HỌC VIỆN NGÂN HÀNG

KHOA CÔNG NGHỆ THÔNG TIN VÀ KINH TẾ SỐ

-----ଓ📖ଌ-----

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN: KHAI PHÁ VÀ PHÂN TÍCH DỮ LIỆU

CHỦ ĐỀ

ỨNG DỤNG THUẬT TOÁN PHÂN LỚP CHẨN ĐOÁN KHẢ NĂNG MẮC BỆNH TIM MẠCH

Giảng viên hướng dẫn : ThS. Nguyễn Dương Hùng

Mã lớp học phần : 241IS23A01

Nhóm thực hiện : Nhóm 08

Hà Nội - 12/2024

HỌC VIỆN NGÂN HÀNG

KHOA CÔNG NGHỆ THÔNG TIN VÀ KINH TẾ SỐ

-----ଓ📖ଌ-----

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN: KHAI PHÁ VÀ PHÂN TÍCH DỮ LIỆU

CHỦ ĐỀ

ỨNG DỤNG THUẬT TOÁN PHÂN LỚP CHẨN ĐOÁN KHẢ NĂNG MẮC BỆNH TIM MẠCH

Giảng viên hướng dẫn	: ThS. Nguyễn Dương Hùng
Mã lớp học phần	: 241IS23A01
Nhóm thực hiện	: Nhóm 08
Họ và tên	: Mã sinh viên
Trịnh Đức Hiền (NT)	: 25A4041497
Nguyễn Thị Mỹ Linh	: 25A4041862
Đồng Thị Hồng Nhung	: 25A4041886
Mai Thu Thảo	: 25A4042230
Nguyễn Thị Thảo	: 25A4042232

Hà Nội - 12/2024

BẢNG PHÂN CÔNG CÔNG VIỆC

STT	Mã sinh viên	Họ và tên	Phân công	Mức độ đóng góp (%)
1	25A4041497	Trịnh Đức Hiền	- Tổng hợp nội dung - Sử dụng thuật toán Random Forest - Sử dụng thuật toán Decision Tree - Mô tả bộ dữ liệu	20%
2	25A4041862	Nguyễn Thị Mỹ Linh	- Trực quan hóa - Làm word - Làm chương 1 - Làm chương 5	19,8%
3	25A4041886	Đồng Thị Hồng Nhung	- Làm chương 2 - Làm thuật toán - Tiền xử lý dữ liệu - Sử dụng thuật toán K-NN	20%
4	25A4042230	Mai Thu Thảo	- Làm chương 2 - Tiền xử lý dữ liệu - Sử dụng thuật toán Naive Bayes - So sánh, đánh giá thuật toán - Chỉnh sửa nội dung, slide	20,4%
5	25A4042232	Nguyễn Thị Thảo	- Trực quan hóa - Làm Slide - Làm chương 1	19,8%

LỜI CẢM ƠN

Lời đầu tiên, nhóm sinh viên chúng em xin được gửi lời cảm ơn đến quý thầy cô và lãnh đạo Học viện Ngân hàng nói chung và Khoa Công nghệ thông tin & Kinh tế số nói riêng đã tạo điều kiện cho chúng em được thực hiện bài báo cáo cho học phần Khai phá và Phân tích dữ liệu, giúp chúng em có thể củng cố, từ đó áp dụng kiến thức đã được học để hiểu biết rộng hơn, nghiên cứu sâu hơn về các bài toán trong thực tế.

Chúng em cũng xin đặc biệt gửi lời cảm ơn đến Thầy Nguyễn Dương Hùng đã tận tâm hướng dẫn, giúp đỡ chúng em trong suốt quá trình thực hiện bài báo cáo, từ những ngày bắt đầu cho đến khi hoàn thành báo cáo. Nếu không có sự chỉ bảo của Thầy thì bài thu hoạch này của chúng em rất khó để đi đúng hướng và hoàn thiện được. Một lần nữa, chúng em xin chân thành cảm ơn Thầy! Trong quá trình thực hiện bài báo cáo, do kiến thức, lý luận cũng như kinh nghiệm thực tiễn của chúng em còn hạn chế nên không thể tránh khỏi những thiếu sót. Chúng em rất mong nhận được góp ý quý báu từ phía Thầy để bài báo cáo của chúng em được hoàn thiện.

Lời cuối, chúng em xin kính chúc thầy Dương Hùng cũng như toàn thể quý thầy cô Khoa Công nghệ thông tin & Kinh tế số thật nhiều sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh trồng người.

Chúng em xin chân thành cảm ơn!

Hà Nội, ngày 19 tháng 12 năm 2024

Nhóm 08

LỜI CAM ĐOAN

Chúng em xin cam đoan kết quả đạt được trong báo cáo là sản phẩm nghiên cứu, tìm hiểu của riêng nhóm. Trong toàn bộ nội dung của báo cáo, những điều được trình bày hoặc là của nhóm hoặc là được tổng hợp từ nhiều nguồn tài liệu. Tất cả các tài liệu tham khảo đều có xuất xứ rõ ràng và được trích dẫn hợp pháp.

Chúng em xin hoàn chịu trách nhiệm và chịu mọi hình thức kỷ luật theo quy định cho lời cam đoan của mình.

Hà Nội, ngày 19 tháng 12 năm 2024

Nhóm 08

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1. Đặt vấn đề và lý do chọn đề tài

Trong kỷ nguyên công nghệ 4.0, sự phát triển mạnh mẽ của công nghệ thông tin và quá trình số hóa đã tác động sâu sắc đến mọi lĩnh vực, từ chính trị, kinh tế đến xã hội. Internet trở thành một kho tàng dữ liệu khổng lồ, chứa đựng mọi dạng thông tin từ cá nhân đến doanh nghiệp. Trong bối cảnh đó, khai phá dữ liệu nổi lên như một công cụ quan trọng, cho phép chúng ta khai thác triệt để nguồn tài nguyên này. Việc phân tích và xử lý lượng dữ liệu khổng lồ giúp chúng ta khám phá ra những giá trị mới, đưa ra tri thức và thông tin quan trọng. Khai phá dữ liệu đã được áp dụng trong nhiều lĩnh vực quan trọng như y tế, kinh doanh, chứng khoán, dự báo thời tiết,... mang lại nhiều ứng dụng thực tiễn. Đặc biệt, trong lĩnh vực y tế, khai phá dữ liệu đóng vai trò quan trọng, hỗ trợ bác sĩ trong quá trình chẩn đoán, nghiên cứu dược phẩm, quản lý hồ sơ bệnh án, và theo dõi sức khỏe bệnh nhân.

Trong số các bệnh lý, bệnh tim mạch là một trong những nguyên nhân hàng đầu gây tử vong và tàn tật trên toàn cầu. Với hệ thống y tế hiện đại ngày nay, việc phát hiện và chẩn đoán sớm bệnh tim mang lại ý nghĩa rất lớn trong việc giảm thiểu hậu quả nghiêm trọng của căn bệnh này. Đông thời, việc phân tích dữ liệu y tế, chúng ta có thể xác định các yếu tố chủ chốt dẫn đến bệnh tim và từ đó đề ra các phương án can thiệp kịp thời, giảm thiểu rủi ro.

1.2. Mục tiêu của đề tài

Mục tiêu của đề tài là phát triển mô hình chẩn đoán nguy cơ mắc bệnh tim mạch chính xác bằng thuật toán phân lớp, dựa trên các yếu tố như tuổi tác, giới tính, và thói quen sinh hoạt. Mô hình này giúp phát hiện bệnh sớm, hỗ trợ nhân viên y tế can thiệp kịp thời, tăng cơ hội hồi phục cho bệnh nhân. Việc khai thác và tối ưu hóa dữ liệu y tế sẽ góp phần cải thiện quy trình điều trị và nâng cao chất lượng chăm sóc sức khỏe.

1.3. Đối tượng và phương pháp nghiên cứu

- Đối tượng nghiên cứu: Đối tượng nghiên cứu là các bệnh nhân có nguy cơ mắc bệnh tim mạch và các chỉ số của bệnh nhân được cung cấp trong bộ dữ liệu được đăng tải trên trang web.
- Phương pháp nghiên cứu

Thu thập dữ liệu: Nhóm đã tìm kiếm và sử dụng các bộ dữ liệu y tế mở mà liên quan đến bệnh tim mạch.

Nghiên cứu mô tả: Mô tả tình hình mắc suy tim trong một quần thể cụ thể, như tỷ lệ mắc, phân bố về độ tuổi, giới tính, v.v.

Phân tích dữ liệu: Sử dụng các mô hình học máy, mô hình phân lớp dữ liệu để tìm ra mối liên hệ giữa các yếu tố trong tập dữ liệu.

1.4. Ý nghĩa của đề tài

- Ý nghĩa khoa học

Thứ nhất, nó đóng góp vào sự phát triển của lĩnh vực khoa học dữ liệu và trí tuệ nhân tạo trong y tế, mở ra hướng nghiên cứu mới về sử dụng các thuật toán học máy để phân tích vàchẩn đoán bệnh tật. Thứ hai, nghiên cứu này giúp cải tiến khả năng xử lý và khai thác dữ liệu y tế phức tạp, từ đó tạo ra những mô hình chẩn đoán chính xác, hỗ trợ bác sĩ trong quá trình chẩn đoán. Thứ ba, kết quả của đề tài có thể được ứng dụng rộng rãi, không chỉ trong lĩnh vực tim mạch mà còn ở nhiều bệnh lý khác, góp phần nâng cao hiệu quả khám chữa bệnh và chất lượng chăm sóc sức khỏe cộng đồng.

- Ý nghĩa thực tiễn

Mô hình giúp nâng cao độ chính xác trong việcchẩn đoán nguy cơ bệnh tim mạch, hỗ trợ bác sĩ đưa ra các quyết định điều trị kịp thời và hiệu quả hơn, từ đó cải thiện khả năng cứu sống bệnh nhân và giảm thiểu các biến chứng nặng nề. Thứ hai, mô hình này còn giúp tối ưu hóa thời gian và nguồn lực y tế, giảm thiểu sai sót trong quá trình chẩn đoán. Ngoài ra, thông qua việc phân tích các yếu tố nguy cơ, mô hình cung cấp thông tin quan trọng cho việc xây dựng các biện pháp phòng ngừa và nâng cao nhận thức cộng đồng về chăm sóc sức khỏe. Kết quả của nghiên cứu có thể được áp dụng rộng rãi trong các cơ sở y tế, góp phần cải thiện chất lượng chăm sóc sức khỏe và phòng ngừa bệnh tim mạch hiệu quả hơn.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về kỹ thuật khai phá dữ liệu (Data Mining)

2.1.1. Khái niệm khai phá dữ liệu

Khai phá dữ liệu (*Data Mining*) là quá trình phân tích, tính toán nhằm tìm ra các thông tin ẩn, hữu ích trong các tập dữ liệu lớn. Khai phá dữ liệu sử dụng các kỹ thuật cũng như kế thừa các khái niệm của nhiều lĩnh vực, điển hình là: học máy (machine learning), dự báo (prediction), thống kê (statistic analysis), hồi quy (regression), phân lớp (classification), phân cụm (clustering… để khám phá ra các mẫu, quy luật hoặc thông tin quan trọng có giá trị từ bộ dữ liệu lớn có sẵn. Đây là một phần của "KDD" (*Knowledge Discovery In Databases - Khai phá tri thức từ dữ liệu).*

2.1.2. Các giai đoạn của quá trình khai phá dữ liệu

Quy trình khai phá dữ liệu là một chuỗi lặp (iterative) và tương tác (interactive) gồm các bước (giai đoạn) bắt đầu với dữ liệu thô (raw data) và kết thúc với tri thức (knowledge of interest) đáp ứng được sự quan tâm của người sử dụng.

Các bước chính trong quy trình khai phá tri thức gồm:

- Làm sạch dữ liệu *(Data Cleaning)*: loại bỏ dữ liệu nhiễu, điều chỉnh những phần dữ liệu không nhất quán (correct data inconsistencies).
- Tích hợp dữ liệu *(Data integration)*: hợp nhất dữ liệu (merge data) từ nhiều nguồn khác nhau vào một kho dữ liệu chung.
- Trích chọn dữ liệu *(Data selection)*: là việc chọn lọc những dữ liệu liên quan từ các hệ thống lưu trữ (Databases) để phục vụ cho nhiệm vụ phân tích đã đề ra. Đồng thời, lựa chọn các thuộc tính có ý nghĩa cho việc khai phá. Giai đoạn này cũng bao gồm việc biến đổi dữ liệu về các định dạng phù hợp với từng bài toán thực tế và thuật toán khai phá dữ liệu.
- Biến đổi dữ liệu *(Data transformation)*: thực hiện việc chuyển đổi dữ liệu sang dạng phù hợp với các kỹ thuật khai phá dữ liệu ở bước sau.
- Khai phá dữ liệu *(Data mining)*: Khai phá dữ liệu là quá trình cốt lõi của quá trình tìm kiếm tri thức từ dữ liệu, tập trung ở việc xác định bài toán, lựa chọn phương pháp/kỹ thuật, thuật toán phù hợp và trích xuất ra thông tin hữu ích hoặc các mẫu đáng chú ý trong dữ liệu, tìm kiếm được càng nhiều thông tin từ dữ liệu càng tốt.
- Đánh giá mẫu *(Knowledge evaluation)*: Quá trình đánh giá các kết quả tìm được thông qua các tiêu chí nhất định.

- Biểu diễn tri thức (*Knowledge representation)*: Sử dụng các kỹ thuật như đồ thị, cây, bảng biểu, luật… để trình bày các thông tin hữu ích (tri thức) trong dữ liệu.

2.2. Bài toán phân lớp trong Khai phá dữ liệu

2.2.1. Khái niệm phân lớp

Phân lớp (Classification) là một phương pháp trong học máy và khai phá dữ liệu, nhằm phân loại các đối tượng hoặc dữ liệu vào các nhóm hay lớp khác nhau dựa trên các đặc trưng của chúng. Cụ thể, mục tiêu của phân lớp là tìm một hàm ánh xạ từ dữ liệu đầu vào (thường là các đặc trưng hoặc thuộc tính của đối tượng) sang một nhãn hoặc lớp đầu ra đã xác định trước. Các nhãn hoặc lớp này đại diện cho các loại khác nhau mà dữ liệu có thể thuộc vào.

2.2.2. Quá trình phân lớp dữ liệu

Quá trình gồm hai bước:

Bước 1: Learning (giai đoạn huấn luyện): Quá trình học nhằm xây dựng một mô hình mô tả một tập các lớp dữ liệu hay các khái niệm định trước. Đầu vào của quá trình này là một tập dữ liệu có cấu trúc được mô tả bằng các thuộc tính và được tạo ra từ tập các bộ giá trị của các thuộc tính đó

Bước 2: Classification (phân loại): Nếu độ chính xác của mô hình được ước lượng dựa trên tập dữ liệu đào tạo thì kết quả thu được là rất khả quan vì mô hình luôn có xu hướng "overfitting" dữ liệu. Do vậy cần sử dụng một tập dữ liệu kiểm tra độc lập với tập dữ liệu đào tạo. Nếu độ chính xác của mô hình là chấp nhận được, thì mô hình được sử dụng để phân lớp những dữ liệu tương lai, hoặc những dữ liệu mà giá trị của thuộc tính phân lớp là chưa biết. (Linh, 2005)

2.2.3. Thuật toán phân lớp Random Forest

2.2.3.1. Giới thiệu thuật toán Random Forest

Random Forest là một thuật toán học có giám sát, được phát triển bởi Leo Breiman vào năm 2001, sử dụng phương pháp tổng hợp (ensemble method). Ý tưởng chính của Random Forest là kết hợp nhiều cây quyết định (decision trees) để tạo thành một "rừng" cây, từ đó tăng cường độ chính xác và ổn định của chẩn đoán. Thay vì sử dụng một cây quyết định duy nhất, Random Forest tạo ra một tập hợp các cây quyết định độc lập thông qua hai kỹ thuật chính:

- **Bootstrap Aggregating (Bagging):** Phương pháp này tạo ra các mẫu dữ liệu ngẫu nhiên có thay thế từ tập dữ liệu huấn luyện ban đầu. Mỗi cây quyết định được huấn luyện trên một tập con khác nhau, giúp mô hình đa dạng hơn và giảm thiểu hiện tượng quá vừa (overfitting). (Breiman, 2001)

- **Random Subspace Method:** Tại mỗi nút phân chia trong từng cây quyết định, một tập hợp ngẫu nhiên các đặc trưng (features) được chọn thay vì sử dụng toàn bộ đặc trưng. Điều này tạo ra sự ngẫu nhiên trong mỗi cây, giúp giảm sự phụ thuộc vào một số đặc trưng nhất định và làm tăng khả năng tổng quát hóa của mô hình.

2.2.3.2. Cách thức hoạt động của Random Forest

Bước 1: Chuẩn bị dữ liệu huấn luyện

Dữ liệu đầu vào sẽ được chia thành hai tập huấn luyện và tập kiểm tra. Tập huấn luyện dùng để xây dựng mô hình, trong khi tập kiểm tra dùng để đánh giá độ chính xác của mô hình.

Bước 2: Xây dựng tập hợp cây quyết định

Từ tập huấn luyện, Random Forest sẽ lấy mẫu ngẫu nhiên có thay thế nhiều lần để tạo thành các tập con. Mỗi tập con này sẽ được sử dụng để huấn luyện một cây quyết định độc lập. Quá trình này được lặp lại cho đến khi đạt số lượng cây quyết định mong muốn.

Bước 3: Tạo dự đoán

Khi có một mẫu dữ liệu mới, mỗi cây trong Random Forest sẽ đưa ra dự đoán riêng lẻ về nhãn của mẫu này. Đối với bài toán phân lớp, mỗi cây sẽ "bỏ phiếu" cho lớp mà nó cho là phù hợp nhất.

Bước 4: Tổng hợp kết quả

Sau khi thu thập các dự đoán từ tất cả các cây, lớp nào có số phiếu cao nhất sẽ được chọn làm nhãn cuối cùng của mẫu đó.

2.2.4. Thuật toán Decision Tree

2.2.4.1. Giới thiệu thuật toán

Thuật toán Decision Tree là một kỹ thuật học máy (machine learning) phổ biến trong các bài toán phân loại và hồi quy. Nó hoạt động bằng cách chia nhỏ dữ liệu dựa trên các thuộc tính và xây dựng một "cây quyết định" để đưa ra dự đoán.

Cây quyết định (Decision Tree) thường được biểu diễn như một cấu trúc phân nhánh, nơi:

- Nút gốc (Root Node) đại diện cho toàn bộ tập dữ liệu.
- Nút bên trong (Internal Node) là các điều kiện kiểm tra trên một thuộc tính.
- Nút lá (Leaf Node) chứa kết quả cuối cùng (nhãn phân loại hoặc giá trị dự đoán).

2.2.4.2. Cách thức hoạt động

Bước 1: Nếu tại nút hiện thời, tất cả các đối tượng huấn luyện đều thuộc vào một lớp. Nút này chính là nút lá có tên là nhãn lớp chung của các đối tượng.

Bước 2: Sử dụng một độ đo → chọn thuộc tính điều kiện phân chia tốt nhất tập mẫu huấn luyện có tại nút.

- *Độ đo Information Gain:*
  - Dùng với ID3
  - Trong thuật toán cây quyết định chúng ta sẽ sử dụng Entropy để đánh giá mức độ tinh khiết của phân phối xác suất của một sự kiện.

    $$\text{Entropy(D)} = \text{Info (D)} = -\sum_{i=1}^{m} p_i * log_2(p_i) \ với\ p_i = \frac{|C_{i,D}|}{|D|}$$

  - Lượng thông tin cần để phân loại một phần tử trong D dựa trên thuộc tính A: $\text{Info}_A(D)$

    $$\text{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} * Info(D_j)$$

    $$\text{Gain(A)} = \text{Info(D)} - \text{Info}_A(D)$$
- *Độ đo Gain Ratio: GainRatio(A)*
  - Dùng với C4.5
  - Chuẩn hoá information gain với trị thông tin phân tách (split information): $\text{SplitInfo}_A(D)$
  - Splitting attribute A tương ứng với trị GainRatio(A) là trị lớn nhất.

    $$\text{SplitInfo}_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} * log_2\left(\frac{|D_j|}{|D|}\right)$$

    $$\text{GainRatio(A)} = \frac{Gain(A)}{\text{SplitInfoA(D)}}$$
- *Độ đo Gini Index*
  - Dùng với CART
  - Splitting attribute tương ứng với gini index nhỏ nhất để tối đa hóa sự suy giảm về độ trùng lắp giữa các phân hoạch.

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

    $$\text{Gini}_A(D) = \frac{|D_1|}{|D|} * \text{Gini}(D_1) + \frac{|D_2|}{|D|} * \text{Gini}(D_2)$$

    $$\Delta Gini(A) = \text{Gini(D)} - \text{Gini}_A(D)$$

Bước 3: Tạo một lượng nút con của nút hiện thời bằng số các giá trị khác nhau của thuộc tính được chọn. Gán cho mỗi nhánh từ nút cha đến nút con một giá trị của thuộc tính rồi phân chia các các đối tượng huấn luyện vào các nút con tương ứng.

Bước 4: Nút con K được gọi là thuần nhất (trở thành lá) nếu tất cả các đối tượng mẫu tại đó đều thuộc vào cùng một lớp.

Bước 5: Lặp lại các bước 1 - 3 đối với mỗi nút chưa thuần nhất.

2.2.5. Thuật toán K-nearest neighbors (K-NN)

2.2.5.1. Giới thiệu thuật toán

KNN (K-Nearest Neighbors) là một trong những thuật toán học có giám sát đơn giản nhất được sử dụng nhiều trong khai phá dữ liệu và học máy. Thuật toán này hoạt động dựa trên nguyên tắc so sánh khoảng cách giữa các điểm dữ liệu.

2.2.5.2. Cách thức hoạt động K-NN

Bước 1: Chuẩn bị dữ liệu

Dữ liệu đầu vào sẽ được chia thành hai tập huấn luyện và tập kiểm tra. Tập huấn luyện dùng để xây dựng mô hình, trong khi tập kiểm tra dùng để đánh giá độ chính xác của mô hình.

Bước 2: Lựa chọn tham số K

K là số điểm dữ liệu lân cận gần nhất để đưa vào quy trình bỏ phiếu đa số. Việc chọn đúng giá trị cho K là một quá trình được gọi là điều chỉnh tham số , giúp cải thiện độ chính xác của thuật toán. Việc tìm ra giá trị của K không hề dễ dàng. (Whitfield, 2024)

K càng nhỏ, mô hình càng nhạy với nhiễu; K càng lớn, mô hình càng tổng quát nhưng có thể bỏ qua chi tiết nhỏ.

Cách xác định giá trị K:

- Trong thực hành chung, việc chọn giá trị K là khi “n” biểu thị số lượng mẫu trong tập dữ liệu đào tạo của bạn. K = SQRT(n)
- Giá trị K phổ biến thường là số lẻ (tránh trường hợp hòa phiếu trong bài toán phân lớp).

Bước 3: Tính khoảng cách

Việc tính khoảng cách đóng vai trò rất quan trọng trong thuật toán KNN, giúp xác định mức độ "gần gũi" giữa điểm cần dự đoán và các điểm trong tập dữ liệu huấn luyện. Khoảng cách này là cơ sở để đánh giá xem điểm nào giống với điểm cần dự đoán nhất. Dưới đây là một số công thức tính khoảng cách thường được sử dụng trong KNN.

Euclid - Phù hợp với dữ liệu liên tục và đã chuẩn hóa.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(pi - qi)^2}$$

Manhattan - Tốt với dữ liệu có sự phân bố đặc thù hoặc không cần chuẩn hóa.

$$d(p,q) = \sum_{i=1}^{n} |pi - qi|$$

Cosine - Dùng khi xử lý dữ liệu dạng vector, ví dụ dữ liệu văn bản.

$$d(p,q) = 1 - \frac{\vec{p}.\vec{q}}{||\vec{p}||\,||\vec{q}||}$$

Bước 4: Lựa chọn K lân cận và gán nhãn phân lớp

Sau khi tính khoảng cách từ điểm mới đến toàn bộ các điểm trong tập huấn luyện, chọn K điểm gần nhất. Sau đó phân loại hoặc chẩn đoán nhãn của điểm mới dựa trên nhãn của K điểm gần nhất.

Với bài toán phân loại: Nhãn của điểm mới sẽ được quyết định bởi nhãn chiếm đa số trong K điểm gần nhất (quy tắc số đông - majority voting).

Bước 5: Đánh giá mô hình

Sử dụng tập kiểm tra: Dự đoán nhãn cho các điểm trong tập kiểm tra, so sánh nhãn dự đoán với nhãn thực tế để đánh giá độ chính xác.

Sử dung các chỉ số đánh giá:

- Độ đo chính xác (Accuracy) $= \frac{TP+TN}{TP+FP+TN+FN}$
- Precision, Recall, F1-score: Áp dụng khi bài toán có nhiều nhãn hoặc dữ liệu mất cân bằng.

Tối ưu hóa mô hình: Dựa trên kết quả đánh giá, điều chỉnh K, chọn công thức khoảng cách phù hợp hoặc áp dụng thêm các kỹ thuật xử lý dữ liệu để cải thiện kết quả

2.2.6. Thuật toán Naive Bayes

2.2.6.1. Giới thiệu thuật toán

Naive Bayes là thuật toán học có giám sát. "Naive" chỉ ra bộ phân loại Naive Bayes sử dụng một giả định đơn giản rằng các đặc trưng (features) trong dữ liệu là độc lập có điều kiện với nhau, dựa trên nhãn lớp. Phần "Bayes" trong tên ám chỉ đến Thomas Bayes, một nhà thống kê và thần học thế kỷ 18, người đã xây dựng định lý Bayes. Thuật toán hoạt động dựa trên xác suất, với giả định rằng sự xuất hiện của một đặc trưng không phụ thuộc vào đặc trưng khác.

Định lý Bayes:

Với P(B) > 0:

$$P(A|B) = \frac{P(AB)}{P(B)}$$

Suy ra:

$$P(AB)=P(A|B)P(B)=P(B|A)P(A)$$

Công thức Bayes:

$$P(B|A) = \frac{P(AB)}{P(A)} = \frac{P(A|B)P(B)}{P(A)} = \frac{P(A|B)P(B)}{P(AB)+P(A\bar{B})} = \frac{P(A|B)P(B)}{P(A|B)P(B)+P(A|\bar{B})P(\bar{B})}$$

Trong đó:

- P(A|B) là xác suất của biến cố A xảy ra khi biết rằng B đã xảy ra.
- P(B|A) là xác suất của biến cố B xảy ra khi biết rằng A đã xảy ra.
- P(A) và P(B) là xác suất của các biến cố A và B.

Các mô hình phân loại Naive Bayes phổ biến gồm:

- **Gaussian Naive Bayes (GaussianNB)**: Được sử dụng với các dữ liệu liên tục tuân theo phân phối chuẩn (Gaussian distribution). Mô hình học bằng cách tính toán giá trị trung bình và độ lệch chuẩn cho mỗi lớp.
- **Multinomial Naive Bayes (MultinomialNB)**: Thích hợp với các biến rời rạc (discrete variables) tuân theo phân phối đa thức, như tần suất xuất hiện từ trong văn bản. Ứng dụng rộng rãi trong xử lý ngôn ngữ tự nhiên (NLP), ví dụ: phân loại email rác.
- **Bernoulli Naive Bayes (BernoulliNB)**: Được sử dụng khi các biến có giá trị nhị phân (Boolean), ví dụ: Đúng/Sai hoặc 1/0. Naïve Bayes còn có **Categorical NB**, một mô hình phân loại được thiết kế để xử lý các thuộc tính dạng danh mục, không chỉ phù hợp với các đặc trưng nhị phân (có/không) mà còn có thể xử lý nhiều giá trị khác nhau của một đặc trưng.

2.2.6.2. Cách thức hoạt động

Bước 1: Biểu diễn dữ liệu

- Gọi D là tập dữ liệu huấn luyện, trong đó mỗi phần tử dữ liệu X được biểu diễn dưới dạng một vector có n giá trị thuộc tính:
  X= (A1, A2, ..., An) = (x1, x2, ..., xn)
- Giả sử có m lớp: C1, C2, ..., Cm.
- Với một phần tử dữ liệu X, nhiệm vụ của bộ phân lớp là gán nhãn cho X vào một trong các lớp Ci sao cho xác suất hậu nghiệm P(Ci|X) là lớn nhất.

Bước 2: Tiêu chí phân loại

- Bộ phân lớp Bayes sẽ dự đoán **X** thuộc lớp Ci nếu và chỉ nếu:

  $P(Ci|X) > P(Cj|X), \forall i \neq j \ (1 \leq i, j \leq m)$.
- Xác suất hậu nghiệm P(Ci|X) được tính theo **định lý Bayes**

Bước 3: Ước lượng xác suất tiền nghiệm P(Ci)

- P(Ci) được ước lượng bằng tỉ lệ phần tử thuộc lớp Ci trong tập dữ liệu huấn luyện:

  $P(Ci) = \frac{P(Di)}{P(D)}$

  Trong đó |Di| là số phần tử dữ liệu thuộc lớp Ci và |D| là tổng số phần tử dữ liệu trong tập huấn luyện.
- Nếu không có thông tin nào về xác suất tiền nghiệm P(Ci), ta giả sử các lớp có xác suất như nhau:

  $P(C1) = P(C2) = ... = P(Cm)$.

  Khi đó, bài toán sẽ chỉ cần tìm lớp có giá trị **P(X | C_i)** lớn nhất.

Bước 4: Giả thiết độc lập có điều kiện (Naive Bayes) và dự đoán lớp

Khi số lượng thuộc tính n của dữ liệu lớn, việc tính xác suất P(X|Ci) là rất tốn kém. Thuật toán Naive Bayes đưa ra giả thiết rằng các thuộc tính là độc lập có điều kiện với nhau, nghĩa là:

$$P(X|Ci) = P(x1, x2, ...,xn|Ci)=P(x1|Ci)\cdot P(x2|Ci)\cdot ...\cdot P(xn|Ci).P(X \mid C\_i)$$

Nhờ giả thiết này, việc tính P(X|Ci) trở nên đơn giản và hiệu quả hơn.

2.3. Cơ sở dữ liệu y khoa

2.3.1. Tổng quan về bệnh tim

Bệnh tim là nhóm các bệnh ảnh hưởng đến tim và mạch máu, như rối loạn nhịp tim, cao huyết áp, bệnh mạch vành, và bệnh van tim. Đây là nguyên nhân chính gây tử vong, với khoảng 17,9 triệu người chết mỗi năm, trong đó 85% do nhồi máu cơ tim và đột quỵ. Ở Việt Nam, gần 200.000 người chết mỗi năm vì bệnh tim.

Bệnh tim mạch đang gia tăng ở người trẻ, nhưng nhiều người không nhận thức được nguy cơ và không thực hiện tầm soát sớm, dẫn đến biến chứng nghiêm trọng và ảnh hưởng đến sức khỏe cộng đồng.

2.3.2. Các triệu chứng liên quan đến bệnh tim

- *Đau ngực (Đau thắt ngực):* Đây là triệu chứng phổ biến nhất của các bệnh lý tim mạch. Cảm giác đau có thể là ngực bị ép, nặng, hoặc cảm giác như có vật nặng đè lên ngực. Đau thường lan ra vai, cổ, lưng hoặc cánh tay.

- *Khó thở:* Nếu tim không bơm đủ máu, có thể dẫn đến thiếu oxy trong cơ thể, gây khó thở, đặc biệt khi gắng sức hoặc khi nghỉ ngơi.
- *Mệt mỏi bất thường:* Cảm giác mệt mỏi quá mức hoặc thiếu năng lượng, thậm chí khi không làm việc nặng, có thể là dấu hiệu của bệnh tim.
- *Phù (sưng tay, chân, mắt cá):* Khi tim không hoạt động hiệu quả, có thể dẫn đến sự tích tụ dịch trong cơ thể, gây sưng tấy ở tay, chân hoặc mắt cá.
- *Nhịp tim không đều (loạn nhịp tim):* Cảm giác tim đập nhanh, đập chậm hoặc đập không đều có thể là dấu hiệu của các vấn đề về nhịp tim.

2.3.3. Các yếu tố ảnh hưởng đến bệnh tim

- *Tuổi tác:* Nguy cơ mắc bệnh tim tăng theo độ tuổi. Nam giới thường có nguy cơ cao hơn phụ nữ trong độ tuổi dưới 55, nhưng phụ nữ có thể có nguy cơ cao hơn sau thời kỳ mãn kinh.
- *Giới tính:* Nam giới có xu hướng mắc bệnh tim ở độ tuổi trẻ hơn so với nữ giới. Tuy nhiên, nguy cơ bệnh tim của phụ nữ tăng lên sau mãn kinh.
- *Lịch sử bệnh gia đình:* Tiền sử gia đình có người mắc bệnh tim có thể làm tăng nguy cơ mắc bệnh tim ở thế hệ sau.
- *Thiếu vận động thể chất:* Lười vận động là một yếu tố nguy cơ lớn đối với bệnh tim.
- *Béo phì:* Béo phì, đặc biệt là béo bụng, làm tăng nguy cơ mắc các bệnh tim mạch do làm tăng lượng cholesterol xấu, huyết áp cao và rối loạn chuyển hóa.

CHƯƠNG 3: XÂY DỰNG MÔ HÌNH CHẨN ĐOÁN

3.1. Cơ sở dữ liệu xây dựng mô hình

3.1.1. Chiến lược phân tích

- Với tham số cơ bản và các cách thức chia tập dữ liệu khác nhau, thực hiện chạy 5 mô hình Decision Tree, Random Forest, C4.5, Naive Bayes, K-NN, tạo bảng số liệu các độ đo.
- Dựa trên bảng số liệu đã tạo, đi vào chi tiết các mô hình và phương pháp chia dữ liệu tương ứng với nó cho kết quả tốt.
- Điều chỉnh các biến độc lập dựa vào mức độ quan trọng của thuộc tính nhằm cải thiện kết quả mô hình.
- Tìm tham số phù hợp nhất với mỗi mô hình để cải thiện kết quả. y khoa của bệnh nhân và chẩn đoán nguy cơ mắc bệnh tim của họ.

3.1.2. Mô tả thuộc tính

Thuộc tính	**Giá trị**
Age	Cho biết tuổi của người đó
Sex	Cho biết giới tính của người đó có 2 giá trị là F và M
Chest Pain Type	Cho biết kiểu đau ngực của người đó gồm 4 giá trị: TA, ATA, NAP, ASY
Resting BP	Cho biết huyết áp khi nghỉ của người đó
Cholesterol	Cho biết Mức cholesterol của người đó
FastingBS	Cho biết mức độ đường huyết khi đói của người đó
Resting Ecg	Cho biết điện tâm đồ của người đó lúc nghỉ được đo bằng các giá trị: Normal, ST.
MaxHR	Nhịp tim tối đa khi tập luyện của người đó
Exercise Angina	Cho biết người đó có đau ngực khi gắng sức không gồm 2 giá trị là N và Y
Old Peak	ST Depression
St Slope	Cho biết độ dốc đoạn ST gồm 2 giá trị là Up và Flat.
Heart Disease	Là biến mục tiêu cho biết người đó có bị mắc bệnh tim hay không.

Bảng 1: Các thuộc tính trong bộ dữ liệu

3.1.3. Triển khai đọc dữ liệu với Python

- **Hiển thị 10 dòng đầu tiên của tập dữ liệu**

```
[27]: df.head(10)
```

[27]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpe
0	40	M	ATA	140	289	0	Normal	172	N	
1	49	F	NAP	160	180	0	Normal	156	N	
2	37	M	ATA	130	283	0	ST	98	N	
3	48	F	ASY	138	214	0	Normal	108	Y	
4	54	M	NAP	150	195	0	Normal	122	N	
5	39	M	NAP	120	339	0	Normal	170	N	
6	45	F	ATA	130	237	0	Normal	170	N	
7	54	M	ATA	110	208	0	Normal	142	N	
8	37	M	ASY	140	207	0	Normal	130	Y	
9	48	F	ATA	120	284	0	Normal	120	N	

Hình 1: Hiển thị 10 dòng dữ liệu đầu tiên

- **Thông tin dữ liệu**

```
: df.shape
: (918, 12)
: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

Hình 2: Thông tin dữ liệu

3.2. Tiền xử lý dữ liệu

Dữ liệu trước khi tiền xử lý:

```
df.head(10)
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0
5	39	M	NAP	120	339	0	Normal	170	N	0.0	Up	0
6	45	F	ATA	130	237	0	Normal	170	N	0.0	Up	0
7	54	M	ATA	110	208	0	Normal	142	N	0.0	Up	0
8	37	M	ASY	140	207	0	Normal	130	Y	1.5	Flat	1
9	48	F	ATA	120	284	0	Normal	120	N	0.0	Up	0

Hình 3: Dữ liệu trước khi tiền xử lý

Thực hiện tiền xử lý dữ liệu:

Bước 1: Làm sạch dữ liệu (Data cleaning)

- Kiểm tra dữ liệu khuyết thiếu với df.isnull().sum()
- Kiểm tra dữ liệu bị trùng với df.duplicated().sum()
- Tìm và xử lý dữ liệu ngoại lai (những giá trị dữ liệu có sự khác biệt bất thường so với những giá trị dữ liệu khác, không theo quy tắc chung)
    - Ở các cột định lượng ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak'], sử dụng hàm tính phân vị thứ nhất (Q1), phân vị thứ ba (Q3), dải phân vị (IQR) và bước ngoại lệ bằng 1,5*IQR. Các điểm dữ liệu nằm dưới Q1 hoặc trên Q3 một khoảng lớn hơn hoặc bằng bước ngoại lệ được đưa vào danh sách ngoại lệ.
    - Kiểm tra các giá trị unique các cột danh mục giúp phát hiện giá trị ‘Unknown’ (nếu có) và xử lý.

Bước 2: Biến đổi dữ liệu (Data Transformation)

Thực hiện mã hóa dữ liệu bằng Label Encoding đối các cột có dữ liệu phân loại [ ]

Dữ liệu sau khi thực hiện tiền xử lý dữ liệu:

```
df.head(10)
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40.0	1	1	140.0	289.0	0	1	172.0	0	0.0	2	0
1	49.0	0	2	160.0	180.0	0	1	156.0	0	1.0	1	1
2	37.0	1	1	130.0	283.0	0	2	98.0	0	0.0	2	0
3	48.0	0	0	138.0	214.0	0	1	108.0	1	1.5	1	1
4	54.0	1	2	150.0	195.0	0	1	122.0	0	0.0	2	0
5	39.0	1	2	120.0	339.0	0	1	170.0	0	0.0	2	0
6	45.0	0	1	130.0	237.0	0	1	170.0	0	0.0	2	0
7	54.0	1	1	110.0	208.0	0	1	142.0	0	0.0	2	0
8	37.0	1	0	140.0	207.0	0	1	130.0	1	1.5	1	1
9	48.0	0	1	120.0	284.0	0	1	120.0	0	0.0	2	0

Hình 4: Dữ liệu sau khi tiền xử lý dữ liệu

3.3. Triển khai mô hình

Phân chia biến độc lập và biến phụ thuộc, tạo tham số để thử nghiệm các phương pháp chia tập dữ liệu. Để lựa chọn phương pháp chia tập huấn luyện và kiểm thử phù hợp nhất với mỗi mô hình, nhóm sử dụng phương pháp chia tập dữ liệu theo tỉ lệ (stratification):

- Stratify (chia theo tỉ lệ): Chia dữ liệu sao cho tỷ lệ phân phối các lớp giữa tập huấn luyện và kiểm thử không biến đổi, nhằm đảm bảo tính đại diện của mẫu cho mỗi lớp trong cả hai tập dữ liệu.
- Tỉ lệ tập huấn luyện / tập kiểm thử: 70:30

```
# Kiểm tra kích thước của từng tập
print(f"Số bản ghi trong X_train: {X_train.shape[0]}")
print(f"Số bản ghi trong y_train: {y_train.shape[0]}")
print(f"Số bản ghi trong X_test: {X_test.shape[0]}")
print(f"Số bản ghi trong y_test: {y_test.shape[0]}")
```

```
Số bản ghi trong X_train: 734
Số bản ghi trong y_train: 734
Số bản ghi trong X_test: 184
Số bản ghi trong y_test: 184
```

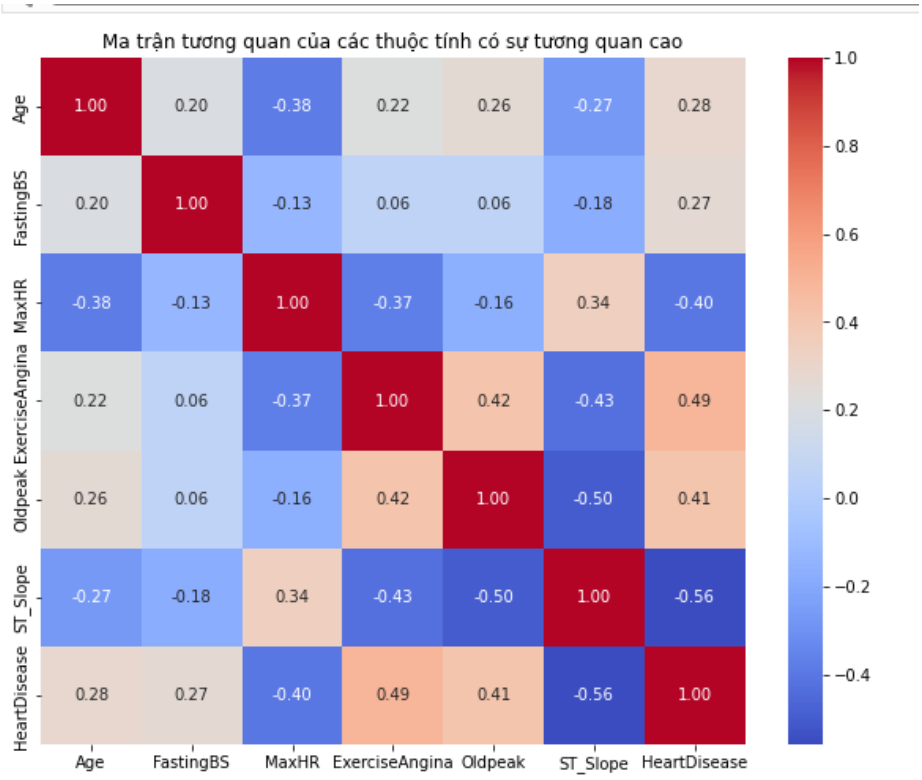
Hình 5: Kiểm tra kích thước từng tập dữ liệu sau khi phân chia

3.4. Trực quan hóa dữ liệu

3.4.1. Ma trận tương quan

- Sử dụng biểu đồ nhiệt để xây dựng ma trận tương quan.
- Nếu dải màu càng tiến về 1, -1: tương quan tuyến tính càng mạnh, càng chặt chẽ.
- Tiến về 1 là tương quan dương, tiến về -1 là tương quan âm.
- Nếu vạch màu tiến về 0: tương quan tuyến tính càng yếu.

- Nếu vạch màu tiến về 1: tương quan tuyến tính tuyệt đối.



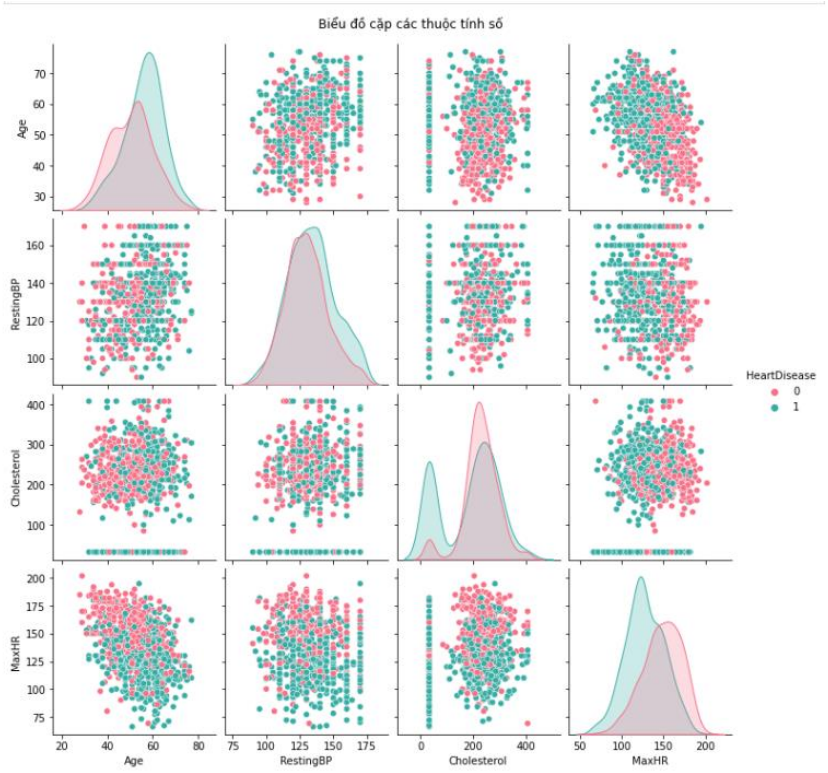
Hình 6: Ma trận tương quan của các thuộc tính có sự tương quan cao

Nhận xét:

- Age (Tuổi): Tương quan dương với FastingBS (0.20), tuổi cao có thể liên quan đến mức đường huyết cao hơn.
- FastingBS: Tương quan dương với HeartDisease (0.27), thể hiện rằng mức đường huyết lúc đói cao có thể là dấu hiệu nguy cơ tim mạch.
- MaxHR: Tương quan dương với Oldpeak (0.34), càng có giá trị Oldpeak cao thì nhịp tim tối đa cũng cao hơn, có thể cho thấy sức khỏe tim mạch tốt hơn trong điều kiện gắng sức.
- ExerciseAngina: Tương quan dương với Oldpeak (0.42), những người có triệu chứng đau ngực khi tập thể dục có thể có Oldpeak cao hơn, cho thấy sự thiếu máu cục bộ.
- ST_Slope: Tương quan dương với MaxHR (0.50), độ dốc ST tốt có thể liên quan đến nhịp tim tối đa cao hơn.

3.4.2. Biểu đồ phân tích các thuộc tính

3.4.2.1 Biểu đồ pairplot



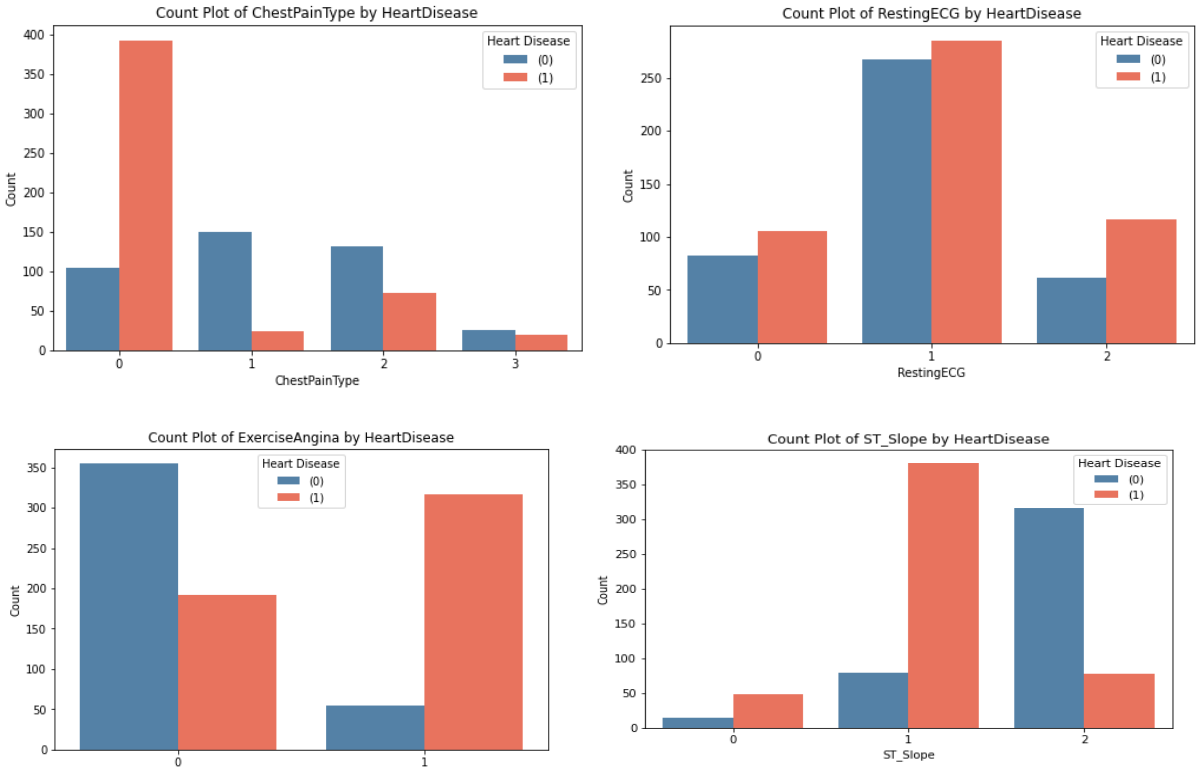
Hình 7: Biểu đồ cặp các thuộc tính số

Nhận xét:

- Age (Tuổi): Trong biểu đồ phân phối, ta thấy rằng nhóm mắc bệnh tim (đường màu hồng) thường có độ tuổi cao hơn so với nhóm không mắc bệnh tim (đường màu xanh). Độ tuổi của bệnh nhân mắc bệnh tim có xu hướng phân phối tập trung ở độ tuổi lớn hơn (trên 50), cho thấy rằng tuổi tác là một yếu tố nguy cơ quan trọng. Biểu đồ phân tán cho thấy một xu hướng giảm: khi tuổi tăng lên, nhịp tim tối đa thường giảm.
- RestingBP: Biểu đồ phân phối chỉ ra rằng huyết áp lúc nghỉ có sự chồng chéo giữa hai nhóm. Tuy nhiên, bệnh nhân mắc bệnh tim có mức huyết áp nghỉ ngơi cao hơn một chút so với nhóm không mắc bệnh tim, cho thấy huyết áp cao có thể là một yếu tố liên quan.
- Cholesterol: Phân phối cholesterol cho thấy sự khác biệt rõ rệt giữa hai nhóm. Nhóm mắc bệnh tim có mức cholesterol cao hơn so với nhóm không mắc bệnh tim, với phần lớn bệnh nhân ở dưới 200 mg/dl không mắc bệnh tim.

- MaxHR: Nhịp tim tối đa cho thấy nhóm mắc bệnh tim có xu hướng thấp hơn so với nhóm không mắc bệnh tim, với nhiều bệnh nhân mắc bệnh tim có nhịp tim tối đa dưới 140 bpm.

3.4.2.2. Biểu đồ Countplot

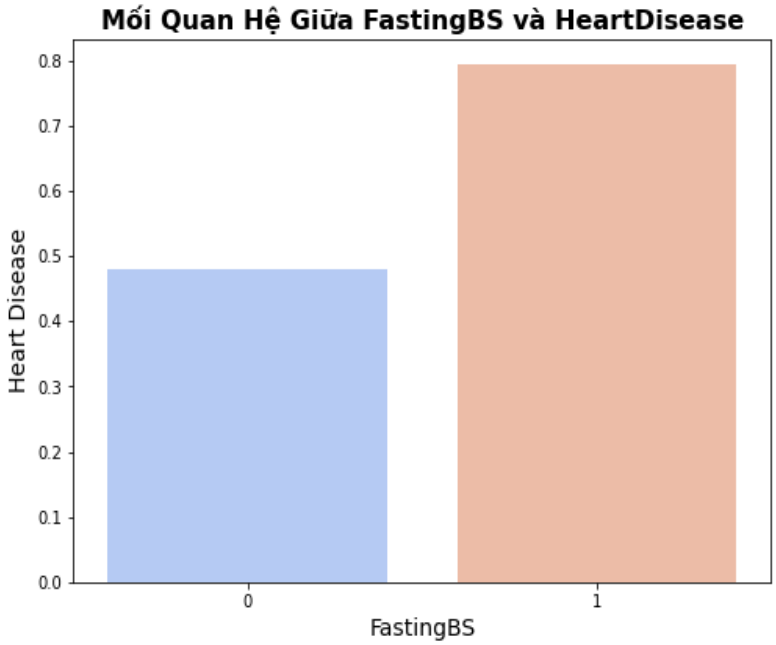


Hình 8: Biểu đồ Countplot giữa các thuộc tính

Nhận xét:

- Nam giới có nguy cơ mắc bệnh tim cao hơn nữ giới.
- Đau thắt ngực điển hình có thể là yếu tố nguy cơ chính cho bệnh tim.
- Điện tâm đồ không bình thường khi nghỉ có thể là dấu hiệu nguy cơ cao.
- Đau thắt ngực khi gắng sức có liên quan mạnh đến nguy cơ mắc bệnh tim.
- Độ dốc ST bằng phẳng liên quan đến nguy cơ mắc bệnh tim.

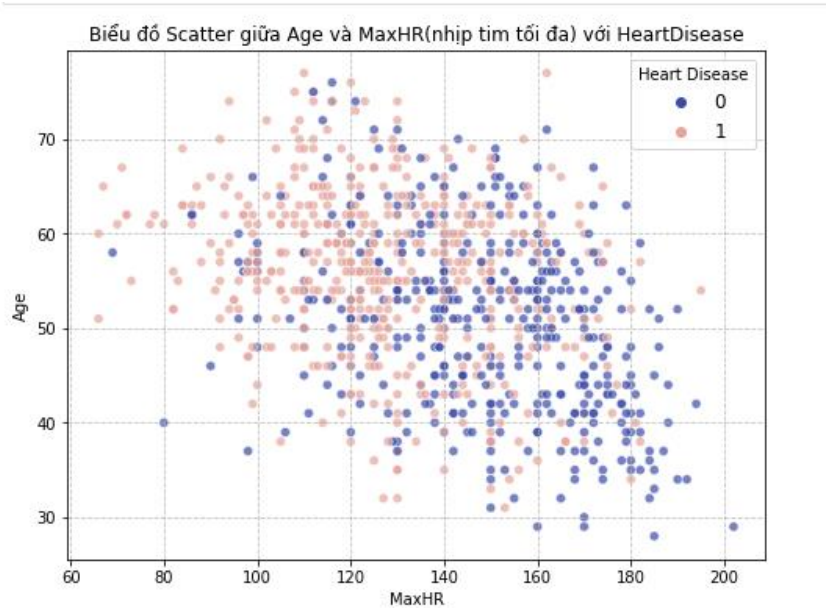
3.4.2.3. Biểu đồ Barplot



Hình 9: Biểu đồ Barplot giữa FastingBS và HeartDisease

Nhận xét: Chỉ số FastingBS (đường huyết lúc đói) < 120 có tỉ lệ mắc bệnh tim cao hơn

3.4.2.4. Biểu đồ Scatter



Hình 10: Biểu đồ Scatter giữa Age và MaxHR với Heart Disease

Nhận xét: Nhóm bệnh nhân mắc bệnh tim thường có nhịp tim tối đa thấp hơn và tuổi cao hơn.

3.4.2.5. Biểu đồ Catplot

Hình 11: Biểu đồ Catplot của tuổi theo giới tính và bệnh tim

Nhận xét: Người lớn tuổi đặc biệt là nam giới có nhiều khả năng mắc bệnh tim.

CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1. Thực nghiệm xây dựng mô hình chẩn đoán bệnh tim dựa trên thuật toán RandomForest

- Chuẩn bị dữ liệu
  - Xác định thuộc tính mô tả X và thuộc tính chẩn đoán Y ta tách thuộc tính HeartDisease thành thuộc tính chẩn đoán Y
  - Tạo mẫu đào tạo thực nghiệm với tỉ lệ 70:30
  - Kiểm tra kích thước của từng tập dữ liệu
  - Chuẩn hóa cả tập huấn luyện và tập kiểm thử bằng cách loại bỏ median và scale theo IQR. Điều này giúp các đặc trưng có cùng tầm quan trọng và cải thiện hiệu suất của các mô hình machine learning.

```
Dữ liệu sau chuẩn hóa (tập huấn luyện):
        Age  Sex  ChestPainType  RestingBP  Cholesterol  FastingBS  \
0  0.307692  0.0            0.0       -1.5     0.120690        0.0
1  0.538462  0.0            0.0       -1.0    -2.193966        1.0
2 -0.076923  0.0            1.0        1.5    -2.193966        1.0
3  0.692308  0.0            0.0        1.5     0.074713        1.0
4  0.692308  0.0            0.0        0.0     1.224138        1.0

   RestingECG     MaxHR  ExerciseAngina   Oldpeak  ST_Slope
0         0.0  0.514286             0.0 -0.266667       1.0
1         0.0 -0.857143             1.0  1.000000      -1.0
2        -1.0 -0.457143             1.0 -0.333333       0.0
3         0.0 -0.942857             1.0  0.333333       0.0
4        -1.0 -0.171429             1.0  0.866667       1.0
```

Hình 12: Random Forest - chuẩn bị dữ liệu

- Xây dựng mô hình Random Forest và in ra kết quả

```
==ma trận nhầm lẫn ===
[[100  12]
 [ 25 139]]


== báo cáo phân loại ===
              precision    recall  f1-score   support

           0       0.80      0.89      0.84       112
           1       0.92      0.85      0.88       164

    accuracy                           0.87       276
   macro avg       0.86      0.87      0.86       276
weighted avg       0.87      0.87      0.87       276
```

Hình 13: Random Forest - kết quả chẩn đoán

Nhận xét:

- Accuracy = 0.87: Độ chính xác của mô hình = 87%
- True Positive = 139: có 139 bệnh nhân mắc chứng suy tim được chẩn đoán đúng.

- True Negative = 100: có 100 bệnh nhân không mắc chứng suy tim được chẩn đoán đúng
- False Positive = 12: có 12 bệnh nhân không mắc chứng suy tim nhưng được chẩn đoán là có chấp nhận.
- False Negative = 25: có 25 bệnh nhân mắc chứng suy tim nhưng được chẩn đoán là không chấp nhận

Lớp 0: Không bị suy tim

- Precison = 0.80: Độ chính xác = 0.80
- Recall = 0.89: Độ hồi tưởng = 0.89
- F1-score = 0.84: Độ đo F = 0.84

Lớp 1: Bị suy tim

- Precison = 0.92: Độ chính xác = 0.92
- Recall = 0.85: Độ hồi tưởng = 0.85
- F1-score = 0.88: Độ đo F = 0.88

➢ **Điều chỉnh các siêu tham số**

Sau khi chạy thuật toán chúng em thấy độ chính xác vẫn còn chưa cao, do đó chúng em muốn giới thiệu bài viết của William Koehrsen, " Hyperparameter Tuning the Random Forest in Python " để có mô tả chi tiết hơn về quy trình. Chúng em sẽ sử dụng RandomizedSearchCV from sklearn để tối ưu hóa các siêu tham số của. Koehrsen sử dụng toàn bộ lưới siêu tham số trong bài báo nhưng điều này có thể mất một thời gian đáng kể để chạy trên thực tế. Chúng em quyết định tập trung vào 3 siêu tham số: n_estimators, max_features và max_depth.

Sau khi chạy ra kết quả của các tham số n_estimators, max_features và max_depth, chúng em đưa các tham số này lại vào mô hình để nâng cao độ chính xác của mô hình.

```
==ma trận nhầm lẫn ===
[[100  12]
 [ 15 149]]


== báo cáo phân loại ===
              precision    recall  f1-score   support

           0       0.87      0.89      0.88       112
           1       0.93      0.91      0.92       164

    accuracy                           0.90       276
   macro avg       0.90      0.90      0.90       276
weighted avg       0.90      0.90      0.90       276
```

Hình 14: Random Forest - kết quả sau khi điều chỉnh siêu tham số

Nhận xét: Sau khi chạy mô hình với tham số tốt nhất, ta nhận thấy độ chính xác đã tăng lên. Ma trận nhầm lẫn cũng đã có sự thay đổi.

- Accuracy = 0.90: Độ chính xác của mô hình = 90%
- True Positive = 149: có 149 bệnh nhân mắc chứng suy tim được chẩn đoán đúng.
- True Negative = 100: có 100 bệnh nhân không mắc chứng suy tim được chẩn đoán đúng
- False Positive = 12: có 12 bệnh nhân không mắc chứng suy tim nhưng được chẩn đoán là mắc bệnh.
- False Negative = 15: có 15 bệnh nhân mắc chứng suy tim nhưng được chẩn đoán là không mắc bệnh

Lớp 0: Không bị suy tim

- Precison = 0.87: Độ chính xác = 0.87
- Recall = 0.89: Độ hồi tưởng = 0.89
- F1-score = 0.88: Độ đo F = 0.88

Lớp 1: Bị suy tim

- Precison = 0.93: Độ chính xác = 0.93
- Recall = 0.91: Độ hồi tưởng = 0.91
- F1-score = 0.92: Độ đo F = 0.92

➢ Chẩn đoán mô hình

Test 1: 1 người có các chỉ số như sau: Tuổi 40, Giới tính 1, Loại đau ngực 1, Huyết áp lúc nghỉ là 140, Mức cholesterol là 289, Đường huyết khi đói là 0,Kết quả điện tâm đồ lúc nghỉ là 1, Nhịp tim tối đa đạt được 172,Cơn đau thắt ngực do gắng sức 0,Độ chênh ST sau vận động 0, Độ dốc đoạn ST2

Kết quả chẩn đoán người này không mắc bệnh

```
new_data1 = np.array([[40.0,1,1,140.0,289.0,0,1,172.0,0,0.0,2]])
```

```
prediction = rfc.predict(new_data1)  # Dự đoán nhãn (0 hoặc 1)
probability = rfc.predict_proba(new_data1)
```

```
print("Prediction:", prediction)  # Dự đoán lớp
print("Probability of Class 1:", prob_prediction)
```

```
Prediction: [0]
Probability of Class 1: [0.3]
```

```
print("Thông tin người cần dự đoán:", new_data1)
if prediction[0] == 0:
    print("Kết quả: Người này KHÔNG bị bệnh.")
else:
    print("Kết quả: Người này CÓ BỆNH.")
```

```
Thông tin người cần dự đoán: [[ 40.   1.   1. 140. 289.   0.   1. 172.   0.   0.   2.]]
Kết quả: Người này KHÔNG bị bệnh.
```

Hình 15: Random Forest - kết quả chẩn đoán mô hình test 1

Test 2: 1 người có các chỉ số như sau: Tuổi 48, Giới tính 0, Loại đau ngực 0, Huyết áp lúc nghỉ là 138, Mức cholesterol là 214, Đường huyết khi đói là 0, Kết quả điện tâm đồ lúc nghỉ là 1, Nhịp tim tối đa đạt được 108, Cơn đau thắt ngực do gắng sức 1, Độ chênh ST sau vận động 1.5, Độ dốc đoạn ST2 1

Kết quả chẩn đoán là người này có bệnh

```
new_data2 = np.array([[48.0,0,0,138.0,214.0,0,1,108.0,1,1.5,1]])
```

```
prediction = rfc.predict(new_data2)  # Dự đoán nhãn (0 hoặc 1)
probability = rfc.predict_proba(new_data2)
```

```
print("Prediction:", prediction)  # Dự đoán lớp
print("Probability of Class 1:", prob_prediction)
```

```
Prediction: [1]
Probability of Class 1: [0.3]
```

```
print("Thông tin người cần dự đoán:", new_data2)
if prediction[0] == 0:
    print("Kết quả: Người này KHÔNG bị bệnh.")
else:
    print("Kết quả: Người này CÓ BỆNH.")
```

```
Thông tin người cần dự đoán: [[ 48.   0.   0. 138. 214.   0.   1. 108.   1.   1.5  1. ]]
Kết quả: Người này CÓ BỆNH.
```

Hình 16: Random Forest - kết quả chẩn đoán mô hình test 2

4.2. Xây dựng mô hình bằng một số thuật toán phân lớp khác

4.2.1 Thuật toán Decision Tree

- Chuẩn bị tập dữ liệu huấn luyện**:** Dựa trên bộ dữ liệu đã được tiền xử lý
  - Tạo mẫu đào tạo thực nghiệm với tỉ lệ 70:30

- Huấn luyện mô hình Decision Tree: Mô hình Decision Tree sẽ học thông qua tập huấn luyện. Quá trình huấn luyện là việc mô hình tìm các thuộc tính và ngưỡng tối ưu để tạo ra cây quyết định.
- Chẩn đoán mô hình Decision Tree: Sau khi huấn luyện xong, sử dụng mô hình đã huấn luyện để chẩn đoán kết quả trên tập dữ liệu mới hoặc tập kiểm thử.
- Đánh giá độ chính xác

➢ Xây dựng mô hình và in ra kết quả

```
Accuracy: 83.15%
Báo cáo:
              precision    recall  f1-score   support

           0       0.77      0.86      0.81        77
           1       0.89      0.81      0.85       107

    accuracy                           0.83       184
   macro avg       0.83      0.84      0.83       184
weighted avg       0.84      0.83      0.83       184

Ma trận nhầm lẫn:
[[66 11]
 [20 87]]
```

Hình 17: Decision Tree – kết quả chẩn đoán

Nhận xét:

- Độ chính xác: Mô hình đạt 83.15% accuracy, là mức khá tốt nhưng còn khả năng cải thiện.
- Hiệu suất từng lớp:
  - Lớp 0: Precision = 0.77, Recall = 0.86, F1-Score = 0.81 (chẩn đoán ổn).
  - Lớp 1: Precision = 0.89, Recall = 0.81, F1-Score = 0.85 (chẩn đoán tốt hơn lớp 0).
- Nhận xét ma trận nhầm lẫn:
  - Lớp 0: 11 mẫu bị nhầm thành lớp 1.
  - Lớp 1: 20 mẫu bị nhầm thành lớp 0 (cần chú ý nếu lớp 1 quan trọng).

4.2.2. Thuật toán K-nearest neighbors (K-NN)

➢ Chuẩn bị dữ liệu:

Phân chia dữ liệu thành 2 tập chính là Tập huấn luyện và kiểm tra với tỉ lệ 70:30

➢ Thực nghiệm thuật toán K-NN với giá trị k= 3 ngẫu nhiên

Nhận thấy chọn ngẫu nhiên k =3 kết quả chạy mô hình đạt được độ chính xác 68% có thể không đạt được hiệu suất cao nhất vì chưa tối ưu hóa được sự cân bằng giữa độ nhạy và độ chính xác tổng thể. Vì vậy em sẽ cải tiến nó bằng việc lựa chọn k tối ưu giúp mô hình hoạt động tốt nhất trên dữ liệu kiểm tra.

- Thực nghiệm thuật toán K-NN với giá trị k tối ưu:
- Đánh giá độ chính xác

```
Accuracy: 73.91%
Confusion Matrix:
[[ 86  37]
 [ 35 118]]

Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.70      0.70       123
           1       0.76      0.77      0.77       153

    accuracy                           0.74       276
   macro avg       0.74      0.74      0.74       276
weighted avg       0.74      0.74      0.74       276
```

Hình 18: KNN - đánh giá độ chính xác

Nhận xét:

- Độ chính xác (Accuracy): Mô hình đạt được 73.91% đạt mức trung bình, cho thấy mô hình có thể phân loại đúng đa số các mẫu.
    - Lớp 0: Có 86 trường hợp thuộc lớp 0 được chẩn đoán chính xác, nhưng có 37 trường hợp bị chẩn đoán nhầm là lớp 1
    - Lớp 1: 118 trường hợp thuộc lớp 1 được chẩn đoán chính xác, nhưng có 35 mẫu bị nhầm sang lớp 0

=> Nhìn chung, tỷ lệ nhầm lẫn giữa hai lớp là đáng kể, đặc biệt với False Negative (35) ở lớp 1, đây là nhược điểm cần lưu ý nếu việc bỏ sót trường hợp này có hậu quả nghiêm trọng

- Precision:
    - Precision của lớp 0 là 0.71: Mô hình chẩn đoán "Không mắc bệnh" chính xác 71%.
    - Precision của lớp 1 là 0.76: Mô hình chẩn đoán "Mắc bệnh" chính xác 76%.

=> Precision lớp 1 cao hơn, cho thấy mô hình hoạt động tốt hơn trong việc phân loại các trường hợp thuộc lớp 1.

- Recall:

- Recall của lớp 0 là 0.70: Mô hình nhận diện đúng 70% trường hợp thực tế thuộc lớp không mắc bệnh.
- Recall của lớp 1 là 0.77: Mô hình nhận diện đúng 77% trường hợp thực tế mắc bệnh.

- F1-score: Phản ảnh sự cân bằng giữa Precision và Recall. Trong mô hình ta thấy F1-score của lớp 0 là 0.70 và F1-score của lớp 1 là 0.77.

=> Mô hình hoạt động tốt hơn trong việc chẩn đoán lớp 1, với sự cân bằng giữa Precision và Recall cao hơn.

4.2.3. Thuật toán Naive Bayes

4.2.3.1. Giới thiệu thuật toán

Dữ liệu trong bài toán của nhóm có cả dữ liệu liên tục (continuous data) và dữ liệu phân loại (category data). Do vậy, chúng em sẽ sử dụng kết hợp 2 mô hình Naive Bayes là Gaussian NB (xử lý dữ liệu liên tục) và Categorical NB (xử lý dữ liệu phân loại).

- Chuẩn bị dữ liệu:
  - Chia các thuộc tính của bộ dữ liệu làm 2 nhóm: Dữ liệu liên tục (bao gồm các thuộc tính 'Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak') và Dữ liệu phân loại (bao gồm các thuộc tính 'Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope')
  - Chia tập huấn luyện và tập kiểm tra với tỉ lệ 70:30
- Huấn luyện riêng từng mô hình
  - Mô hình GaussianNB: Thực hiện huấn luyện trên tập dữ liệu liên tục (X_train_cont) để tính toán xác suất cho các thuộc tính liên tục dựa trên phân phối Gaussian.
  - Mô hình CategoricalNB: Thực hiện huấn luyện trên tập dữ liệu danh mục (X_train_cate) để tính toán xác suất của các thuộc tính danh mục dựa trên bảng tần suất.
- Chẩn đoán và kết hợp xác suất
  - Dùng *predict_log_proba* để tính xác suất logarit của mỗi lớp cho mỗi mẫu dữ liệu.
  - Sau đó kết hợp xác suất của hai mô hình bằng cách cộng chúng lại.
  - Dùng *np.argmax* để tìm chỉ số của lớp có xác suất cao nhất.

➢ Độ chính xác của mô hình

```
Accuracy: 0.8586956521739131
Ma trận nhầm lẫn:
 [[ 96  16]
 [ 23 141]]
Báo cáo phân loại:
              precision    recall  f1-score   support

           0       0.81      0.86      0.83       112
           1       0.90      0.86      0.88       164

    accuracy                           0.86       276
   macro avg       0.85      0.86      0.85       276
weighted avg       0.86      0.86      0.86       276
```

Hình 19: Naive Bayes - kết quả chẩn đoán

Mô hình đạt 86% độ chính xác tổng thể. Thực hiện tối ưu cách kết hợp xác suất thông qua tối ưu trọng số kết hợp xác suất bằng Grid Search (Grid Search thử nghiệm nhiều giá trị khác nhau của α (từ 0 đến 1) và chọn giá trị mang lại độ chính xác cao nhất).

```
Accuracy: 0.894927536231884
Ma trận nhầm lẫn:
 [[100  12]
 [ 17 147]]
Báo cáo phân loại:
              precision    recall  f1-score   support

           0       0.85      0.89      0.87       112
           1       0.92      0.90      0.91       164

    accuracy                           0.89       276
   macro avg       0.89      0.89      0.89       276
weighted avg       0.90      0.89      0.90       276
```

Hình 180: Naive Bayes - kết quả chẩn đoán sau khi tối ưu

Nhận xét:

- Độ chính xác tổng thể (Accuracy): Mô hình đạt 89%, kết quả này ở mức khá cao.
  - Lớp 0: Có 100 trường hợp không mắc bệnh được chẩn đoán chính xác (True Negative). Có 12 trường hợp không mắc bệnh nhưng bị chẩn đoán nhầm là mắc bệnh (False Positive)
  - Lớp 1: Có 17 trường hợp mắc bệnh nhưng bị chẩn đoán nhầm là không mắc bệnh (False Negative). Con số này đã giảm so với 23 trường hợp trước khi tối ưu mô hình nhưng đây vẫn là một nhược điểm cần lưu ý vì có thể bỏ sót bệnh nhân cần được điều trị. Có 147 trường hợp mắc bệnh được chẩn đoán chính xác (True Positive)
- Precision: Phản ánh tỷ lệ chẩn đoán đúng so với tổng số chẩn đoán cho lớp đó.

- Precision của lớp 0 là 0.85: mô hình chẩn đoán "Không mắc bệnh" chính xác 85%, thấp hơn so với lớp 1
- Precision của lớp 1 là 0.92: mô hình chẩn đoán "Mắc bệnh" chính xác 92%

- Recall: Phản ánh tỷ lệ chẩn đoán đúng trên tổng số mẫu thực tế thuộc lớp đó. Recall của lớp 0 là 0.89 và của lớp 1 là 0.90, có nghĩa là mô hình nhận diện đúng 89% trường hợp "Không mắc bệnh" và 90% trường hợp "Mắc bệnh" trong thực tế, tỉ lệ này là khá tốt.
- F1-score: Phản ảnh sự cân bằng giữa Precision và Recall. Ở đây F1-score cao hơn ở lớp 1 (0.91) so với lớp 0 (0.87), mô hình hoạt động tốt hơn trong việc chẩn đoán người mắc bệnh (lớp 1).

4.3. So sánh Random Forest với 3 thuật toán còn lại và đánh giá chung

Thuật toán	Accuracy	Precision (Lớp 0)	Precision (Lớp 1)	Recall (Lớp 0)	Recall (Lớp 1)	F1-score (Lớp 0)	F1-score (Lớp 1)
Random Forest	90%	0.87	0.93	0.89	0.91	0.88	0.92
Naive Bayes	89%	0.85	0.92	0.89	0.90	0.87	0.91
Decision Tree	83.15%	0.77	0.89	0.86	0.81	0.81	0.85
K-NN	73.91%	0.71	0.76	0.70	0.77	0.70	0.77

Hình 21: So sánh hiệu suất các thuật toán

Dựa trên giá trị của các độ đo chính, có thể thấy:

- Accuracy: Random Forest có độ chính xác tổng thể khá cao (90%) và cao nhất trong 4 thuật toán. Theo sau là Naive Bayes với 89%, Decision Tree với 83% và K-NN chỉ khoảng 74%
- Precision: Random Forest có tỷ lệ chẩn đoán cao hơn ở cả hai lớp "Mắc bệnh" và "Không mắc bệnh" so với 3 thuật toán còn lại.
- Recall: Nhìn chung, Random Forest và Naive Bayes là hai thuật toán có recall cao nhất trong bốn thuật toán, đối với chẩn đoán "Không mắc bệnh" (lớp 0), hai

thuật toán này đều có độ chính xác 89%. Tuy nhiên đối với lớp "Mắc bệnh (lớp 1) thì Random Forest có độ chính xác 91%, nhỉnh hơn 1% so với Naive Bayes.

- F1-score: Random Forest và Naive Bayes vẫn là hai thuật toán có sự cân bằng giữa Precision và Recall cao hơn cả. Random Forest với 0.88 cho lớp 0 và 0.92 cho lớp 1, Naive Bayes chỉ nhỏ hơn 0.01 so với Random Forest ở mỗi lớp

Dựa trên hiệu suất tổng thể ở cả 4 chỉ số trên thì Random Forest thể hiện sự vượt trội hơn so với các thuật toán còn lại trong việc chẩn đoán khả năng mắc bệnh tim mạch. Naïve Bayes và Decision Tree có hiệu suất khá ổn, K-NN chưa thực sự phù hợp. Tuy nhiên, việc xác định và lựa chọn thuật toán phù hợp không chỉ dựa vào hiệu suất, mà còn phải cân nhắc các yếu tố khác như: tài nguyên hệ thống, độ phức tạp của dữ liệu và đặc biệt là mục tiêu của bài toán.

Random Forest có độ phức tạp cao hơn so với một số thuật toán như K-NN hay Decision Tree, đặc biệt khi xử lý dữ liệu lớn. Quá trình xây dựng và tổng hợp nhiều cây quyết định yêu cầu nhiều bước lặp lại để tối ưu hóa các tiêu chí phân chia, dẫn đến thời gian huấn luyện dài hơn, đòi hỏi tài nguyên tính toán cao và không phù hợp với hệ thống có giới hạn về phần cứng. Tuy nhiên, Random Forest có khả năng song song hóa tốt, cho phép chia nhỏ việc xây dựng các cây quyết định để xử lý trên nhiều bộ xử lý (parallel computing). Điều này giúp giảm thời gian xử lý trên các tập dữ liệu lớn.

Random Forest có khả năng xử lý tốt dữ liệu lớn và hỗ trợ phát hiện các đặc trưng quan trọng, nhờ cơ chế ngẫu nhiên hóa trong việc chọn mẫu và thuộc tính. Điều này giúp giảm thiểu overfitting, đặc biệt trong bài toán có nhiều thuộc tính (12 thuộc tính như trong bộ dữ liệu hiện tại). K-NN và Naive Bayes lại gặp khó khăn hơn trong việc đối phó với các tập dữ liệu lớn hoặc mất cân bằng. K-NN dễ bị ảnh hưởng bởi số lượng mẫu và Naive Bayes có thể giả định sai về phân phối dữ liệu.

Quan trọng hơn hết, mục đích của chúng em là tìm ra thuật toán giúp chẩn đoán chính xác nhất có thể khả năng một người có mắc chứng suy tim hay không. Nếu một bệnh nhân mắc chứng suy tim nhưng bị bỏ qua, chẩn đoán sai thì sẽ có thể bỏ lỡ mất cơ hội, thời điểm thích hợp nhất để chữa trị. Để đạt được điều đó thì cần giảm thiểu hết mức có thể trường hợp False Negative (mắc chứng suy tim nhưng lại được chẩn đoán là không suy tim), hay nói cách khác là tối đa hóa Recall cho lớp "Mắc bệnh" (lớp 1). Như vậy, đối với bài toán chẩn đoán mắc chứng suy tim của một người thì **Random Forest** trả về kết quả tốt nhất.

CHƯƠNG 5: TỔNG KẾT

5.1. Kết luận

Đề tài "Ứng dụng thuật toán phân lớp vào chẩn đoán khả năng mắc bệnh tim mạch" đã đạt được mục tiêu là xây dựng một mô hình chẩn đoán hiệu quả. Trong số các thuật toán được áp dụng, Random Forest thể hiện hiệu suất vượt trội về độ chính xác và khả năng cân bằng giữa các chỉ số Precision, Recall và F1-Score. Đề tài đã chứng minh rằng việc sử dụng thuật toán phân lớp có thể hỗ trợ đắc lực trong việc chẩn đoán bệnh lý, góp phần vào việc phát hiện sớm và điều trị kịp thời cho bệnh nhân.

5.2. Hạn chế của đề tài

Tuy nhiên, hạn chế của đề tài là dữ liệu không đủ lớn cũng có thể dẫn đến việc mô hình không được huấn luyện đầy đủ và chưa đạt được hiệu quả tối ưu khi áp dụng vào môi trường thực tế. Vì vậy, để cải thiện độ chính xác và khả năng tổng quát hóa của mô hình, cần thu thập thêm các bộ dữ liệu lớn hơn và phong phú hơn, phản ánh đúng đặc điểm của bệnh nhân trong các tình huống thực tế. Việc giải quyết vấn đề này sẽ giúp mô hình chẩn đoán trở nên hiệu quả hơn, có khả năng xử lý đa dạng các trường hợp bệnh lý và đạt được độ chính xác cao hơn khi triển khai trong thực tế.

5.3. Hướng phát triển cho đề tài

Qua quá trình nghiên cứu, đề tài có thể mở rộng và cải tiến nếu được thu thập và mở rộng bộ dữ liệu. Bộ dữ liệu cần được thu thập thêm từ các bệnh viện, cơ sở y tế để nâng cao tính chính xác và khả năng áp dụng của mô hình. Bộ dữ liệu lớn và đa dạng hơn sẽ giúp mô hình phản ánh đầy đủ hơn các đặc điểm bệnh lý trong thực tế, đồng thời cải thiện khả năng tổng quát hóa của mô hình.

Thu thập dữ liệu từ nhiều nguồn khác nhau không chỉ giúp nâng cao tính chính xác của mô hình mà còn giúp phát hiện các đặc điểm và mối liên quan mà dữ liệu hiện tại chưa thể làm rõ. Đồng thời, bộ dữ liệu lớn và phong phú hơn sẽ giúp mô hình chẩn đoán khả năng mắc bệnh tim mạch trở nên mạnh mẽ và hiệu quả hơn, phù hợp với các đối tượng bệnh nhân đa dạng.

Với bộ dữ liệu thực tế đa dạng và đầy đủ hơn, mô hình sẽ có thể hỗ trợ các bác sĩ và nhân viên y tế trong việc chẩn đoán sớm, chính xác hơn, đồng thời giảm thiểu rủi ro chẩn đoán sai, từ đó góp phần nâng cao chất lượng công tác điều trị và chăm sóc sức khỏe.

TÀI LIỆU THAM KHẢO

[1]. *AIcandy*. (không ngày tháng). Được truy lục từ Thuật toán Random Forest: Giải thích chi tiết và ứng dụng: https://aicandy.vn/thuat-toan-random-forest-giai-thich-chi-tiet-va-ung-dung/

[2]. Breiman, L. (2001). *Random Forest* . The Netherlands : Kluwer Academic .

[3]. Hop, N. T. (2019, September 14). *Thuật toán phân lớp Naive Bayes*. Được truy lục từ VIBLO: https://viblo.asia/p/thuat-toan-phan-lop-naive-bayes-924lJWPm5PM

[4]. Linh, N. T. (2005). *NGHIÊN CỨU CÁC THUẬT TOÁN PHÂN LỚP DỮ LIỆU DỰA TRÊN CÂY QUYẾT ĐỊNH.* ĐẠI HỌC QUỐC GIA HÀ NỘI TRƯỜNG ĐẠI HỌC CÔNG NGHỆ.

[5]. Whitfield, B. (2024, 11 26). *Random Forest: A Complete Guide for Machine Learning*. Được truy lục từ Built in: https://builtin.com/data-science/random-forest-algorithm

[6]. Slide bài giảng học phần Khai phá và Phân tích dữ liệu Khoa Công nghệ thông tin và Kinh tế số - Học viện Ngân hàng