mang lại khi phân chia dữ liệu.

⇨ Điều này giúp:

- Phát hiện các triệu chứng có vai trò quyết định trong việc chẩn đoán bệnh.
- Tập trung vào các triệu chứng có giá trị dự đoán cao, từ đó tối ưu hóa quy trình chẩn đoán.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms



Hình 1. Biểu đồ tầm quan trọng của của triệu chứng trong dự đoán bệnh của mô hình.

Ví dụ:

Từ tập dữ liệu, CART có thể xác định triệu chứng **fatigue** quan trọng nhất trong dự đoán bệnh **Anemia**, hoặc triệu chứng **skin rash** liên quan mạnh đến bệnh **Chickenpox**.

❖ **Hỗ trợ phòng ngừa bệnh:**

CART không chỉ dừng lại ở việc dự đoán bệnh mà còn hỗ trợ phòng ngừa bệnh:

⇨ Sau khi dự đoán, CART có thể liên kết với tập dữ liệu phòng ngừa (Disease precaution.csv) để đưa ra các gợi ý phù hợp.

⇨ Ví dụ, nếu hệ thống dự đoán bệnh Urinary Tract Infection (UTI), nó sẽ đề xuất các biện pháp như: Uống nhiều nước, tăng cường vitamin C.

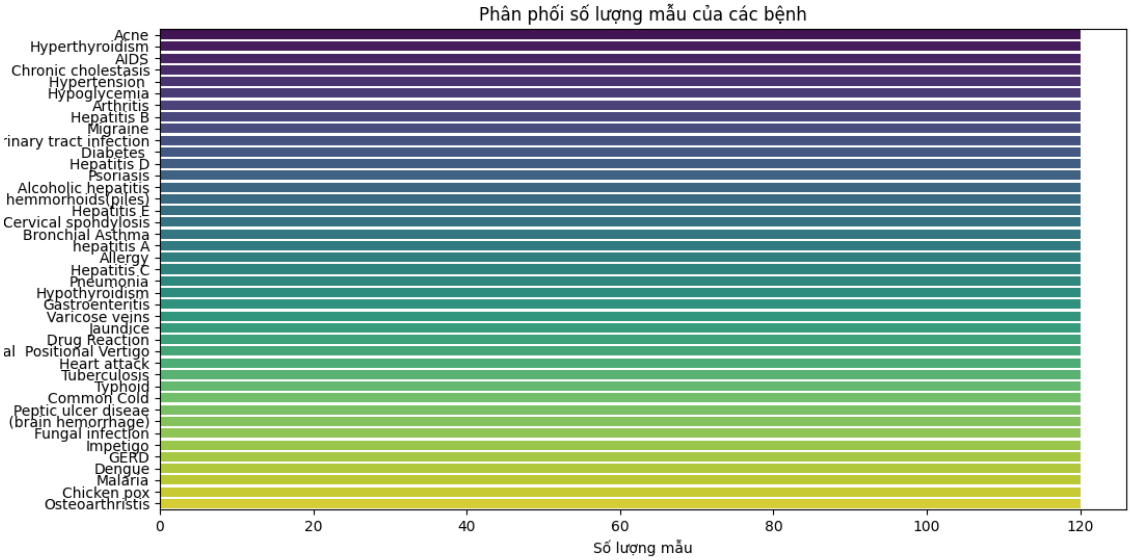
⇨ Vai trò này giúp người dùng không chỉ hiểu về tình trạng sức khỏe hiện tại mà còn biết cách giảm nguy cơ bệnh trong tương lai.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Xử lý dữ liệu phức tạp trong y tế:**

Bài toán dự đoán bệnh trong y tế đi kèm với nhiều thách thức trong xử lý dữ liệu, tuy nhiên dữ liệu ban đầu trong đề tài này đã có sự phân bố khá cân bằng giữa các lớp bệnh. Điều này giúp giảm thiểu nhu cầu sử dụng các phương pháp cân bằng dữ liệu như SMOTE, vốn thường được áp dụng trong các bài toán có dữ liệu mất cân bằng nghiêm trọng.

⇨ **Dữ liệu cân bằng:** Nhờ phân phối tương đối đồng đều của các mẫu bệnh, mô hình CART có thể trực tiếp huấn luyện mà không cần tạo thêm dữ liệu giả lập. Điều này giúp đảm bảo tính chính xác và giảm nguy cơ overfitting khi sử dụng dữ liệu nhân tạo.



Hình 2. Biểu đồ phân phối số lượng mẫu bệnh theo ***DiseaseAndSymptoms.csv***

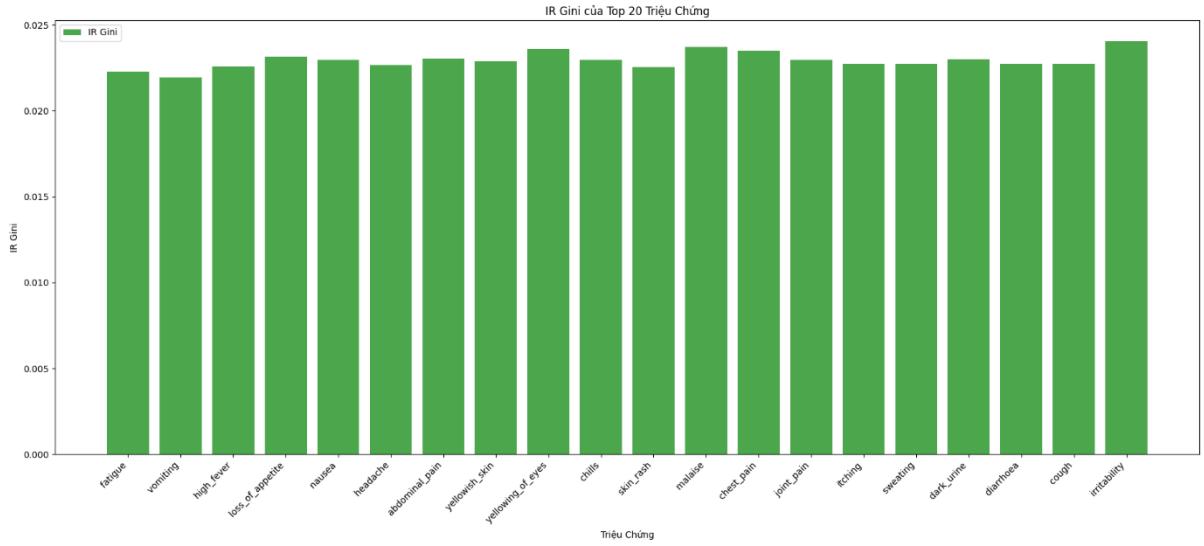
Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

- ⇨ **Dữ liệu danh mục (categorical data):** Các triệu chứng được biểu diễn dưới dạng danh mục (có/không) rất phù hợp với thuật toán CART. Dữ liệu không cần phải chuẩn hóa hoặc chuyển đổi phức tạp, giúp thuật toán dễ dàng xây dựng cây quyết định dựa trên các giá trị thực tế.

❖ **Đưa ra các quyết định tối ưu:**

CART chọn triệu chứng tốt nhất ở mỗi bước dựa trên:

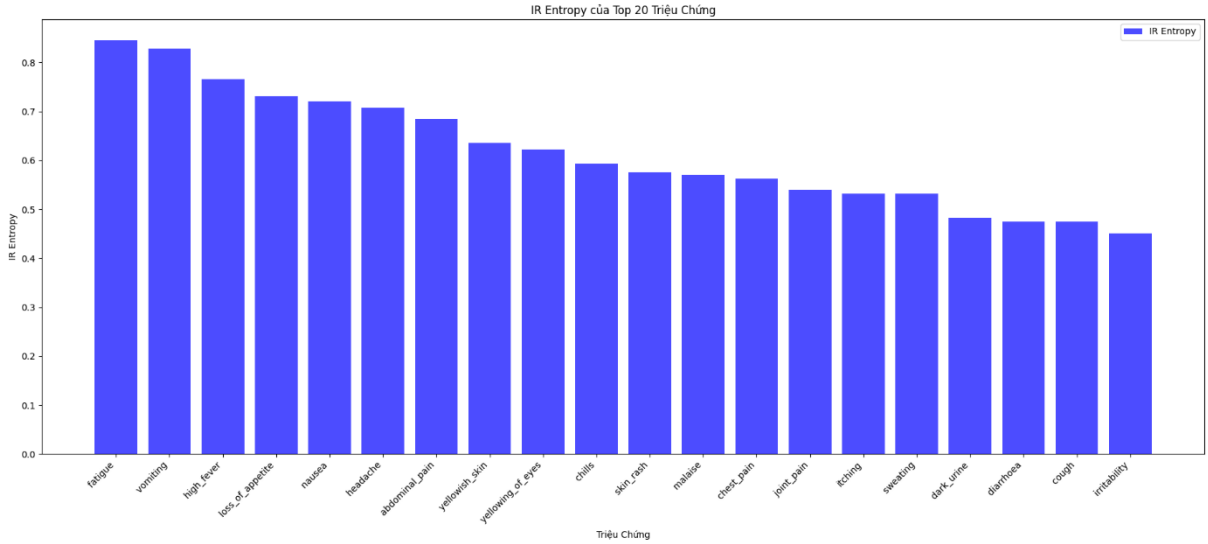
- ⇨ Gini Index: Đo lường mức độ bất thuần của dữ liệu.



Hình 3. Biểu đồ IR_{gini} của top 20 loại triệu chứng phổ biến trong ***DiseaseAndSymptoms.csv.***

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

⇨ Entropy: Đánh giá mức độ không chắc chắn của dữ liệu.



Hình 4. Biểu đồ $IR_{entropy}$ của top 20 loại triệu chứng phổ biến trong ***DiseaseAndSymptoms.csv****.*

Quá trình tối ưu này giúp cây quyết định đạt được hai mục tiêu:

⇨ Tối đa hóa độ chính xác trong phân loại.

Hình 5 biểu đồ độ chính xác của mô hình.

⇨ Giữ cho cây quyết định có cấu trúc gọn gàng, tránh overfitting.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

- **Cải thiện quy trình chẩn đoán và điều trị:**

  Thuật toán CART giúp tự động hóa quy trình chẩn đoán:

  - ⇨ Người dùng nhập danh sách các triệu chứng hiện tại, hệ thống sẽ đưa ra dự đoán và đề xuất phòng ngừa.
  - ⇨ Với độ chính xác cao, hệ thống hỗ trợ bác sĩ trong các quyết định điều trị ban đầu, giúp tiết kiệm thời gian và nâng cao hiệu quả điều trị.

2.3. Quy trình ứng dụng CART:

Quá trình ứng dụng thuật toán CART trong bài toán dự đoán bệnh được thực hiện qua các bước:

Xử lý và cân bằng dữ liệu:

- Dữ liệu từ tập **DiseaseAndSymptoms.csv** được làm sạch (xử lý các giá trị rỗng) và mã hóa dưới dạng số để sử dụng trong thuật toán.
- Dữ liệu mất cân bằng (một số bệnh xuất hiện ít hơn) được cân bằng bằng kỹ thuật SMOTE.

Huấn luyện mô hình CART:

- Sử dụng thư viện *DecisionTreeClassifier* của *Scikit-learn* để xây dựng cây quyết định.
- Các thông số tối ưu được lựa chọn:
  - *max_depth=15*: Giới hạn độ sâu của cây để tránh quá khớp.
  - *min_samples_split=5*: Số lượng mẫu tối thiểu để tiếp tục phân chia nút.
  - *min_samples_leaf=2*: Số lượng mẫu tối thiểu trong mỗi nhánh lá.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Dự đoán bệnh:

- ❖ Với các triệu chứng đầu vào, mô hình sử dụng cây quyết định để phân loại và dự đoán bệnh.
- ❖ Ví dụ: Triệu chứng đầu vào **itching: Yes, skin_rash: Yes, nodal_skin_eruptions: Yes** được dự đoán là **Fungal infection**.

Gợi ý biện pháp phòng ngừa:

Dựa trên tập dữ liệu Disease precaution.csv, mô hình cung cấp các biện pháp phòng ngừa tương ứng với bệnh được dự đoán.

Đánh giá mô hình:

- ❖ Hiệu suất được đo lường qua độ chính xác, báo cáo phân loại (classification report) và ma trận nhầm lẫn (confusion matrix).
- ❖ Độ chính xác đạt được trên tập kiểm tra: **>90% (với dữ liệu đã cân bằng)**.

2.4. Kết quả đạt được từ ứng dụng CART:

Dựa trên những gì đã thực hiện, kết quả chính bao gồm:

Độ chính xác cao:

Với tập dữ liệu đã cân bằng, CART đạt được độ chính xác đáng kể trên tập kiểm tra (90%) và hiệu suất tốt trên tập validation.

Phân tích đặc trưng quan trọng:

Các đặc trưng (triệu chứng) được mô hình đánh giá mức độ quan trọng, giúp nhận diện các triệu chứng có ảnh hưởng lớn đến việc phân loại bệnh.

Dự đoán và gợi ý hiệu quả:

Hệ thống không chỉ dự đoán chính xác bệnh mà còn cung cấp các gợi ý phòng ngừa cụ thể, ví dụ:

- o *Bệnh:* Fungal infection.
- o *Phòng ngừa:* "Rửa tay thường xuyên", "Giữ vùng bị ảnh hưởng sạch và khô".

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Trực quan hóa mô hình:

Cây quyết định được vẽ và trực quan hóa để phân tích cách mô hình đưa ra quyết định.

2.5. Ví dụ minh họa:

Triệu chứng đầu vào:

Itching	Yes
Skin rash	Yes
Nodal skin eruption	No
Dischromic patches	Yes

Kết quả dự đoán:

Bệnh dự đoán: Fungal infection

Biện pháp phòng ngừa: Giữ vùng bị ảnh hưởng sạch và khô, rửa tay thường xuyên, tránh ẩm ướt.

3. Dữ liệu sử dụng:

3.1. Tập Dữ Liệu Triệu Chứng và Bệnh (DiseaseAndSymptoms.csv):

Cấu trúc:

- ⇨ Tập dữ liệu chứa danh sách các bệnh lý và các triệu chứng liên quan.
- ⇨ Bao gồm 17 cột tương ứng với các triệu chứng (***Symptom_1 đến Symptom_17***) và 1 cột nhãn (***Disease***) xác định bệnh tương ứng.
- ⇨ Một số giá trị trong các cột triệu chứng có thể rỗng hoặc không đầy đủ, đại diện cho các triệu chứng không xảy ra.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Mục đích sử dụng:

- ⇨ Làm dữ liệu đầu vào để huấn luyện mô hình CART.
- ⇨ Phân tích vai trò và mức độ ảnh hưởng của từng triệu chứng trong dự đoán bệnh.
- ⇨ Đánh giá độ bất thuần (Entropy, Gini) của các nhánh dựa trên dữ liệu này.

3.2. Tập Dữ Liệu Biện Pháp Phòng Ngừa (Disease precaution.csv):

Cấu trúc:

- ⇨ Cung cấp thông tin các biện pháp phòng ngừa tương ứng với từng loại bệnh.
- ⇨ Gồm 2 cột chính:
  - *Disease:* Tên bệnh.
  - *Precautions:* Danh sách các biện pháp phòng ngừa được liệt kê theo dạng chuỗi văn bản.

Mục đích sử dụng:

- ⇨ Tích hợp vào mô hình để cung cấp thêm các thông tin hữu ích cho người dùng sau khi dự đoán bệnh.
- ⇨ Tăng tính ứng dụng thực tế của hệ thống thông qua việc đề xuất biện pháp phòng ngừa.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

3.3. Quy trình xử lý dữ liệu:

❖ **Tiền xử lý dữ liệu:**

Điền giá trị rỗng:

⇨ Các cột ***Symptom_1 đến Symptom_17*** trong tập dữ liệu **DiseaseAndSymptoms.csv** chứa thông tin triệu chứng liên quan đến từng bệnh. Tuy nhiên, không phải bệnh nào cũng có đủ 17 triệu chứng.

⇨ Những ô trống được thay thế bằng chuỗi trống (“”) để đảm bảo tính đồng nhất và không làm gián đoạn quá trình mã hóa.

Mã hóa dữ liệu:

⇨ Triệu chứng:

- Sử dụng ***LabelEncoder*** để chuyển đổi tên triệu chứng (dạng chuỗi) thành các số nguyên.
- Việc này giúp thuật toán CART xử lý dữ liệu dễ dàng hơn, vì nó làm việc hiệu quả hơn với dữ liệu số.
- Tất cả triệu chứng từ 17 cột được gộp lại thành một danh sách duy nhất trước khi mã hóa để đảm bảo mỗi triệu chứng chỉ được gán một giá trị duy nhất.

⇨ Tên bệnh:

- Tương tự, tên bệnh từ cột ***Disease*** cũng được mã hóa bằng ***LabelEncoder***.
- Điều này giúp chuẩn bị dữ liệu đầu ra cho cây quyết định, nơi mỗi giá trị số đại diện cho một loại bệnh.

❖ **Chuẩn hóa triệu chứng và bệnh:**

Triệu chứng:

Các triệu chứng được liệt kê trong các cột từ ***Symptom_1 đến Symptom_17*** có thể chứa dữ liệu không đồng nhất, bao gồm các giá trị rỗng hoặc không chuẩn. Để đảm bảo tính nhất quán và hỗ

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

trợ quá trình huấn luyện mô hình, mỗi cột triệu chứng được chuẩn hóa bằng cách sử dụng **LabelEncoder**. Quy trình này gán một giá trị số duy nhất cho mỗi triệu chứng, đảm bảo các giá trị tương ứng là dễ hiểu cho mô hình. Sau khi chuẩn hóa, các ánh xạ giữa giá trị số và triệu chứng ban đầu được lưu trữ trong file ***symptom_encoders.pkl*** để hỗ trợ việc khôi phục sau này. Dữ liệu sau mã hóa được lưu vào file ***encoded_data.csv.***

Tên bệnh:

Cột ***Disease*** chứa các tên bệnh dưới dạng văn bản, có thể gây khó khăn cho mô hình trong việc xử lý. Do đó, tên bệnh được mã hóa thành các giá trị số duy nhất bằng cách sử dụng **LabelEncoder**. Quá trình này không chỉ đảm bảo tính đồng nhất mà còn cho phép mô hình làm việc hiệu quả hơn. Ánh xạ giữa các giá trị số và tên bệnh ban đầu được lưu trong file ***disease_encoder.pkl***. Cùng với dữ liệu triệu chứng đã chuẩn hóa, các mã bệnh được lưu vào ***encoded_data.csv, symptom_mapping.csv và disease_mapping.csv*** để đảm bảo tính liên kết chặt chẽ giữa dữ liệu đầu vào và đầu ra của mô hình.

❖ **Cân bằng dữ liệu (nếu cần):**

Xử lý mất cân bằng lớp:

⇨ Trong tập dữ liệu, số lượng mẫu của mỗi loại bệnh có thể không đồng đều. Điều này có thể khiến cây quyết định thiên vị về các lớp có số lượng mẫu lớn hơn.

⇨ Sử dụng kỹ thuật **SMOTE (Synthetic Minority Over-sampling Technique)** để tạo thêm dữ liệu cho các lớp ít xuất hiện, cân bằng lại phân phối lớp trước khi huấn luyện mô hình.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Biểu đồ phân phối:

- ⇨ Trước và sau khi sử dụng SMOTE, biểu đồ phân phối lớp được vẽ để trực quan hóa sự thay đổi, đảm bảo rằng dữ liệu đã cân bằng.
- ⇨ Việc này không chỉ giúp cải thiện hiệu suất mô hình mà còn tăng tính công bằng khi dự đoán các bệnh hiếm gặp.

Nhưng đối với tập dữ liệu **DiseaseAndSymptoms.csv** thì dữ liệu đầu vào đã được cân bằng (có thể xem chi tiết ở **Hình 2**)

❖ **Phân chia dữ liệu:**

Tách tập huấn luyện và kiểm tra:

- ⇨ Dữ liệu sau khi tiền xử lý được chia thành 2 tập:
  - o Tập huấn luyện (80%): Để huấn luyện mô hình cây quyết định.
  - o Tập kiểm tra (20%): Để đánh giá hiệu suất mô hình trên dữ liệu chưa từng thấy.
- ⇨ Phương pháp chia dữ liệu đảm bảo tính ngẫu nhiên nhưng vẫn giữ phân phối tương đồng với dữ liệu gốc.

Lưu dữ liệu kiểm tra:

Tập kiểm tra được lưu vào file test_data.csv để dễ dàng sử dụng cho các bước phân tích và đánh giá sau này.

3.4. Đánh giá chất lượng dữ liệu:

Dữ liệu được xử lý để đảm bảo tính đầy đủ, đồng nhất và đa dạng trước khi đưa vào mô hình. Các giá trị rỗng trong cột triệu chứng (***Symptom_1 đến Symptom_17***) được điền và chuẩn hóa bằng **LabelEncoder**. Phân phối dữ liệu ban đầu không đồng đều giữa các bệnh đã được cân bằng bằng **SMOTE**, giảm thiểu lệch mẫu.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Chất lượng dữ liệu được đánh giá thông qua các chỉ số như **Entropy, Gini Index, và Impurity Reduction (IR)**, giúp xác định mức độ quan trọng của từng triệu chứng đối với dự đoán bệnh. Kết quả sau xử lý và đánh giá được lưu trong các file như **encoded_data.csv** và **symptom_metrics_results.csv**, đảm bảo dữ liệu đạt chuẩn để hỗ trợ mô hình hoạt động hiệu quả.

3.5. Mối liên hệ giữa dữ liệu và thuật toán CART:

Vai trò của triệu chứng và bệnh trong CART:

- ⇨ Cột triệu chứng (Symptom_1 đến Symptom_17):
  - Các triệu chứng đóng vai trò là các đặc trưng (features) trong cây quyết định.
  - CART sử dụng các triệu chứng để chia dữ liệu thành các nhánh, giảm thiểu độ bất thuần (Gini hoặc Entropy) tại mỗi bước.
  - Mỗi triệu chứng được xem xét như một tiêu chí phân nhánh, nơi CART tìm điểm chia tốt nhất để tối ưu hóa cấu trúc cây.
- ⇨ Cột bệnh (Disease):
  - Đây là biến mục tiêu (target) mà mô hình dự đoán tại các nút lá của cây quyết định.
  - Các nhánh cây được xây dựng dựa trên việc giảm bất thuần của biến này thông qua các đặc trưng.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Phù hợp của dữ liệu với CART:**

⇨ Dữ liệu danh mục:

- Tập dữ liệu bao gồm các triệu chứng và bệnh danh mục (categorical), phù hợp tự nhiên với cách mà CART xử lý dữ liệu.
- Thuật toán CART không yêu cầu chuẩn hóa dữ liệu, điều này phù hợp với dữ liệu triệu chứng dạng chuỗi hoặc số nguyên sau khi mã hóa.

⇨ Giảm bất thuần:

- CART sử dụng các chỉ số như Gini Index hoặc Entropy để đánh giá chất lượng của mỗi nhánh phân chia.
- Với dữ liệu triệu chứng và bệnh, việc giảm bất thuần tại mỗi nhánh cho phép mô hình tập trung vào việc phân biệt các bệnh dựa trên triệu chứng.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Chương 2: Phương pháp nghiên cứu:

1. Xử Lý và Chuẩn Hóa Dữ Liệu:

❖ **Nguồn dữ liệu:**

⇨ Tập dữ liệu DiseaseAndSymptoms.csv: Bao gồm danh sách triệu chứng của từng bệnh.

⇨ Tập dữ liệu Disease precaution.csv: Cung cấp các biện pháp phòng ngừa tương ứng cho từng bệnh.

❖ **Chuẩn hóa dữ liệu triệu chứng:**

⇨ Dữ liệu triệu chứng được mã hóa bằng LabelEncoder để chuyển đổi các giá trị rời rạc (như "itching", "skin_rash") thành số nguyên. Điều này giúp các thuật toán có thể xử lý dễ dàng.

⇨ Quá trình mã hóa được lưu lại dưới dạng file ***symptom_encoders.pkl*** để đảm bảo việc khôi phục dữ liệu về dạng ban đầu sau khi huấn luyện.

❖ **Chuẩn hóa dữ liệu bệnh:**

⇨ Tương tự, các tên bệnh (ví dụ: "AIDS", "Diabetes") cũng được mã hóa bằng LabelEncoder và lưu trữ dưới dạng file **disease_encoder.pkl**.

⇨ Việc mã hóa này đồng nhất dữ liệu bệnh, giúp tối ưu hóa hiệu quả khi huấn luyện và đánh giá mô hình.

❖ **Phát hiện và xử lý dữ liệu mất cân bằng:**

⇨ Bằng cách sử dụng biểu đồ **countplot**, phân phối số lượng mẫu giữa các bệnh được trực quan hóa để phát hiện bất kỳ sự mất cân bằng nào.

⇨ Nếu dữ liệu không cân bằng, kỹ thuật **SMOTE (Synthetic Minority Oversampling Technique)** được áp dụng để tạo ra các mẫu dữ liệu nhân tạo, giúp cân bằng phân phối lớp bệnh.

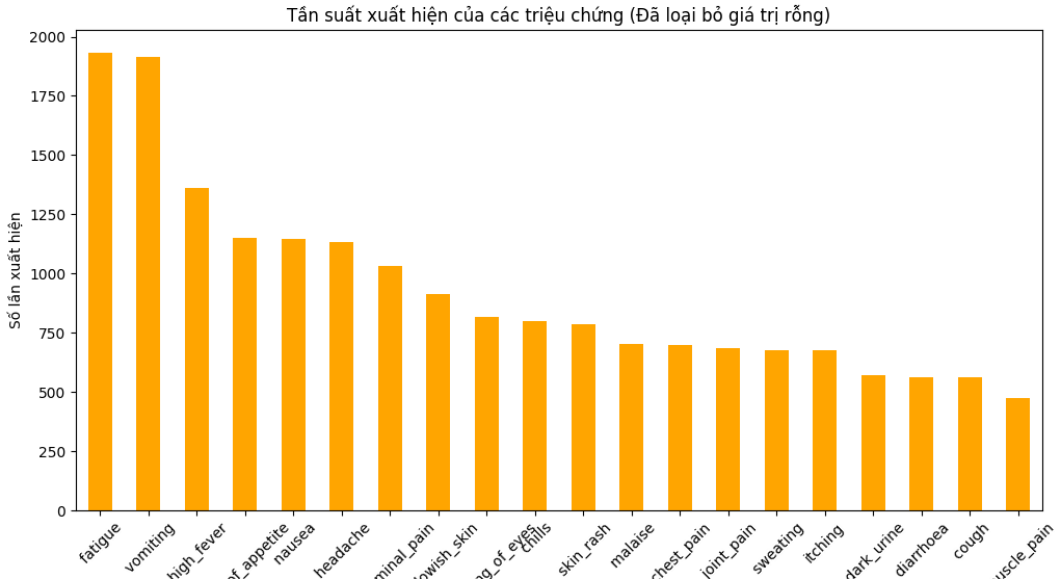
⇨ Nhưng đối với dự liệu đầu vào **DiseaseAndSymptoms.csv** ban đầu đã được cần bằng (Có thể tham khảo ***Hình 2***)

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Đánh giá chất lượng dữ liệu:**

Các thông tin quan trọng như số lượng triệu chứng, sự đồng nhất trong dữ liệu bệnh, và tỷ lệ phân phối triệu chứng đã được kiểm tra và đưa ra thành các biểu đồ để trực quan hóa dữ liệu đầu vào.

⇨ Có thể truy cập **thư mục Chart** để xem các biểu đồ cần thiết.



Hình 6. Biểu đồ tần suất xuất hiện các bệnh (đã được loại bỏ giá trị NaN)

2. Xây Dựng và Huấn Luyện Mô Hình:

2.1. Chuẩn hóa dữ liệu (Mã hóa triệu chứng và bệnh):

❖ **Triệu chứng:** Đầu tiên, ta chuyển đổi các triệu chứng trong các cột từ *dạng văn bản* (*string*) thành *dạng số* (*integer*). Sử dụng **LabelEncoder**, các giá trị như “fever”, “headache”, ... sẽ được ánh xạ thành các số nguyên tương ứng.

❖ **Bệnh:** Tiếp theo, các nhãn bệnh trong cột **Disease** cũng được mã hóa thành các số nguyên tương ứng, nhờ vào LabelEncoder.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Ví dụ: Trong file **DiseaseAndSymptoms.csv**, triệu chứng như "fever" có thể được mã hóa thành số 0, "headache" thành số 1, …

⇨ Có thể xem chi tiết trong file **encoded_data.csv, symptom_mapping.csv và disease_mapping.csv** để hiểu thêm về chi tiết.

Công thức Mã hóa:

$$X_{encoded} = LabelEncoder(X)$$

Trong đó:

$X_{encoded}$ là dữ liệu đầu vào đã mã hóa dưới dạng số.

2.2. Áp dụng thuật toán CART (Cây quyết định):

Sau khi mã hóa dữ liệu, thuật toán CART (Classification and Regression Tree) sẽ được sử dụng để huấn luyện mô hình. CART xây dựng một cây quyết định bằng cách chọn triệu chứng phù hợp nhất tại mỗi bước chia (split). Cây quyết định chọn triệu chứng tốt nhất dựa trên việc tính toán chỉ số Gini hoặc Entropy.

❖ **Công thức tính Gini index**:

$$Gini(D) = 1 - \sum_{i=1}^{k} {p_i}^2$$

Trong đó:

D là tập dữ liệu hiện tại.

p_i là tỉ lệ mẫu thuộc lớp i trong dữ liệu D.

k là số lớp bệnh (các loại bệnh).

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Công thức tính Entropy**:

$$Entropy(D) = -\sum_{i=1}^{k} p_i \times \log_2 p_i$$

Trong đó:

D là tập dữ liệu hiện tại.

p_i là tỉ lệ mẫu thuộc lớp i trong dữ liệu D.

k là số lớp bệnh (các loại bệnh).

⇨ **Giải thích:**

- **Gini index:** Càng nhỏ giá trị Gini, sự phân chia trong tập con càng đồng nhất. Cây quyết định sẽ tìm cách chia sao cho Gini nhỏ nhất.
- **Entropy:** Cây quyết định sẽ tìm cách chia sao cho Entropy giảm đi nhiều nhất, nghĩa là làm giảm sự không chắc chắn trong tập con.

Symptom_1	**Symptom_2**	**Disease**
Fever	Headache	Flu
Headache	Cough	Cold
Fever	Cough	Flu
Fever	Headache	Flu

Để chia cây, CART sẽ tính Gini hoặc Entropy của các nút để xác định triệu chứng nào (fever, headache, cough) sẽ là tiêu chí phân chia.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Phân chia (split)**: CART sẽ kiểm tra giá trị của các triệu chứng và chọn triệu chứng có giá trị **Gini** hoặc **Entropy** giảm nhiều nhất. Dưới đây là cách phân chia:

⇨ Trước khi phân chia: Tính Gini hoặc Entropy của toàn bộ tập dữ liệu.

⇨ Sau khi phân chia: Chia tập dữ liệu thành hai nhóm con (ví dụ: có triệu chứng "fever" và không có triệu chứng "fever") và tính Gini hoặc Entropy cho mỗi nhóm con. CART sẽ tính *IR (Impurity Reduction)*, là sự giảm bất thuần *(impurity)* giữa trước và sau khi phân chia.

Công thức tính IR (Impurity Reduction):

$$IR = Impurity_{Before} - Impurity_{After}$$

⇨ CART sẽ chọn các phân chia có IR lớn nhất, tức là giảm độ bất thuần nhiều nhất.

❖ **Quá trình hình thành cây quyết định:**

Giả sử bạn có dữ liệu đầu vào là các triệu chứng và bệnh, với cột triệu chứng là các giá trị phân loại (*categorical values*). Cây quyết định CART sẽ:

⇨ **Tính toán Entropy và Gini** cho toàn bộ tập dữ liệu (trước khi phân chia). (Các kết quả có thể xem ở file **symptom_metrics_results.csv**)

⇨ **Chọn một triệu chứng** để chia dữ liệu, ví dụ triệu chứng "fever".

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Chia dữ liệu thành hai nhóm:

- Nhóm có triệu chứng “fever”.
- Nhóm không có triệu chứng “fever”.

⇨ **Tính toán lại Entropy và Gini cho từng nhóm con** (sau khi phân chia).

⇨ **Chọn triệu chứng có sự giảm Entropy hoặc Gini lớn nhất** (tức là sự giảm bất thuần lớn nhất) để chia tiếp.

Quá trình này tiếp tục cho đến khi:

⇨ Cây đạt độ sâu tối đa (max depth).

⇨ Hoặc mỗi nhánh con chỉ chứa một lớp bệnh duy nhất (không thể phân chia nữa).

❖ **Ví dụ minh họa cụ thể từ dữ liệu DiseaseAndSymptoms.csv**:

Symptom_1	**Symptom_2**	**Disease**
Fever	Headache	Flu
Cough	Fever	Cold
Fever	Headache	Flu
Headache	Cough	Cold
Fever	Headache	Flu

Cây quyết định sẽ bắt đầu với việc tính **Entropy** hoặc **Gini** cho toàn bộ dữ liệu. Sau đó, nó sẽ thử chia dữ liệu theo các triệu chứng (ví dụ: “fever”, “headache”, “cough”) để tìm ra triệu chứng nào mang lại sự giảm Entropy hoặc Gini lớn nhất.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Trước khi phân chia: Tính Entropy hoặc Gini cho toàn bộ tập dữ liệu.

Sau khi phân chia theo triệu chứng "fever":

- Nhóm 1 (có triệu chứng fever): [Flu, Flu, Flu] -> Entropy hoặc Gini thấp.
- Nhóm 2 (không có triệu chứng fever): [Cold, Cold] -> Entropy hoặc Gini thấp.

Cây quyết định sẽ tính **IR (Impurity Reduction)** cho các nhánh này. Nếu việc phân chia này làm giảm đáng kể Entropy hoặc Gini, "fever" sẽ được chọn làm tiêu chí chia.

2.3. Huấn Luyện Mô Hình:

- Với dữ liệu đầu vào là file **DiseaseAndSymptoms.csv** và đã được mã hóa ở bước chuẩn hóa dữ liệu.
- Sau đó chia dữ liệu thành 2 phần là **X (features)** và **Y (labels)** với:

  X chứa các cột triệu chứng đã mã hóa.

  Y chứa các nhãn bệnh đã mã hóa.
- Chia dữ liệu thành tập huấn luyện và kiểm tra: là một bước quan trọng trong quy trình huấn luyện để huấn luyện mô hình.
  - ⇨ **Tập huấn luyện (training set)** sẽ được sử dụng để huấn luyện mô hình.
  - ⇨ **Tập kiểm tra (test set)** sẽ được dùng để đánh giá hiệu quả của mô hình sau khi huấn luyện.
- Quy trình này sử dụng **train_test_split** từ **sklearn** để chia dữ liệu thành 80% cho tập huấn luyện và 20% cho tập kiểm tra. Việc này giúp kiểm tra độ tổng quát của mô hình và tránh hiện tượng overfitting.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

- ❖ Sau khi chia xong dữ liệu, chúng ta sử dụng thuật toán Decision Tree để huấn luyện mô hình. Cụ thể, mô hình sử dụng thuật toán CART (Classification and Regression Tree) để xây dựng cây quyết định, phân loại bệnh dựa trên triệu chứng đầu vào.
- ❖ Các tham số chính của mô hình bao gồm:
  - ⇨ *max_depth:* Giới hạn chiều sâu tối đa của cây quyết định. Việc này giúp tránh overfitting bằng cách không để cây quá phức tạp.
  - ⇨ *min_samples_split:* Quy định số lượng mẫu tối thiểu cần có để tiếp tục phân chia một nút.
  - ⇨ *min_samples_leaf:* Quy định số lượng mẫu tối thiểu ở mỗi lá cây (nút cuối cùng).

  Với cái tham số được sử dụng trong model lần lượt là: 15, 2, 2.
  - ⇨ Có thể tham khảo fle **Disease_and_Symptoms.py** để hiểu rõ hơn về mô hình.
- ❖ Khi huấn luyện, mô hình sẽ tự động tạo ra các phân nhánh dựa trên các triệu chứng (tính năng) sao cho mỗi nhánh dẫn đến các kết quả (bệnh) cụ thể. Cây quyết định được tạo ra dựa trên việc lựa chọn triệu chứng nào sẽ được chia trước tiên để giảm thiểu độ "impurity" (sự không thuần nhất) trong dữ liệu.

3. Dự đoán bệnh và phòng ngừa:

3.1. Dự đoán bệnh từ các triệu chứng:

Cây quyết định (CART) sử dụng các triệu chứng mà người dùng nhập vào để dự đoán bệnh. Cơ chế hoạt động của thuật toán như sau:

Quá trình ra quyết định trong cây quyết định:

- ❖ **Đầu vào:** Các triệu chứng như “itching”, “skin_rash”, “dischromic _patches”, ... được nhập vào dưới dạng dữ liệu phân loại (categorical data) (hoặc đã được mã hóa thành các giá trị số nếu sử dụng Label Encoding).

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

- **Chuyển đổi dữ liệu:** Các triệu chứng này được mã hóa thành các giá trị số để cây quyết định có thể xử lý (sử dụng LabelEncoder trong mô hình).
- **Cấu trúc cây quyết định:**
  - Mỗi nút trong cây đại diện cho một đặc trưng (triệu chứng), tức là một triệu chứng cụ thể mà mô hình quyết định có thể phân loại.
  - Các nhánh cây mô tả các phân nhánh dựa trên các giá trị của triệu chứng. Ví dụ, nếu triệu chứng là "skin_rash", cây sẽ quyết định xem bệnh có thể là gì dựa trên giá trị của triệu chứng này.
  - Các lá của cây đại diện cho kết quả dự đoán cuối cùng, tức là bệnh mà mô hình đưa ra.
- **Thuật toán CART** sử dụng tiêu chí phân chia (splitting criteria) như Gini Index hoặc Entropy để quyết định cách phân chia dữ liệu tại mỗi nút trong cây.
  - Gini Index đo lường độ phân tán của dữ liệu tại mỗi phân nhánh và lựa chọn các đặc trưng phân chia giúp giảm thiểu sự không đồng nhất trong nhóm.
  - Entropy là một tiêu chí khác giúp đo lường độ không chắc chắn trong nhóm. CART sẽ chọn đặc trưng có giá trị Entropy thấp nhất.
- **Ra quyết định dựa trên cây:**
  - Khi dữ liệu (triệu chứng) được đưa vào mô hình, cây quyết định sẽ di chuyển từ gốc cây xuống các lá, kiểm tra từng triệu chứng tại mỗi nút cây và phân loại bệnh cho đến khi gặp lá của cây.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

⇨ Ví dụ: Nếu triệu chứng là “itching” và cây quyết định xác định đây là triệu chứng chủ yếu liên quan đến bệnh “Allergy”, mô hình sẽ dự đoán bệnh là “Allergy” và đưa ra các biện pháp phòng ngừa cho bệnh này.

❖ **Cơ chế dự đoán:**

⇨ Tiêu chí lựa chọn triệu chứng: Mô hình dựa vào chỉ số Gini hoặc Entropy để chọn triệu chứng tốt nhất tại mỗi bước. Những triệu chứng giúp chia dữ liệu thành các nhóm đồng nhất nhất sẽ được ưu tiên.

⇨ **Ví dụ minh họa**:

Nếu đầu vào là các triệu chứng như “itching”, “skin rash”, mô hình sẽ bắt đầu từ nút gốc, kiểm tra các điều kiện để quyết định rẽ nhánh. Nếu triệu chứng “itching” làm giảm Gini Index nhiều nhất, nó sẽ được chọn trước. Tiếp tục, các triệu chứng khác sẽ được đánh giá để hoàn thiện dự đoán.

3.2. Cung cấp biện pháp phòng ngừa:

Sau khi cây quyết định đưa ra dự đoán về bệnh, bước tiếp theo là cung cấp các biện pháp phòng ngừa cho bệnh đó. Quá trình này như sau:

❖ **Tra cứu thông tin phòng ngừa:**

Trong dữ liệu của bạn, mỗi bệnh đều có một tập các biện pháp phòng ngừa đi kèm. Sau khi mô hình dự đoán được bệnh (ví dụ: “Allergy”), ta sẽ tra cứu biện pháp phòng ngừa tương ứng từ bộ dữ liệu **Disease precaution.csv**.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Biện pháp phòng ngừa:**

Những biện pháp này có thể bao gồm các hành động như dùng thuốc, thay đổi thói quen sống, tránh tiếp xúc với các yếu tố gây bệnh, v.v. Ví dụ: Nếu bệnh dự đoán là "Allergy", biện pháp phòng ngừa có thể là "sử dụng thuốc chống dị ứng", "tránh tiếp xúc với phấn hoa", "giữ môi trường sống sạch sẽ", ...

❖ **Cung cấp thông tin cho người dùng:**

Sau khi tra cứu các biện pháp phòng ngừa cho bệnh, mô hình trả lại một danh sách biện pháp phòng ngừa cho người dùng. Các biện pháp này giúp người bệnh có thể chủ động hơn trong việc ngăn ngừa các triệu chứng tái phát hoặc kiểm soát bệnh tốt hơn.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Chương 3: Kết quả thực nghiệm và phân tích đánh giá:

1. Phân tích kết quả từ Quá trình huấn luyện Mô hình (80% Dữ liệu):

1.1. Hiểu biết về mô hình:

- Cây quyết định (Decision Tree) xây dựng dựa trên thuật toán CART, sử dụng các chỉ số như Entropy hoặc Gini Index để đánh giá chất lượng phân chia dữ liệu. Điều này đảm bảo rằng mỗi nhánh trong cây sẽ tối ưu hóa độ thuần khiết (impurity) của dữ liệu sau khi phân chia.
- Trong bài toán này, **tập dữ liệu triệu chứng** được mã hóa và chuẩn hóa thành dạng số. Quá trình xây dựng cây dựa trên việc chọn triệu chứng tốt nhất tại mỗi bước để phân chia dữ liệu, từ đó giảm độ bất thuần và tối ưu hóa dự đoán.
- **Quá trình huấn luyện:**

⇨ Dữ liệu được phân chia thành tập huấn luyện (80%) và kiểm tra (20%).

⇨ Mô hình được thiết lập với các tham số tối ưu như *max_depth = 15, min_samples_split = 5, min_samples_leaf = 2*.

⇨ Trong mỗi bước, thuật toán xác định triệu chứng tốt nhất để giảm độ bất thuần, dựa vào chỉ **số Gini hoặc Entropy**, và xây dựng cấu trúc nhánh cây.

- Mô hình có khả năng tự động lựa chọn các triệu chứng quan trọng để tối ưu hóa kết quả.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

1.2. Hiệu suất huấn luyện:

1.2.1. Độ chính xác (accuracy_score): là tỷ lệ giữa số lượng dự đoán đúng của mô hình và tổng số mẫu.

❖ **Công thức:**

$$Accuracy = \frac{Số\ dự\ đoán\ đúng}{Tổng\ số\ mẫu}$$

❖ **Ý nghĩa:**

⇨ Độ chính xác cao thể hiện mô hình hoạt động tốt trong việc phân loại chính xác các nhãn.

⇨ Tuy nhiên, với dữ liệu không cân bằng, độ chính xác có thể không phải là thước đo tốt nhất.

❖ **Độ chính xác của mô hình** là 0.9989837398373984

⇨ Có thể tham khảo ***Hình 5***

1.2.2. Báo cáo phân loại (classification_report): Báo cáo phân loại bao gồm các chỉ số đánh giá chính như **Precision, Recall, và F1-Score** cho từng lớp nhãn.

❖ **Các chỉ số chính:**

⇨ Precision: Tỷ lệ mẫu được dự đoán đúng trên tổng số mẫu được dự đoán vào lớp đó.

⇨ Recall: Tỷ lệ mẫu được dự đoán đúng trên tổng số mẫu thực tế thuộc lớp đó.

⇨ F1-Score: Trung bình điều hòa giữa Precision và Recall.

❖ **Macro Average (macro avg):**

⇨ Là giá trị trung bình của Precision, Recall, và F1-Score được tính riêng biệt cho từng lớp, sau đó lấy trung bình cộng.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

- ⇨ Macro avg không quan tâm đến kích thước của từng lớp, vì vậy mỗi lớp đóng góp trọng số bằng nhau. Nếu có lớp dữ liệu nhỏ hoặc không cân bằng, macro avg sẽ thể hiện rõ mức độ chênh lệch.

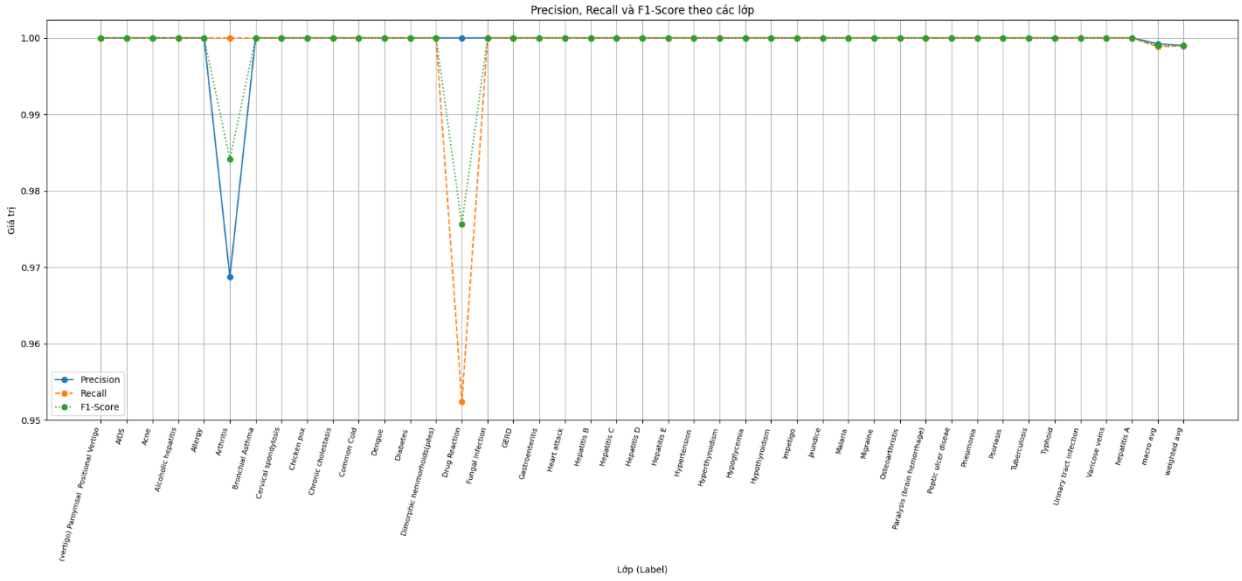
❖ **Weighted Average (weighted avg):**

- ⇨ Là giá trị trung bình của Precision, Recall, và F1-Score, nhưng có trọng số theo kích thước (số lượng mẫu) của từng lớp.
- ⇨ Weighted avg phản ánh mức độ hiệu suất tổng thể của mô hình dựa trên mức độ quan trọng của các lớp (dựa vào số lượng mẫu trong lớp đó). Điều này phù hợp nếu mô hình đang xử lý dữ liệu không cân bằng.

❖ **Ý nghĩa:** Cho cái nhìn tổng quan chi tiết về hiệu suất mô hình trên từng nhãn cụ thể, đặc biệt hữu ích với dữ liệu nhiều lớp.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Báo cáo phân loại của mô hình:** Xem chi tiết ở file **classification_report.csv**



Hình 7: Biểu đồ biểu diễn 3 thuộc tính của báo cáo phân loại

❖ **Nhận xét:**

⇨ **Tính ổn định:** Biểu đồ cho thấy các giá trị Precision, Recall và F1-Score của hầu hết các lớp (labels) duy trì ở mức gần 1 (hoàn hảo), thể hiện mô hình có khả năng phân loại tốt trên nhiều lớp khác nhau.

⇨ **Hiệu suất thấp hơn ở một số lớp:** Một số lớp như **Arthritis, Fungal Infection** có sự giảm sút đáng kể về Precision, Recall và F1-Score. Điều này chỉ ra rằng mô hình gặp khó khăn trong việc phân loại đúng các bệnh này, có thể do dữ liệu không đủ đại diện hoặc các triệu chứng của các lớp này bị trùng lặp nhiều với các lớp khác.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

- ⇨ **Các lớp trung bình (macro avg, weighted avg):** Các giá trị trung bình (macro avg, weighted avg) duy trì ở mức rất cao, thể hiện mô hình tổng thể hoạt động tốt. Điều này cũng phản ánh mô hình không bị lệch quá mức bởi một số lớp cụ thể.
- ⇨ **Những lớp có hiệu suất tốt:** Các lớp như **Dengue**, **Diabetes**, **Typhoid**, **Malaria**, và các bệnh phổ biến khác có Precision, Recall, và F1-Score rất cao. Điều này cho thấy rằng mô hình có thể dễ dàng phân biệt và dự đoán đúng các bệnh này, có thể do chúng có đặc điểm triệu chứng đặc trưng hoặc dữ liệu đại diện tốt hơn.
- ⇨ **Nguyên nhân của sự sai lệch:** Các lớp như **Arthritis và Fungal Infection** có thể có dữ liệu triệu chứng không đặc trưng rõ ràng hoặc phân phối dữ liệu không cân bằng ngay cả khi đã sử dụng SMOTE. Điều này có thể dẫn đến khó khăn trong việc dự đoán.

1.2.3. Độ chính xác trên tập kiểm định (Accuracy on Validation set):

- ❖ **Mục đích:**
  - ⇨ Đánh giá khả năng tổng quát hóa của mô hình trên dữ liệu chưa từng gặp trong quá trình huấn luyện.
  - ⇨ So sánh độ chính xác trên tập huấn luyện và tập kiểm định giúp phát hiện hiện tượng overfitting hoặc underfitting.
- ❖ **Ý nghĩa:** Độ chính xác cao trên tập kiểm định cho thấy mô hình đã học tốt từ dữ liệu huấn luyện và áp dụng hiệu quả trên dữ liệu mới.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

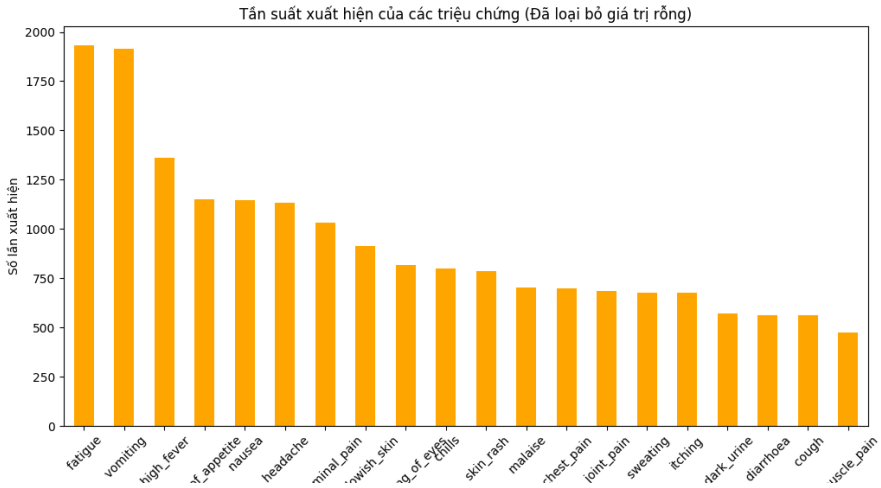
- **Độ chính xác trên tập kiểm định của mô hình là:** 0.9949238578680203
- **Nhận xét:**

Độ chính xác trên tập validation đạt **99.5%**, rất gần với độ chính xác trên tập kiểm tra (**99.9%**). Điều này cho thấy mô hình tổng quát hóa tốt và không bị **overfitting** nghiêm trọng.

Sự khác biệt nhỏ giữa hai giá trị này là chấp nhận được và chứng tỏ mô hình có thể hoạt động ổn định trên dữ liệu mới không được huấn luyện.

1.2.4. Biểu đồ tần suất Triệu chứng (Đã loại bỏ Giá trị rỗng):

- **Mục đích:** Hiển thị tần suất xuất hiện của từng triệu chứng trong tập dữ liệu.
- **Ý nghĩa:**
  - ⇨ Triệu chứng có tần suất cao thường chiếm ưu thế trong mô hình.
  - ⇨ Giúp nhận biết các triệu chứng phổ biến và triệu chứng hiếm gặp, từ đó đánh giá độ cân bằng trong dữ liệu.
- **Biểu đồ tần suất Triệu chứng của mô hình:**



Hình 6. Biểu đồ tần suất xuất hiện các bệnh (đã được loại bỏ giá trị NaN)

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Nhận xét:**

⇨ **Phân bố tần suất triệu chứng:**

- Triệu chứng “fatigue” (mệt mỏi) và “vomiting” (nôn mửa) xuất hiện với tần suất cao nhất, gần 2000 lần. Điều này có thể cho thấy chúng là các triệu chứng phổ biến nhất trong dữ liệu.
- Các triệu chứng như “high_fever” (sốt cao), “loss_of_appetite” (mất cảm giác thèm ăn), “nausea” (buồn nôn) cũng có tần suất tương đối cao, nằm trong nhóm triệu chứng phổ biến.

⇨ **Triệu chứng ít xuất hiện hơn:** Các triệu chứng như “muscle_pain” (đau cơ), “cough” (ho) có tần suất thấp hơn đáng kể. Điều này có thể phản ánh rằng các triệu chứng này chỉ liên quan đến một số bệnh cụ thể trong dữ liệu.

⇨ **Mối quan hệ với chất lượng dự đoán:**

- Các triệu chứng phổ biến hơn như “fatigue” và “vomiting” thường dễ góp phần vào quá trình dự đoán bệnh chính xác hơn, vì chúng cung cấp nhiều thông tin trong mô hình.
- Ngược lại, các triệu chứng ít xuất hiện hơn có thể làm tăng độ không chắc chắn trong mô hình, đặc biệt khi chúng không được liên kết mạnh mẽ với bất kỳ bệnh nào.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

⇨ **Ý nghĩa thực tế:**

- Tính phổ biến: Các triệu chứng phổ biến có thể là dấu hiệu cho thấy dữ liệu tập trung vào một số bệnh hoặc điều kiện y tế cụ thể.
- Dữ liệu bổ sung: Cần đảm bảo rằng các triệu chứng ít xuất hiện vẫn được xử lý và huấn luyện đầy đủ trong mô hình để tránh việc bỏ sót các dự đoán liên quan đến chúng.

1.2.5. Biểu đồ Phân phối Số lượng mẫu của các Bệnh:

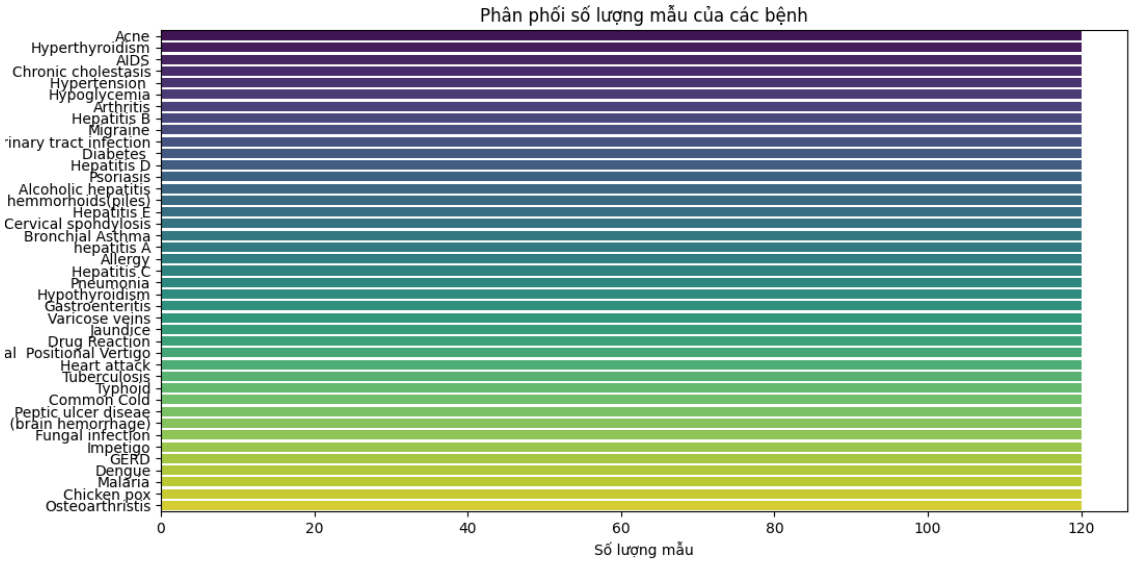
❖ **Mục đích:** Hiển thị số lượng mẫu của từng loại bệnh trong tập dữ liệu.

❖ **Ý nghĩa:**

⇨ Nhận biết dữ liệu có cân bằng giữa các nhãn bệnh không.

⇨ Nhãn bệnh ít dữ liệu dễ gây nhầm lẫn, có thể cần bổ sung hoặc sử dụng kỹ thuật cân bằng dữ liệu như SMOTE.

❖ **Biểu đồ Phân phối Số lượng mẫu của các Bệnh:**



Hình 2. Biểu đồ phân phối số lượng mẫu bệnh theo ***DiseaseAndSymptoms.csv***

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Nhận xét:**

⇨ **Phân phối đồng đều:** Biểu đồ cho thấy số lượng mẫu của các bệnh được phân bố tương đối đồng đều, với số mẫu dao động từ 100 đến hơn 120 mẫu cho mỗi bệnh. Điều này giúp giảm thiểu vấn đề dữ liệu không cân bằng trong quá trình huấn luyện mô hình.

⇨ **Đảm bảo chất lượng dự đoán:** Với sự phân phối đồng đều như vậy, mô hình có thể học tốt từ các bệnh mà không bị thiên lệch về bất kỳ lớp nào. Điều này góp phần cải thiện độ chính xác tổng thể và độ tin cậy của mô hình.

⇨ **Ý nghĩa thực tế:** Phân phối dữ liệu như trên cho thấy rằng tập dữ liệu đã được chuẩn bị tốt, có tính đại diện cho từng loại bệnh. Điều này giúp mô hình dự đoán chính xác hơn trong các tình huống thực tế khi đối mặt với các bệnh đa dạng.

⇨ **Khả năng mở rộng:** Với số lượng mẫu tương đối đồng đều, tập dữ liệu có thể dễ dàng mở rộng bằng cách bổ sung thêm các bệnh hoặc triệu chứng mới mà không làm ảnh hưởng đến cân bằng dữ liệu hiện tại.

1.2.6. Ma trận nhầm lẫn đa lớp (Multiclass Confusion Matrix):

❖ **Định nghĩa:**

⇨ Ma trận nhầm lẫn hiển thị số lượng mẫu được dự đoán đúng và nhầm lẫn giữa các lớp.

⇨ Mỗi hàng trong ma trận biểu diễn số mẫu thực tế thuộc một lớp cụ thể.

⇨ Mỗi cột biểu diễn số mẫu được dự đoán là thuộc một lớp cụ thể.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Các thành phần chính:**

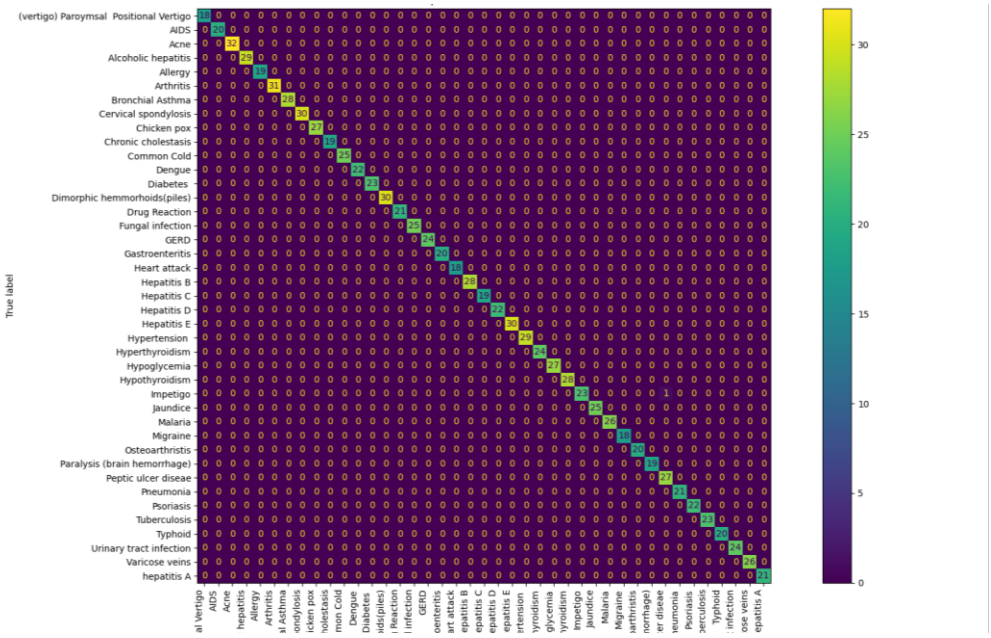
- ⇨ True Positive (TP): Số lượng mẫu được mô hình dự đoán đúng thuộc lớp cụ thể (giá trị trên đường chéo).
- ⇨ False Positive (FP): Số lượng mẫu bị dự đoán nhầm vào lớp cụ thể từ các lớp khác (tổng cột trừ đi giá trị TP).
- ⇨ False Negative (FN): Số lượng mẫu thực sự thuộc lớp cụ thể nhưng bị dự đoán nhầm sang lớp khác (tổng hàng trừ đi giá trị TP).
- ⇨ True Negative (TN): Số lượng mẫu không thuộc lớp cụ thể và không bị dự đoán nhầm vào lớp đó (tổng số mẫu trừ tổng của hàng và cột cho lớp đó, cộng lại giá trị TP).

❖ **Ý nghĩa:**

- ⇨ Hiển thị hiệu suất tổng thể: Cho thấy mô hình phân loại chính xác như thế nào cho từng lớp.
- ⇨ Nhận diện các nhãn thường xuyên bị nhầm lẫn: Ví dụ, từ ma trận nhầm lẫn của bạn, có thể thấy các lớp có triệu chứng tương tự thường bị nhầm lẫn với nhau.
- ⇨ Đánh giá hiệu suất riêng lẻ của từng lớp: Mỗi lớp có thể có các chỉ số Precision, Recall, và F1-Score khác nhau, từ đó giúp hiểu rõ hơn về nhược điểm của mô hình.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Ma trận nhầm lẫn của mô hình:**



Hình 8. Ma trận nhầm lẫn đa lớp của mô hình.

❖ **Nhận xét:**

⇨ **Tổng quan:**

- Ma trận nhầm lẫn là công cụ hữu ích để đánh giá chất lượng mô hình trong các bài toán phân loại.
- Các ô chính trên đường chéo đại diện cho số lượng mẫu được dự đoán đúng cho từng lớp.
- Các ô ngoài đường chéo thể hiện số lượng mẫu bị dự đoán nhầm lẫn sang lớp khác.

⇨ **Quan sát:**

- Đa phần các ô trên ma trận tập trung vào đường chéo chính, điều này chứng tỏ mô hình có độ chính xác cao, phần lớn các mẫu được phân loại đúng.
- Các lớp bệnh như “Acne”, “Arthritis”, và “Cervical spondylosis” có số lượng dự đoán đúng cao (ví dụ: 32, 31, 30 mẫu tương ứng).

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

- Một số ô ngoài đường chéo có giá trị nhỏ, ví dụ: "Hypothyroidism" bị nhầm lẫn với "Diabetes". Điều này có thể do các triệu chứng của các bệnh này có sự tương đồng.

⇨ **Những phát hiện quan trọng:**

- **Tỷ lệ nhầm lẫn thấp:** Các ô ngoài đường chéo có giá trị rất thấp, cho thấy mô hình phân loại chính xác trong hầu hết các trường hợp.
- **Đồng nhất dữ liệu:** Các giá trị không đồng đều giữa các lớp, phản ánh sự phân bố dữ liệu ban đầu trong tập dữ liệu.
- **Các bệnh khó phân biệt:** Một số cặp bệnh có sự nhầm lẫn lẫn nhau, điều này có thể do mô hình gặp khó khăn trong việc phân biệt các triệu chứng tương tự giữa các lớp.

1.2.7. Biểu đồ tầm quan trọng của triệu chứng:

❖ **Định nghĩa:** Chỉ số tầm quan trọng đo lường mức độ đóng góp của mỗi triệu chứng trong quá trình xây dựng cây quyết định.

❖ **Ý nghĩa:**

⇨ Trục hoành (X-axis): Đại diện cho mức độ quan trọng của từng triệu chứng đối với mô hình quyết định (Decision Tree).

⇨ Trục tung (Y-axis): Liệt kê các triệu chứng được sử dụng trong bài toán phân loại bệnh.

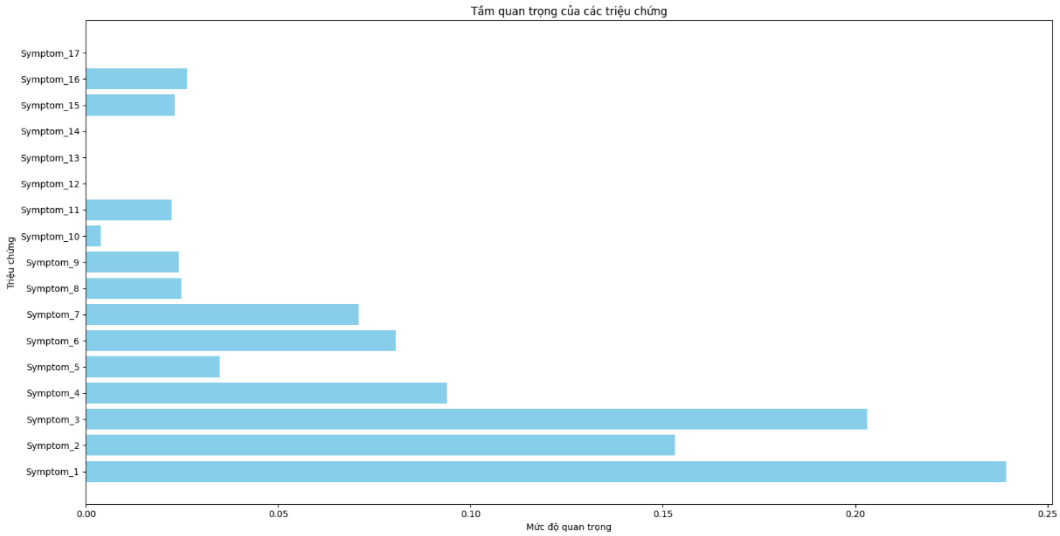
⇨ Mức độ quan trọng cho biết ảnh hưởng của từng triệu chứng đến việc mô hình phân biệt giữa các lớp bệnh.

⇨ Giúp xác định các triệu chứng có ảnh hưởng lớn đến quá trình phân loại.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

⇨ Các triệu chứng quan trọng thường được chọn ở các nút cao trong cây quyết định.

❖ **Biểu đồ tầm quan trọng của triệu chứng của mô hình:**



Hình 1. Biểu đồ tầm quan trọng của của triệu chứng trong dự đoán bệnh của mô hình.

❖ **Nhận xét:**

⇨ **Triệu chứng có mức độ quan trọng cao:**

- *Symptom_1, Symptom_4, và Symptom_2* có mức độ quan trọng lớn nhất, cho thấy chúng đóng vai trò rất lớn trong việc dự đoán bệnh.
- Những triệu chứng này có thể liên quan chặt chẽ đến nhiều bệnh hoặc có giá trị phân biệt cao giữa các lớp bệnh.

⇨ Triệu chứng có mức độ quan trọng thấp:

- *Symptom_11 và Symptom_10* có mức độ quan trọng gần như không đáng kể. Điều này có thể do:

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

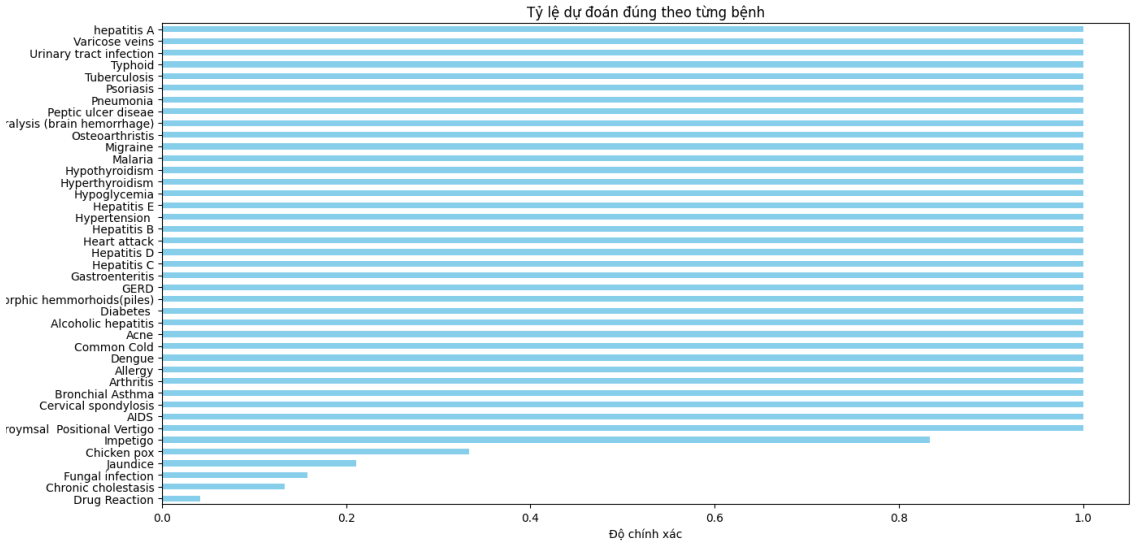
- Chúng không đặc trưng hoặc không liên quan nhiều đến các bệnh trong tập dữ liệu.
- Sự trùng lặp hoặc thiếu thông tin trong các triệu chứng này.

⇨ Phân bổ không đồng đều: Một số triệu chứng có mức độ quan trọng vượt trội so với các triệu chứng khác, điều này thể hiện rằng mô hình chủ yếu dựa vào một số yếu tố quan trọng nhất thay vì phân bổ đều trọng số.

2. Phân tích kết quả từ Quá trình kiểm tra Mô hình (20% Dữ liệu):

2.1. Tỷ lệ dự đoán đúng theo từng bệnh:

2.1.1. Biểu đồ tỉ lệ dự đoán đúng theo từng bệnh của mô hình (giai đoạn kiểm tra vớ 20% dữ liệu):



Hình 9. Biểu đồ dự đoán đúng theo từng bệnh của mô hình (giai đoạn kiểm tra với 20% dữ liệu).

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

2.1.2.Tổng quan:

- ❖ Biểu đồ hiển thị tỷ lệ dự đoán đúng (Accuracy) cho từng bệnh.
- ❖ Hầu hết các bệnh phổ biến như Hepatitis A, Typhoid, Psoriasis, và Malaria đạt tỷ lệ dự đoán đúng cao (gần 100%).
- ❖ Một số bệnh có tỷ lệ dự đoán thấp, như:
  - ⇨ Drug Reaction.
  - ⇨ Fungal Infection.
  - ⇨ Chronic Cholestasis.

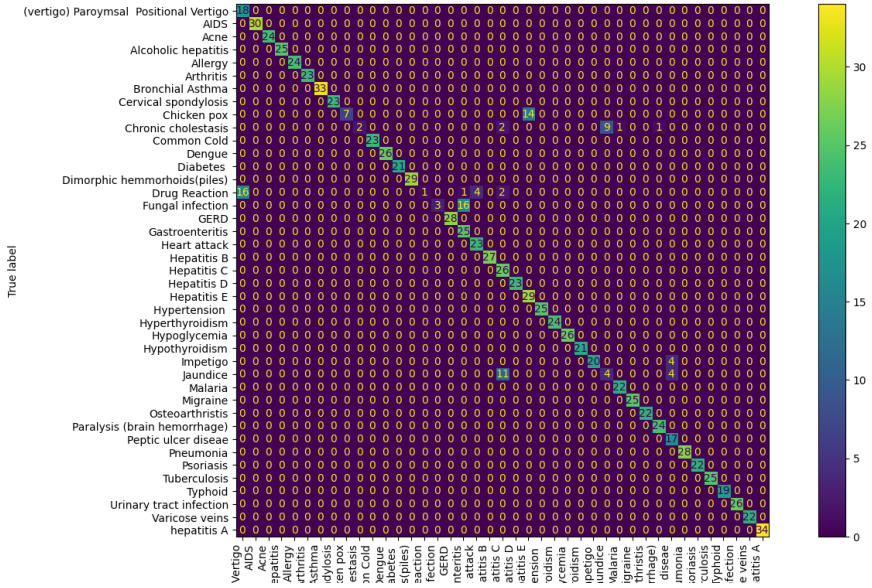
2.1.3. Phân tích chi tiết:

- ❖ **Drug Reaction:**
  - ⇨ Tỷ lệ chính xác rất thấp (gần 0%). Điều này có thể do số lượng mẫu quá ít hoặc triệu chứng không đủ đặc trưng để phân biệt.
  - ⇨ Triệu chứng thường gặp như Rash hoặc Itching dễ gây nhầm lẫn với các bệnh da liễu khác.
- ❖ **Fungal Infection:**
  - ⇨ Đạt tỷ lệ chính xác dưới 50%. Nguyên nhân có thể đến từ triệu chứng như Yellowish_skin và High_fever, dễ trùng lặp với các bệnh gan như Hepatitis E.
- ❖ **Chronic Cholestasis:**
  - ⇨ Tỷ lệ chính xác thấp. Điều này có thể do các triệu chứng chồng chéo với Hepatitis C, ví dụ Yellowish_skin.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

2.2. Ma trận nhầm lẫn đa lớp:

2.2.1. Ma trận nhầm lẫn đa lớp của mô hình (giai đoạn kiểm tra với 20% dữ liệu):



Hình 1. Ma trận nhầm lẫn đa lớp của mô hình (giai đoạn kiểm tra với 20% dữ liệu)

2.2.2. Tổng quan:

- ❖ Ma trận nhầm lẫn cho thấy số lượng dự đoán đúng (giá trị trên đường chéo) và dự đoán sai (các ô ngoài đường chéo).
- ❖ Một số bệnh có nhầm lẫn nổi bật:
  - ⇨ Drug Reaction bị nhầm với Paroymsal Positonal Vertigo, Heart Attack .
  - ⇨ Fungal Infection bị nhầm với Gastroenteritis.
  - ⇨ Chronic Cholestasis nhầm với Jaundice.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

2.2.3. Phân tích chi tiết:

- **Drug Reaction:**
  - ⇨ Dự đoán đúng: 1 mẫu.
  - ⇨ Nhầm lẫn:
    - Với (Vertigo) Paroxysmal Positional Vertigo: 16 mẫu.
    - Với Heart Attack: 4 mẫu.
    - Với Hepatitis C: 2 mẫu.
  - ⇨ Lý do nhầm lẫn:
    - Triệu chứng chung: Một số triệu chứng của Drug Reaction như Fever, Nausea, và Vomiting xuất hiện ở nhiều bệnh khác, đặc biệt là (Vertigo) và Heart Attack.
    - Dữ liệu mất cân bằng: Drug Reaction có số lượng mẫu rất thấp, làm mô hình không đủ khả năng học được các đặc trưng riêng biệt của bệnh.
    - Hạn chế trong đặc trưng: Các triệu chứng đầu vào không đủ mạnh để phân biệt Drug Reaction với các bệnh khác, đặc biệt khi chúng có triệu chứng giống nhau.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Fungal Infection:**

⇨ Dự đoán đúng: 3 mẫu.

⇨ Nhầm lẫn:

- Với Gastroenteritis: 16 mẫu.

⇨ Lý do nhầm lẫn:

- Triệu chứng trùng lặp: Fungal Infection và Gastroenteritis đều có triệu chứng như Abdominal Pain, Diarrhea, và Fever. Điều này khiến mô hình khó phân biệt giữa hai bệnh.
- Thiếu triệu chứng đặc trưng: Fungal Infection cần các triệu chứng khác biệt hơn, như Skin Rash hoặc các xét nghiệm cụ thể, nhưng không có trong dữ liệu.
- Số lượng mẫu không đồng đều: Gastroenteritis có số mẫu vượt trội hơn so với Fungal Infection, làm lệch độ chính xác của mô hình.

❖ **Chicken Pox:**

⇨ Dự đoán đúng: 7 mẫu.

⇨ Nhầm lẫn:

- Với Hepatitis E: 5 mẫu.

⇨ Lý do nhầm lẫn:

- Triệu chứng chung: Triệu chứng như Fever và Skin Rash xuất hiện ở cả hai bệnh, làm mô hình dễ nhầm lẫn giữa Chicken Pox và Hepatitis E.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

- Không đủ dữ liệu phân biệt: Dữ liệu hiện tại không đủ để mô hình phân biệt giữa hai bệnh dựa trên các triệu chứng hiện có.

❖ **Chronic Cholestasis:**

⇨ Dự đoán đúng: 2 mẫu.

⇨ Nhầm lẫn:

- Với Hepatitis C: 2 mẫu.
- Với Malaria: 1 mẫu.
- Với Jaundice: 9 mẫu.
- Với Paralysis (Brain Hemorrhage): 1 mẫu.

⇨ Lý do nhầm lẫn:

- Triệu chứng phổ biến: Các triệu chứng như Yellowish Skin, Dark Urine, và Nausea rất giống nhau ở các bệnh liên quan đến gan như Hepatitis C, Jaundice, và Chronic Cholestasis.
- Sự chồng lấn triệu chứng: Mô hình không thể phân biệt rõ khi nhiều bệnh chia sẻ các triệu chứng chính.
- Dữ liệu mất cân bằng: Số lượng mẫu của Chronic Cholestasis thấp hơn so với các bệnh khác, khiến độ chính xác thấp.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

❖ **Jaundice:**

⇨ Dự đoán đúng: 4 mẫu.

⇨ Nhầm lẫn:

- Với Hepatitis C: 11 mẫu.
- Với Peptic Ulcer Disease: 4 mẫu.

⇨ Lý do nhầm lẫn:

- Triệu chứng chung: Yellowish Skin, Fatigue, và Dark Urine là những triệu chứng điển hình của Jaundice nhưng cũng xuất hiện trong các bệnh như Hepatitis C.
- Hạn chế trong dữ liệu: Thiếu thông tin cụ thể hơn để mô hình có thể phân biệt được các nguyên nhân gây Yellowish Skin như viêm gan (Hepatitis) hoặc bệnh vàng da (Jaundice).

❖ **Nguyên nhân chính:**

⇨ **Triệu chứng trùng lặp:** Các bệnh có triệu chứng giống nhau như Drug Reaction và Vertigo, hoặc Jaundice và Hepatitis C, gây khó khăn cho mô hình phân loại.

⇨ **Mất cân bằng dữ liệu:** Các bệnh như Drug Reaction, Chronic Cholestasis, và Fungal Infection có số lượng mẫu thấp, dẫn đến độ chính xác thấp hơn cho những bệnh này.

⇨ **Thiếu triệu chứng đặc trưng:** Dữ liệu hiện tại không đủ các triệu chứng đặc trưng để mô hình phân biệt tốt giữa các bệnh, đặc biệt là những bệnh dễ nhầm lẫn như Jaundice, Hepatitis C, và Chronic Cholestasis.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

⇨ **Thiếu thông tin bổ sung:** Các xét nghiệm hoặc thông tin chuyên sâu khác (như kết quả xét nghiệm máu hoặc bệnh sử) có thể giúp mô hình cải thiện đáng kể khả năng phân loại.

2.3. Phân tích từ báo cáo (detailed_analysis_report.txt):

2.3.1. Nội dung báo cáo:

Xem ở file **detailed_analysis_report.txt**

2.3.2. Phân tích báo cáo:

❖ Độ chính xác tổng thể:

⇨ Độ chính xác của mô hình là 91.36%. Mặc dù mức này khá cao, nhưng vẫn có khoảng 8.64% dự đoán sai, tương ứng với 85 mẫu.

⇨ Điều này cho thấy mô hình hoạt động tốt, nhưng vẫn còn dư địa để cải thiện.

❖ Tỷ lệ dự đoán đúng theo từng bệnh:

⇨ Các bệnh có tỷ lệ dự đoán đúng tuyệt đối: Bao gồm các bệnh như:

- Paroymsal Positional Vertigo
- AIDS
- Acne
- Alcoholic hepatitis
- Bronchial Asthma

⇨ Các bệnh có tỷ lệ dự đoán thấp: Gồm:

- Drug Reaction: 4.17%
- Chronic cholestasis: 13.33%
- Fungal infection: 15.79%
- Jaundice: 21.05%

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

- ❖ Triệu chứng phổ biến trong các dự đoán sai:
  - ⇨ Các triệu chứng xuất hiện nhiều trong dự đoán sai:
    - o Yellowish_skin (107 lần).
    - o Skin_rash (51 lần).
    - o High_fever (32 lần).
    - o Fatigue (29 lần).
    - o Vomiting (28 lần).
  - ⇨ Đây là những triệu chứng phổ biến và không đặc trưng, dẫn đến sai sót trong phân loại.
- ❖ Phân phối dữ liệu theo bệnh:
  - ⇨ Các bệnh có số mẫu không đồng đều, dẫn đến mô hình không học tốt với những bệnh có ít dữ liệu.
  - ⇨ Chronic cholestasis chỉ có 15 mẫu, Peptic ulcer disease có 17 mẫu, trong khi Hepatitis A có 34 mẫu.
- ❖ Nguyên nhân tiềm ẩn sai sót:
  - ⇨ Dữ liệu không cân bằng: Một số bệnh có ít mẫu gây khó khăn trong việc học.
  - ⇨ Triệu chứng không đặc trưng: Các triệu chứng như Yellowish_skin xuất hiện ở nhiều bệnh, gây nhầm lẫn.
  - ⇨ Tương đồng về triệu chứng giữa các bệnh: Ví dụ: Drug Reaction và Vertigo có thể có triệu chứng tương tự nhau.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

3. Đánh Giá Mô Hình:

3.1. Ưu điểm:

Hiệu suất cao:

- Độ chính xác tổng thể đạt **91.36%,** cho thấy mô hình có khả năng dự đoán tốt trên tập dữ liệu kiểm tra.
- Một số bệnh được dự đoán chính xác tuyệt đối như AIDS, Acne, Alcoholic hepatitis, phản ánh mô hình hoạt động tốt với các bệnh có triệu chứng đặc trưng.

Khả năng xử lý nhiều bệnh: Mô hình có thể phân loại chính xác một danh sách dài các bệnh với triệu chứng đầu vào đa dạng.

Ma trận nhầm lẫn và phân tích chi tiết: Nhận diện rõ các cặp bệnh dễ nhầm lẫn, từ đó có cơ sở cải thiện mô hình hoặc quy trình thu thập dữ liệu.

3.2. Nhược điểm:

Nhầm lẫn ở các bệnh có triệu chứng tương đồng: Các bệnh như *Drug Reaction, Fungal Infection, Jaundice* thường xuyên bị nhầm lẫn với các bệnh khác do triệu chứng không đặc trưng.

Dữ liệu không cân bằng: Các bệnh như *Chronic cholestasis (15 mẫu), Peptic ulcer disease (17 mẫu)* có ít dữ liệu, dẫn đến tỷ lệ dự đoán sai cao.

Hạn chế trong phân tích triệu chứng phổ biến: Triệu chứng như *Yellowish_skin hoặc Skin_rash* xuất hiện ở nhiều bệnh gây giảm độ chính xác của mô hình.

Hạn chế của thuật toán Decision Tree: Cây quyết định có thể bị *overfitting*, đặc biệt với dữ liệu không cân bằng hoặc có nhiễu. Không phù hợp với các bệnh có triệu chứng phức tạp hoặc các mối quan hệ phi tuyến tính.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

3.3. Đề xuất cải thiện:

Thu thập và cân bằng dữ liệu:

- ❖ Tăng số lượng mẫu cho các bệnh có ít dữ liệu như Chronic cholestasis.
- ❖ Sử dụng các kỹ thuật oversampling nâng cao như ADASYN để cải thiện cân bằng dữ liệu.

Sử dụng mô hình nâng cao:

- ❖ Thay thế Decision Tree bằng Random Forest hoặc Gradient Boosting để tăng khả năng tổng quát hóa và giảm overfitting.
- ❖ Áp dụng các thuật toán học sâu (Deep Learning) để xử lý tốt hơn các triệu chứng không đặc trưng.

Tăng cường xử lý dữ liệu:

- ❖ Phân cụm các triệu chứng trước khi huấn luyện mô hình để giảm nhiễu và tăng đặc trưng.
- ❖ Áp dụng kỹ thuật chọn lọc đặc trưng (Feature Selection) để chỉ giữ lại các triệu chứng có ý nghĩa phân biệt cao.

Sử dụng biểu diễn triệu chứng tiên tiến: Thay vì mã hóa triệu chứng dạng số, có thể thử dùng các phương pháp embedding để biểu diễn triệu chứng dưới dạng vector liên tục, giúp mô hình hiểu rõ hơn mối quan hệ giữa các triệu chứng.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

PHẦN KẾT LUẬN

Bài toán "Ứng dụng thuật toán cây quyết định trong dự đoán và phòng ngừa bệnh" đã được giải quyết thành công, đáp ứng các mục tiêu đề ra. Với việc áp dụng thuật toán CART (Classification and Regression Tree), bài toán đạt được kết quả nổi bật trong việc xây dựng một hệ thống dự đoán bệnh từ triệu chứng và cung cấp các biện pháp phòng ngừa tương ứng. Hệ thống đã đạt được độ chính xác cao trên tập kiểm tra và mang lại nhiều thông tin chi tiết về hiệu suất mô hình thông qua các công cụ phân tích.

Hiệu suất mô hình:

- ❖ Mô hình đạt độ chính xác 91.36% trên tập kiểm tra, phản ánh năng lực tổng quát hóa tốt.
- ❖ Các bệnh như AIDS, Acne, Alcoholic hepatitis được dự đoán chính xác nhờ đặc điểm triệu chứng đặc trưng.

Phân tích kết quả:

- ❖ Phân tích chi tiết ma trận nhầm lẫn cho thấy một số cặp bệnh dễ nhầm lẫn như:
  - ⇨ Drug Reaction thường nhầm với Vertigo (16 mẫu) và Heart Attack (4 mẫu).
  - ⇨ Jaundice nhầm với Hepatitis C (11 mẫu) và Peptic Ulcer Disease (4 mẫu).
- ❖ Triệu chứng phổ biến gây nhầm lẫn bao gồm các biểu hiện như Yellowish_skin hoặc Skin_rash, xuất hiện ở nhiều bệnh khác nhau.

Hạn Chế:

- ❖ Dữ liệu không cân bằng:
  - ⇨ Một số bệnh hiếm gặp như Chronic cholestasis, Peptic ulcer disease có số lượng mẫu hạn chế, dẫn đến tỷ lệ dự đoán sai cao.
  - ⇨ Sự chênh lệch trong phân phối dữ liệu ảnh hưởng đến khả năng tổng quát hóa của mô hình.
- ❖ Triệu chứng gây nhầm lẫn: Nhiều triệu chứng không đủ đặc trưng, xuất hiện ở nhiều bệnh, gây khó khăn trong việc phân biệt.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

- ❖ Hạn chế từ thuật toán:
  - ⇨ Thuật toán CART có xu hướng overfitting khi dữ liệu không cân bằng hoặc chứa nhiều nhiễu.
  - ⇨ Khả năng xử lý các mối quan hệ phi tuyến tính và tương tác phức tạp giữa các triệu chứng còn hạn chế.

Bài toán đã chứng minh tính khả thi và hiệu quả của thuật toán CART trong dự đoán bệnh và phân tích các triệu chứng y tế. Tuy nhiên, để đạt hiệu quả cao hơn, cần thực hiện các cải tiến đã đề xuất nhằm nâng cao độ chính xác và khả năng ứng dụng trong các bài toán y tế phức tạp hơn, góp phần cải thiện chất lượng chăm sóc sức khỏe trong tương lai.

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

TÀI LIỆU THAM KHẢO

1. **Aurelien Geron**, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*, O'Reilly, 2019.

2. **GeeksforGeeks**, *"CART (Classification and Regression Tree) in Machine Learning"*, truy cập ngày 03 tháng 12 năm 2024, https://www.geeksforgeeks.org/cart-classification-and-regression-tree-in-machine-learning/

3. **Medium - GeekCulture**, *"Decision Trees with CART Algorithm"*, truy cập ngày 03 tháng 12 năm 2024, https://medium.com/geekculture/decision-trees-with-cart-algorithm-7e179acee8ff

4. **Kaggle**, *Disease and Symptoms Dataset*, https://www.kaggle.com/datasets/choongqianzheng/disease-and-symptoms-dataset

5. **Scikit-Learn Documentation**, *"DecisionTreeClassifier – User Guide"*, https://scikit-learn.org/stable/modules/tree.html

6. **Seaborn Documentation**, *"Statistical data visualization"*, https://seaborn.pydata.org/

7. **Pandas Documentation**, *"Advanced Data Analysis and Manipulation"*, https://pandas.pydata.org/docs/

8. **Matplotlib Documentation**, *"Visualization Techniques and Customization"*, https://matplotlib.org/stable/users/index.html

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms

9. **Scikit-Learn**, *"SMOTE Oversampling Technique – Implementation Guide"*, https://imbalanced-learn.org/stable/over_sampling.html#smote

10. **Joblib Documentation**, *"Efficient Serialization of Python Objects"*, https://joblib.readthedocs.io/

11. **Kaggle Discussion**, *"Improving Accuracy in Disease Prediction Models"*, https://www.kaggle.com/discussions/general/

12. **GitHub Repositories**, *"Machine Learning Projects for Disease Prediction"*, https://github.com/topics/disease-prediction

Đường dẫn đến Github chứa source code, dataset và các file được nhắc đến trong bài:
https://github.com/Killig3110/DiseaseAndSymptoms