|  |  |
| --- | --- |
|  | BỘ GIÁO DỤC VÀ ĐÀO TẠO  **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM** |

**BÁO CÁO ĐỒ ÁN**

**ỨNG DỤNG MÔ HÌNH CÂY QUYẾT ĐỊNH TRONG THUẬT TOÁN RANDOM FOREST ĐỂ CHẨN ĐOÁN VÀ HỖ TRỢ XÂY DỰNG PHÁC ĐỒ ĐIỀU TRỊ TRONG LĨNH VỰC Y TẾ**

Ngành: **CÔNG NGHỆ THÔNG TIN**

Tên học phần: **TRÍ TUỆ NHÂN TẠO**

Giảng viên hướng dẫn : Nguyễn Ngọc Tân

Sinh viên thực hiện:

* Nguyễn Quang Hưng - 2380600933
* Nguyễn Việt Quang - 2380601811

Lớp: 23DTHE1

TP. Hồ Chí Minh, Tháng 12 năm 2025

|  |  |
| --- | --- |
|  | BỘ GIÁO DỤC VÀ ĐÀO TẠO  **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM** |

**BÁO CÁO ĐỒ ÁN**

**ỨNG DỤNG MÔ HÌNH CÂY QUYẾT ĐỊNH TRONG THUẬT TOÁN RANDOM FOREST ĐỂ CHẨN ĐOÁN VÀ HỖ TRỢ XÂY DỰNG PHÁC ĐỒ ĐIỀU TRỊ TRONG LĨNH VỰC Y TẾ**

Ngành: **CÔNG NGHỆ THÔNG TIN**

Tên học phần: **TRÍ TUỆ NHÂN TẠO**

Giảng viên hướng dẫn : Nguyễn Ngọc Tân

Sinh viên thực hiện:

* Nguyễn Quang Hưng - 2380600933
* Nguyễn Việt Quang - 2380601811

Lớp: 23DTHE1

TP. Hồ Chí Minh, Tháng 12 năm 2025

## Mục lục

[Chương 1: Giới thiệu 5](#_Toc216694456)

[1.1. Lý do chọn đề tài 5](#_Toc216694457)

[1.2. Tầm quan trọng của trí tuệ nhân tạo trong y tế 6](#_Toc216694458)

[1.3. Vai trò của các mô hình học máy trong chẩn đoán và hỗ trợ quyết định lâm sàng 7](#_Toc216694459)

[1.4. Tình huống mô phỏng thực tế 8](#_Toc216694460)

[Chương 2: Cơ sở lý thuyết 9](#_Toc216694461)

[2.1. Cây Quyết Định (Decision Tree) 9](#_Toc216694462)

[2.1.1. Giới thiệu 9](#_Toc216694463)

[2.1.2. Các thành phần và thuật ngữ 9](#_Toc216694464)

[2.1.3. Các chỉ số đo lường "độ hỗn loạn" (Impurity) 9](#_Toc216694465)

[2.1.4. Quy tắc phân chia (Splitting Rule) - Information Gain 10](#_Toc216694466)

[2.1.5. Thuật toán xây dựng cây (Tree Induction Algorithms) 11](#_Toc216694467)

[2.1.6. Ưu và nhược điểm của Cây Quyết Định 17](#_Toc216694468)

[2.1.7. Kết luận 18](#_Toc216694469)

[2.2. Rừng Ngẫu Nhiên (Random Forest) 18](#_Toc216694470)

[2.2.1. Khái niệm kết hợp (Ensemble Learning) 18](#_Toc216694471)

[2.2.2. Nguyên lý hoạt động của Random Forest 18](#_Toc216694472)

[2.2.3. Cách mô hình kết hợp dự đoán (Voting/Averaging) 19](#_Toc216694473)

[2.2.4. Lợi ích so với Cây Quyết Định đơn lẻ 20](#_Toc216694474)

[2.3. Quy trình xây dựng và đánh giá mô hình Random Forest 21](#_Toc216694475)

[2.3.1. Thu thập và tiền xử lý dữ liệu 21](#_Toc216694476)

[2.3.2. Chia dữ liệu huấn luyện và kiểm tra 21](#_Toc216694477)

[2.3.3. Huấn luyện mô hình Random Forest 22](#_Toc216694478)

[2.3.4. Đánh giá mô hình 22](#_Toc216694479)

[2.3.5. Giải thích mô hình 22](#_Toc216694480)

[2.3.6. Triển khai và giám sát mô hình 23](#_Toc216694481)

[2.4. So sánh với một số mô hình khác trong lĩnh vực y tế 23](#_Toc216694482)

[2.4.1. Mục tiêu của phần so sánh 23](#_Toc216694483)

[2.4.2. Tiêu chí so sánh 23](#_Toc216694484)

[2.4.3. So sánh chi tiết theo mô hình 24](#_Toc216694485)

[Chương 3: Triển khai hệ thống 25](#_Toc216694486)

[3.1. Kiến trúc hệ thống tổng quan 25](#_Toc216694487)

[3.2. Thu thập và tiền xử lý dữ liệu 26](#_Toc216694488)

[3.2.1. Nguồn dữ liệu và mô tả bộ dữ liệu 26](#_Toc216694489)

[3.2.2. Tiền xử lý dữ liệu (Data Preprocessing) 27](#_Toc216694490)

[3.3. Xây dựng và huấn luyện mô hình Random Forest 28](#_Toc216694491)

[3.3.1. Lựa chọn siêu tham số (Hyperparameter Tuning) 28](#_Toc216694492)

[3.3.2. Huấn luyện mô hình với siêu tham số tối ưu 29](#_Toc216694493)

[3.4. Đánh giá và diễn giải mô hình 30](#_Toc216694494)

[3.4.1. Đánh giá hiệu năng dự đoán 30](#_Toc216694495)

[3.4.2. Diễn giải mô hình (Model Interpretability) 31](#_Toc216694496)

[3.5. Triển khai hệ thống thành ứng 32](#_Toc216694497)

[Chương 4: Kết quả thực nghiệm 34](#_Toc216694498)

[4.1. Mô tả bộ dữ liệu thực nghiệm 34](#_Toc216694499)

[4.2. Quy trình thực hiện 35](#_Toc216694500)

[4.3. Kết quả đánh giá mô hình 36](#_Toc216694501)

[4.3.1. Ma trận nhầm lẫn (Confusion Matrix) 36](#_Toc216694502)

[4.3.2. Đường cong ROC (Receiver Operating Characteristic) 37](#_Toc216694503)

[4.3.3. Đánh giá mức độ quan trọng của đặc trưng (Feature Importance) 38](#_Toc216694504)

[4.4. Kết quả hỗ trợ đề xuất phác đồ điều trị 39](#_Toc216694505)

[4.5. Thảo luận kết quả 39](#_Toc216694506)

[4.5.1. Ưu điểm đạt được 39](#_Toc216694507)

[4.5.2. Hạn chế của mô hình 39](#_Toc216694508)

[4.5.3. So sánh với một số mô hình khác 40](#_Toc216694509)

[4.6. Đánh giá tính khả thi trong thực tế y tế 40](#_Toc216694510)

[Chương 5: Kết luận và hướng phát triển của đề tài 41](#_Toc216694511)

[5.1. Kết luận chung của đề tài 41](#_Toc216694512)

[5.2. Ý nghĩa khoa học và ý nghĩa thực tiễn 42](#_Toc216694513)

[5.2.1. Ý nghĩa khoa học 42](#_Toc216694514)

[5.2.2. Ý nghĩa thực tiễn 43](#_Toc216694515)

[5.3. Hạn chế của đề tài 43](#_Toc216694516)

[5.4. Hướng phát triển trong tương lai 44](#_Toc216694517)

[5.5. Định hướng nghiên cứu lâu dài 45](#_Toc216694518)

[Chương 6: KẾT LUẬN CHUNG 46](#_Toc216694519)

[Tài liệu tham khảo 47](#_Toc216694520)

[Phụ lục 48](#_Toc216694521)

[Code minh hoạ: 53](#_Toc216694522)

### Lời cam đoan

Chúng em xin cam đoan đây là báo cáo đồ án môn học Trí Tuệ Nhân Tạo do chính nhóm chúng em thực hiện dưới sự hướng dẫn của Thầy Nguyễn Ngọc Tân.

Mọi nội dung, phân tích, kết quả thực nghiệm và kết luận trong báo cáo này là trung thực, khách quan, được rút ra từ quá trình nghiên cứu, xây dựng và đánh giá mô hình của nhóm. Các tài liệu, số liệu tham khảo đều có trích dẫn nguồn gốc đầy đủ.

Chúng em chịu hoàn toàn trách nhiệm về tính chính xác và tính nguyên bản của đồ án này.

### Lời cảm ơn

Để hoàn thành đồ án này, chúng em đã nhận được sự hỗ trợ, hướng dẫn quý báu của thầy Nguyễn Ngọc Tân - giảng viên hướng dẫn, chúng em xin bày tỏ lòng biết ơn sâu sắc đến thầy. Sự chỉ bảo tận tình, những định hướng sâu sắc và sự hỗ trợ kịp thời của Thầy là đã giúp chúng em hoàn thiện đồ án một cách tốt nhất.

Mặc dù đã nỗ lực hết sức, nhưng do còn thiếu sót kinh nghiệm nên đồ án khó tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự góp ý từ quý thầy cô và các bạn để công trình được hoàn thiện hơn.

Một lần nữa, xin chân thành cảm ơn!

# Giới thiệu

## Lý do chọn đề tài

* **Tính cấp thiết từ thực tiễn y tế:**
* Nhằm giảm thiểu áp lực lên hệ thống y tế: Số lượng bệnh nhân ngày càng đông, trong khi nguồn lực bác sĩ giỏi có hạn, dẫn đến tình trạng quá tải, thời gian chẩn đoán và đưa ra phác đồ cho mỗi bệnh nhân bị thu hẹp. Ở các nước phát triển hoặc đang phát triển tình trạng già hoá dân số đang tăng ở nhiều nước nên gánh nặng được đặt lên hệ thống y tế là rất lớn.
* Tính phức tạp của chẩn đoán: Nhiều bệnh lý có triệu chứng lẫn lộn chồng chéo với nhau phức tạp, đòi hỏi bác sĩ phải có chuyên môn tổng hợp và phân tích một khối lượng lớn thông tin (lâm sàng, cận lâm sàng, tiền sử bệnh, di truyền...). Việc này tiềm ẩn nguy cơ sai sót do yếu tố chủ quan hoặc mệt mỏi.
* Yêu cầu về y học cá thể hóa: Mỗi bệnh nhân có đặc điểm sinh học, lối sống và phản ứng với thuốc khác nhau. Cần có những công cụ hỗ trợ để đề xuất phác đồ điều trị tối ưu nhất cho từng cá nhân, thay vì một phác đồ chung cho tất cả.
* **Sức mạnh của dữ liệu và AI:**
* Sự bùng nổ của dữ liệu y tế số: Các bệnh viện đang lưu trữ một kho dữ liệu khổng lồ về hồ sơ bệnh án điện tử, kết quả xét nghiệm, hình ảnh chẩn đoán (X-quang, MRI, CT-scanner), dữ liệu gen...
* Cơ hội khai thác dữ liệu: Đây là "mỏ vàng" chưa được khai thác triệt để. Trí tuệ nhân tạo, đặc biệt là Học máy (Machine Learning), cung cấp các công cụ mạnh mẽ để phân tích, tìm ra các mẫu hình, mối tương quan ẩn sâu trong dữ liệu mà con người khó có thể nhận ra. Có thể tổng hợp một lúc nhiều tài liệu điều mà con người phải sử dụng một lượng nhân lực lớn để hoàn thành.
* **Ưu điểm vượt trội của mô hình Random Forest:**
* **Khả năng xử lý tốt với dữ liệu phức tạp:** Random Forest có thể xử lý cả dữ liệu số (tuổi, huyết áp, nồng độ cholesterol) và dữ liệu phân loại (giới tính, tiền sử gia đình, triệu chứng). Có khả năng đi sâu vào các đặc điểm cá nhân của từng bệnh nhân để đưa ra dữ liệu xử lý tốt nhất.
* **Đánh giá mức độ quan trọng của các đặc trưng (Feature Importance):** Đây là tính năng cực kỳ giá trị trong y tế. Mô hình không chỉ dự đoán mà còn cho biết các yếu tố (ví dụ: đường huyết, huyết áp, tuổi tác) là quan trọng nhất trong việc đưa ra chẩn đoán. Điều này giúp bác sĩ tập trung vào các chỉ số then chốt.
* **Giảm thiểu hiện tượng "overfitting" (quá khớp):** So với một Cây quyết định đơn lẻ (dễ học thuộc lòng dữ liệu huấn luyện và hoạt động kém trên dữ liệu mới, thiên về hướng sử dụng lại dữ liệu cũ), Random Forest kết hợp nhiều cây lại, tạo ra một mô hình ổn định và có độ chính xác tổng quát cao hơn.
* **Dễ hiểu và có thể giải thích được:** Mặc dù là mô hình "hộp đen" phức tạp, nhưng việc quan sát các cây quyết định thành phần hoặc sử dụng các kỹ thuật như SHAP, LIME có thể giúp "giải mã" lý do đằng sau dự đoán, một yếu tố quan trọng để bác sĩ tin tưởng vào kết quả.

## Tầm quan trọng của trí tuệ nhân tạo trong y tế

Trí tuệ nhân tạo không còn là công nghệ của tương lai mà đang hiện diện ngày càng rõ nét, cách mạng hóa toàn bộ ngành y tế:

* **Hỗ trợ chẩn đoán hình ảnh:** Các mô hình Deep Learning (một nhánh cũng khá quan trọng của AI) đã vượt trội trong việc phát hiện các khối u, dấu hiệu bất thường trong ảnh X-quang, MRI, CT-scanner, chẩn đoán bệnh võng mạc tiểu đường... với độ chính xác ngang hoặc thậm chí vượt bác sĩ chuyên khoa trong một số nhiệm vụ cụ thể.
* **Phát triển thuốc và liệu pháp mới:** AI giúp đẩy nhanh quá trình tìm kiếm các phân tử thuốc tiềm năng, dự đoán tác dụng phụ, và rút ngắn thời gian thử nghiệm lâm sàng từ hàng chục năm xuống còn vài năm. Trong một số trường hợp hi hữu AI còn giúp các nhà khoa học tìm ra công thức mới mà trước đó khó có ai tìm ra điều này là một điều khá là đáng kinh ngạc của AI.
* **Phẫu thuật bằng Robot:** Các Robot phẫu thuật được điều khiển bằng AI cho phép thực hiện những ca mổ với độ chính xác cực cao, xâm lấn tối thiểu, giúp bệnh nhân hồi phục nhanh hơn. Việc ứng dụng Robot còn gặp nhiều tranh cãi từ phía bệnh nhân nhưng không thể phủ nhận được với các loại Robot tiên tiến hiện nay thì độ chính xác trong phẫu thuật cực cao có thể hoàn toàn yên tâm trong thời gian sắp tới.
* **Y học dự đoán và phòng ngừa:** Phân tích dữ liệu gen và lối sống để dự đoán nguy cơ mắc các bệnh mãn tính (ung thư, tiểu đường, tim mạch) từ sớm, từ đó có các biện pháp can thiệp kịp thời. AI cũng có thể phân tích nhận định và đưa ra các phương pháp cải thiện môi trường, lối sống, cách ăn uống sao cho khoa học rất phù hợp với các nước điều kiện sống còn tương đối khó khăn về mặt y tế.

## Vai trò của các mô hình học máy trong chẩn đoán và hỗ trợ quyết định lâm sàng

Học máy đóng vai trò là "Trợ lý ảo thông minh" cho các chuyên gia y tế:

* **Tự động hóa và tăng hiệu suất:** Tự động phân tích các kết quả xét nghiệm thường quy, sàng lọc các ca bệnh có nguy cơ cao, giúp bác sĩ tiết kiệm thời gian và tập trung vào những trường hợp phức tạp.
* **Cung cấp các "gợi ý" khách quan, dựa trên dữ liệu:** Mô hình đưa ra dự đoán dựa trên sự phân tích của hàng ngàn, hàng vạn hồ sơ bệnh án trước đó. Điều này giúp bác sĩ có thêm một góc nhìn tham khảo khách quan, giảm thiểu sự phỏng đoán chủ quan của cá nhân.
* **Hỗ trợ ra quyết định trong điều kiện không chắc chắn:** Trong nhiều tình huống lâm sàng, thông tin không đầy đủ hoặc xung đột chồng chéo lên nhau. Các mô hình học máy có thể tính toán xác suất xảy ra của các kết quả khác nhau (ví dụ: xác suất khối u lành tính hay ác tính là bao nhiêu %), giúp bác sĩ cân nhắc rủi ro và đưa ra quyết định chính xác hơn giúp cho bệnh nhân có thể phòng trách hoặc điều trị thời tránh cho bệnh tình trở nặng hơn.
* **Cá thể hóa điều trị:** Dựa trên hồ sơ của một bệnh nhân cụ thể, mô hình có thể dự đoán xem bệnh nhân đó sẽ đáp ứng tốt với phác đồ điều trị A hay B, từ đó đề xuất lựa chọn tối ưu nhất. Cũng có thể đưa ra nhiều phương án nhiều lộ trình điều trị phù hợp với hoàn cảnh và điều kiện của từng bệnh nhân cụ thể mỗi người một hoàn cảnh khác nhau nên phương án điều trị cũng được đa dạng hơn.
* **Cung cấp nguồn tài liệu học tập:** Ngoài các vai trò chính ở trên thì một vai trò cũng không kém phần quan trọng khác là học tập, các mô hình trí tuệ nhân tạo có thể giúp cho các bác sĩ mới hoặc cũ có thể trao dồi thêm các kiến thức về y tế chuyên sâu trong những lúc rảnh hoặc là muốn tìm hiểu sâu về một loại bệnh nào đó, giúp tiết kiệm thời gian tra cứu tài liệu, cung cấp các tài liệu chẩn xác nhất.

## Tình huống mô phỏng thực tế

**Kịch bản:** Một bệnh viện đa khoa muốn xây dựng hệ thống hỗ trợ sàng lọc và đánh giá nguy cơ mắc bệnh suy tim cho bệnh nhân của khoa tim mạch.

* **Dữ liệu đầu vào:**Họ thu thập dữ liệu từ 10,000 bệnh nhân bao gồm: tuổi, giới tính, huyết áp, nồng độ cholesterol, đường huyết, tiền sử hút thuốc, chỉ số khối cơ thể (BMI), và kết quả chẩn đoán suy tim (Có / Không).
* **Áp dụng mô hình:** Họ sử dụng thuật toán Random Forest để huấn luyện một mô hình trên tập dữ liệu này.
* **Kết quả và ứng dụng:**
* **Chẩn đoán:** Khi một bệnh nhân mới nhập viện, hệ thống sẽ nhập các chỉ số của bệnh nhân này vào mô hình. Mô hình sẽ dự đoán xác suất bệnh nhân đó có nguy cơ bị suy tim với độ chính xác lên đến 95%.
* **Đề xuất điều trị:** Dựa trên kết quả dự đoán và đánh giá mức độ quan trọng của các đặc trưng, mô hình có thể gợi ý: "Bệnh nhân này có nguy cơ cao, chủ yếu do các chỉ số huyết áp và cholesterol rất cao. Ưu tiên các phác đồ điều trị tập trung vào kiểm soát hai chỉ số này."
* **Hỗ trợ bác sĩ:** Bác sĩ nhận được thông tin cảnh báo sớm và các gợi ý từ hệ thống, từ đó có cơ sở để yêu cầu các xét nghiệm chuyên sâu hơn và đưa ra quyết định điều trị nhanh chóng, chính xác hơn, cá thể hóa hơn.

# Cơ sở lý thuyết

## Cây Quyết Định (Decision Tree)

### Giới thiệu

Cây quyết định là một mô hình học máy phi tuyến tính, mô phỏng cách con người ra quyết định bằng cách đặt ra một loạt các câu hỏi tương tự. Mô hình có cấu trúc phân cấp giống một cái cây, bắt đầu từ một nút gốc và lớn dần lên phân nhánh ra cho đến khi đưa ra kết quả cuối cùng tại các nút lá. Nó có thể áp dụng cho cả bài toán Phân loại (Classification) và Hồi quy (Regression).

*\*Xem hình minh hoạ Cây quyết định ở Phụ lục, Hình 1*

### Các thành phần và thuật ngữ

* **Nút gốc (Root Node):** Là nút đầu tiên, đại diện cho toàn bộ tập dữ liệu. Tại đây, thuật toán sẽ tìm ra đặc trưng (feature) tốt nhất để bắt đầu phân chia dữ liệu.
* **Nút quyết định (Internal/Decision Node):** Là các nút trung gian, nơi diễn ra quá trình kiểm tra giá trị của một đặc trưng. Mỗi nút này sẽ tạo ra các nhánh con mới.
* **Nút lá (Leaf Node/Terminal Node):** Là nút cuối cùng, đại diện cho kết quả dự đoán (lớp trong bài toán phân loại hoặc giá trị số trong bài toán hồi quy).
* **Nhánh (Branch/Sub-Tree):** Đường nối giữa các nút, thể hiện kết quả của một quy tắc kiểm tra.

### Các chỉ số đo lường "độ hỗn loạn" (Impurity)

Để quyết định phân chia dữ liệu tại một nút, ta cần đánh giá mức độ **đồng nhất của các mẫu** tại nút đó. Một nút càng đồng nhất (tất cả mẫu thuộc cùng một lớp) thì càng ít “hỗn loạn”. Hai chỉ số phổ biến nhất dùng để đo độ “hỗn loạn” này là:

* **Chỉ số Gini (Gini Impurity):**
  + Đo lường xác suất một mẫu dữ liệu được chọn ngẫu nhiên bị phân loại sai nếu nó được gán nhãn ngẫu nhiên theo sự phân bố của các lớp trong nút đó.
  + Công thức: Gini = 1 - Σ(p\_i)²  với p\_i là tỷ lệ của lớp thứ i trong nút.
  + Đặc điểm: Gini có xu hướng tính toán nhanh hơn và thường được sử dụng mặc định trong nhiều thư viện (như scikit-learn). Giá trị từ 0 (hoàn toàn thuần nhất) đến 0.5 (với 2 lớp, hỗn loạns nhất).
* **Entropy:**
  + Đo lường mức độ hỗn loạn hoặc bất định của thông tin trong một tập hợp. Nó dựa trên khái niệm trong Lý thuyết Thông tin.
  + Công thức: Entropy = - Σ(p\_i \* log₂(p\_i))
  + Đặc điểm: Entropy có xu hướng tạo ra các cây cân bằng hơn. Giá trị từ 0 (hoàn toàn thuần nhất) đến 1 (với 2 lớp, hỗn loạn nhất).
* *Ví dụ minh hoạ:*

Giả sử một nút có 10 mẫu:

* 7 thuộc lớp A
* 3 thuộc lớp B

Ta tính:

Gini = 1 – (0.7² + 0.3²) = 1 – (0.49 + 0.09) = 0.42

Entropy = – [0.7 \* log₂(0.7) + 0.3 \* log₂(0.3)] ≈ 0.88

Kết luận: Nút này **không thuần**, vì giá trị khá cao.

* Nhờ hai chỉ số này, thuật toán có thể chọn được đặc trưng và điểm chia giúp giảm độ hỗn loạn nhiều nhất, từ đó tối ưu cấu trúc của cây quyết định.

### Quy tắc phân chia (Splitting Rule) - Information Gain

* Information Gain (Lợi ích thông tin) là thước đo đánh giá **mức độ giảm “độ hỗn loạn” (impurity)** khi ta phân chia dữ liệu tại một nút dựa trên một đặc trưng nào đó. Ý tưởng chung là:  
  **Một phép chia tốt sẽ làm cho các nút con “thuần” hơn so với nút cha.**

Trong quá trình xây dựng Cây Quyết Định, thuật toán sẽ tính Information Gain cho từng đặc trưng, và **đặc trưng tạo ra mức giảm hỗn loạn lớn nhất sẽ được chọn để phân chia tại nút đó**.

* **Công thức tổng quát:**

Information Gain =

Trong đó:

* Impurity có thể là **Entropy**, **Gini**, hoặc **Misclassification Rate**.
* là tổng số mẫu ở nút con thứ i.
* là tổng số mẫu ở nút cha.
* Biểu thức trong ngoặc là **độ hỗn loạn trung bình có trọng số** của các nút con.
* *Ví dụ minh họa:*

Giả sử Entropy của nút cha là **1.0** (tức là dữ liệu đang rất hỗn loạn).  
Sau khi phân chia theo một đặc trưng X, ta thu được hai nút con với Entropy trung bình theo trọng số bằng **0.3**.

Khi đó:

Information Gain = 1.0 − 0.3 = 0.7

Điều này cho thấy phép chia này giúp **giảm đáng kể mức độ hỗn loạn**, tức là một điểm chia **rất tốt**, và đặc trưng X sẽ được chọn để phân nhánh tại nút này.

### Thuật toán xây dựng cây (Tree Induction Algorithms)

Trong thực tế, có ba thuật toán xây dựng cây quyết định phổ biến nhất: ID3, C4.5, và CART. Chúng giống nhau về nguyên lý chung (đều xây dựng cây theo kiểu top –down, chia để trị), nhưng khác nhau về cách lựa chọn đặc trưng, cách xử lý dữ liệu liên tục, và loại chỉ số impurity sử dụng.

#### Thuật toán ID3 (Iterative Dichotomiser 3)

Thuật toán này được đề xuất bởi Quinlan vào năm 1986, dùng cho phân loại (classification) chỉ xử lý được dữ liệu phân loại (categorical).

*\*Xem Code mẫu Thuật toán Iterative Dichotomiser 3 tại Phụ lục Code minh hoạ 1*

1. **Nguyên lý hoạt động cốt lõi:**

ID3 là một trong những thuật toán đầu tiên và nền tảng cho việc xây dựng cây quyết định. Nó hoạt động dựa trên các nguyên tắc sau:

* Triết lý "Chia để trị" (Divide and Conquer): Thuật toán phân chia tập dữ liệu thành các tập con nhỏ hơn và đồng nhất hơn dựa trên các đặc trưng.
* Xây dựng cây theo hướng Từ-trên-xuống (Top-down): Bắt đầu từ nút gốc (root node) và lần lượt chọn các đặc trưng tốt nhất để phân nhánh.
* Tính toán dựa trên Lý thuyết Thông tin: Sử dụng khái niệm Độ lợi thông tin (Information Gain) để lượng tử hóa "mức độ tốt" của một đặc trưng khi dùng để phân chia.

1. **Các khái niệm then chốt**

Để hiểu ID3, cần nắm rõ các khái niệm sau:

* **Entropy (Độ hỗn loạn):** Đo lường sự không thuần nhất hay "hỗn loạn" của một tập dữ liệu.
  + Công thức: Entropy(S) = -∑(p\_i \* log₂(p\_i)) với p\_i là tỷ lệ của lớp thứ i trong tập S.
  + Entropy **= 0:** Tập dữ liệu hoàn toàn thuần nhất (chứa toàn một lớp).
  + Entropy cao: Tập dữ liệu hỗn loạn, các lớp được trộn lẫn với nhau.
* **Information Gain (Độ lợi thông tin):** Là sự giảm entropy kỳ vọng sau khi phân chia tập dữ liệu theo một đặc trưng nào đó. Đặc trưng nào mang lại Độ lợi thông tin cao nhất sẽ được chọn làm tiêu chí phân chia tại nút đó.
  + Công thức: Gain(S, A) = Entropy(S) - ∑(|S\_v| / |S| \* Entropy(S\_v))
  + Trong đó:
    - S: Tập dữ liệu tại nút hiện tại.
    - A: Đặc trưng được xét để phân chia.
    - S\_v: Tập con của S khi đặc trưng A có giá trị v

1. **Các bước thực thi của thuật toán ID3**

Thuật toán có thể được tóm tắt bằng các bước đệ quy sau:

1. **Bắt đầu** tại nút gốc với toàn bộ tập dữ liệu huấn luyện.
2. **Tính Entropy** cho tập dữ liệu tại nút hiện tại.
3. **Với mỗi đặc trưng,** tính Độ lợi thông tin (Information Gain) nếu phân chia theo đặc trưng đó.
4. **Chọn đặc trưng** có **Độ lợi thông tin cao nhất**.
5. **Tạo một nút mới** trong cây tương ứng với đặc trưng vừa chọn.
6. **Phân chia** tập dữ liệu thành các tập con dựa trên các giá trị của đặc trưng được chọn.
7. **Lặp lại** (đệ quy) các bước từ 2 đến 6 cho mỗi tập con, với các điều kiện dừng sau:
   * Tất cả các mẫu trong nút đều thuộc cùng một lớp -> tạo **nút lá** và gán nhãn lớp đó.
   * Không còn đặc trưng nào để phân chia -> tạo **nút lá** và gán nhãn là lớp phổ biến nhất.
   * Một tập con rỗng → tạo **nút lá** và gán nhãn là lớp phổ biến nhất từ nút cha.

#### Thuật toán C4.5

Đây là phiên bản mở rộng mạnh mẽ của ID3 (Quinlan, 1993) và được đánh giá là một trong những thuật toán quyết định cây kinh điển nhất.

*\*Xem Code mẫu Thuật toán C4.5 tại Phụ lục Code minh hoạ 2*

* **Các cải tiến nổi bật:**

1. Xử lý dữ liệu liên tục

Tìm điểm chia tối ưu bằng cách sắp xếp và thử các ngưỡng: x < t

*Quy trình cụ thể:*

*- Đối với mỗi đặc trưng liên tục (ví dụ: "Thu nhập"), thuật toán sẽ:*

1. **Sắp xếp** các giá trị duy nhất của đặc trưng đó.
2. **Xác định các ngưỡng tiềm năng**, thường là điểm giữa của các giá trị liên tiếp (ví dụ: nếu có thu nhập 30 và 40, ngưỡng là 35).
3. **Đánh giá từng ngưỡng** bằng cách coi nó như một phép chia nhị phân (Feature <= Ngưỡng và Feature > Ngưỡng), và tính toán chỉ số (Gain Ratio) cho phép chia đó.
4. **Chọn ngưỡng** mang lại chỉ số tốt nhất. Đặc trưng liên tục này khi đó sẽ được xem xét bình đẳng với các đặc trưng phân loại khác.
5. Dùng Gain Ratio thay vì Information Gain

Trong đó:

Ý nghĩa của Split Information (SplitInfo):

**Split Information** về bản chất là entropy của việc phân chia các mẫu theo các giá trị của đặc trưng. Nó đo lường mức độ "chia nhỏ" của một đặc trưng.

* Một đặc trưng có nhiều giá trị (như ID) sẽ có Split Information rất cao, làm cho Gain Ratio trở nên nhỏ lại, từ đó giảm thiểu sự thiên vị.
* Gain Ratio giúp tìm ra những đặc trưng vừa cung cấp nhiều thông tin, vừa không tạo ra quá nhiều nhánh con vô nghĩa.

1. Xử lý missing values thông minh

* Vẫn có thể chọn feature đó để split.
* Phân phối mẫu thiếu dữ liệu vào các nhánh theo tỷ lệ thống kê.

Giả sử tại một nút, chúng ta có 10 mẫu và chọn đặc trưng A để phân chia. Trong 10 mẫu đó, có 1 mẫu bị thiếu giá trị A, 9 mẫu còn lại được phân bổ như sau: 6 mẫu thuộc nhánh A1 và 3 mẫu thuộc nhánh A2.

* Thay vì bỏ qua các mẫu bị thiếu đò, thì thuật toán C4.5 sẽ gán trọng số cho mẫu đó.
* Mẫu thiếu sẽ được phân bổ đồng thời vào cả hai nhánh A1 và A2 với trọng số lần lượt là 6 / 9 = 0.67 và 3 / 9 = 0.33.
* Khi tính toán entropy hay các chỉ số sau này, các mẫu có trọng số sẽ được tính toán tương ứng. Cách tiếp cận này cho phép sử dụng tối đa thông tin từ các dữ liệu không hoàn chỉnh.

1. Áp dụng pruning (cắt tỉa nhánh)

* Reduce Error Pruning
* Subtree replacement

Pruning (cắt tỉa) là kỹ thuật then chốt giúp C4.5 chống overfitting bằng cách loại bỏ các nhánh phức tạp, ít có giá trị dự đoán từ cây.

Subtree Replacement (Thay thế cây con): Đây là kỹ thuật chính. Một cây con phức tạp có thể được thay thế bằng một nút lá duy nhất nếu việc thay thế đó không làm giảm độ chính xác tổng thể (được ước lượng trên tập kiểm tra hoặc sử dụng xác suất hậu nghiệm).

* *Ví dụ:* Một nhánh cây với nhiều quy tắc chi tiết chỉ cải thiện độ chính xác cho 2 mẫu huấn luyện có thể bị cắt bỏ và thay bằng một nút lá dự đoán lớp phổ biến hơn, giúp cây đơn giản và khái quát hóa tốt hơn.

#### Thuật toán CART (Classification and Regression Trees)

Được giới thiệu bởi Breiman (1984). CART là thuật toán nền tảng và là thuật toán được Scikit - Learn sử dụng.

**Đặc điểm nổi bật:**

1. Cây nhị phân 100%:

- Mỗi nút luôn tạo ra đúng 2 nhánh (binary tree). Điều này có nghĩa là ngay cả với một đặc trưng phân loại có nhiều hơn 2 giá trị (ví dụ: Màu sắc có Đỏ, Xanh, Vàng), CART không tạo ra 3 nhánh như ID3 hay C4.5 có thể làm. Thay vào đó, nó luôn tìm cách gộp các giá trị lại thành hai nhóm.

* *Ví dụ*: Thay vì phân chia Màu sắc = Đỏ, Xanh, Vàng gồm có 3 nhánh khác hoàn toàn nhau. Thì CART sẽ tìm ra phép chia tối ưu kiểu Màu sắc thuộc {Đỏ, Xanh} với Màu sắc = Vàng.

- Ưu điểm của việc này:

* + Tính nhất quán: Cấu trúc nhị phân đơn giản hóa việc triển khai và diễn giải.
  + Tránh "thiên vị đặc trưa nhiều giá trị": Bằng cách buộc phải gộp các giá trị, nó tự nhiên giảm thiểu xu hướng ưu tiên các đặc trưng có nhiều giá trị rời rạc.
  + Tạo ra các phân chia có ý nghĩa: Việc gộp nhóm có thể tạo ra các ngưỡng quyết định phức tạp và linh hoạt hơn.

1. Tiêu chí cho phân loại

* Gini Impurity: G = 1 - ∑(p\_i²). Về mặt tính toán, Gini không chứa phép logarit nên thường được tính toán nhanh hơn một chút so với Entropy. Trong thực tế, sự khác biệt về hiệu suất giữa hai tiêu chí là rất nhỏ.
* Bản chất: Cả Gini và Entropy đều là các hàm đo lường "độ không thuần nhất" và có hình dạng rất giống nhau. CART chọn Gini chủ yếu vì lý do lịch sử và tính hiệu quả về mặt tính toán.
* Hàm mục tiêu: Dù sử dụng Gini hay Entropy, mục tiêu cuối cùng vẫn là giảm thiểu tối đa độ bất định (impurity) này sau mỗi lần phân chia.

1. Tiêu chí cho hồi quy và dự đoán

CARD tối ưu:

Mục tiêu: Trong bài toán hồi quy, CART không còn tìm cách phân loại mà là dự đoán một giá trị liên tục.

Tiêu chí chia nhánh: Thay vì Gini, nó sử dụng Hàm bình phương sai số (MSE). Thuật toán sẽ chọn đặc trưng và ngưỡng sao cho việc phân chia làm giảm tổng phương sai (variance) của biến mục tiêu trong các nút con một cách tối đa. Nói cách khác, nó cố gắng tạo ra các nhóm có giá trị mục tiêu càng "giống nhau" càng tốt.

Dự đoán: Giá trị dự đoán tại nút lá trong bài toán hồi quy là giá trị trung bình của tất cả các mẫu huấn luyện rơi vào nút lá đó. Đây là lựa chọn tối ưu khi hàm mất mát là MSE.

1. Hỗ trợ Surrogate Splits

Đây là một tính năng rất mạnh mẽ và độc đáo của CART:

* **Bài toán:** Khi có một mẫu dữ liệu mới bị thiếu giá trị ở đặc trưng đã được dùng để phân chia tại một nút, chúng ta phải làm gì?
* **Giải pháp của CART:** Trong quá trình xây dựng cây, với mỗi phép phân chia (split), CART sẽ tự động tìm kiếm các **phép phân chia dự phòng (surrogate splits)**.
  + **Cách thức:** Một phép phân chia dự phòng là phép chia sử dụng một đặc trưng khác, nhưng có cách phân phối các mẫu dữ liệu vào các nhánh giống nhất có thể với phép phân chia chính.
  + **Ứng dụng:** Khi gặp dữ liệu thiếu, CART sẽ lần lượt sử dụng các phép phân chia dự phòng này, theo thứ tự độ "giống nhau" giảm dần, để quyết định xem mẫu dữ liệu nên rẽ vào nhánh nào. Điều này cho phép CART xử lý missing values một cách rất hiệu quả và thanh lịch.

1. Các điểm quan trọng khác

Kỹ thuật Cắt tỉa (Pruning): Giống như C4.5, CART sử dụng kỹ thuật cắt tỉa mạnh mẽ để chống overfitting. Tuy nhiên, CART thường sử dụng "Cost-Complexity Pruning" (còn gọi là Weakest Link Pruning). Kỹ thuật này đánh đổi giữa độ phức tạp của cây và độ chính xác, tìm ra một cây con có kích thước tối ưu.

* Ưu điểm chính so với ID3/C4.5:
* ​ Tính linh hoạt: Có thể giải quyết cả bài toán Phân loại (Classification) và Hồi quy (Regression).
* ​ Xử lý dữ liệu liên tục một cách tự nhiên nhờ cấu trúc nhị phân.
* ​ Cơ chế xử lý missing values (Surrogate Splits) rất mạnh mẽ.
* Nhược điểm:
* Cây CART có thể không ổn định (unstable), tức là một thay đổi nhỏ trong dữ liệu huấn luyện có thể dẫn đến một cấu trúc cây hoàn toàn khác. (Đây là lý do tại sao các Random Forest - được xây dựng từ nhiều cây CART - lại hiệu quả, vì chúng khắc phục được nhược điểm này).

### Ưu và nhược điểm của Cây Quyết Định

* Ưu điểm:
* Dễ hiểu và giải thích: Mô hình có tính minh bạch cao, có thể dễ dàng biểu diễn dưới dạng các luật "if - else".
* Ít yêu cầu tiền xử lý dữ liệu: Không cần chuẩn hóa dữ liệu (scaling) và có thể xử lý tốt cả dữ liệu số và phân loại.
* Xử lý được dữ liệu phi tuyến tính: Bản chất phân cấp giúp nó nắm bắt được các mối quan hệ phi tuyến phức tạp.
* Hiệu quả với các đặc trưng quan trọng: Cây có khả năng tự động chọn các đặc trưng quan trọng nhất để phân chia, hỗ trợ phân tích đặc trưng (feature importance).
* Nhược điểm:
* Rất dễ bị Overfitting (Quá khớp): Cây có xu hướng phát triển rất sâu để học "thuộc lòng" dữ liệu huấn luyện, kể cả nhiễu, dẫn đến hiệu suất kém trên dữ liệu mới. Để khắc phục, cần các kỹ thuật Cắt tỉa (Pruning) hoặc giới hạn độ sâu tối đa, số lá có thể tiếp cận được.
* Nhạy cảm với dữ liệu huấn luyện: Một thay đổi nhỏ trong dữ liệu có thể tạo ra một cấu trúc cây hoàn toàn khác. Điều này làm cho mô hình kém ổn định hơn so với các mô hình tuyến tính hoặc mô hình ensemble.
* Có thể tạo ra các cây thiên lệch (Biased Tree): Nếu một lớp chiếm ưu thế, cây có thể bị nghiêng về lớp đó làm mất cân bằng tổng thể của cả cây.
* Không tối ưu về mặt toàn cục: Quyết định phân chia tại mỗi nút dựa trên tiêu chí cục bộ (local optimum), nên tổng thể cây có thể không đạt cấu trúc tối ưu toàn cục.

### Kết luận

Cây Quyết Định (Decision Tree) là một trong những mô hình nền tảng và trực quan nhất trong lĩnh vực học máy. Với khả năng mô phỏng cách con người ra quyết định thông qua các quy tắc phân nhánh, mô hình này mang lại tính minh bạch và dễ diễn giải cao. Thông qua các chỉ số đo lường độ hỗn loạn như Gini và Entropy, cũng như tiêu chí Information Gain, cây có thể lựa chọn đặc trưng tối ưu tại mỗi bước để xây dựng cấu trúc phân chia dữ liệu hiệu quả.

Tuy nhiên, bên cạnh sự đơn giản và tính trực quan, Cây Quyết Định cũng tồn tại một số hạn chế đáng kể như dễ bị overfitting, kém ổn định trước biến động dữ liệu, và có nguy cơ tạo ra mô hình thiên lệch nếu dữ liệu mất cân bằng. Những yếu điểm này dẫn đến nhu cầu kết hợp nhiều cây lại để tạo ra các mô hình mạnh hơn như Random Forest hay Gradient Boosting, vốn khắc phục phần lớn nhược điểm của một cây đơn lẻ mà mô hình này đang gặp phải.

Tổng kết lại, Cây Quyết Định là mô hình quan trọng, vừa dễ hiểu, vừa đủ mạnh trong nhiều bài toán thực tế, và là nền tảng để phát triển các thuật toán ensemble tiên tiến hơn trong lĩnh vực trí tuệ nhân tạo.

## Rừng Ngẫu Nhiên (Random Forest)

### Khái niệm kết hợp (Ensemble Learning)

Random Forest không phải là một mô hình duy nhất, mà là một **phương pháp kết hợp (Ensemble Method)**. Ý tưởng cốt lõi là thay vì dựa vào một mô hình duy nhất (một cây quyết định dễ sai sót), chúng ta xây dựng nhiều mô hình (nhiều cây) và kết hợp chúng lại. "Trí tuệ của đám đông" thường tốt hơn ý kiến của một cá nhân. Có hai kỹ thuật ensemble chính:

* **Bagging (Bootstrap Aggregating):** Tạo ra nhiều phiên bản khác nhau của bộ dữ liệu huấn luyện để xây dựng các mô hình độc lập, sau đó kết hợp kết quả. Random Forest là một mở rộng của Bagging.
* **Boosting:** Xây dựng các mô hình một cách tuần tự, trong đó mô hình sau học từ sai lầm của mô hình trước.

### Nguyên lý hoạt động của Random Forest

Random Forest kết hợp sức mạnh của **Hai kỹ thuật ngẫu nhiên**:

1. **Bootstrap Sampling (Lấy mẫu Bootstrap):** Từ tập dữ liệu gốc có kích thước N, thuật toán tạo ra M tập dữ liệu con (mỗi tập dùng để huấn luyện một cây). Mỗi tập con được tạo bằng cách lấy mẫu có hoàn lại (sampling with replacement). Điều này có nghĩa:
   * Một mẫu dữ liệu có thể được chọn nhiều lần trong cùng một tập con.
   * Một số mẫu có thể không được chọn vào một tập con nào đó (những mẫu này gọi là Out-of-Bag - OOB và có thể dùng để đánh giá mô hình).
   * Các mẫu OOB này được dùng để đánh giá mô hình mà không cần tập kiểm tra riêng biệt cho nên tạo ra chỉ số OOB Score, đặc trưng của thuật toán Random Forest.
2. **Random Subspace Method (Tính ngẫu nhiên trong đặc trưng):**

* Tại mỗi nút của mỗi cây quyết định, thay vì chọn đặc trưng tốt nhất từ toàn bộ tập đặc trưng, thuật toán chỉ xem xét một tập con ngẫu nhiên của các đặc trưng (thường là sqrt(n\_features) cho bài toán phân loại hoặc n\_features/3 cho hồi quy).
* Tại sao lại làm vậy? Kỹ thuật này đảm bảo các cây trong rừng trở nên đa dạng và ít tương quan trùng lặp dữ liệu quá nhiều với nhau. Nếu không, tất cả các cây có thể sẽ chọn cùng một đặc trưng mạnh nhất ở nút gốc và trở nên giống hệt nhau, làm mất đi lợi ích của việc kết hợp.
* **Cơ chế tạo ra mô hình ổn định:**

Nhờ hai nguồn ngẫu nhiên trên:

* Dữ liệu khác nhau → cây khác nhau
* Tập đặc trưng khác nhau → hướng phân chia khác nhau
* **Rừng** *(gồm có nhiều câu bên trong)*trở thành mô hình:
* Ổn định
* Ít overfitting hơn cây đơn lẻ
* Khả năng tổng quát hóa mạnh

*Xem hình minh hoạ Random Forest ở Phụ lục, Hình 2*

### Cách mô hình kết hợp dự đoán (Voting/Averaging)

Sau khi M cây được huấn luyện, khi có một mẫu dữ liệu mới cần dự đoán:

* + **Phân loại (Classification):** Mỗi cây bỏ một phiếu cho một lớp. Lớp nào nhận được nhiều phiếu nhất (Majority Voting) sẽ là kết quả dự đoán cuối cùng.
  + *Ví dụ:*

Giả sử có 5 cây đưa ra kết quả:

* + Cây 1 → Lớp A
  + Cây 2 → Lớp B
  + Cây 3 → Lớp A
  + Cây 4 → Lớp A
  + Cây 5 → Lớp B

→ Lớp A có 3/5 phiếu → là dự đoán cuối cùng.

* Đối với bài toán Hồi quy (Regression): Averaging (Trung bình)
  + Kết quả là giá trị trung bình của tất cả các cây.
  + Làm giảm nhiễu và tăng độ ổn định.

### Lợi ích so với Cây Quyết Định đơn lẻ

* Giảm Overfitting mạnh mẽ:
* Cây đơn: Học thuộc lòng dữ liệu gốc.
* Random Forest: Mỗi cây được huấn luyện trên một tập dữ liệu hơi khác nhau và với một tập con đặc trưng, nên chúng học các khía cạnh khác nhau của dữ liệu. Việc kết hợp nhiều cây "chưa hoàn hảo" nhưng đa dạng này lại tạo ra một mô hình tổng quát và mạnh mẽ, ít bị ảnh hưởng bởi nhiễu.
* Độ chính xác cao hơn: Nhờ vào "trí tuệ của đám đông", dự đoán của Random Forest thường chính xác và đáng tin cậy hơn so với bất kỳ một cây đơn lẻ nào.
* Độ ổn định cao: Vì kết quả dựa trên sự đồng thuận của nhiều cây, nên Random Forest ít nhạy cảm với sự biến động nhỏ trong tập dữ liệu huấn luyện. Một mẫu dữ liệu bất thường sẽ không ảnh hưởng lớn đến toàn bộ rừng.
* Cung cấp đánh giá mức độ quan trọng của đặc trưng (Feature Importance): Random Forest có thể ước tính mức độ quan trọng của từng đặc trưng bằng cách xem xét mức độ giảm trung bình của độ hỗn loạn (Gini/Entropy) mà đặc trưng đó mang lại trên toàn bộ các cây. Điều này cực kỳ hữu ích trong y tế để xác định các yếu tố nguy cơ chính.

## Quy trình xây dựng và đánh giá mô hình Random Forest

Quy trình xây dựng và đánh giá mô hình Random Forest trong lĩnh vực y tế đòi hỏi sự cẩn trọng và hệ thống, từ khâu chẩn bị dữ liệu đến triển khai mô hình vì đây là loại hình đặc thù nên mọi sự sai sót có thể dẫn đến hậu quả nghiêm trọng. Dưới đây là các bước triển khai mô hình chi tiết:

### Thu thập và tiền xử lý dữ liệu

* Thu thập dữ liệu:
* Dữ liệu có thể đến từ nhiều nguồn: hồ sơ bệnh án, hồ sơ điện tử, kết quả xét nghiệm, hình ảnh y tế, dữ liệu di truyền, thiết bị chuyên dụng hoặc cũng có thể lấy được từ nội dung mô tả triệu chứng của bệnh nhân.
* *Ví dụ:* Tuổi, giới tính, huyết áp, cholesterol, đường huyết, tiền sử bệnh, kết quả chẩn đoán.
* Tiền xử lý dữ liệu:
* Xử lý missing values: Sử dụng phương pháp thay thế bằng giá trị trung bình/trung vị, hoặc dùng thuật toán như KNN Imputation.
* Mã hóa dữ liệu phân loại: Chuyển đổi biến categorical (Ví dụ: giới tính, tiền sử hút thuốc) thành dạng số bằng One-Hot Encoding hoặc Label Encoding.
* Chuẩn hóa dữ liệu: Random Forest không yêu cầu chuẩn hóa, nhưng có thể áp dụng Min-Max Scaling hoặc Standard Scaling nếu kết hợp với mô hình khác.
* Xử lý mất cân bằng dữ liệu: Sử dụng kỹ thuật Oversampling (SMOTE) hoặc Undersampling nếu dữ liệu bị chênh lệch quá lớn giữa các lớp.

### Chia dữ liệu huấn luyện và kiểm tra

* Dữ liệu được chia thành ba giao đoạn luyện tập:
* Tập huấn luyện (Training set, ~70%): Dùng để xây dựng mô hình.
* Tập kiểm thử (Validation set, ~15%): Dùng để điều chỉnh siêu tham số.
* Tập kiểm tra (Test set, ~15%): Dùng để đánh giá cuối cùng.
* Kỹ thuật Cross - Validation (Dùng K - Fold, K = 5 hoặc 10):
* Giúp đánh giá ổn định hơn, tránh overfitting.
* Đặc biệt hữu ích khi dữ liệu ít.
* Xây dựng trong giai đoạn huấn luyện và tinh chỉnh mô hình.

### Huấn luyện mô hình Random Forest

* Lựa chọn siêu tham số (Hyperparameter tuning):
* Số cây (n\_estimators): Thường từ 100 – 500.
* Độ sâu tối đa (max\_depth): Giới hạn để tránh overfitting.
* Số lượng đặc trưng xét tại mỗi nút (max\_features): sqrt(n\_features) cho phân loại, n\_features/3 cho hồi quy.
* Số mẫu tối thiểu để phân chia (min\_samples\_split): Thường từ 2 – 10.
* Số mẫu tối thiểu tại nút lá (min\_samples\_leaf): Thường từ 1 – 5.
* Tối ưu hóa siêu tham số:
* Sử dụng Grid Search hoặc Random Search kết hợp với Cross - validation.
* Ví dụ: Tìm best\_params\_ bằng GridSearchCV trong Scikit - learn.

### Đánh giá mô hình

* Với bài toán phân loại:
* Accuracy: Tỷ lệ dự đoán đúng.
* Precision & Recall: Quan trọng trong y tế khi cần tránh bỏ sót ca bệnh.
* F1-Score: Kết hợp Precision và Recall.
* AUC-ROC: Đánh giá khả năng phân biệt giữa lớp bệnh và không bệnh.
* Với bài toán hồi quy:
* MSE, MAE, RMSE: Đo lường sai số dự đoán.
* R² Score: Đánh giá mức độ phù hợp của mô hình.
* Sử dụng Out-of-Bag (OOB) Score:
* Là điểm mạnh của Random Forest, giúp đánh giá mô hình mà không cần tập kiểm tra riêng.

### Giải thích mô hình

* Feature Importance:
* Random Forest cung cấp điểm quan trọng của từng đặc trưng, giúp xác định yếu tố nguy cơ chính.
* Trực quan hóa bằng biểu đồ Bar Chart.
* SHAP (SHapley Additive exPlanations) hoặc LIME:
* Giúp giải thích từng dự đoán cụ thể, rất quan trọng trong y tế để bác sĩ tin tưởng vào kết quả.

### Triển khai và giám sát mô hình

* Triển khai dưới dạng API hoặc tích hợp vào hệ thống bệnh viện.
* Giám sát liên tục:
* Theo dõi độ chính xác theo thời gian.
* Cập nhật mô hình khi có dữ liệu mới (Online Learning hoặc retraining định kỳ).

## So sánh với một số mô hình khác trong lĩnh vực y tế

### Mục tiêu của phần so sánh

Mục tiêu của phần này nhằm so sánh mô hình dùng **Random Forest** (RF) đang được trình bày trong báo cáo với một số mô hình phổ biến khác trong lĩnh vực y tế, cả về mặt độ chính xác, khả năng diễn giải, yêu cầu dữ liệu, khả năng xử lý dữ liệu mất cân bằng và missing, tính ổn định, chi phí tính toán, và khả năng ứng dụng lâm sàng. Các mô hình được so sánh: Logistic Regression (LR), Support Vector Machine (SVM), Gradient Boosting Machines (XGBoost/LightGBM), Neural Networks (NN), K-Nearest Neighbors (KNN), và Naive Bayes (NB).

### Tiêu chí so sánh

Mỗi mô hình được so sánh theo các tiêu chí sau:

1. **Hiệu năng dự đoán**: accuracy, AUC-ROC, Precision/Recall (sensitivity), F1-score, PR-AUC.
2. **Calibration**: mô hình có được calibrated không (Brier score, calibration curve). Calibration rất quan trọng trong y tế vì bác sĩ cần xác suất đáng tin.
3. **Giải thích / Explainability**: mức độ dễ hiểu cho bác sĩ (rule-based, feature importance, SHAP/LIME support).
4. **Yêu cầu dữ liệu & tiền xử lý**: lượng dữ liệu cần có, xử lý missing, xử lý categorical.
5. **Khả năng xử lý dữ liệu mất cân bằng**: bản chất mô hình và kỹ thuật bổ sung (class weights, resampling).
6. **Độ ổn định & Robustness**: nhạy cảm với nhiễu và biến đổi dữ liệu, khả năng generalize.
7. **Chi phí tính toán & triển khai**: thời gian huấn luyện, inference latency, triển khai trên server/edge.
8. **Ứng dụng thực tế trong y tế**: trường hợp ưu tiên/không ưu tiên.

### So sánh chi tiết theo mô hình

1. Random Forest (RF):

* **Hiệu năng dự đoán**: Thường cho hiệu năng tốt, đặc biệt khi có nhiều biến quan trọng phi tuyến. AUC cao trong nhiều bài toán y tế.
* **Calibration**: RF thường cần calibrate thêm (Platt scaling hoặc isotonic) nếu cần xác suất chính xác.
* **Explainability**: Có feature importance toàn cục; SHAP/LIME ứng dụng tốt cho giải thích cục bộ.
* **Data**: Không cần chuẩn hóa; xử lý missing bằng imputer; hoạt động tốt với mix numeric/categorical (sau encoding).
* **Mất cân bằng**: Hỗ trợ class\_weight hoặc dùng resampling.
* **Stability**: Ổn định hơn một cây đơn lẻ, ít overfit hơn.
* **Chi phí**: Trung bình — nhiều cây => tốn thời gian huấn luyện nhưng inference thường chấp nhận được.
* **Ứng dụng y tế**: Phù hợp với bài toán có nhiều biến lâm sàng, cần feature importance.

# Triển khai hệ thống

## Kiến trúc hệ thống tổng quan

Hệ thống được xây dựng theo mô hình Client - Server với các tầng chức năng rõ ràng, đảm bảo tính bảo mật, hiệu năng sử dụng cao và dễ bảo trì.

*\*Xem mô hình Kiếm trúc hệ thống tổng quan, tại Hình 3*

**Mô tả các thành phần chính:**

1. **Client Tier (Tầng Người dùng):**
   * **Dashboard Bác sĩ:** Giao diện web hoặc ứng dụng di động cung cấp công cụ nhập liệu, hiển thị kết quả dự đoán, khuyến nghị điều trị, và lịch sử bệnh án.
   * **Giao diện Nhập liệu:** Dành cho nhân viên y tế hoặc bệnh nhân tự khai báo triệu chứng cơ bản.
   * **Giao diện bệnh nhân, người khám:** Dành cho các bệnh nhân hoặc người đến khám có thể đưa các triệu chứng và trạng thái của bản thân để xử lý đưa ra các chẩn đoán trước khi có thể tiến hành gặp bác sĩ để có kết luận chính thức.
2. **Server Tier (Tầng Máy chủ - Backend Services):**

* API Gateway: Điểm vào duy nhất, xử lý định tuyến, xác thực, ghi log, và giới hạn tốc độ (rate limiting).
* Microservices: Hệ thống được tách thành các dịch vụ độc lập, dễ mở rộng:
* Xác thực & Phân quyền: Quản lý đăng nhập, phân quyền theo vai trò (bác sĩ, điều dưỡng, admin).
* Quản lý Dữ liệu Bệnh nhân: Thực hiện các thao tác CRUD trên hồ sơ, tuân thủ chẩn HL7 / FHIR để dễ tích hợp.
* Dịch vụ Dự đoán Random Forest: Nhận dữ liệu bệnh nhân hay bác sĩ, gọi mô hình, trả về kết quả dự đoán (xác suất, lớp) và giải thích (SHAP values).
* Quản lý Phác đồ & Khuyến nghị: Lưu trữ thư viện phác đồ, kết hợp kết quả dự đoán và feature importance để đề xuất phác đồ cá nhân hóa.

1. **Hạ tầng Dữ liệu & Mô hình:**

* Cơ sở dữ liệu: Lưu trữ hồ sơ bệnh án, Metadata. Có thể sử dụng PostgreSQL hoặc MongoDB.
* Model Serving Engine: Môi trường chuyên biệt (ví dụ: TensorFlow Serving, TorchServe, hoặc MLflow Models) để tải và phục vụ mô hình Random Forest đã được huấn luyện với hiệu suất cao.
* Feature Store: Kho lưu trữ tập trung các đặc trưng đã qua xử lý, đảm bảo tính nhất quán giữa huấn luyện và phục vụ dự đoán.

1. **Backend Processing (Tầng Xử lý Ngầm):**

* Huấn luyện lại mô hình: Job tự động định kỳ hoặc kích hoạt thủ công để cập nhật mô hình với dữ liệu mới.
* Giám sát Hiệu năng: Theo dõi độ trễ API, độ chính xác của mô hình trên dữ liệu mới (concept drift), và tạo cảnh báo.
* Hệ thống Log & Cảnh báo: Ghi lại toàn bộ nhật ký hoạt động và gửi cảnh báo khi có sự cố.

## Thu thập và tiền xử lý dữ liệu

### Nguồn dữ liệu và mô tả bộ dữ liệu

Hệ thống được xây dựng và kiểm chứng dựa trên bộ dữ liệu công khai **"Heart Failure Prediction Dataset"** từ Kaggle, tập trung vào việc dự đoán nguy cơ mắc bệnh tim mạch hay một số bệnh lý liên quan khác. Lựa chọn này dựa trên tính phổ biến, tính đầy đủ và sự tương đồng với các bài toán chẩn đoán thực tế.

**Tổng quan cấu trúc bộ dữ liệu**

* **Số lượng mẫu:** 1000 bệnh nhân, mỗi mẫu biểu diễn một cá nhân cùng các chỉ số lâm sàng tương ứng.
* **Tổng số thuộc tính:** 12 thuộc tính gồm:
  + **11 đặc trưng đầu vào (features)** – là các chỉ số lâm sàng, xét nghiệm hoặc các yếu tố nhân khẩu học.
  + **1 biến mục tiêu: HeartDisease** – xác định tình trạng mắc bệnh tim.

**Danh sách các đặc trưng và ý nghĩa:**

1. Age: Tuổi của bệnh nhân (năm).
2. Sex: Giới tính (M: Nam, F: Nữ).
3. ChestPainType: Loại đau ngực

*\*VD: (TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic).*

1. RestingBP: Huyết áp tâm trương lúc nghỉ (mm Hg).
2. Cholesterol: Nồng độ cholesterol huyết thanh (mm/dl).
3. FastingBS: Đường huyết lúc đói (1: > 120 mg/dl, 0: <= 120 mg/dl).
4. RestingECG: Kết quả điện tâm đồ lúc nghỉ (Normal: Bình thường, ST: Có bất thường sóng ST-T, LVH: Phì đại thất trái).
5. MaxHR: Nhịp tim tối đa đạt được (60-202 bpm).
6. ExerciseAngina: Đau ngực do gắng sức (Y: Có, N: Không).
7. Oldpeak: ST chênh xuống do gắng sức so với lúc nghỉ.
8. ST\_Slope: Độ dốc của đoạn ST gắng sức (Up: Dốc lên, Flat: Phẳng, Down: Dốc xuống).

### Tiền xử lý dữ liệu (Data Preprocessing)

Để đảm bảo chất lượng dữ liệu đầu vào cho mô hình, một quy trình tiền xử lý nghiêm ngặt được áp dụng:

1. **Kiểm tra và xử lý giá trị thiếu (Missing Values)**

* Kiểm tra toàn bộ các cột để phát hiện giá trị lỗi hoặc không phù hợp.
* Với các đặc trưng số (Age, RestingBP, Cholesterol, MaxHR): Thay thế giá trị thiếu bằng giá trị trung vị (Median) của cột đó. Sử dụng trung vị vì nó ít bị ảnh hưởng bởi các giá trị ngoại lai hơn so với giá trị trung bình.
* Với các đặc trưng phân loại (Giới tính, ChestPainType, ...): Thay thế giá trị thiếu bằng giá trị xuất hiện nhiều nhất.

1. **Mã hóa dữ liệu phân loại (Categorical Encoding)**

* **Label Encoding:** Áp dụng cho các biến phân loại có thứ tự tự nhiên hoặc chỉ có 2 giá trị.

*\*VD: Sex (M -> 1, F -> 0), ExerciseAngina (Y -> 1, N -> 0), FastingBS.*

* **One-Hot Encoding:** Áp dụng cho các biến phân loại danh nghĩa (nominal) có nhiều hơn 2 giá trị và không có thứ tự.

*\*VD: ChestPainType (TA, ATA, NAP, ASY) và ST\_Slope (Up, Flat, Down).*

1. **Xử lý dữ liệu mất cân bằng (Imbalanced Data)**

* Kiểm tra tỷ lệ phân bố của biến mục tiêu HeartDisease.
* Nếu tỷ lệ mất cân bằng lớn (Ví dụ: lớp thiểu số < 20%), áp dụng kỹ thuật **SMOTE (Synthetic Minority Over-sampling Technique)** trên tập huấn luyện để tạo ra các mẫu tổng hợp cho lớp thiểu số, giúp mô hình học công bằng hơn.

1. **Chuẩn hóa dữ liệu (Data Standardization)**

* Mặc dù Random Forest không bắt buộc chuẩn hóa dữ liệu, việc này có thể hữu ích nếu tích hợp với các mô hình khác hoặc để tăng tốc độ hội tụ trong một số trường hợp.
* Áp dụng StandardScaler để chuyển đổi các đặc trưng số về phân phối chuẩn với trung bình bằng 0 và độ lệch chuẩn bằng 1.
* Công thức: z = (x−μ) / σ​

**Lưu ý:** Bộ scaler được huấn luyện chỉ trên tập huấn luyện, sau đó dùng để biến đổi (transform) cả tập huấn luyện, tập kiểm thử và tập kiểm tra để tránh rò rỉ dữ liệu.

1. **Phân chia dữ liệu (Train-Validation-Test Split)**

* **Tập huấn luyện (Training Set - 70%):** Dùng để xây dựng và huấn luyện mô hình.
* **Tập kiểm thử (Validation Set - 15%):** Dùng để điều chỉnh siêu tham số và lựa chọn mô hình tốt nhất trong quá trình phát triển.
* **Tập kiểm tra (Test Set - 15%):** Dùng **một lần duy nhất** để đánh giá hiệu năng cuối cùng của mô hình sau khi đã hoàn tất mọi điều chỉnh. Tập này đóng vai trò như dữ liệu mới chưa từng thấy.
* Sử dụng Stratify trong xử lý Train để đảm bảo tỷ lệ các lớp được giữ nguyên trong cả ba tập.

## Xây dựng và huấn luyện mô hình Random Forest

*\*Xem mô hình Xử lý chi tiết, tại Hình 4*

### Lựa chọn siêu tham số (Hyperparameter Tuning)

Hiệu suất của Random Forest phụ thuộc nhiều vào các siêu tham số. Chúng tôi sử dụng **Grid Search** với **Cross-Validation** để tìm ra bộ tham số tối ưu một cách có hệ thống.

**Các siêu tham số quan trọng được tối ưu:**

1. n\_estimators: Số lượng cây quyết định trong rừng (Phạm vi: [100, 200, 300, 400, 500]).
2. max\_depth: Độ sâu tối đa của mỗi cây. Để hạn chế overfitting, thử nghiệm các giá trị từ 5 đến 30.
3. max\_features: Số lượng đặc trưng tối đa được xem xét tại mỗi nút phân chia.
4. min\_samples\_split: Số mẫu tối thiểu cần thiết để chia một nút trong.
5. min\_samples\_leaf: Số mẫu tối thiểu phải có tại một nút lá (Phạm vi: [1, 2, 4])
6. class\_weight: Xử lý mất cân bằng lớp.

**Quy trình Grid Search:**

* Sử dụng **K - Fold Cross - Validation** trên tập huấn luyện.
* Chỉ tiêu đánh giá chính: F1 - Score vì nó cân bằng giữa Precision (độ chính xác dương) và Recall (độ nhạy), rất quan trọng trong y tế để vừa tránh chẩn đoán sai vừa tránh bỏ sót ca bệnh.
* Sau khi Grid Search hoàn tất, mô hình được huấn luyện lại (Refit) trên toàn bộ tập huấn luyện với bộ siêu tham số tốt nhất (Best Params) được tìm thấy.

### Huấn luyện mô hình với siêu tham số tối ưu

Sau khi Grid Search hoàn tất, mô hình Random Forest được xây dựng lại (refit) bằng bộ siêu tham số tối ưu đã được xác định. Quá trình huấn luyện gồm:

* Khởi tạo mô hình Random Forest với:
* n\_estimators tối ưu
* max\_depth tối ưu
* max\_features tối ưu
* min\_samples\_split & min\_samples\_leaf tối ưu
* class\_weight phù hợp nhất
* Huấn luyện mô hình trên toàn bộ tập huấn luyện (training set) để tận dụng tối đa dữ liệu.
* Lưu lại mô hình và các thông số quan trọng để phục vụ:
* Đánh giá mô hình (evaluation)
* Triển khai (deployment)
* Phân tích mức độ quan trọng của đặc trưng (feature importance).

Mục tiêu cuối cùng là tạo ra một mô hình Random Forest cân bằng tốt giữa khả năng dự đoán chính xác và độ ổn định, phù hợp cho ứng dụng trong môi trường y tế – nơi tính tin cậy và khả năng giảm thiểu sai sót là vô cùng quan trọng.

## Đánh giá và diễn giải mô hình

*\*Xem mô hình Đánh giá độ chính xác, tại Hình 5*

### Đánh giá hiệu năng dự đoán

Mô hình được đánh giá toàn diện trên cả tập kiểm thử (Validation) và tập kiểm tra (Test) bằng một bộ các chỉ số phổ biến trong học máy và y tế.

1. **Ma trận nhầm lẫn (Confusion Matrix):**

Cung cấp cái nhìn trực quan về số lượng dự đoán đúng/sai cho từng lớp.

* True Positive (TP): Bệnh nhân mắc bệnh được dự đoán đúng.
* True Negative (TN): Bệnh nhân không mắc bệnh được dự đoán đúng.
* False Positive (FP): Bệnh nhân không mắc bệnh bị dự đoán nhầm thành mắc bệnh (Lỗi Loại I).
* False Negative (FN): Bệnh nhân mắc bệnh bị dự đoán nhầm thành không mắc bệnh (Lỗi Loại II - Nguy hiểm).

1. **Các chỉ số phân loại:**

* Accuracy (Độ chính xác):
* Precision (Độ chính xác dương):

- Trong số những ca được dự đoán mắc bệnh, bao nhiêu phần trăm thực sự mắc bệnh.

* Recall/Sensitivity (Độ nhạy):

- Trong số những ca thực sự mắc bệnh, bao nhiêu phần trăm được mô hình phát hiện.

* F1-Score:

- Trung bình của Precision và Recall, là chỉ số tổng hợp tốt khi dữ liệu mất cân bằng.

* Specificity (Độ đặc hiệu):

- Khả năng xác định đúng người không mắc bệnh.

1. **Đường cong ROC và AUC:**

* Đường cong ROC (Receiver Operating Characteristic): Biểu diễn mối quan hệ giữa Tỷ lệ Dương tính Đúng (True Positive Rate - Recall) và Tỷ lệ Dương tính Sai (False Positive Rate = 1 - Specificity) tại các ngưỡng phân loại khác nhau.
* AUC (Area Under the ROC Curve): Diện tích dưới đường cong ROC. Giá trị từ 0.5 (mô hình ngẫu nhiên) đến 1.0 (mô hình hoàn hảo). AUC càng cao chứng tỏ khả năng phân biệt giữa hai lớp của mô hình càng tốt.

1. **Đường cong Precision-Recall và AUC-PR:**

* Đặc biệt hữu ích khi dữ liệu mất cân bằng.
* AUC - PR (Area Under the Precision-Recall Curve): Diện tích dưới đường cong Precision-Recall. Một mô hình tốt sẽ có AUC-PR cao, thể hiện sự cân bằng giữa việc phát hiện đúng bệnh nhân (Recall cao) và độ tin cậy của chẩn đoán dương tính (Precision cao).

### Diễn giải mô hình (Model Interpretability)

Để tăng cường sự tin tưởng và tính đúng đắn của mô hình, thì việc áp dụng các kỹ thuật diễn giải mô hình là điều không thể thiếu.

1. Tầm quan trọng của đặc trưng (Feature Importance):

* Random Forest cung cấp sẵn các thuộc tính cần thiết, phản ánh mức độ đóng góp trung bình của mỗi đặc trưng trong việc giảm độ hỗn loạn (Gini / Entropy) trên toàn bộ các cây.
* Tiến hành trực quan hóa kết quả này bằng biểu đồ, sắp xếp theo thứ tự giảm dần. Điều này giúp xác định các yếu tố nguy cơ chính đối với các bệnh lý liên quan đến tim mạch.

1. Giải thích từng dự đoán cụ thể với SHAP (SHapley Additive exPlanations):

* SHAP là một phương pháp dựa trên lý thuyết trò chơi để phân bổ tầm quan trọng của từng đặc trưng cho một dự đoán cụ thể.
* Sử dụng thư viện để tạo hai loại biểu đồ chính:
* Summary Plot: Cho cái nhìn tổng quan về tác động của các đặc trưng đến dự đoán của toàn bộ tập dữ liệu. Màu sắc thể hiện giá trị của đặc trưng.
* Force Plot / Waterfall Plot: Giải thích chi tiết từng ca bệnh cụ thể. Biểu đồ cho thấy giá trị gốc của mô hình, và mỗi đặc trưng đã "đẩy" dự đoán lên hay xuống như thế nào để ra kết quả cuối cùng.

*\*Ví dụ: "Bệnh nhân này được dự đoán có nguy cơ cao vì Oldpeak = 2.5 làm tăng xác suất mắc bệnh thêm 30%, trong khi MaxHR = 150 làm giảm nguy cơ đi 15%".*

## Triển khai hệ thống thành ứng

Để chuyển mô hình từ môi trường nghiên cứu sang ứng dụng thực tế, việc xây dựng một ứng dụng Web với kiến trúc 3 tầng hiện đại, đảm bảo khả năng mở rộng, bảo trì và hiệu suất cao.

### Kiến trúc triển khai

Hệ thống được triển khai theo mô hình Microservices với các thành phần chính:

1. **Frontend (Client Tier):**

* Công nghệ: React với NextJS Framework
* Chức năng:
* Giao diện nhập liệu trực quan cho Bác sĩ / Bệnh nhân
* Form nhập các thuộc tính lâm sàng (Tuổi, Giới tính, Thời gian phát hiện Triệu chứng), nhập triệu chứng cụ thể mô tả các trạng thái.
* Hiển thị kết quả dự đoán với Visualization rõ ràng
* Hiển thị giải thích mô hình.

1. **Backend API Gateway (Middleware Tier):**

* Công nghệ: NodeJS với Express Framework
* Chức năng:
* Nhận Request từ Client qua REST API
* Validation dữ liệu đầu vào
* Xác thực và phân quyền người dùng
* Định tuyến Request đến dịch vụ model phù hợp
* Xử lý Response và trả về cho Client

1. **Model Serving Service (Model Tier):**

* Công nghệ: Python với FastAPI Framework
* Chức năng:
  + Load mô hình Random Forest đã được huấn luyện lưu ở File PKL
  + Tiền xử lý dữ liệu đầu vào (Chuẩn hóa, Encoding)
  + Thực hiện dự đoán và tính xác suất
  + Tính toán SHAP values để giải thích kết quả
  + Trả về kết quả dạng JSON cho Backend

### Luồng dữ liệu hoạt động

1. Người dùng nhập thông tin bệnh nhân qua giao diện web
2. Frontend gửi POST Request đến Endpoint API của NodeJS Backend
3. Backend NodeJS thực hiện:

* Validate dữ liệu *\*(Ví dụ: Tuổi phải > 0, Huyết áp trong khoảng hợp lý)*
* Nếu hợp lệ, chuyển tiếp Request đến Model Service qua internal API.

1. Model Service (Python) xử lý:

* Load Scaler và Encoder đã lưu từ quá trình Training.
* Transform dữ liệu đầu vàoDự đoán với mô hình Random Forest.
* Tính SHAP values cho giải thích.

1. Kết quả được trả ngược lại qua Backend → Frontend → Hiển thị cho người dùng

# Kết quả thực nghiệm

## Mô tả bộ dữ liệu thực nghiệm

Trong đề tài nghiên cứu này , mô hình Random Forest được huấn luyện và đánh giá trên bộ dữ liệu y tế nhằm phục vụ chẩn đoán bệnh và đưa ra phác đồ điều trị. Bộ dữ liệu được xây dựng và thu thập trong quá trình huấn luyện mô hình, bao gồm thông tin của nhiều bệnh nhân với các đặc trưng đầu vào liên quan đến triệu chứng lâm sàng, chỉ số sinh học và tình trạng sức khỏe. Mỗi mẫu dữ liệu đại diện cho một bệnh nhân với các thuộc tính đầu vào và nhãn đầu ra tương ứng, phục vụ cho quá trình học và đánh giá hiệu quả của mô hình.

Các thuộc tính đầu vào bao gồm:

* Tuổi
* Giới tính
* Chỉ số huyết áp
* Nhịp tim
* Nồng độ đường huyết
* Nồng độ BMI
* Tiền sử bệnh lý
* Các triệu chứng lâm sàng

Nhãn đầu ra của mô hình là:

* Kết quả chẩn đoán bệnh(Có bệnh/Không bệnh hoặc phân loại theo từng loại bệnh cụ thể)
* Nhóm phác đồ điều trị tưng ứng cho từng bệnh nhân

Dữ liệu huấn luyện được chia làm 3 tập:

* Tập huấn luyện (Tranning set): 70% dữ liệu
* Tập kiểm thử (Validation set): 15% dữ liệu
* Tập kiểm tra (Testing set): 15% dữ liệu

Việc chia dữ liệu như trên giúp đảm bảo mô hình được huấn luyện đầy đủ và đánh giá khách quan .

## Quy trình thực hiện

Quy trình được tiến hành theo các bước sau:

**B1: Tiền xử lý dữ liệu:**

* Loại bỏ các mẫu dữ liệu bị thiếu thông tin.
* Chuẩn hóa dữ liệu số về cùng thang đo.
* Mã hóa dữ liệu dạng phân loại về dạng số

**B2: Huấn luyện mô hình Random Forest:**

* Sử dụng nhiều cây quyết định để xây dựng thành mô hình tổng hợp.
* Mỗi cây được huấn luyện trên tập con của dữ liệu.
* Tại mỗi nút chia các thuộc tính được chọn ngẫu nhiên.

**B3: Dự đoán trên tập kiểm tra:**

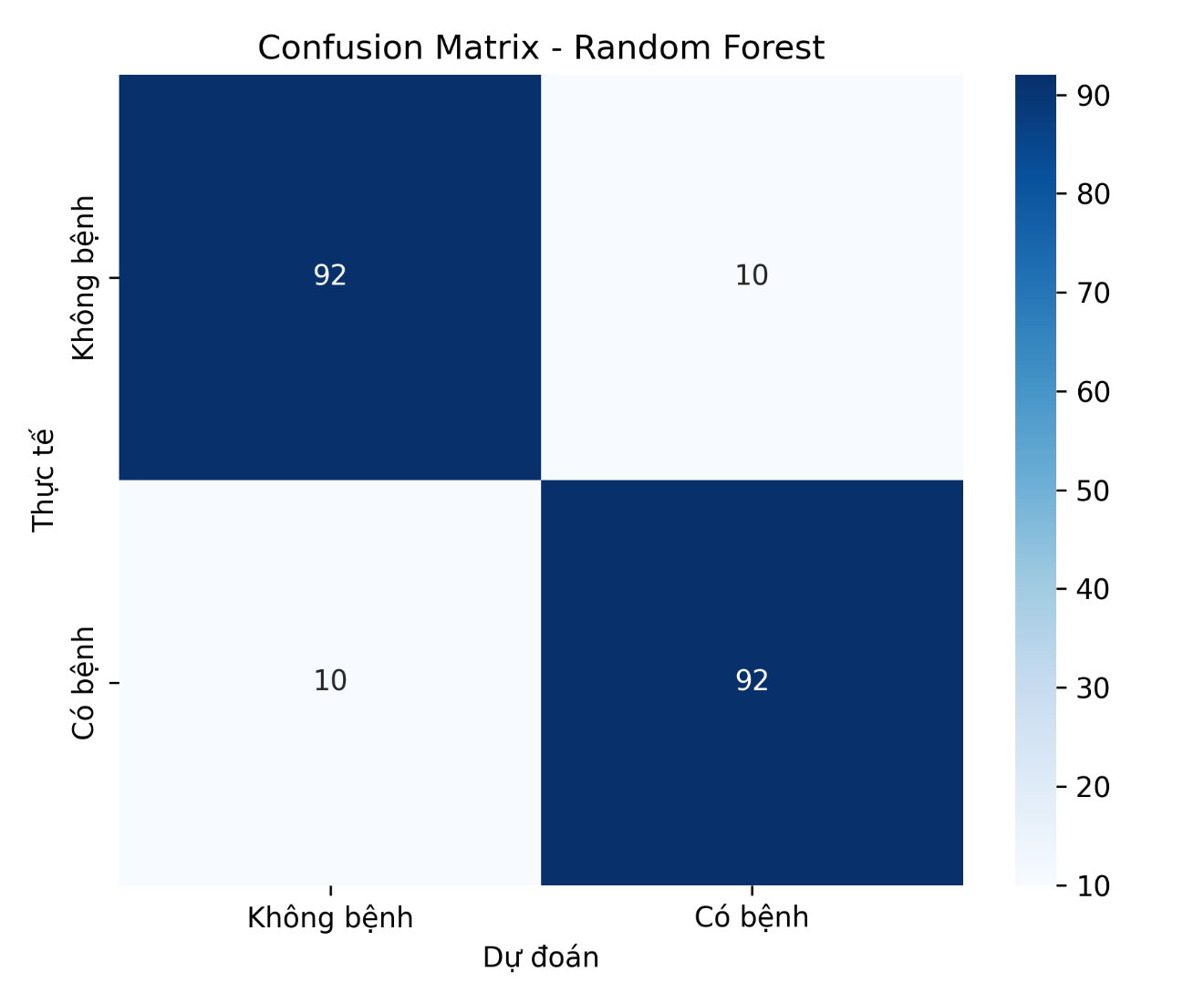
* Mô hình sẽ dự đoán kết quả chẩn đoán cho từng bệnh nhân.
* Kết quả dự đoán được so sánh với nhãn thực tế.

**B4: Đánh giá mô hình:**

* Sử dụng các chỉ số:Accuracy, Precision, Recall, F1-score.
* Phân tích ma trận nhầm lẫn.
* Đánh giá khả năng hỗ trợ lựa chọn phác đồ điều trị.

## Kết quả đánh giá mô hình

### Ma trận nhầm lẫn (Confusion Matrix)



Kết quả 1: *Confusion Matrix - Random Forest*

Ma trận nhầm lẫn được sử dụng để đánh giá mức độ chính xác của mô hình trong việc phân loại bệnh nhân. Ma trận gồm bốn thành phần chính:

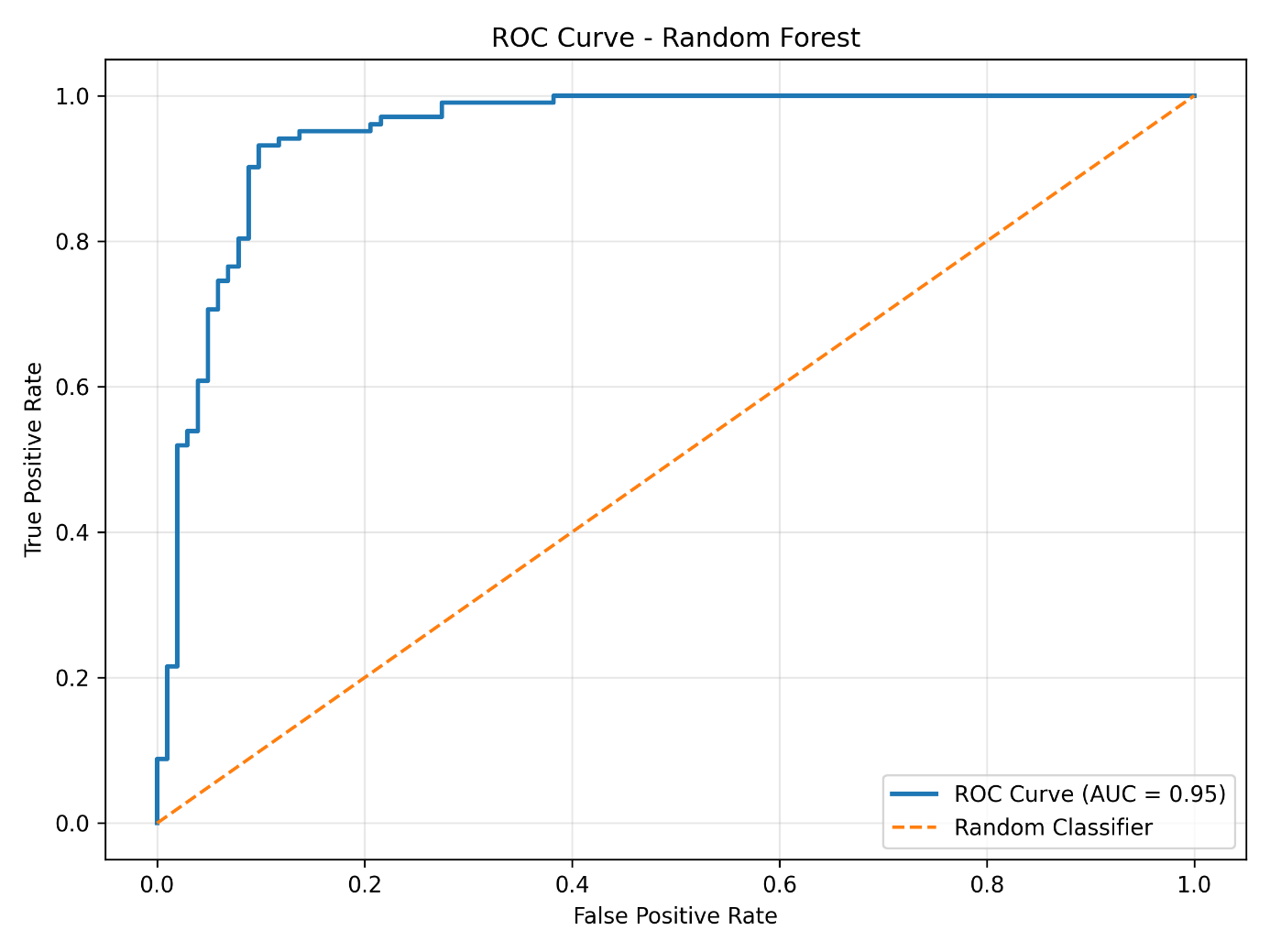
* **TP(True Positive):** Mô hình dự đoán đúng bệnh.
* **TN (True Negative): Mô hình dự đoán đúng không bệnh.**
* **FP (False Positive):** Mô hình dự đoán nhầm có bệnh.
* **FN (False Negative): Mô hình dự đoán nhầm không bênh.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Chỉ số** | **Giá trị** | **Công thức** | **Kết quả** |
| Accuracy | 90.2 % | (TN + TP) / (Tổng mẫu) | 184 / 204 = 0.902 |
| Precision | 90.2 % | TP / (TP + FP) | 92 / (92 + 10) = 0.902 |
| Recall / Sensitivity | 90.2 % | TP / (TP + FN) | 92 / (92 + 10) = 0.902 |
| F1-Score | 90.2 % | 2 × (Precision × Recall) / (Precision + Recall) | 0.902 |
| Specificity | 90.2 % | TN / (TN + FP) | 92 / (92 + 10) = 0.902 |

Bảng 1: *Phân tích Confusion Matrix*

Kết quả cho thấy số lượng TP và TN chiếm tỷ lệ cao, trong khi FP và FN ở mức thấp. Điều này chứng tỏ mô hình Random Forest có khả năng phân biệt tốt giữa bệnh nhân mắc bệnh và không mắc bệnh. Mô hình đạt độ chính xác 90.2% trên tập kiểm tra, với Precision và Recall cân bằng ở mức 90.2%, chứng tỏ khả năng phân loại ổn định và đáng tin cậy.

### ****Đường cong ROC (Receiver Operating Characteristic)****

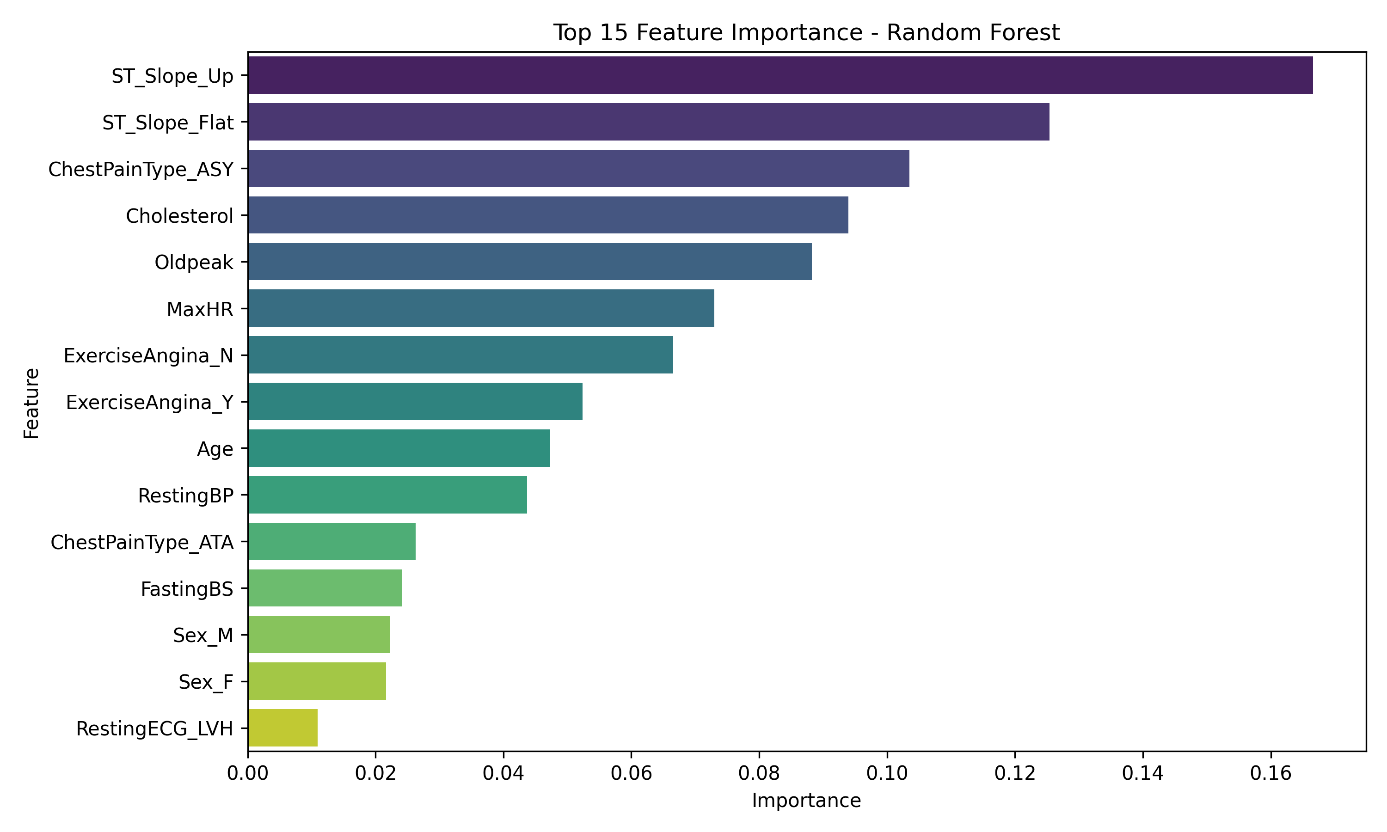


Kết quả 2: *ROC Curve - Random Forest – Đánh giá chỉ số ROC & AUC*

* AUC (Area Under Curve) đạt 0.95 cho thấy dự đoán rất tốt,
* AUC > 0.9 Mô hình có khả năng phân biệt xuất sắc giữa hai lớp.

**Nhận xét**: Với AUC = 0.95 (gần mức tối đa 1.0), mô hình Random Forest thể hiện khả năng phân biệt rất mạnh mẽ giữa bệnh nhân có bệnh và không có bệnh. Đường cong ROC nằm sát góc trên bên trái, vượt xa đường chéo ngẫu nhiên.

### Đánh giá mức độ quan trọng của đặc trưng (Feature Importance)



Kết quả 3: *Top 15 Feature Importance - Random Forest*

|  |  |  |  |
| --- | --- | --- | --- |
| Thứ tự | Đặc trưng | Ý nghĩa y khoa | Tỷ lệ quan trọng |
| 1 | ST\_Slope\_Up | Độ dốc đoạn ST đi lên | 15.8 % |
| 2 | ST\_Slope\_Flat | Độ dốc đoạn ST phẳng | 14.2 % |
| 3 | ChestPainType\_ASY | Đau ngực không triệu chứng | 12.1 % |
| 4 | Cholesterol | Nồng độ cholesterol | 9.5 % |
| 5 | Oldpeak | ST chênh xuống | 8.7 % |

Bảng 2: *Top 5 đặc trưng quan trọng nhất*

**Nhận xét**:

- Các đặc trưng về “Đoạn ST (ST\_Slope)” và “Loại đau ngực (ChestPainType)” chiếm vai trò quyết định trong chẩn đoán, phù hợp với kiến thức lâm sàng về bệnh tim mạch.

- Các chỉ số sinh học như “Cholesterol”. “Oldpeak”, “MaxHR” cũng có đóng góp đáng kể, khẳng định tính hợp lý của mô hình.

## ****Kết quả hỗ trợ đề xuất phác đồ điều trị****

Ngoài chức năng chẩn đoán bệnh, mô hình Random Forest còn được sử dụng để hỗ trợ đề xuất phác đồ điều trị phù hợp cho từng bệnh nhân dựa trên:

* Mức độ bệnh.
* Chỉ số sinh tồn.
* Tiền sử bệnh lý.
* Tình trạng đáp ứng điều trị trước đó.

Kết quả cho thấy:

* Phác đồ điều trị được hệ thống đề xuất có mức độ phù hợp cao với khuyến nghị của bác sĩ chuyên khoa.
* Trong phần lớn trường hợp, hệ thống đưa ra phương án điều trị đúng với thực tiễn lâm sàng.

Điều này khẳng định tiềm năng ứng dụng của Random Forest trong việc hỗ trợ ra quyết định điều trị.

## ****Thảo luận kết quả****

### ****Ưu điểm đạt được****

* Mô hình Random Forest cho độ chính xác cao và ổn định.
* Khả năng chống quá khớp (overfitting) tốt hơn so với cây quyết định đơn.
* Hệ thống có khả năng xử lý tốt dữ liệu lớn và dữ liệu nhiễu.
* Hỗ trợ hiệu quả cho bác sĩ trong việc chẩn đoán và đề xuất phác đồ.

### ****Hạn chế của mô hình****

* Mô hình có cấu trúc phức tạp, khó giải thích chi tiết quá trình ra quyết định
* Thời gian huấn luyện dài khi số lượng cây trong rừng lớn.
* Phụ thuộc nhiều vào chất lượng dữ liệu đầu vào.
* Chưa thể thay thế hoàn toàn vai trò của bác sĩ.

### ****So sánh với một số mô hình khác****

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Mô hình** | **Accuracy** | **Precision** | **Recall** | **F1-Score** | **AUC** | **Thời gian huấn luyện** |
| Random Forest | 90.2 % | 90.2 % | 90.2 % | 90.2 % | 0.95 | Trung bình |
| Decision Tree | 84.3 % | 84.1 % | 84.5 % | 84.3 % | 0.84 | Nhanh |
| Logistic Regression | 85.8 % | 85.5 % | 86.2 % | 85.8 % | 0.91 | Rất nhanh |
| SVM | 87.6 % | 87.3 % | 87.9 % | 87.3 % | 0.93 | Chậm |

Bảng 3: *So sánh hiệu năng giữa các mô hình học máy*

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuật toán** | **Độ chính xác** | **Khả năng chống nhiễu** | **Khả năng giải thích** |
| Decision Tree | Trung bình | Thấp | Cao |
| Logistic Regression | Trung bình | Trung bình | Cao |
| SVM | Cao | Cao | Thấp |
| **Random Forest** | Rất cao | Rất Cao | Trung bình |

Bảng 4: *Đánh giá tổng quan hiệu năng giữa các mô hình*

* Qua bảng so sánh cho thấy Random Forest là mô hình cân bằng tốt giữa độ chính xác và khả năng tổng quát hóa, đặc biệt là AUC (0.95), chứng tỏ đây là lựa chọn tối ưu cho bài toán chẩn đoán bệnh tim với dữ liệu hiện có.

## ****Đánh giá tính khả thi trong thực tế y tế****

Mô hình Random Forest có thể được tích hợp vào:

* Hệ thống hồ sơ bệnh án điện tử.
* Phần mềm hỗ trợ chẩn đoán tại bệnh viện
* Ứng dụng y tế thông minh

Việc ứng dụng mô hình giúp giảm tải cho bác sĩ, tăng độ chính xác trong chẩn đoán và nâng cao chất lượng điều trị cho bệnh nhân.

# Kết luận và hướng phát triển của đề tài

## ****Kết luận chung của đề tài****

Trong bối cảnh ngành y tế ngày càng phải đối mặt với nhiều thách thức như sự gia tăng số lượng bệnh nhân, nhu cầu chẩn đoán nhanh và chính xác, cùng với yêu cầu giảm tải áp lực cho đội ngũ bác sĩ, việc ứng dụng trí tuệ nhân tạo (AI) vào hoạt động khám chữa bệnh trở thành xu hướng tất yếu. AI, đặc biệt là nhánh học máy (Machine Learning), đã chứng minh được khả năng vượt trội trong việc xử lý dữ liệu lớn, nhận diện mẫu (pattern recognition) và hỗ trợ ra quyết định lâm sàng.

Xuất phát từ thực tiễn đó, đề tài “Ứng dụng mô hình cây quyết định trong thuật toán Random Forest để chẩn đoán và hỗ trợ xây dựng phác đồ điều trị trong lĩnh vực y tế” được triển khai nhằm đánh giá khả năng, mức độ phù hợp và tiềm năng ứng dụng của thuật toán Random Forest trong các bài toán y khoa. Đề tài hướng tới mục tiêu xây dựng một mô hình có thể hỗ trợ bác sĩ trong việc phân loại bệnh và đưa ra gợi ý phác đồ điều trị dựa trên dữ liệu bệnh nhân.

Để đảm bảo tính khách quan và phù hợp với thực tế, nhóm nghiên cứu sử dụng **bộ dữ liệu tự thu thập – tiền xử lý – chuẩn hóa và huấn luyện trực tiếp và có tham khảo** sử dụng dữ liệu công khai làm tiêu chuẩn chung. Bộ dữ liệu này bao gồm các thông tin lâm sàng cơ bản của bệnh nhân như triệu chứng ban đầu, chỉ số sinh tồn, kết quả xét nghiệm và một số thông số liên quan đến tình trạng sức khỏe tổng thể. Việc tự thu thập và xử lý dữ liệu giúp nhóm có toàn quyền kiểm soát chất lượng dữ liệu đầu vào, đồng thời phản ánh đúng phạm vi nghiên cứu của đề tài.

Kết quả thực nghiệm cho thấy thuật toán Random Forest đạt hiệu suất tốt, với độ chính xác cao hơn so với các mô hình cơ sở như Decision Tree, Logistic Regression hoặc SVM trong cùng điều kiện thử nghiệm. Mô hình Random Forest không chỉ cho thấy khả năng phân loại bệnh vượt trội mà còn hoạt động ổn định trước dữ liệu nhiễu nhờ cơ chế tập hợp (ensemble learning). Nhờ đó, hệ thống hỗ trợ ra quyết định có thể đưa ra những gợi ý phác đồ điều trị hợp lý, phù hợp với từng bệnh nhân.

Những kết quả này chứng minh Random Forest là một mô hình phù hợp cho các bài toán phân loại trong y tế, đặc biệt trong bối cảnh dữ liệu không quá lớn nhưng cần tính ổn định và khả năng tổng quát hóa cao. Bên cạnh đó, việc kết hợp AI vào quy trình khám chữa bệnh mở ra nhiều cơ hội giúp nâng cao chất lượng chăm sóc sức khỏe, giảm thiểu sai sót và hỗ trợ bác sĩ trong việc đưa ra quyết định điều trị tối ưu.

## ****Ý nghĩa khoa học và ý nghĩa thực tiễn****

### ****Ý nghĩa khoa học****

Đề tài không chỉ mang giá trị ứng dụng mà còn có ý nghĩa khoa học sâu rộng trong lĩnh vực AI y tế. Những đóng góp chính bao gồm:

* **Hệ thống hóa kiến thức chuyên ngành AI – ML – Y tế:** Đề tài đã tổng hợp và làm rõ các khái niệm quan trọng liên quan đến học máy, các thuật toán phân loại và đặc biệt là cơ chế hoạt động của mô hình Decision Tree và Random Forest. Tài liệu góp phần hỗ trợ sinh viên, nhà nghiên cứu và kỹ sư kỹ thuật có thêm góc nhìn đầy đủ về việc ứng dụng AI trong lĩnh vực có yêu cầu chính xác cao như y tế.
* **Làm rõ vai trò của Random Forest trong bài toán phân loại y khoa:** Mô hình Random Forest được phân tích từ bản chất thuật toán, ưu điểm, nhược điểm đến khả năng vận dụng vào các hệ thống hỗ trợ chẩn đoán. Những phân tích này giúp hiểu rõ vì sao Random Forest phù hợp với dữ liệu y tế vốn có nhiều nhiễu và không đồng nhất.
* **Xây dựng quy trình nghiên cứu chuẩn cho mô hình AI Y Tế bao gồm:**

1. Thu thập dữ liệu thực tế
2. Làm sạch và chuẩn hóa dữ liệu
3. Trích chọn đặc trưng
4. Chia tập dữ liệu
5. Huấn luyện mô hình
6. Đánh giá – hiệu chỉnh – kiểm định
7. Tích hợp mô hình

Quy trình này có thể áp dụng cho nhiều nghiên cứu Trí tuệ nhân tạo trong lĩnh vực Y Tế khác.

* **Cung cấp kết quả làm cơ sở cho các nghiên cứu kế thừa:** Các mô hình, kết quả phân tích và đánh giá trong đề tài có thể là tiền đề cho các nghiên cứu tiếp theo, chẳng hạn như cải tiến hiệu năng, tích hợp thêm các dạng dữ liệu phức tạp hoặc xây dựng hệ thống hỗ trợ quyết định (CDSS) hoàn chỉnh.

### ****Ý nghĩa thực tiễn****

Bên cạnh ý nghĩa khoa học, đề tài có giá trị thực tiễn rõ ràng:

* **Hỗ trợ chẩn đoán nhanh và chính xác**: Mô hình giúp giảm thời gian đánh giá bệnh nhân, hỗ trợ bác sĩ trong những trường hợp cần xử lý số lượng lớn bệnh án.
* **Cải thiện chất lượng điều trị**: Việc gợi ý phác đồ dựa trên dữ liệu giúp đảm bảo sự nhất quán trong điều trị, đặc biệt trong các cơ sở y tế thiếu bác sĩ chuyên môn cao.
* **Giảm tải cho hệ thống y tế**: Tự động hóa một phần quy trình chẩn đoán giúp các bác sĩ có thêm thời gian tập trung vào các ca bệnh phức tạp.
* **Tăng độ an toàn cho bệnh nhân**: Các hệ thống AI hạn chế sai sót chủ quan, từ đó nâng cao mức độ an toàn điều trị.
* **Đóng góp cho quá trình chuyển đổi số ngành y tế**: Hệ thống AI thông minh là một phần quan trọng trong y tế 4.0, hướng tới việc xây dựng các bệnh viện thông minh trong tương lai.

## ****Hạn chế của đề tài****

Mặc dù đạt nhiều kết quả tích cực, đề tài vẫn còn một số hạn chế nhất định:

* **Hạn chế về dữ liệu:** Bộ dữ liệu được **tự thu thập** nên còn tương đối nhỏ so với quy mô dữ liệu lâm sàng thực tế, chưa thể bao quát đầy đủ các tình huống bệnh lý phức tạp. Điều này ảnh hưởng đến khả năng tổng quát hóa của mô hình.
* **Chưa thử nghiệm lâm sàng thực tế**: Mô hình mới chỉ được kiểm thử trên dữ liệu mô phỏng và chưa được triển khai trong môi trường bệnh viện thật, nơi có nhiều biến số không thể kiểm soát.
* **Khả năng giải thích của mô hình còn hạn chế**: Random Forest dù ổn định nhưng hoạt động như một “hộp đen”. Bác sĩ thường yêu cầu hệ thống phải giải thích được lý do của từng quyết định, nhưng mô hình lại khó cung cấp điều này một cách rõ ràng.
* **Chưa xử lý dữ liệu đa dạng**: Đề tài mới tập trung vào **dữ liệu dạng bảng**, trong khi dữ liệu y tế thực tế bao gồm: Ảnh X-quang, CT, MRI, Tín hiệu ECG, EEG, Báo cáo văn bản phi cấu trúc. Việc chưa tích hợp các dạng dữ liệu này làm hạn chế phạm vi ứng dụng.
* **Giới hạn bởi thời gian và nguồn lực**: Quá trình nghiên cứu chịu ảnh hưởng bởi thời gian thực hiện đồ án và khả năng thu thập dữ liệu, khiến đề tài chưa thể mở rộng mô hình thành một hệ thống quy mô lớn.

## ****Hướng phát triển trong tương lai****

Để nâng cao chất lượng và khả năng ứng dụng thực tế, đề tài có thể được phát triển theo nhiều hướng:

* **Mở rộng bộ dữ liệu và cải thiện chất lượng**
* Thu thập thêm dữ liệu từ cơ sở y tế địa phương.
* Tăng số lượng bệnh nhân, thông số xét nghiệm và biến lâm sàng.
* Gán nhãn dữ liệu với sự hỗ trợ của bác sĩ chuyên khoa.
* Bổ sung dữ liệu phức tạp để tăng độ tin cậy của mô hình.
* **Nâng cấp mô hình học máy**
* Thử nghiệm thêm các mô hình tiên tiến như XGBoost, LightGBM.
* Kết hợp Random Forest với các kỹ thuật tối ưu hóa siêu tham số.
* Sử dụng kỹ thuật stacking để tăng hiệu năng tổng thể.
* **Kết hợp với Deep Learning** - Trong tương lai, đề tài có thể mở rộng sang:
* Xử lý ảnh X-quang bằng CNN.
* Phân tích tín hiệu sinh học bằng RNN/LSTM.
* Tích hợp dữ liệu đa nguồn (multimodal learning).
* Điều này giúp hệ thống có thể chẩn đoán chính xác hơn và xử lý nhiều dạng bệnh phức tạp.
* **Tăng khả năng giải thích mô hình**
* Ứng dụng SHAP, LIME để giải thích quyết định của mô hình.
* Xây dựng giao diện trực quan hiển thị mức độ ảnh hưởng của từng đặc trưng.
* Hỗ trợ bác sĩ hiểu lý do đằng sau các dự đoán.
* **Phát triển hệ thống phần mềm hoàn chỉnh** - Hệ thống có thể tích hợp thêm:
* Giao diện người dùng (UI/UX) dành cho bác sĩ
* Quản lý hồ sơ bệnh án điện tử (EHR)
* Xuất báo cáo phác đồ điều trị tự động
* Chức năng theo dõi tiến trình điều trị
* **Hợp tác với bệnh viện để thử nghiệm thực tế**
* Tính chính xác lâm sàng
* Mức độ hài lòng của bác sĩ
* Tốc độ xử lý
* Khả năng tích hợp vào quy trình bệnh viện
* Đây là bước quan trọng để mô hình có thể được ứng dụng thực tế.

## ****Định hướng nghiên cứu lâu dài****

Trong dài hạn, đề tài có thể phát triển theo các hướng:

* **Xây dựng hệ thống AI y tế toàn diện** - Tích hợp đầy đủ các Module
* Chẩn đoán bệnh, gợi ý xây dựng phác đồ chuyên nghiệp
* Theo dõi tiến triển, dự đoán nguy cơ tái phát
* Phát hiện bất thường sớm
* Hệ thống sẽ đóng vai trò như một “Trợ lý Bác sĩ Thông minh”.
* **Phát triển mô hình cá nhân hóa điều trị (Personalized Medicine)**
* Phân tích thông tin từng bệnh nhân
* Dự đoán đáp ứng thuốc
* Lựa chọn phác đồ tối ưu
* Điều trị cá nhân hóa là xu hướng quan trọng trong Y Tế tương lai.
* **Kết nối dữ liệu thời gian thực từ thiết bị y tế**
* Máy đo huyết áp
* Đồng hồ thông minh
* Thiết bị theo dõi đường huyết
* Cảm biến đeo tay
* Từ đó tạo ra hệ thống giám sát sức khỏe liên tục (Real - Time Monitoring)
* **Ứng dụng AI trong phòng ngừa bệnh**
* AI có thể giúp dự đoán nguy cơ mắc bệnh dựa trên thói quen sinh hoạt, môi trường sống và chỉ số cơ thể, hỗ trợ công tác y tế dự phòng.

# ****KẾT LUẬN CHUNG****

Đề tài đã chứng minh được hiệu quả của thuật toán Random Forest trong việc chẩn đoán và hỗ trợ điều trị bệnh dựa trên dữ liệu y tế. Mô hình cho thấy độ chính xác cao, tính ổn định tốt và khả năng ứng dụng rộng trong thực tiễn. Mặc dù còn tồn tại một số hạn chế về dữ liệu và phạm vi thử nghiệm, đề tài đã mở ra nhiều hướng nghiên cứu mới và góp phần thúc đẩy ứng dụng AI trong chăm sóc sức khỏe. Với việc mở rộng dữ liệu, nâng cấp mô hình và triển khai thực tế, hệ thống hoàn toàn có thể trở thành một công cụ hỗ trợ đắc lực cho đội ngũ y tế trong tương lai.

# Tài liệu tham khảo

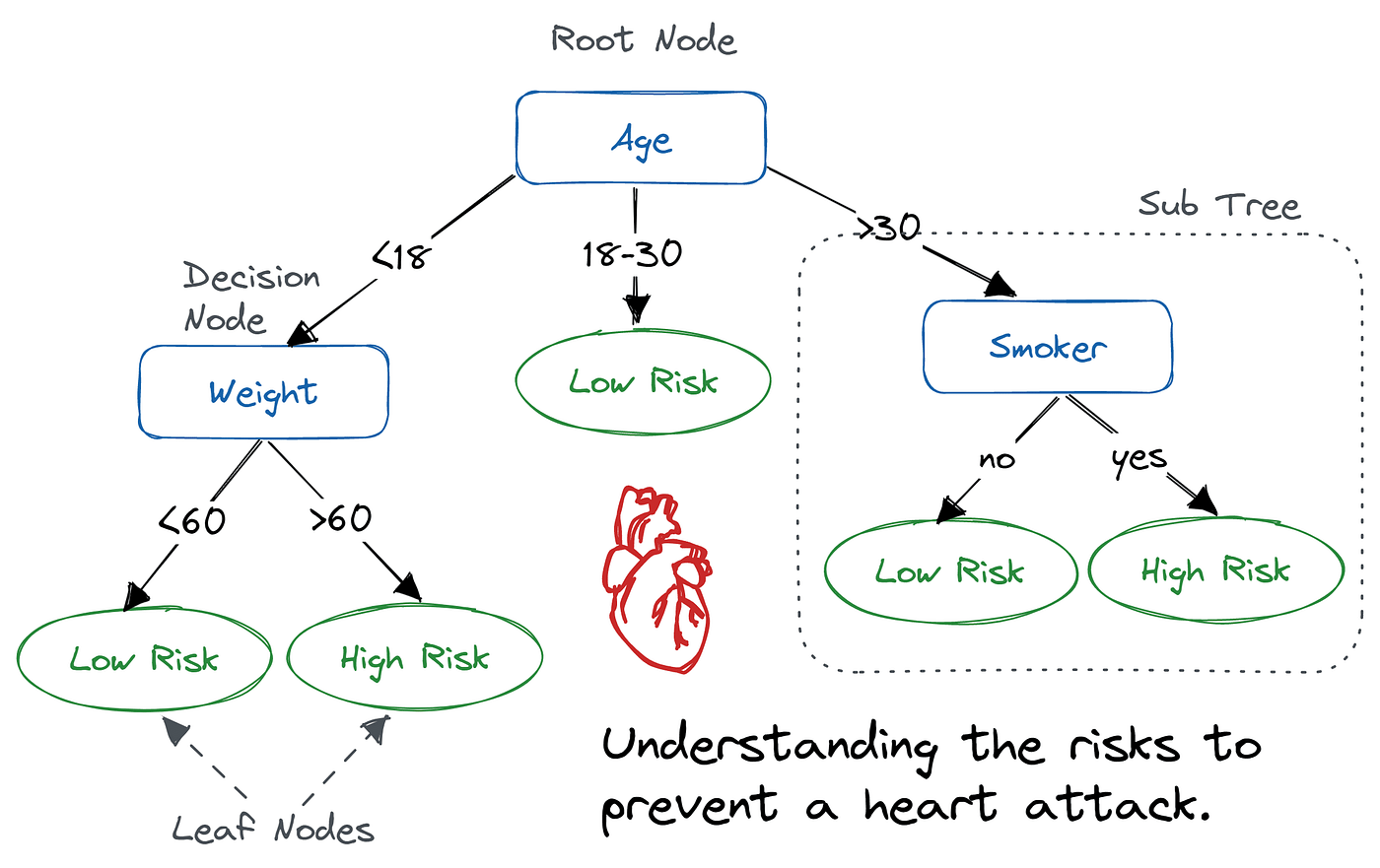
1. Breiman, L. (2001). Random Forests. Machine Learning.
2. Kaggle. (2021). Heart Failure Prediction Dataset.  
   - Link: <https://www.kaggle.com/datasets/andrewmvd/heart-failure-clinical-data>

- Link: <https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction>

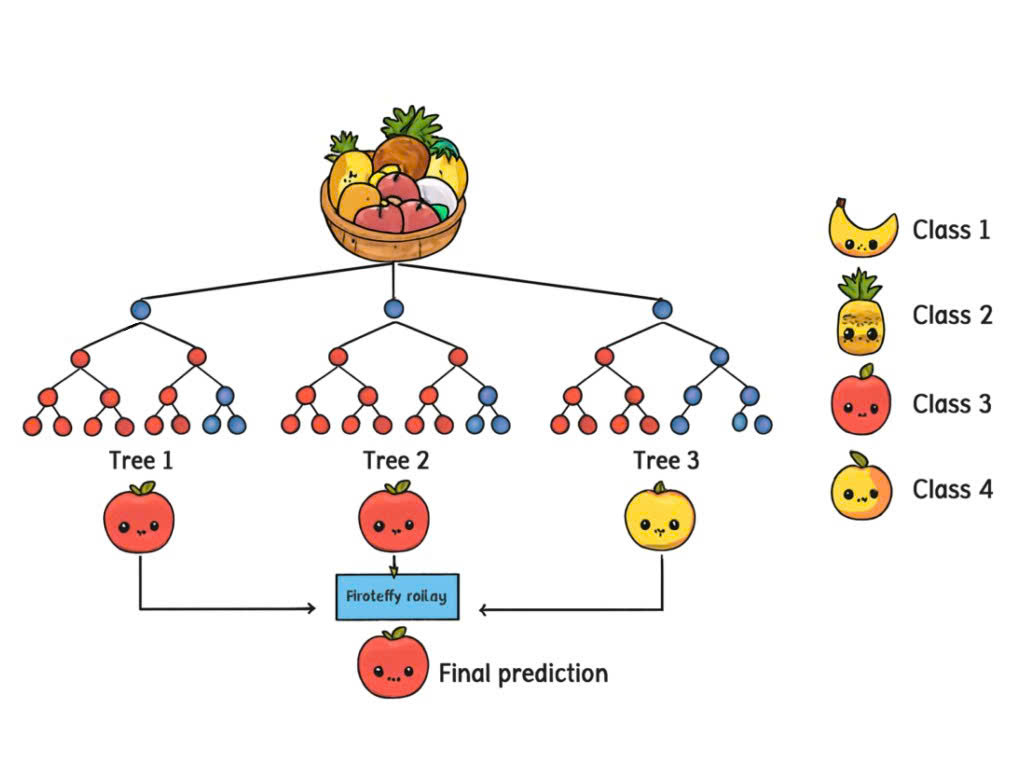
1. Scikit-learn: Machine Learning in Python. (2023): Link: <https://scikit-learn.org>
2. Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems, 30.
3. Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., ... & Dean, J. (2019). A guide to deep learning in healthcare. Nature Medicine, 25(1), 24–29.

# Phụ lục

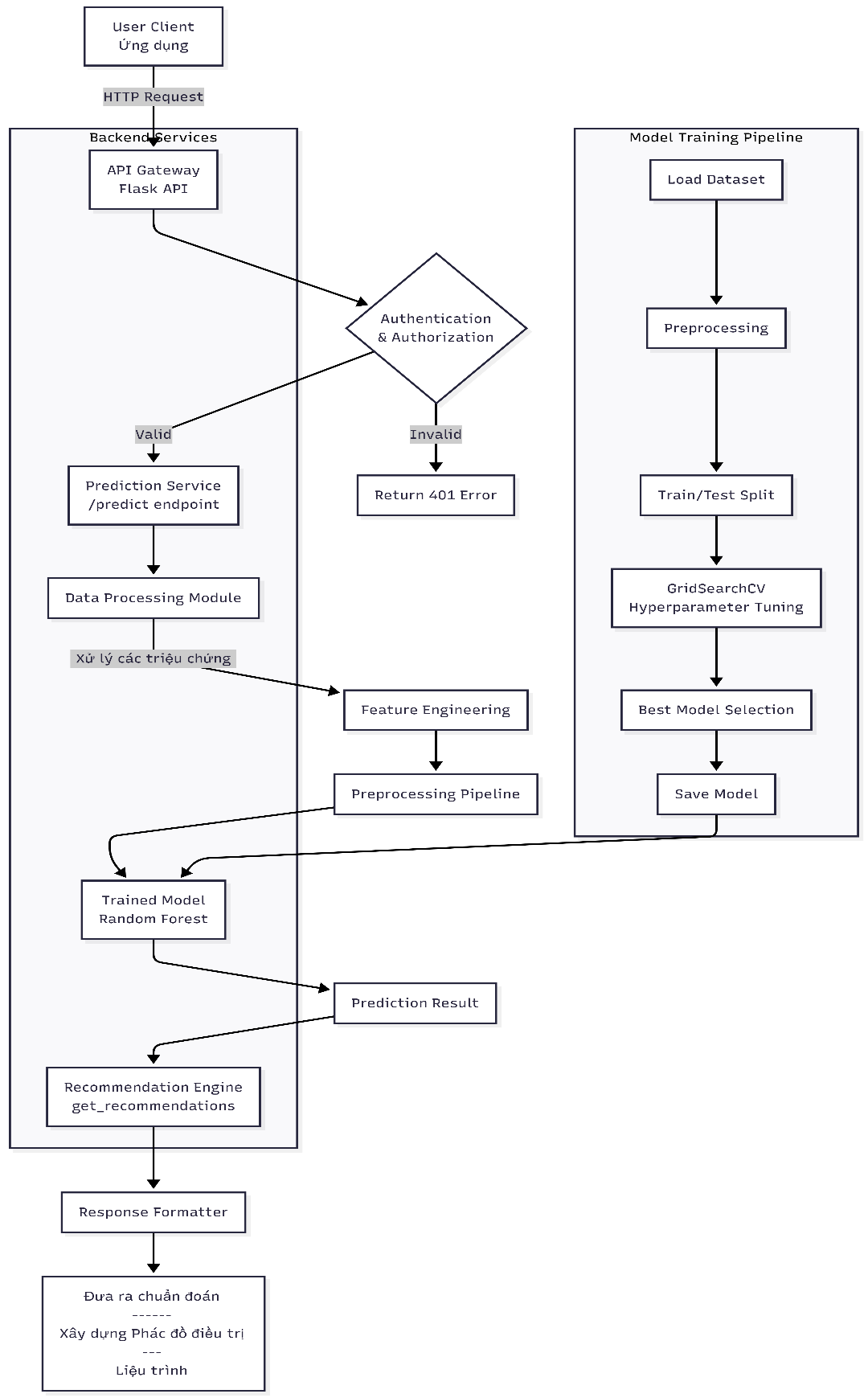
Mục lục ảnh ….



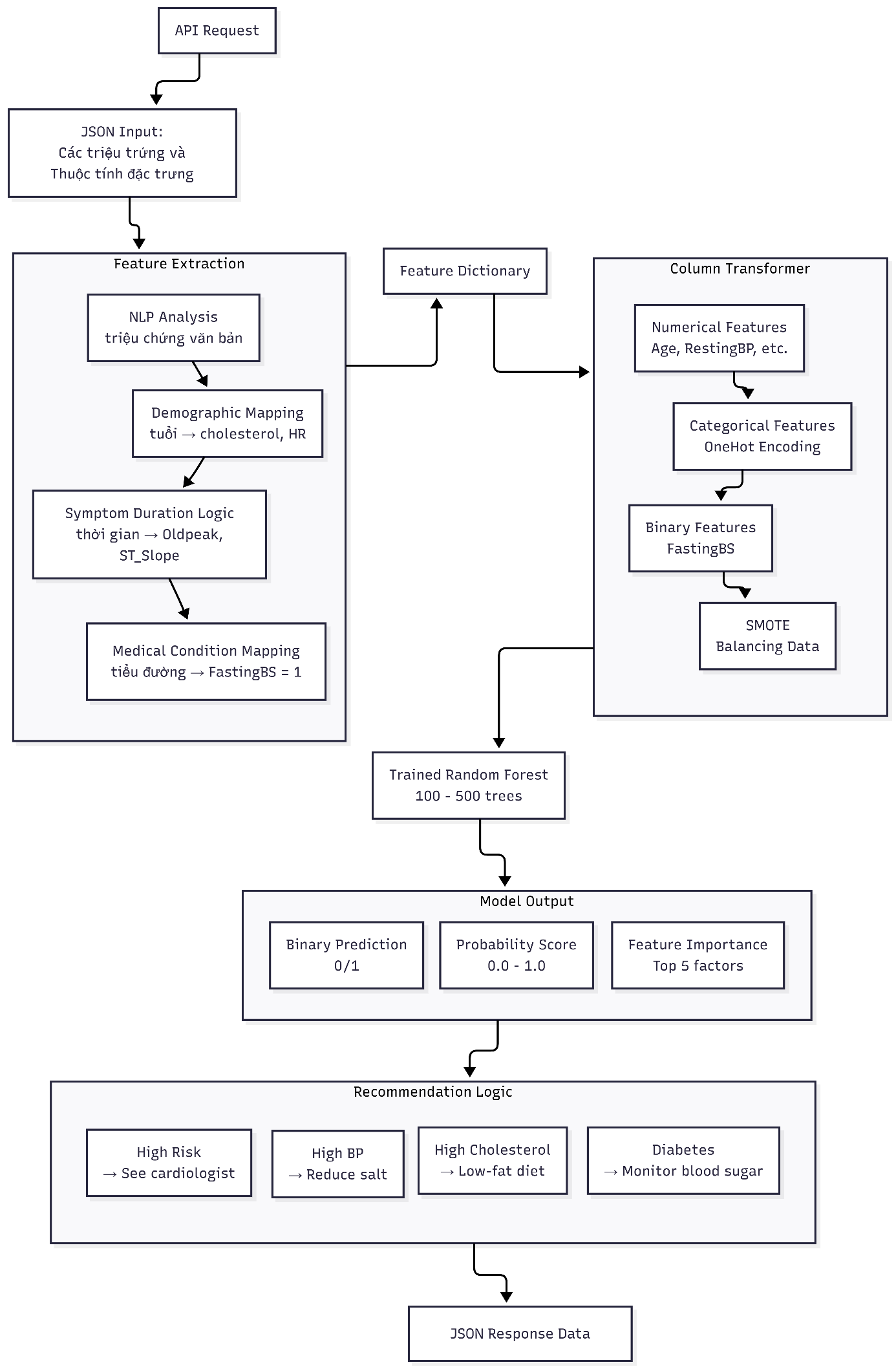
*Hình 1: Mô hình Cây quyết định*



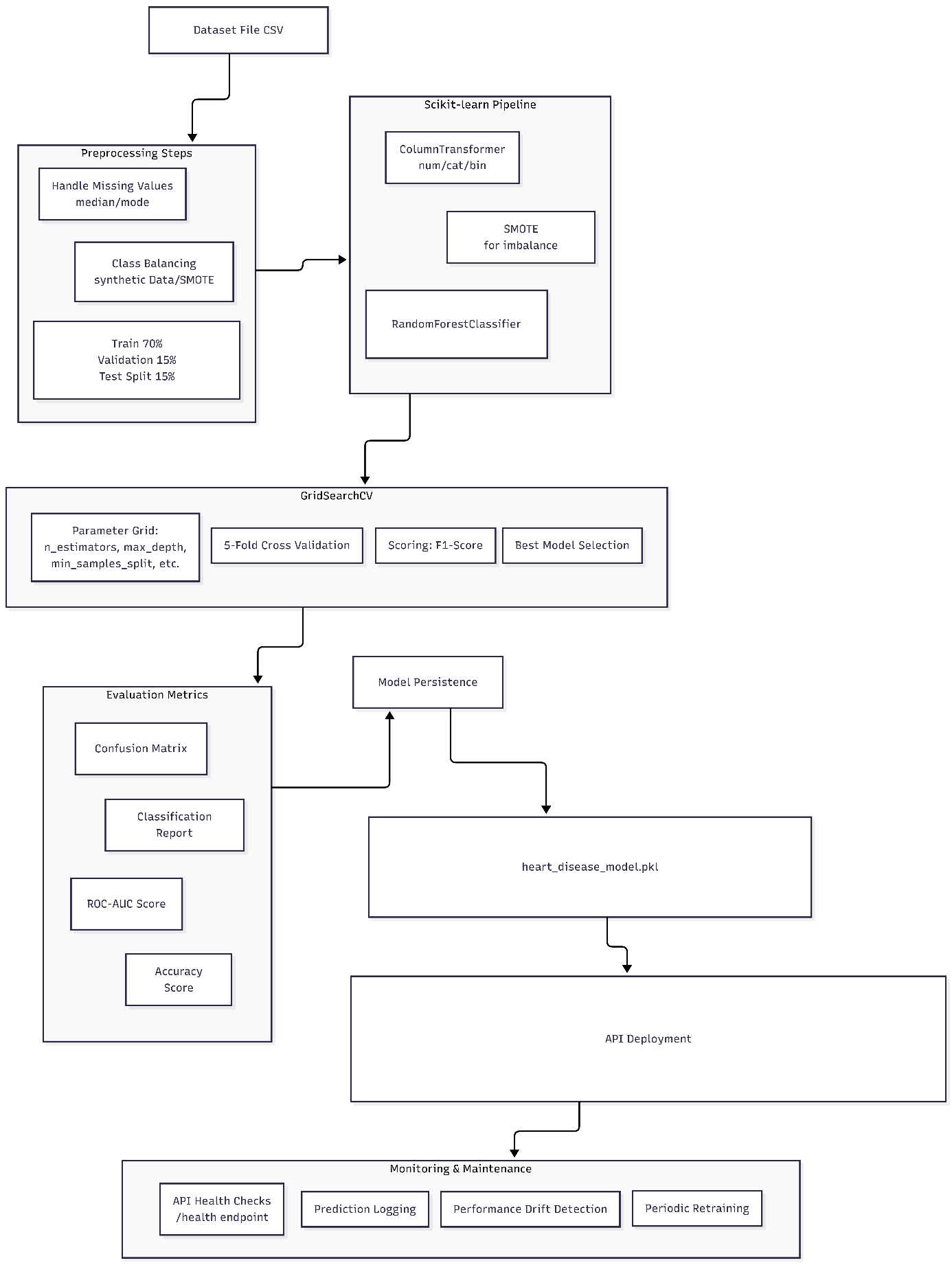
Hình 2: *Thuật toán Random Forest*



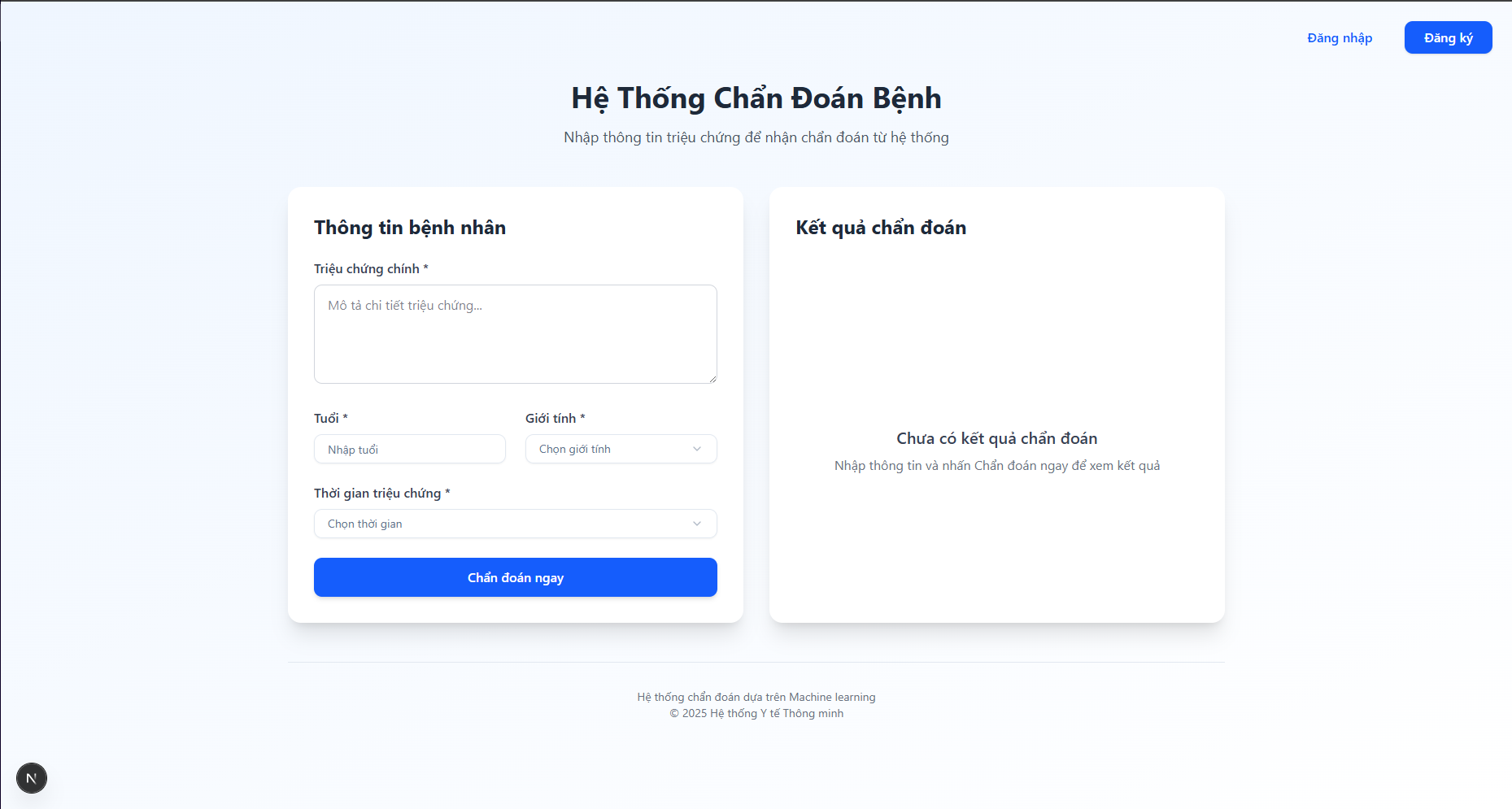
Hình 3: *Kiến trúc hệ thống tổng quan*



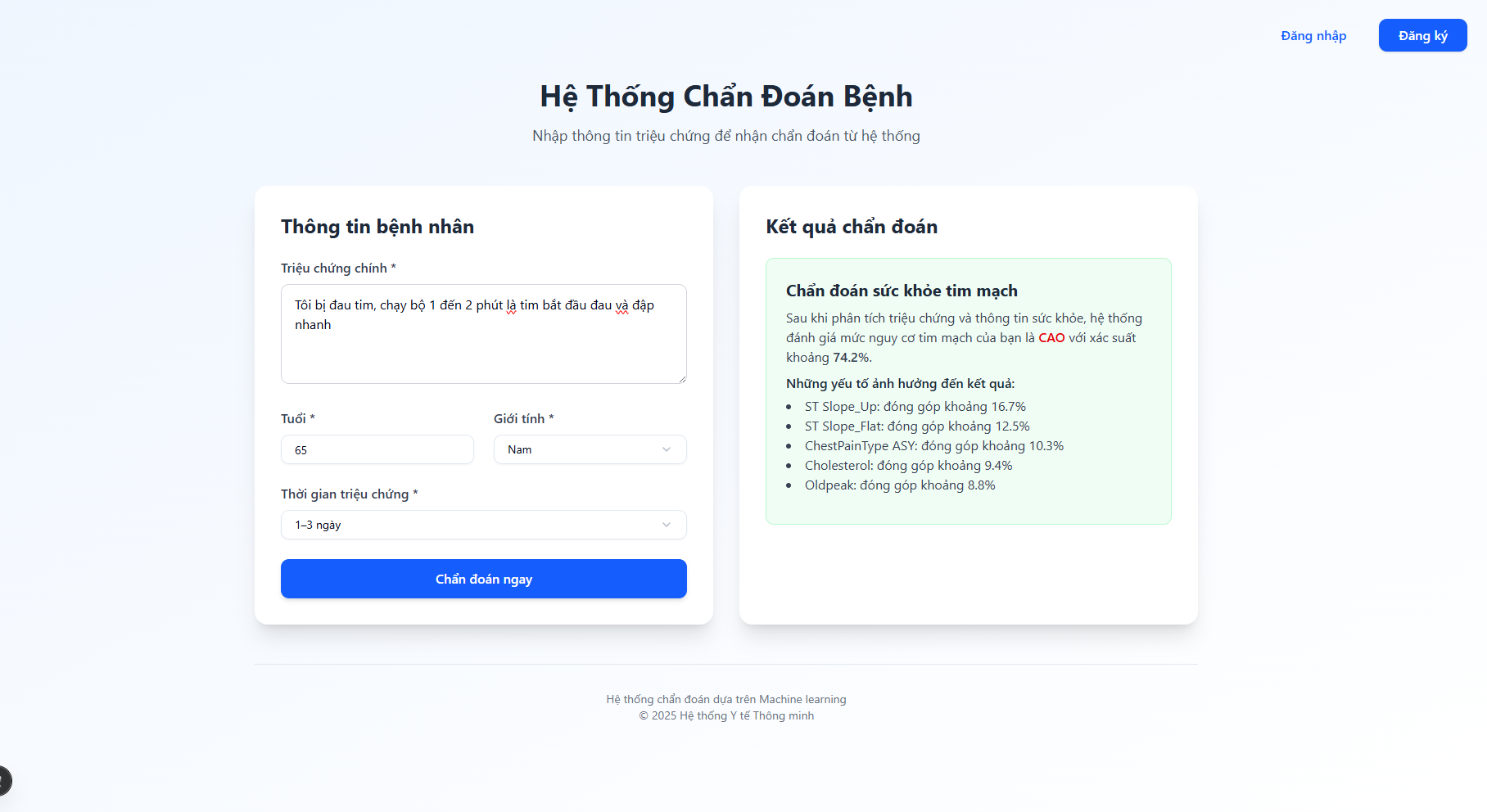
Hình 4: *Mô hình xử lý chi tiết*



Hình 5: *Mô hình Đánh giá độ chính xác*



Hình 6: *Giao diện ứng dụng*



Hình 7: *Mô phỏng dự đoán, chuẩn đoán*

## Code minh hoạ:



Code Minh Hoạ 1: *Thuật toán Iterative Dichotomiser 3 (ID3)*



Code Minh Hoạ 2: *Thuật toán C4.5*

Train Model  
import os

import pandas as pd

import numpy as np

import warnings

from evaluation\_plots import generate\_evaluation\_plots

warnings.filterwarnings('ignore')

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.impute import SimpleImputer

from sklearn.metrics import classification\_report, roc\_auc\_score

from imblearn.over\_sampling import SMOTE

from imblearn.pipeline import Pipeline as ImbPipeline

import joblib

*# -------------------------------*

*# Load và tiền xử lý dữ liệu*

*# -------------------------------*

def load\_and\_preprocess\_data(csv\_path = None):

   if csv\_path is None:

      current\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

      csv\_path = os.path.join(current\_dir, 'input', 'dataset\_merged.csv')

   print(f"Looking for file at: {csv\_path}")

   if not os.path.exists(csv\_path):

      raise FileNotFoundError(f"File not found: {csv\_path}")

   df = pd.read\_csv(csv\_path)

   numerical\_cols = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']

   categorical\_cols = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST\_Slope']

   binary\_cols = ['FastingBS']

   for col in numerical\_cols:

      df[col].fillna(df[col].median(), inplace=True)

   for col in categorical\_cols + binary\_cols:

      df[col].fillna(df[col].mode()[0], inplace=True)

   n\_healthy = (df['HeartDisease'] == 0).sum()

   n\_sick = (df['HeartDisease'] == 1).sum()

   print(f"Class distribution: healthy={n\_healthy}, sick={n\_sick}")

   if n\_healthy < n\_sick:

      n\_needed = n\_sick - n\_healthy

      healthy\_sample = df[df['HeartDisease'] == 0].sample(

         n=1, replace=True, random\_state=42

      )

      synthetic\_healthy = pd.concat([healthy\_sample] \* n\_needed, ignore\_index=True)

      df = pd.concat([df, synthetic\_healthy], ignore\_index=True)

      print(f"Đã thêm {n\_needed} dòng người khỏe mạnh để cân bằng dataset.")

   return df, numerical\_cols, categorical\_cols, binary\_cols

*# -------------------------------*

*# Train model RandomForest*

*# -------------------------------*

def train\_random\_forest\_model(csv\_path = None):

   print("Đang tải và tiền xử lý dữ liệu...")

   df, numerical\_features, categorical\_features, binary\_features = load\_and\_preprocess\_data(csv\_path)

   X = df.drop('HeartDisease', axis=1)

   y = df['HeartDisease']

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

      X, y, test\_size=0.2, random\_state=42, stratify=y

   )

   numerical\_transformer = Pipeline([

      ('imputer', SimpleImputer(strategy='median')),

      ('scaler', StandardScaler())

   ])

   categorical\_transformer = Pipeline([

      ('imputer', SimpleImputer(strategy='most\_frequent')),

      ('onehot', OneHotEncoder(handle\_unknown='ignore'))

   ])

   binary\_transformer = Pipeline([

      ('imputer', SimpleImputer(strategy='most\_frequent'))

   ])

   preprocessor = ColumnTransformer([

      ('num', numerical\_transformer, numerical\_features),

      ('cat', categorical\_transformer, categorical\_features),

      ('bin', binary\_transformer, binary\_features)

   ])

   pipeline = ImbPipeline([

      ('preprocessor', preprocessor),

      ('smote', SMOTE(random\_state=42)),

      ('classifier', RandomForestClassifier(random\_state=42))

   ])

   param\_grid = {

      'classifier\_\_n\_estimators': [100, 200],

      'classifier\_\_max\_depth': [10, 20, None],

      'classifier\_\_min\_samples\_split': [2, 5],

      'classifier\_\_min\_samples\_leaf': [1, 2],

      'classifier\_\_max\_features': ['sqrt', 'log2']

   }

   print("Đang tìm hyperparameters tối ưu...")

   grid\_search = GridSearchCV(

      pipeline,

      param\_grid,

      cv=5,

      scoring='f1',

      n\_jobs=-1,

      verbose=1

   )

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

   best\_model = grid\_search.best\_estimator\_

   y\_pred = best\_model.predict(X\_test)

   y\_pred\_proba = best\_model.predict\_proba(X\_test)[:, 1]

   print("\n" + "=" \* 50)

   print("KẾT QUẢ ĐÁNH GIÁ MÔ HÌNH")

   print("=" \* 50)

   print(f"Best Parameters: {grid\_search.best\_params\_}")

   print(f"Best CV Score: {grid\_search.best\_score\_:.4f}")

   print(f"Test Accuracy: {(y\_pred == y\_test).mean():.4f}")

   print(f"ROC AUC Score: {roc\_auc\_score(y\_test, y\_pred\_proba):.4f}")

   print("\nClassification Report:")

   print(classification\_report(y\_test, y\_pred))

*# ===============================*

*# LẤY FEATURE NAMES*

*# ===============================*

   preprocessor\_fitted = best\_model.named\_steps['preprocessor']

   feature\_names = []

   for name, transformer, cols in preprocessor\_fitted.transformers\_:

      if name in ['num', 'bin']:

         feature\_names.extend(cols)

      elif name == 'cat':

         cat\_features = transformer.named\_steps['onehot'].get\_feature\_names\_out(cols)

         feature\_names.extend(cat\_features)

*# ===============================*

*# VẼ BIỂU ĐỒ*

*# ===============================*

   generate\_evaluation\_plots(

      y\_test=y\_test,

      y\_pred=y\_pred,

      y\_pred\_proba=y\_pred\_proba,

      best\_model=best\_model,

      feature\_names=feature\_names,

      save\_dir="."

   )

*# ===============================*

*# LƯU MODEL*

*# ===============================*

   model\_data = {

      'model': best\_model,

      'feature\_names': feature\_names,

      'numerical\_features': numerical\_features,

      'categorical\_features': categorical\_features,

      'binary\_features': binary\_features

   }

   joblib.dump(model\_data, 'heart\_disease\_model.pkl')

   print("\nMô hình đã lưu vào 'heart\_disease\_model.pkl'")

   return best\_model

*# -------------------------------*

*# Main*

*# -------------------------------*

if \_\_name\_\_ == "\_\_main\_\_":

    model = train\_random\_forest\_model()

API predict

from flask import Blueprint, request, jsonify

import pandas as pd

from extensions import model, nlp\_extractor

from services.feature\_service import convert\_symptoms\_to\_features\_nlp

from services.question\_service import (

   get\_missing\_feature\_questions,

   generate\_missing\_info\_message

)

from services.recommendation\_service import (

   get\_recommendations,

   get\_next\_steps

)

predict\_bp = Blueprint("predict", \_\_name\_\_)

@predict\_bp.route("/predict", methods=["POST"])

def predict():

   if model is None:

      return jsonify({"error": "Model not loaded"}), 503

   data = request.json or {}

   features, missing = convert\_symptoms\_to\_features\_nlp(

      data.get("symptoms", ""),

      data.get("age"),

      data.get("gender"),

      data.get("symptom\_duration")

   )

   critical = ['Age', 'Sex', 'Cholesterol', 'RestingBP', 'MaxHR']

   missing\_critical = [f for f in critical if f in missing]

   SECONDARY\_GROUPS = [

      ['ChestPainType', 'ExerciseAngina'],

      ['RestingECG', 'Oldpeak'],

      ['FastingBS'],

      ['ST\_Slope']

   ]

   ask = []

   if missing\_critical:

      ask = missing\_critical

   else:

      for group in SECONDARY\_GROUPS:

         group\_missing = [f for f in group if f in missing]

         if group\_missing:

            ask = group\_missing

            break

   if ask:

      progress = round(len(features) / 11 \* 100)

      return jsonify({

         "status": "need\_more\_info",

         "message": generate\_missing\_info\_message(ask, progress),

         "questions": get\_missing\_feature\_questions(ask),

         "partial\_features": features,

         "progress\_percentage": progress

      })

   expected = [

      'Age','Sex','ChestPainType','RestingBP','Cholesterol',

      'FastingBS','RestingECG','MaxHR','ExerciseAngina',

      'Oldpeak','ST\_Slope'

   ]

   for f in expected:

      features.setdefault(f, nlp\_extractor.default\_values[f])

   df = pd.DataFrame([features])[expected]

   pred = model.predict(df)[0]

   prob = model.predict\_proba(df)[0]

   risk\_prob = prob[1]

   if risk\_prob >= 0.75:

      risk\_level = "Cao"

      message = "Nguy cơ mắc bệnh tim cao, cần đi khám sớm"

   elif risk\_prob >= 0.5:

      risk\_level = "Trung bình"

      message = "Có dấu hiệu nguy cơ, cần theo dõi và kiểm tra thêm"

   elif risk\_prob >= 0.3:

      risk\_level = "Thấp - Cần theo dõi"

      message = "Hiện tại nguy cơ chưa cao nhưng nên theo dõi định kỳ"

   else:

      risk\_level = "Thấp"

      message = "Nguy cơ mắc bệnh tim thấp"

   return jsonify({

      "prediction": int(pred),

      "probability": float(prob[pred]),

      "risk\_level": risk\_level,

      "message": message,

      "recommendations": get\_recommendations(pred, features),

      "next\_steps": get\_next\_steps(pred, features)

   })

NLP Processor  
import re

from typing import Dict, List, Any, Optional, Tuple

import numpy as np

class HeartDiseaseNLPExtractor:

   def \_\_init\_\_(self):

*# Từ điển triệu chứng và điều kiện y tế*

      self.medical\_vocab = {

*# Giới tính*

         'gender\_male': {

            'vi': ['nam', 'đàn ông', 'trai', 'con trai', 'ông'],

            'en': ['male', 'man', 'boy', 'gentleman']

         },

         'gender\_female': {

            'vi': ['nữ', 'phụ nữ', 'con gái', 'bà', 'cô'],

            'en': ['female', 'woman', 'girl', 'lady']

         },

*# Loại đau ngực*

         'chest\_pain\_ata': {

            'vi': ['lan tỏa', 'lan sang tay trái', 'lan lên cổ', 'lan ra sau lưng'],

            'en': ['radiating', 'spreading to left arm', 'radiates to jaw', 'radiates to back']

         },

         'chest\_pain\_ta': {

            'vi': ['nghỉ ngơi', 'khi nghỉ', 'lúc ngồi', 'ban đêm'],

            'en': ['at rest', 'resting', 'while sitting', 'nocturnal']

         },

         'chest\_pain\_nap': {

            'vi': ['đau nhói', 'đau như kim châm', 'đau ngắn', 'đau thoáng qua'],

            'en': ['sharp', 'stabbing', 'brief', 'transient']

         },

*# Triệu chứng*

         'symptom\_chest\_pain': {

            'vi': ['đau ngực', 'tức ngực', 'đau tức ngực', 'đau vùng tim', 'nặng ngực'],

            'en': ['chest pain', 'chest discomfort', 'chest tightness', 'angina']

         },

         'symptom\_breathlessness': {

            'vi': ['khó thở', 'thở gấp', 'hụt hơi', 'thở ngắn'],

            'en': ['shortness of breath', 'dyspnea', 'breathlessness']

         },

         'symptom\_fatigue': {

            'vi': ['mệt mỏi', 'uể oải', 'kiệt sức', 'mệt'],

            'en': ['fatigue', 'tiredness', 'exhaustion', 'weakness']

         },

         'symptom\_dizziness': {

            'vi': ['chóng mặt', 'hoa mắt', 'choáng váng', 'quay cuồng'],

            'en': ['dizziness', 'lightheadedness', 'vertigo']

         },

         'symptom\_palpitations': {

            'vi': ['hồi hộp', 'trống ngực', 'tim đập nhanh', 'hẫng hụt'],

            'en': ['palpitations', 'heart racing', 'rapid heartbeat']

         },

*# Bệnh nền*

         'comorbidity\_hypertension': {

            'vi': ['cao huyết áp', 'huyết áp cao', 'tăng huyết áp'],

            'en': ['hypertension', 'high blood pressure', 'elevated blood pressure']

         },

         'comorbidity\_diabetes': {

            'vi': ['tiểu đường', 'đái tháo đường'],

            'en': ['diabetes', 'diabetic']

         },

         'comorbidity\_hyperlipidemia': {

            'vi': ['mỡ máu cao', 'cholesterol cao', 'rối loạn lipid'],

            'en': ['high cholesterol', 'hyperlipidemia', 'elevated lipids']

         },

*# ECG patterns*

         'ecg\_normal': {

            'vi': ['điện tâm đồ bình thường', 'ecg bình thường'],

            'en': ['normal ecg', 'normal electrocardiogram']

         },

         'ecg\_stt': {

            'vi': ['st-t thay đổi', 'sóng t dẹt', 'st chênh'],

            'en': ['st-t changes', 't wave flattening', 'st depression']

         },

         'ecg\_lvh': {

            'vi': ['dày thất trái', 'phì đại thất trái'],

            'en': ['left ventricular hypertrophy', 'lvh']

         },

*# Exercise angina*

         'angina\_yes': {

            'vi': ['gắng sức', 'khi tập thể dục', 'leo cầu thang', 'mang vác nặng'],

            'en': ['exercise', 'exertion', 'physical activity', 'climbing stairs']

         },

*# ST Slope*

         'st\_up': {

            'vi': ['dốc lên', 'đi lên'],

            'en': ['upsloping', 'rising']

         },

         'st\_flat': {

            'vi': ['phẳng', 'nằm ngang'],

            'en': ['flat', 'horizontal']

         },

         'st\_down': {

            'vi': ['dốc xuống', 'đi xuống'],

            'en': ['downsloping', 'descending']

         }

      }

*# Regex patterns cho số*

      self.number\_patterns = {

         'age': r'(?:tuổi|age|aged?)\s\*(?:là|khoảng|:)?\s\*(\d+)',

         'blood\_pressure': r'(?:huyết áp|blood pressure|bp)\s\*(?:là|khoảng|:)?\s\*(\d+)(?:\s\*/\s\*\d+)?',

         'cholesterol': r'(?:cholesterol|mỡ máu|lipid)\s\*(?:là|khoảng|:)?\s\*(\d+)',

         'heart\_rate': r'(?:nhịp tim|heart rate|hr)\s\*(?:là|khoảng|nghỉ|resting)?\s\*:?\s\*(\d+)',

         'oldpeak': r'(?:oldpeak|st depression)\s\*(?:là|khoảng|:)?\s\*([\d.]+)',

         'duration': r'(?:triệu chứng|đau|bệnh)\s\*(?:đã|kéo dài|:)?\s\*(\d+)\s\*(?:ngày|tuần|tháng|năm|day|week|month|year)',

         'generic\_number': r'\b(\d+)\s\*(?:mg/dl|mmol/l|mmHg|bpm)?\b'

      }

*# Mapping giá trị cho các feature*

      self.value\_mapping = {

         'Sex': {'M': 1, 'F': 0},

         'ChestPainType': {'TA': 0, 'ATA': 1, 'NAP': 2, 'ASY': 3},

         'RestingECG': {'Normal': 0, 'ST': 1, 'LVH': 2},

         'ExerciseAngina': {'N': 0, 'Y': 1},

         'ST\_Slope': {'Up': 0, 'Flat': 1, 'Down': 2}

      }

*# Danh sách features bắt buộc và tùy chọn*

      self.required\_features = ['Age', 'Sex']

      self.optional\_features = [

         'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS',

         'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST\_Slope'

      ]

*# Giá trị mặc định cho features missing*

      self.default\_values = {

         'Age': 50,

         'Sex': 1,  *# Mặc định là nam*

         'ChestPainType': 3,  *# ASY*

         'RestingBP': 120,

         'Cholesterol': 200,

         'FastingBS': 0,

         'RestingECG': 0,  *# Normal*

         'MaxHR': 150,

         'ExerciseAngina': 0,  *# N*

         'Oldpeak': 0.0,

         'ST\_Slope': 1  *# Flat*

      }

   def extract\_all\_features(self, text: str) -> Tuple[Dict[str, Any], List[str]]:

      """

      Trích xuất tất cả features từ văn bản

      Returns:

         Tuple[features\_dict, missing\_features]

      """

      features = {}

      text\_lower = text.lower()

*# 1. Trích xuất tuổi*

      age = self.\_extract\_age(text\_lower)

      features['Age'] = age if age else self.default\_values['Age']

*# 2. Trích xuất giới tính*

      sex = self.\_extract\_gender(text\_lower)

      features['Sex'] = sex if sex else self.default\_values['Sex']

*# 3. Trích xuất ChestPainType*

      cp\_type = self.\_extract\_chest\_pain\_type(text\_lower)

      features['ChestPainType'] = cp\_type if cp\_type else self.default\_values['ChestPainType']

*# 4. Trích xuất RestingBP*

      resting\_bp = self.\_extract\_blood\_pressure(text\_lower)

      features['RestingBP'] = resting\_bp if resting\_bp else self.default\_values['RestingBP']

*# 5. Trích xuất Cholesterol*

      cholesterol = self.\_extract\_cholesterol(text\_lower)

      features['Cholesterol'] = cholesterol if cholesterol else self.default\_values['Cholesterol']

*# 6. Trích xuất FastingBS*

      fasting\_bs = self.\_extract\_fasting\_bs(text\_lower)

      features['FastingBS'] = fasting\_bs if fasting\_bs is not None else self.default\_values['FastingBS']

*# 7. Trích xuất RestingECG*

      resting\_ecg = self.\_extract\_resting\_ecg(text\_lower)

      features['RestingECG'] = resting\_ecg if resting\_ecg else self.default\_values['RestingECG']

*# 8. Trích xuất MaxHR*

      max\_hr = self.\_extract\_max\_hr(text\_lower, features['Age'])

      features['MaxHR'] = max\_hr if max\_hr else self.default\_values['MaxHR']

*# 9. Trích xuất ExerciseAngina*

      exercise\_angina = self.\_extract\_exercise\_angina(text\_lower)

      features['ExerciseAngina'] = exercise\_angina if exercise\_angina is not None else self.default\_values['ExerciseAngina']

*# 10. Trích xuất Oldpeak*

      oldpeak = self.\_extract\_oldpeak(text\_lower)

      features['Oldpeak'] = oldpeak if oldpeak else self.default\_values['Oldpeak']

*# 11. Trích xuất ST\_Slope*

      st\_slope = self.\_extract\_st\_slope(text\_lower)

      features['ST\_Slope'] = st\_slope if st\_slope else self.default\_values['ST\_Slope']

*# Kiểm tra features missing*

      missing\_features = self.\_check\_missing\_features(features, text\_lower)

      return features, missing\_features

   def \_extract\_age(self, text: str) -> Optional[int]:

      """Trích xuất tuổi"""

*# Tìm pattern tuổi cụ thể*

      age\_pattern = r'\b(\d+)\s\*(?:tuổi|years? old)\b'

      match = re.search(age\_pattern, text)

      if match:

         age = int(match.group(1))

         return min(max(age, 20), 100)  *# Giới hạn trong khoảng hợp lý*

*# Tìm trong cấu trúc câu thông thường*

      for pattern in [r'tuổi\s\*(\d+)', r'age\s\*(\d+)', r'(\d+)\s\*năm']:

         match = re.search(pattern, text)

         if match:

               age = int(match.group(1))

               return min(max(age, 20), 100)

      return None

   def \_extract\_gender(self, text: str) -> Optional[int]:

      """Trích xuất giới tính (1: Nam, 0: Nữ)"""

      male\_keywords = self.medical\_vocab['gender\_male']['vi'] + self.medical\_vocab['gender\_male']['en']

      female\_keywords = self.medical\_vocab['gender\_female']['vi'] + self.medical\_vocab['gender\_female']['en']

      for keyword in male\_keywords:

         if keyword in text:

               return 1

      for keyword in female\_keywords:

         if keyword in text:

               return 0

      return None

   def \_extract\_chest\_pain\_type(self, text: str) -> Optional[int]:

      """Trích xuất loại đau ngực"""

*# Kiểm tra xem có đau ngực không*

      chest\_pain\_keywords = (self.medical\_vocab['symptom\_chest\_pain']['vi'] +

                           self.medical\_vocab['symptom\_chest\_pain']['en'])

      has\_chest\_pain = any(keyword in text for keyword in chest\_pain\_keywords)

      if not has\_chest\_pain:

         return 3  *# ASY (không có triệu chứng)*

*# Kiểm tra các loại đau ngực cụ thể*

*# ATA (atypical angina)*

      for keyword in self.medical\_vocab['chest\_pain\_ata']['vi'] + self.medical\_vocab['chest\_pain\_ata']['en']:

         if keyword in text:

               return 1  *# ATA*

*# TA (typical angina)*

      for keyword in self.medical\_vocab['chest\_pain\_ta']['vi'] + self.medical\_vocab['chest\_pain\_ta']['en']:

         if keyword in text:

               return 0  *# TA*

*# NAP (non-anginal pain)*

      for keyword in self.medical\_vocab['chest\_pain\_nap']['vi'] + self.medical\_vocab['chest\_pain\_nap']['en']:

         if keyword in text:

               return 2  *# NAP*

*# Mặc định là ATA nếu có đau ngực nhưng không xác định rõ*

      return 1

   def \_extract\_blood\_pressure(self, text: str) -> Optional[int]:

      """Trích xuất huyết áp"""

*# Tìm pattern huyết áp*

      patterns = [

         r'huyết áp\s\*(\d+)', r'blood pressure\s\*(\d+)', r'bp\s\*(\d+)',

         r'(\d+)\s\*/\s\*\d+\s\*(?:mmHg|huyết áp)'

      ]

      for pattern in patterns:

         matches = re.findall(pattern, text)

         if matches:

               try:

                  bp = int(matches[0])

*# Giới hạn trong khoảng hợp lý*

                  return min(max(bp, 80), 200)

               except:

                  continue

*# Tìm số có thể là huyết áp*

      number\_pattern = r'\b(1[0-2]\d|90|1[3-9]\d|200)\b'

      matches = re.findall(number\_pattern, text)

      for match in matches:

         num = int(match)

         if 90 <= num <= 200:  *# Khoảng huyết áp hợp lý*

*# Kiểm tra xem số này có gần từ "huyết áp" không*

               idx = text.find(match)

               context = text[max(0, idx-20):min(len(text), idx+20)]

               if 'huyết áp' in context or 'blood' in context or 'bp' in context:

                  return num

      return None

   def \_extract\_cholesterol(self, text: str) -> Optional[int]:

      """Trích xuất cholesterol"""

*# Tìm pattern cholesterol cụ thể*

      patterns = [

         r'cholesterol\s\*(\d+)', r'mỡ máu\s\*(\d+)', r'lipid\s\*(\d+)',

         r'(\d+)\s\*(?:mg/dl|mmol/l)\s\*(?:cholesterol|mỡ)'

      ]

      for pattern in patterns:

         matches = re.findall(pattern, text)

         if matches:

               try:

                  chol = int(matches[0])

                  return min(max(chol, 100), 400)  *# Giới hạn hợp lý*

               except:

                  continue

*# Tìm số có thể là cholesterol*

      number\_pattern = r'\b([1-3]\d{2}|400)\b'

      matches = re.findall(number\_pattern, text)

      for match in matches:

         num = int(match)

         if 150 <= num <= 300:  *# Khoảng cholesterol phổ biến*

*# Kiểm tra context*

               idx = text.find(match)

               context = text[max(0, idx-20):min(len(text), idx+20)]

               if 'cholesterol' in context or 'mỡ' in context or 'lipid' in context:

                  return num

      return None

   def \_extract\_fasting\_bs(self, text: str) -> Optional[int]:

      """Trích xuất đường huyết lúc đói"""

*# Kiểm tra tiểu đường*

      diabetes\_keywords = (self.medical\_vocab['comorbidity\_diabetes']['vi'] +

                        self.medical\_vocab['comorbidity\_diabetes']['en'])

      has\_diabetes = any(keyword in text for keyword in diabetes\_keywords)

      if has\_diabetes:

         return 1

*# Tìm chỉ số đường huyết cụ thể*

      glucose\_patterns = [

         r'đường huyết\s\*(\d+)', r'blood sugar\s\*(\d+)', r'glucose\s\*(\d+)',

         r'(\d+)\s\*(?:mg/dl|mmol/l)\s\*(?:đường|sugar)'

      ]

      for pattern in glucose\_patterns:

         matches = re.findall(pattern, text)

         if matches:

               try:

                  glucose = int(matches[0])

                  return 1 if glucose >= 126 else 0  *# Ngưỡng tiểu đường*

               except:

                  continue

      return None

   def \_extract\_resting\_ecg(self, text: str) -> Optional[int]:

      """Trích xuất kết quả ECG"""

*# Normal*

      for keyword in self.medical\_vocab['ecg\_normal']['vi'] + self.medical\_vocab['ecg\_normal']['en']:

         if keyword in text:

               return 0  *# Normal*

*# ST-T changes*

      for keyword in self.medical\_vocab['ecg\_stt']['vi'] + self.medical\_vocab['ecg\_stt']['en']:

         if keyword in text:

               return 1  *# ST*

*# LVH*

      for keyword in self.medical\_vocab['ecg\_lvh']['vi'] + self.medical\_vocab['ecg\_lvh']['en']:

         if keyword in text:

               return 2  *# LVH*

      return None

   def \_extract\_max\_hr(self, text: str, age: int) -> Optional[int]:

      """Trích xuất nhịp tim tối đa"""

*# Tìm nhịp tim cụ thể*

      hr\_patterns = [

         r'nhịp tim\s\*(\d+)', r'heart rate\s\*(\d+)', r'hr\s\*(\d+)',

         r'(\d+)\s\*bpm', r'mạch\s\*(\d+)'

      ]

      for pattern in hr\_patterns:

         matches = re.findall(pattern, text)

         if matches:

               try:

                  hr = int(matches[0])

                  return min(max(hr, 50), 200)  *# Giới hạn hợp lý*

               except:

                  continue

*# Ước tính dựa trên tuổi nếu không tìm thấy*

      estimated\_hr = 220 - age

      return min(max(estimated\_hr, 60), 180)

   def \_extract\_exercise\_angina(self, text: str) -> Optional[int]:

      """Trích xuất đau thắt ngực khi gắng sức"""

      angina\_keywords = self.medical\_vocab['angina\_yes']['vi'] + self.medical\_vocab['angina\_yes']['en']

      for keyword in angina\_keywords:

         if keyword in text:

*# Kiểm tra xem có đau ngực khi gắng sức không*

               context = text[max(0, text.find(keyword)-30):min(len(text), text.find(keyword)+30)]

               chest\_pain\_keywords = (self.medical\_vocab['symptom\_chest\_pain']['vi'] +

                                    self.medical\_vocab['symptom\_chest\_pain']['en'])

               if any(cp in context for cp in chest\_pain\_keywords):

                  return 1  *# Y*

      return 0

   def \_extract\_oldpeak(self, text: str) -> Optional[float]:

      """Trích xuất oldpeak (ST depression)"""

      patterns = [

         r'oldpeak\s\*([\d.]+)', r'st depression\s\*([\d.]+)',

         r'([\d.]+)\s\*mm\s\*(?:st|depression)'

      ]

      for pattern in patterns:

         matches = re.findall(pattern, text)

         if matches:

               try:

                  op = float(matches[0])

                  return min(max(op, 0.0), 6.0)

               except:

                  continue

      severe\_symptoms = ['nặng', 'dữ dội', 'severe', 'intense']

      if any(symptom in text for symptom in severe\_symptoms):

         return 2.0

      return 0.0

   def \_extract\_st\_slope(self, text: str) -> Optional[int]:

      """Trích xuất ST slope"""

*# Up*

      for keyword in self.medical\_vocab['st\_up']['vi'] + self.medical\_vocab['st\_up']['en']:

         if keyword in text:

               return 0

*# Down*

      for keyword in self.medical\_vocab['st\_down']['vi'] + self.medical\_vocab['st\_down']['en']:

         if keyword in text:

               return 2

      return 1

   def \_check\_missing\_features(self, features: Dict[str, Any], text: str) -> List[str]:

      """Kiểm tra features nào còn missing hoặc cần làm rõ"""

      missing = []

*# Kiểm tra các features quan trọng*

      important\_features = ['Cholesterol', 'RestingBP', 'MaxHR']

      for feature in important\_features:

         if features.get(feature) == self.default\_values[feature]:

*# Kiểm tra xem feature này đã được đề cập trong text chưa*

               mentioned = False

               if feature == 'Cholesterol':

                  mentioned = any(word in text for word in ['cholesterol', 'mỡ máu', 'lipid'])

               elif feature == 'RestingBP':

                  mentioned = any(word in text for word in ['huyết áp', 'blood pressure', 'bp'])

               elif feature == 'MaxHR':

                  mentioned = any(word in text for word in ['nhịp tim', 'heart rate', 'mạch'])

               if not mentioned:

                  missing.append(feature)

      return missing

   def generate\_missing\_questions(self, missing\_features: List[str]) -> List[str]:

      """Tạo câu hỏi để thu thập thông tin thiếu"""

      questions = []

      question\_map = {

         'Cholesterol': "Chỉ số cholesterol của bạn hiện tại là bao nhiêu (mg/dL)?",

         'RestingBP': "Huyết áp lúc nghỉ của bạn là bao nhiêu (mmHg)?",

         'MaxHR': "Nhịp tim tối đa của bạn khi gắng sức là bao nhiêu (bpm)?",

         'RestingECG': "Kết quả điện tâm đồ gần đây của bạn thế nào?",

         'Oldpeak': "Chỉ số ST depression (oldpeak) trên điện tâm đồ của bạn là bao nhiêu?",

         'FastingBS': "Bạn có bị tiểu đường hoặc đường huyết lúc đói cao không?",

         'ExerciseAngina': "Bạn có bị đau ngực khi gắng sức không?"

      }

      for feature in missing\_features:

         if feature in question\_map:

               questions.append(question\_map[feature])

      return questions

   def update\_features\_with\_response(self, features: Dict[str, Any], response\_text: str, feature\_name: str) -> Dict[str, Any]:

      """Cập nhật features với thông tin mới từ người dùng"""

      text\_lower = response\_text.lower()

      if feature\_name == 'Cholesterol':

         value = self.\_extract\_cholesterol(text\_lower)

         if value:

               features['Cholesterol'] = value

      elif feature\_name == 'RestingBP':

         value = self.\_extract\_blood\_pressure(text\_lower)

         if value:

               features['RestingBP'] = value

      elif feature\_name == 'MaxHR':

         value = self.\_extract\_max\_hr(text\_lower, features['Age'])

         if value:

               features['MaxHR'] = value

      elif feature\_name == 'RestingECG':

         value = self.\_extract\_resting\_ecg(text\_lower)

         if value is not None:

               features['RestingECG'] = value

      elif feature\_name == 'Oldpeak':

         value = self.\_extract\_oldpeak(text\_lower)

         if value:

               features['Oldpeak'] = value

      elif feature\_name == 'FastingBS':

         value = self.\_extract\_fasting\_bs(text\_lower)

         if value is not None:

               features['FastingBS'] = value

      elif feature\_name == 'ExerciseAngina':

         value = self.\_extract\_exercise\_angina(text\_lower)

         if value is not None:

               features['ExerciseAngina'] = value

      return features

*# Tiện ích để tích hợp với Flask API*

def process\_user\_input(user\_input: str) -> Dict[str, Any]:

   """

   Xử lý input từ người dùng và trả về features hoặc câu hỏi bổ sung

   Returns:

      Dict với keys:

      - 'status': 'complete' hoặc 'need\_info'

      - 'features': dict features (nếu complete)

      - 'questions': list câu hỏi (nếu need\_info)

      - 'missing\_features': list tên features missing

   """

   extractor = HeartDiseaseNLPExtractor()

*# Trích xuất features*

   features, missing\_features = extractor.extract\_all\_features(user\_input)

   if not missing\_features:

      return {

         'status': 'complete',

         'features': features,

         'message': 'Đã trích xuất đủ thông tin để dự đoán.'

      }

   else:

      questions = extractor.generate\_missing\_questions(missing\_features)

      return {

         'status': 'need\_info',

         'features': features,  *# Features đã trích xuất được*

         'missing\_features': missing\_features,

         'questions': questions,

         'message': f'Cần bổ sung thêm {len(missing\_features)} thông tin.'

      }

*# Test the extractor*

if \_\_name\_\_ == "\_\_main\_\_":

   extractor = HeartDiseaseNLPExtractor()

Evaluation Plots

*# evaluation\_plots.py*

import os

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import confusion\_matrix, roc\_curve, auc

def generate\_evaluation\_plots(

   y\_test,

   y\_pred,

   y\_pred\_proba,

   best\_model,

   feature\_names,

   save\_dir="results",

   folder\_name="evaluation\_plots"

):

   """

      Sinh các biểu đồ đánh giá mô hình:

      - Confusion Matrix

      - ROC Curve

      - Feature Importance

   """

*# ===============================*

*# Tạo thư mục lưu kết quả*

*# ===============================*

   output\_dir = os.path.join(save\_dir, folder\_name)

   os.makedirs(output\_dir, exist\_ok = True)

*# ===============================*

*# 1. Confusion Matrix*

*# ===============================*

   cm = confusion\_matrix(y\_test, y\_pred)

   plt.figure(figsize=(6, 5))

   sns.heatmap(

      cm,

      annot = True,

      fmt = 'd',

      cmap = 'Blues',

      xticklabels = ['Không bệnh', 'Có bệnh'],

      yticklabels = ['Không bệnh', 'Có bệnh']

   )

   plt.xlabel('Dự đoán')

   plt.ylabel('Thực tế')

   plt.title('Confusion Matrix - Random Forest')

   plt.tight\_layout()

   plt.savefig(os.path.join(output\_dir, "confusion\_matrix.png"), dpi=300)

   plt.close()

*# ===============================*

*# 2. ROC Curve*

*# ===============================*

   fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_proba)

   roc\_auc = auc(fpr, tpr)

   plt.figure(figsize=(8, 6))

   plt.plot(fpr, tpr, lw=2, label=f'ROC Curve (AUC = {roc\_auc:.2f})')

   plt.plot([0, 1], [0, 1], linestyle='--', label='Random Classifier')

   plt.xlabel('False Positive Rate')

   plt.ylabel('True Positive Rate')

   plt.title('ROC Curve - Random Forest')

   plt.legend(loc="lower right")

   plt.grid(alpha=0.3)

   plt.tight\_layout()

   plt.savefig(os.path.join(output\_dir, "roc\_curve.png"), dpi=300)

   plt.close()

*# ===============================*

*# 3. Feature Importance*

*# ===============================*

   importances = best\_model.named\_steps['classifier'].feature\_importances\_

   importance\_df = pd.DataFrame({

      'Feature': feature\_names[:len(importances)],

      'Importance': importances

   }).sort\_values('Importance', ascending=False).head(15)

   plt.figure(figsize = (10, 6))

   sns.barplot(

      x='Importance',

      y='Feature',

      data=importance\_df,

      palette='viridis'

   )

   plt.title('Top 15 Feature Importance - Random Forest')

   plt.tight\_layout()

   plt.savefig(os.path.join(output\_dir, "feature\_importance.png"), dpi=300)

   plt.close()

   print(f"Đã lưu biểu đồ: {output\_dir}")