HỌC VIỆN NGÂN HÀNG

KHOA HỆ THỐNG THÔNG TIN QUẢN LÝ

BÁO CÁO BÀI TẬP LỚN

Học phần:

KHAI PHÁ DỮ LIỆU

Đề tài:

ỨNG DỤNG KHAI PHÁ DỮ LIỆU XÂY DỰNG MÔ HÌNH DỰ ĐOÁN KHẢ NĂNG MẮC BỆNH TIM MẠCH CỦA NGƯỜI DÂN

Giảng viên hướng dẫn : TS. Bùi Thị Hồng Nhung

Lớp tín chỉ : 231IS30A03

Nhóm sinh viên thực hiện : Nhóm 5

Hà Nội, 12/2023

HỌC VIỆN NGÂN HÀNG

KHOA HỆ THỐNG THÔNG TIN QUẢN LÝ

BÁO CÁO BÀI TẬP LỚN

Học phần:

KHAI PHÁ DỮ LIỆU

Đề tài:

ỨNG DỤNG KHAI PHÁ DỮ LIỆU XÂY DỰNG MÔ HÌNH DỰ ĐOÁN KHẢ NĂNG MẮC BỆNH TIM MẠCH CỦA NGƯỜI DÂN

Giảng viên hướng dẫn : TS. Bùi Thị Hồng Nhung

Lớp tín chỉ : 231IS30A03

Nhóm sinh viên thực hiện : Nhóm 5

Hà Nội, 12/2023

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành nhất đến TS. Bùi Thị Hồng Nhung, là giảng viên của chúng em trong học phần Khai phá dữ liệu. Cảm ơn cô đã luôn tận tình trong công việc và giúp đỡ chúng em trong quá trình hoàn thiện bài làm của nhóm.

Đồng thời, chúng em xin cảm ơn ơn khoa Hệ thống thông tin quản lý đã đưa môn học Khai phá dữ liệu vào chương trình đào tạo. Đây là môn học rất bổ ích với nhiều kiến thức và rất gần gũi với thực tiễn. Thông qua môn học này, chúng em đã biết những nguyên tắc và những bước cơ bản để khai phá những thông tin quan trọng, ẩn sâu trong những tập dữ liệu của các tổ chức. Đây là những kiến thức rất hữu ích, có tính ứng dụng cao trong thực tế công việc. Việc được tiếp cận những nội dung này trong chương trình đào tạo ngành Hệ thống thông tin quản lý sẽ là nền tảng để sinh viên chúng em làm quen và dễ dàng tiếp cận được với công việc sau này.

Trong quá trình thực hiện bài làm, với sự hướng dẫn của giảng viên, chúng em đã cố gắng vận dụng những kiến thức đã học kết hợp những sự tìm hiểu của cá nhân các thành viên với mong muốn hoàn thành bài làm một cách tốt nhất. Song, do lượng kiến thức là rất lớn, khả năng tìm hiểu còn hạn chế và còn chưa có nhiều kinh nghiệm thực tế nên bài làm của nhóm có thể còn tồn tại những thiếu sót. Chúng em rất mong sẽ nhận được ý kiến đóng góp của cô để bài làm được hoàn thiện hơn!

Chúng em xin chân thành cảm ơn!

Hà Nội, ngày 28 tháng 12 năm 2023

NHÓM SINH VIÊN THỰC HIỆN

BẢNG PHÂN CÔNG CÔNG VIỆC

Họ và tên	Mã sinh viên	Nội dung công việc
Lưu Quang Hà	22A4050260	- Phân cụm dữ liệu
Bùi Tú Phương	23A4040113	- Khai phá luật kết hợp
Nguyễn Thảo Vy	23A4040158	- Khai phá luật kết hợp
Nguyễn Thanh Bình	23A4040012	- Tiền xử lý dữ liệu - Phân lớp dữ liệu
Bùi Minh Tuấn	23A4040168	- Tiền xử lý dữ liệu - Phân lớp dữ liệu

DANH SÁCH THÀNH VIÊN NHÓM

Họ và tên	Tỷ lệ đóng góp	Chữ ký
Lưu Quang Hà	20%	
Bùi Tú Phương	20%	
Nguyễn Thảo Vy	20%	
Nguyễn Thanh Bình	20%	
Bùi Minh Tuấn	20%	

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

LỜI MỞ ĐẦU

Bệnh tim mạch là căn bệnh đang dần trở nên phổ biến ngày nay. Trong cuộc sống hiện đại, số lượng người trẻ mắc những căn bệnh liên quan đến tim mạch đang có xu hướng gia tăng. Theo thống kê từ Tổ chức Y tế Thế giới (WHO), hàng năm có khoảng 17,9 triệu người tử vong do bệnh tim, trong đó có 85% trường hợp gây ra bởi bệnh nhồi máu cơ tim và đột quỵ. Tại Việt Nam, hàng năm có gần 200.000 người chết vì bệnh tim, con số này cao hơn so với tỷ lệ tử vong do ung thư. Đáng chú ý, các loại bệnh như động mạch não, mạch vành và động mạch ngoại biên đang trở nên phổ biến ở những người trẻ trong khi trước đây, các bệnh trên thường xuất hiện ở người cao tuổi (Bệnh viện Đa khoa Vinmec, n.d.).

Có rất nhiều nguyên nhân dẫn đến những căn bệnh về tim mạch. Hiện nay, trước những áp lực của công việc và cuộc sống, đời sống sức khoẻ về vật chất và tinh thần của con người thường không được đảm bảo. Nhiều người, đặc biệt là người trẻ, thường xuyên sử dụng những chất kích thích như rượu bia, thuốc lá, cà phê, … là những chất không tốt cho sức khoẻ tim mạch. Căn bệnh này thường biểu hiện thành một số triệu chứng như khó thở, tức ngực, kiệt sức, …

Nhận thức được sự nguy hiểm của căn bệnh đối với con người, nhóm chúng em quyết định lựa chọn đề tài ***"Ứng dụng khai phá dữ liệu xây dựng mô hình dự đoán khả năng mắc bệnh tim mạch của người dân"*** để thực hiện bài tập lớn trong môn học này. Bài làm sẽ hướng tới việc tìm ra những triệu chứng thường gặp của người mắc bệnh tim mạch, xây dựng mô hình chẩn đoán khả năng mắc bệnh và phân nhóm bệnh nhân mắc bệnh tim mạch dựa trên một số tiêu chí.

CHƯƠNG 1: PHÁT BIỂU BÀI TOÁN

1.1. Tính cấp thiết của đề tài

Theo CDC (Trung tâm kiểm soát và phòng ngừa dịch bệnh Hoa Kỳ), bệnh tim là nguyên nhân hàng đầu gây tử vong cho người dân thuộc hầu hết các chủng tộc ở Hoa Kỳ (người Mỹ gốc Phi, người Mỹ da đỏ và thổ dân Alaska, và người da trắng). Các chỉ số quan trọng gây bệnh tim: bao gồm hút thuốc, tình trạng bệnh tiểu đường, béo phì (chỉ số BMI cao), không hoạt động thể chất đầy đủ hoặc uống quá nhiều rượu. Việc xác định và ngăn ngừa các yếu tố có tác động lớn nhất đến bệnh tim là rất quan trọng trong chăm sóc sức khỏe. Đổi lại, sự phát triển trong điện toán cho phép ứng dụng các phương pháp học máy để phát hiện các "mẫu" trong dữ liệu có thể dự đoán tình trạng của bệnh nhân.

Việc dự đoán khả năng mắc bệnh tim có ý nghĩa cực kỳ quan trọng trong lĩnh vực chẩn đoán y học. Ngày nay, khi khai phá dữ liệu được ứng dụng trong hầu hết lĩnh vực của cuộc sống, y tế cũng không phải ngoại lệ. Áp dụng kiến thức khai phá dữ liệu dựa trên cơ sở lý thuyết rõ ràng và bộ dữ liệu thực, kết quả thu được có thể ứng dụng thực tiễn y học, đem lại giá trị chẩn đoán cho y học. Tuy nhiên cần xem xét ngữ cảnh để bài toán này có thể áp dụng tại thời điểm, không gian nào cho phù hợp, dựa vào thời điểm, không gian thu thập dữ liệu.

Nhận thấy được tầm quan trọng của việc ứng dụng khai phá dữ liệu vào việc chẩn đoán nguy cơ mắc bệnh tim của người dân, nhóm chúng em đã nghiên cứu khả năng mắc bệnh tim của người dân, nhằm hỗ trợ dự đoán khả năng mắc bệnh tim dựa trên những đặc điểm (dữ liệu nhãn) được thu thập bởi những người tham gia khảo sát - khảo sát được cung cấp bởi CDC - một tổ chức y tế uy tín tại Hoa Kỳ.

1.2. Một số kết quả nghiên cứu tại Việt Nam

Việc xây dựng mô hình chẩn đoán một số căn bệnh dựa trên kiến thức về khai phá dữ liệu đã được một số nhà nghiên cứu lựa chọn cho đề tài của mình:

Thứ nhất, luận văn của Thạc sĩ Hoàng Thị Thanh Hiền (2016) đã chỉ ra ý tưởng về việc vận dụng một số kỹ thuật khai phá dữ liệu để phát hiện bệnh trầm cảm của học sinh phổ thông. Đề tài đã chỉ ra việc ứng dụng mô hình cây quyết định và kỹ thuật phân cụm K-Means để phân loại những triệu chứng của bệnh trầm cảm.

Thứ hai, bài viết của các tác giả Trần Đinh Toàn & Dương Thị Mộng Thuỳ (2022) chỉ ra một số kỹ thuật học máy được ứng dụng vào việc phân loại bệnh tim. Đề tài sử dụng kỹ thuật phân lớp bằng thuật toán Naïve Bayes và ANN để nhận diện những đối tượng mắc và không mắc bệnh tim.

Nhận xét khoảng trống nghiên cứu

Việc ứng dụng khai phá dữ liệu trong việc chẩn đoán một số căn bệnh là không mới tại Việt Nam. Đây là vấn đề được nhiều nhà khoa học quan tâm và đưa ra ý tưởng thông qua các đề tài nghiên cứu. Tuy nhiên, chưa có nhiều đề tài nghiên cứu về việc chẩn đoán bệnh tim bằng các kỹ thuật khai phá dữ liệu. Các kỹ thuật như phân cụm, khai phá luật kết hợp cũng ít được đề cập tới.

Vì thế, đề tài ***"Ứng dụng khai phá dữ liệu xây dựng mô hình dự đoán khả năng mắc bệnh tim mạch của người dân"*** được nhóm chúng em lựa chọn để nghiên cứu trong bài tập lớn cuối kỳ nhằm ứng dụng những kiến thức đã học vào bài toán nhận diện bệnh tim, dựa trên bộ dữ liệu với số lượng mẫu và thuộc tính lớn. Đề tài hướng tới việc ứng dụng khai phá dữ liệu để tìm ra những triệu chứng phổ biến nhất của bệnh tim để người dân chủ động kiểm tra và phòng tránh từ sớm.

1.3. Đối tượng và phương pháp nghiên cứu

- **Đối tượng nghiên cứu:** Đề tài thực hiện nghiên cứu các kỹ thuật phân lớp, phân cụm và khai phá luật kết hợp dựa trên bộ dữ liệu khảo sát các triệu chứng của những người mắc và không mắc bệnh tim, tiến hành bởi CDC Hoa Kỳ.
- **Phương pháp nghiên cứu:** Đề tài được thực hiện dựa trên 02 phương pháp nghiên cứu:
  + Phương pháp nghiên cứu lý luận: Thu thập, đọc hiểu thông tin từ các tài liệu, giáo trình liên quan đến khai phá dữ liệu; Nghiên cứu các kỹ thuật phân lớp dữ liệu, phân cụm dữ liệu, luật kết hợp, ứng dụng các kỹ thuật đó để chẩn đoán bệnh tim dựa vào các thông tin đầu vào; Tìm hiểu các kỹ thuật khai phá luật kết hợp để phân tích những đặc trưng của bệnh tim.
  + Phương pháp nghiên cứu thực tiễn: Sử dụng kiến thức khai phá dữ liệu cộng với tri thức y học trong quá trình khai phá dữ liệu y khoa.

1.4. Ý nghĩa đề tài

Đề tài có ý nghĩa quan trọng về mặt khoa học và thực tiễn.

- **Ý nghĩa khoa học**: Đề tài góp phần mở rộng kiến thức về các yếu tố, nguy cơ gây bệnh tim. Bên cạnh đó, kết quả và kinh nghiệm thu được khi thực hiện đề tài này sẽ giúp các cán bộ y tế phát hiện sớm nguy cơ về bệnh tim cho bệnh nhân.
- **Ý nghĩa thực tiễn:** Đề tài có thể được ứng dụng để phát triển các hệ thống dự đoán người bị bệnh tim. Các hệ thống này sẽ dựa trên các yếu tố nguy cơ để dự đoán xem một người có nguy cơ mắc bệnh tim hay không. Khi phát hiện sớm bệnh tim, người bệnh có thể được điều trị và phòng ngừa kịp thời, giảm nguy cơ tử vong và biến chứng. Ngoài ra, đề tài có thể giúp phát hiện sớm bệnh tim ở những người có nguy cơ cao. Việc dự đoán được sớm nguy cơ mắc bệnh tim sẽ giúp cán bộ y tế đưa ra các biện pháp điều trị, phòng ngừa hiệu quả đồng thời theo dõi, cảnh báo và tư vấn giúp bệnh nhân tránh những biến chứng nguy hiểm, giảm gánh nặng kinh tế cho gia đình và xã hội.

1.5. Kết cấu đề tài

Chương 1: Phát biểu bài toán

Chương 1 nêu ra tính cấp thiết của đề tài trong việc xây dựng mô hình dự đoán khả năng mắc bệnh tim mạch của người dân; nêu ra đối tượng và phạm vi nghiên cứu, ý nghĩa và kết cấu của đề tài.

Chương 2: Cơ sở lý thuyết

Chương 2 chú trọng trình bày cơ sở lý thuyết về các mô hình, kỹ thuật được sử dụng trong bài làm, bao gồm các mô hình phân lớp (Cây quyết định, Rừng ngẫu nhiên), phân cụm (K-Means) và khai phá luật kết hợp (Apriori và ECLAT).

Chương 3: Xây dựng mô hình dự báo

Chương 3 tiến hành ứng dụng những kiến thức trong Chương 2 để xây dựng các mô hình dựa trên bộ dữ liệu có sẵn; tiến hành đánh giá về hiệu suất cũng như kết quả của từng mô hình

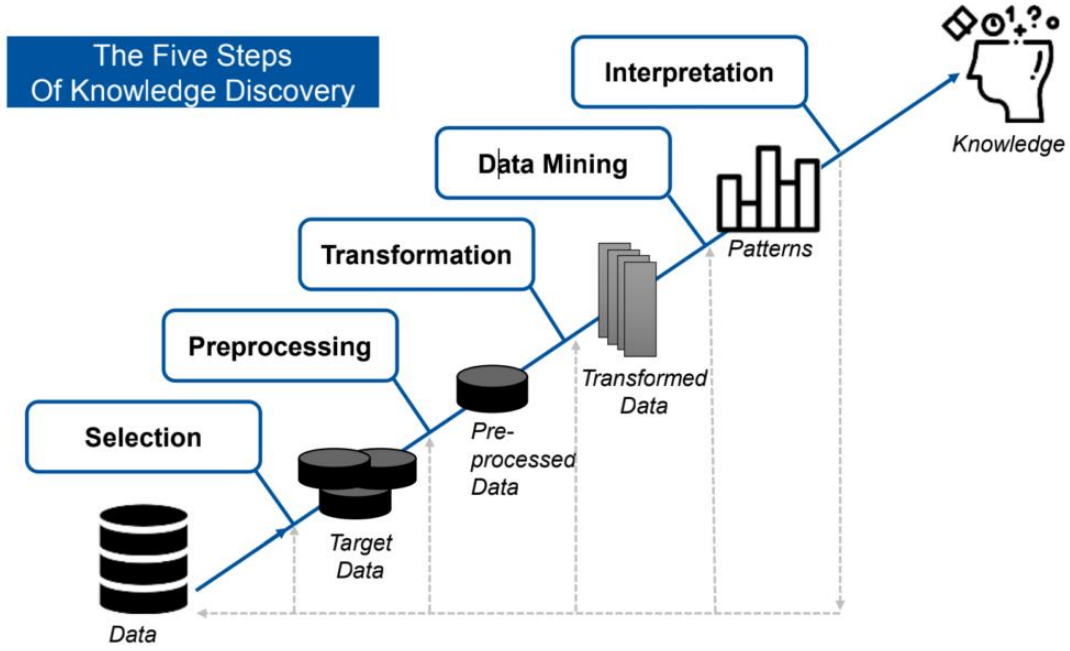
Chương 4: Một số đề xuất

Chương 4 nêu ra những ý tưởng cải tiến các mô hình được trình bày trong bài làm, nêu ra những hạn chế và định hướng phát triển của đề tài trong tương lai.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về kỹ thuật khai phá dữ liệu

2.1.1. Khái niệm khai phá dữ liệu



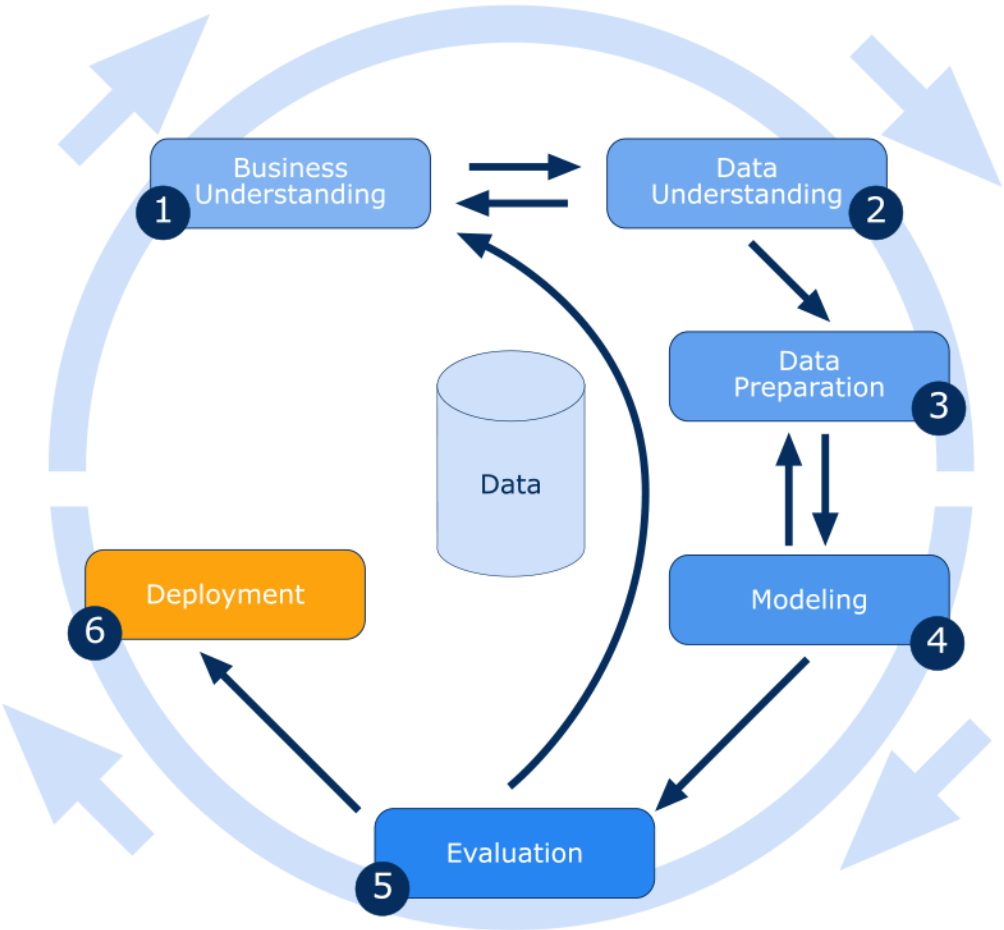
Hình 2-1: Quy trình khai phá tri thức (KDD)

Khai phá dữ liệu (Data mining) là một bước trong quy trình khám phá tri thức, nhằm:

- Rút trích thông tin hữu ích, chưa biết, tiềm ẩn trong khối dữ liệu lớn.
- Phân tích dữ liệu bán tự động.
- Giải thích dữ liệu trên các tập dữ liệu lớn.

Tóm lại, khai phá dữ liệu (Data mining) là quá trình tìm kiếm các mẫu, xu hướng và mối quan hệ có ý nghĩa trong các bộ dữ liệu lớn.

2.1.2. Các giai đoạn khai phá dữ liệu



Hình 2-2: Quy trình khai phá dữ liệu

Quy trình CRISP-DM là một quy trình lặp, có khả năng quay lui (backtracking) gồm 6 giai đoạn:

- *Tìm hiểu nghiệp vụ (Business understanding):* bao gồm xác định vấn đề kinh doanh, hiểu mục tiêu và mục đích của dự án và xác định KPI sẽ được sử dụng để đo lường thành công. Điều này rất quan trọng vì nó giúp đảm bảo rằng dự án khai thác dữ liệu phù hợp với mục đích và mục đích kinh doanh.
- *Tìm hiểu dữ liệu (Data understanding):* bao gồm việc hiểu nguồn dữ liệu, xác định mọi vấn đề về chất lượng dữ liệu và khám phá dữ liệu để xác định các mẫu và mối quan hệ.
- *Chuẩn bị dữ liệu (Data preparation):* bao gồm việc làm sạch dữ liệu để loại bỏ mọi lỗi hoặc sự không nhất quán, chuyển đổi dữ liệu để phù hợp cho việc phân tích và tích hợp dữ liệu từ các nguồn khác nhau để tạo một tập dữ liệu duy nhất.

- *Mô hình hoá (Modeling):* bước này liên quan đến việc xây dựng mô hình dự đoán bằng thuật toán học máy. bao gồm việc chọn một thuật toán thích hợp, huấn luyện mô hình trên dữ liệu và đánh giá hiệu suất của nó.
- *Đánh giá (Evaluation):* sử dụng các biện pháp thống kê để đánh giá mức độ mô hình có thể dự đoán kết quả trên dữ liệu mới
- *Triển khai (Deployment):* tích hợp mô hình vào các hệ thống và quy trình hiện có để đưa ra dự đoán theo thời gian thực.

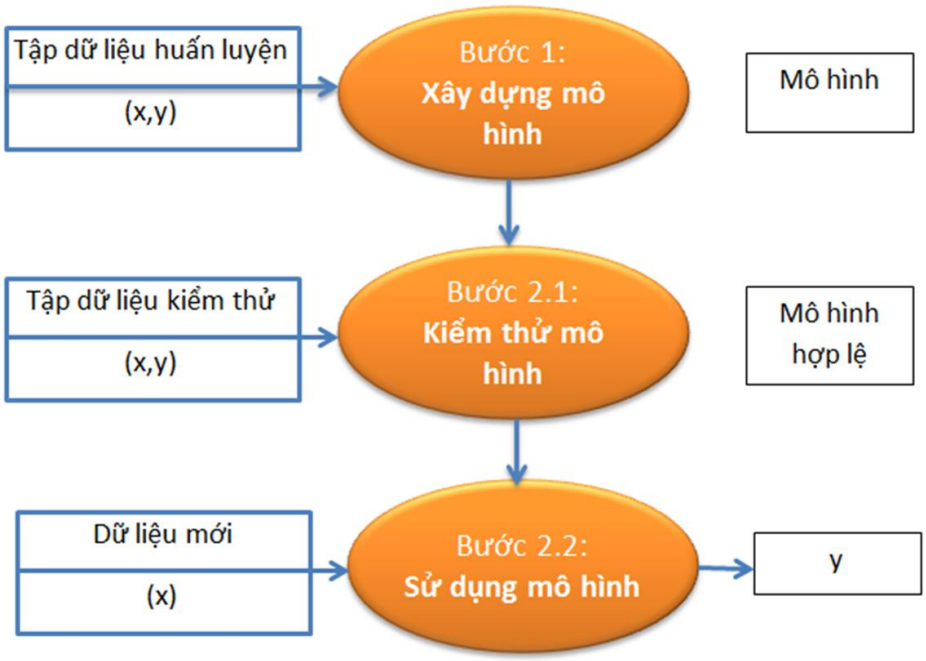
2.2. Bài toán phân lớp trong Khai phá dữ liệu

2.2.1. Khái niệm về phân lớp

Phân lớp dữ liệu là quá trình học có giám sát trên một tập dữ liệu đầu vào nhằm xây dựng một mô hình để có thể dự đoán xu hướng cho các dữ liệu mới.

- Đầu vào: Tập các dữ liệu có dạng (x,y) = (x1,x2,…,xn, y)
  - x là biến độc lập (Independent variable) mô tả các thuộc tính của một đối tượng.
  - y là biến phụ thuộc (Dependent variable) cần tìm hiểu, phân loại. y còn gọi là thuộc tính nhãn.
- Đầu ra: Một mô hình có khả năng phân loại đúng đắn cho tập các dữ liệu đầu vào.

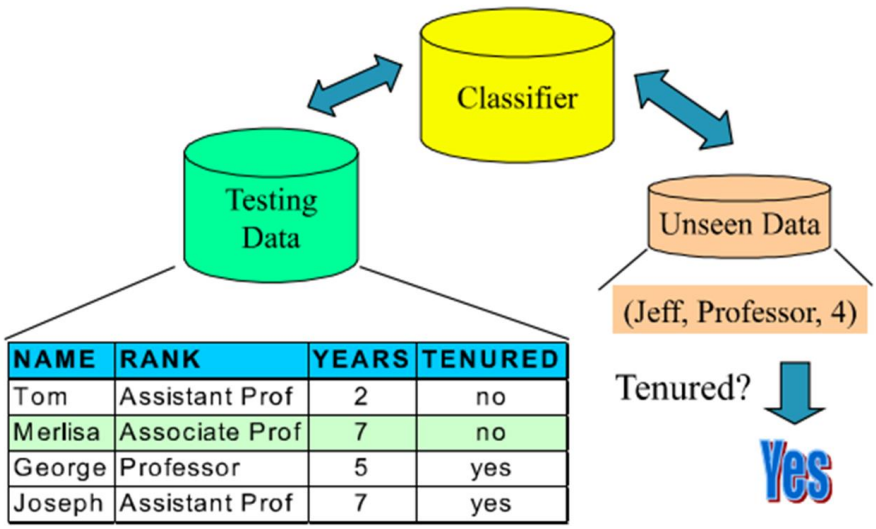
2.2.2. Quá trình phân lớp dữ liệu



Hình 2-3: Quá trình phân lớp dữ liệu

Bước học (bước huấn luyện): Xây dựng mô hình

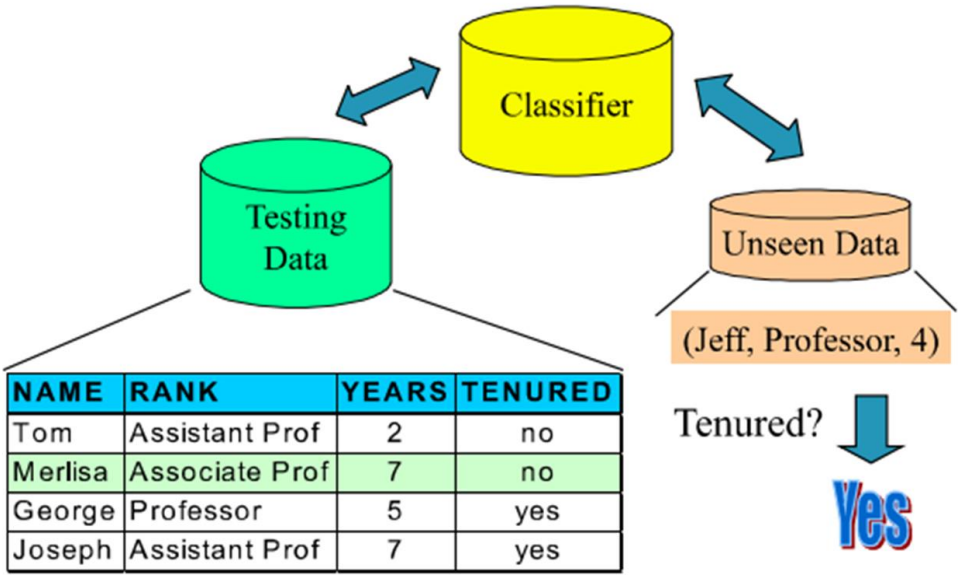
- Xác định tập dữ liệu huấn luyện gồm các mẫu đã được gán nhãn y.
- Chạy một thuật toán phân lớp trên tập dữ liệu huấn luyện.
- Mô hình được biểu diễn dưới dạng các luật phân lớp, các cây quyết định hoặc các công thức toán.



Hình 2-4: Bước học trong quá trình phân lớp dữ liệu

Bước phân loại: Sử dụng mô hình để gán nhãn thích hợp cho các dữ liệu chưa được gán nhãn.

- Ước lượng độ chính xác của mô hình:
- Xác định tập dữ liệu kiểm thử gồm các mẫu đã được gán nhãn y (dữ liệu kiểm thử và dữ liệu huấn luyện phải khác nhau để tránh tình trạng quá khớp over-fitting)
- Chạy mô hình với tập dữ liệu kiểm thử thu được nhãn y'. Sau đó xác định ma trận nhầm lẫn của mô hình.
- So sánh y và y' để xác định độ chính xác của mô hình.
- Nếu mô hình chính xác, sử dụng nó để dự đoán nhãn cho các dữ liệu cần gán nhãn.



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Hình 2-5: Bước phân loại trong quy trình phân lớp dữ liệu

2.2.3. Một số thuật toán phân lớp

Để phân loại các thuật toán phân lớp, có thể dựa trên các yếu tố khác nhau để phân loại thành các nhóm thuật toán.

Nhóm thuật toán dựa trên xác suất và lý thuyết thông tin:

- Phân lớp Naive Bayes.
- Phân lớp cực đại Entropy

Nhóm thuật toán dựa trên cây:

- Phân lớp dựa trên cây quyết định.
- Phân lớp rừng ngẫu nhiên (Random Forest)

Nhóm thuật toán dựa trên hồi quy:

- Hồi quy tuyến tính.
- Hồi quy logistic

Nhóm thuật toán dựa trên khoảng cách

- Phân lớp K láng giềng gần nhất.
- Phân lớp sử dụng máy vectơ hỗ trợ SVM
- Phân lớp Rocchio

Nhóm thuật toán khác: Mạng Neural, … (Khoa Hệ thống thông tin quản lý, Học viện Ngân hàng, 2023)

Trong bài nghiên cứu nhóm chúng em, hai thuật toán phân lớp được áp dụng là Phân lớp dựa trên cây quyết định và Phân lớp rừng ngẫu nhiên (Thuộc nhóm thuật toán phân lớp dựa trên cây).

Một số đặc điểm của Random Forest

- Lấy mẫu ngẫu nhiên: Random Forest bắt đầu bằng cách lấy mẫu ngẫu nhiên từ tập dữ liệu huấn luyện, thường theo phương pháp "bootstrap sampling" (lấy mẫu với hoán vị). Điều này đảm bảo rằng mỗi cây được huấn luyện trên một tập dữ liệu con khác nhau.
- Xây dựng các cây quyết dịnh: Mỗi cây quyết định trong Random Forest được xây dựng độc lập từ nhau. Đối với mỗi cây, một số lượng ngẫu nhiên các đặc trưng được chọn để xây dựng cây, giúp tăng tính đa dạng của các cây.
- Dự đoán bằng bầu chọn đa số: Khi cần dự đoán, mỗi cây trong Random Forest đưa ra một dự đoán riêng. Dự đoán cuối cùng của Random Forest được xác định bằng cách áp dụng bầu chọn đa số. Nếu hơn nửa số cây dự đoán một lớp, thì Random Forest chọn lớp đó là kết quả.
- Ứng dụng trung bình hoặc bầu chọn đa số: Trong trường hợp của regression tasks (dự đoán giá trị liên tục), kết quả cuối cùng thường là trung bình của kết quả của tất cả các cây. Trong trường hợp của classification tasks (dự đoán lớp), sử dụng bầu chọn đa số.

- Chống Overfitting: Sự ngẫu nhiên trong việc lấy mẫu và xây dựng cây giúp giảm nguy cơ overfitting*, đặc biệt là khi có nhiễu hoặc khi tập dữ liệu nhỏ.

** Overfitting: Chính xác trên tập dữ liệu train (huấn luyện) nhưng lại kém chính xác trên tập dữ liệu test (tập kiểm tra).*

2.3. Bài toán phân cụm trong Khai phá dữ liệu

Bài toán phân cụm dữ liệu là một nhánh ứng dụng chính của lĩnh vực học không giám sát, mà dữ liệu mô tả trong bài toán là không được dán nhãn. Thuật toán sẽ tìm cách phân cụm dữ liệu thành từng nhóm có đặc điểm tương tự nhau, nhưng đồng thời đặc tính giữa các nhóm đó lại phải càng khác biệt càng tốt. Số các cụm dữ liệu có thể được xác định trước theo kinh nghiệm hoặc có thể được tự động xác định theo thuật toán (Hoàng Huyền Trang, 2016). Đây là kỹ thuật được áp dụng trong rất nhiều lĩnh vực như kinh doanh, sinh học, bảo hiểm, ngân hàng, …

2.3.1. Một số phương pháp phân cụm và các thuật toán sử dụng

- Phân cụm phẳng (phân cụm phân vùng): Xây dựng từng bước phân hoạch các cụm và đánh giá chúng theo các tiêu chí tương ứng. Một số thuật toán phân cụm phẳng bao gồm K-mean, k-mediod, CLARANS, …
- Phân cụm phân cấp: Xây dựng hợp (tách) dần các cụm tạo cấu trúc phân cấp và đánh giá theo các tiêu chí tương ứng. Các kỹ thuật bao gồm CHAMELEON, BIRRCH và CURE, …
- Phân cụm dựa theo mật độ: Sử dụng hàm mật độ để tìm các phần tử chính tại nơi có mật độ cao và hàm liên kết để xác định cụm là lân cận phần tử chính. Một số thuật toán bao gồm DBSCAN, OPTICS…
- Phân cụm dựa theo lưới: Sử dụng lưới các ô cùng cỡ và tạo phân cấp ô lưới theo một số tiêu chí. Các kỹ thuật bao gồm: STING, CLIQUE, WaweCluster…
- Phân cụm dựa theo mô hình: Giả sử tồn tại một số mô hình dữ liệu cho phân cụm, cần xác định mô hình tốt nhất phù hợp với dữ liệu. Thuật toán nổi bật là MCLUST
- Phân cụm mờ: Giả sử không có phân cụm "cứng" cho dữ liệu và đối tượng có thể thuộc một số cụm. Cần sử dụng hàm mờ từ các đối tượng tới các cụm. Thuật toán nổi bật của phân cụm mờ là Fuzzy CMEANS (Hà Quang Thuỵ, 2018).

2.3.2. Thuật toán phân cụm K-means

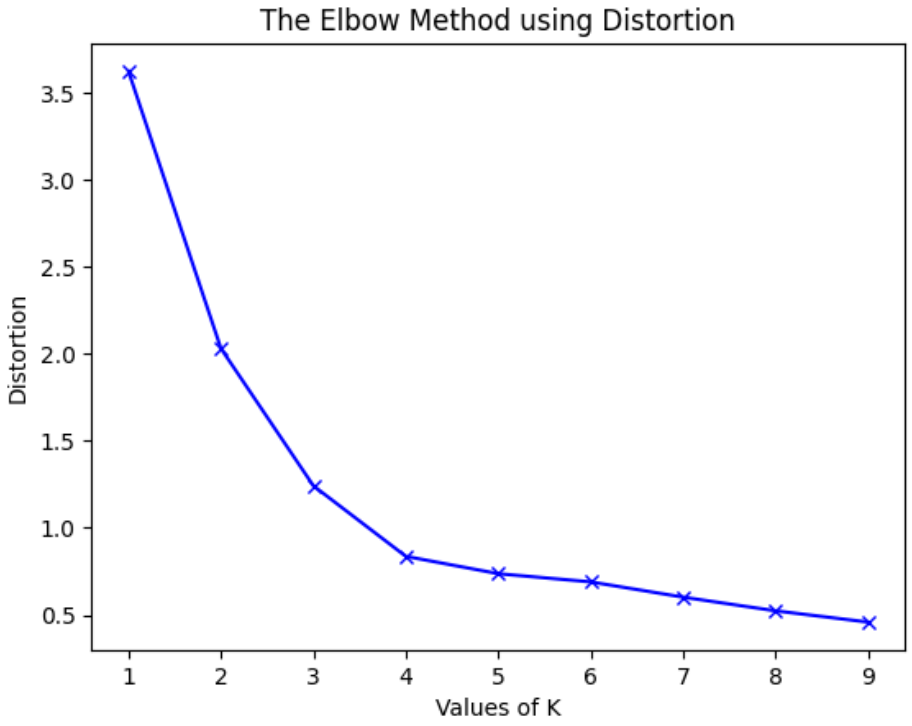
Thuật toán K-Means là một trong những thuật toán học không giám sát đơn giản nhất để giải quyết vấn đề phân cụm dữ liệu nổi tiếng, với số cụm được xác định trước là k cụm.

Thuộc nhóm phân cụm dữ liệu cứng/rõ, ý tưởng chính là để xác định k trọng tâm cho k cụm, một trọng tâm cho mỗi cụm. Những trọng tâm nên được đặt ở vị trí thích hợp nhất vì vị trí khác nhau gây ra kết quả khác nhau. Vì vậy, sự lựa chọn tốt hơn là đặt chúng càng nhiều càng tốt và cách xa nhau. Bước tiếp theo là với mỗi điểm thuộc tập dữ liệu cho trước và liên kết nó với trọng tâm gần nhất (Hoàng Huyền Trang, 2016).

- Đầu vào: Số cụm (k) và tập dữ liệu X chứa n đối tượng
- Đầu ra: một tập dữ liệu được phân thành k cụm

2.3.3. Phương pháp xác định số cụm k tối ưu trong thuật toán K-means

Để tìm được số cụm tối ưu cho thuật toán K-means, ta có thể sử dụng phương pháp "khuỷu tay" (elbow method). Elbow method sử dụng vòng lặp từ giá trị 1 đến n (n là tham số mà người dùng lựa chọn). Tại mỗi giá trị của k, phương pháp này sẽ tính tổng bình phương (WCSS) trong cụm.



Hình 2-6: Biểu đồ hình khuỷu tay

Nguồn: GeeksforGeeks (2019)

Để xác định số cụm tối ưu, người dùng sẽ chọn giá trị k ở "khuỷu tay", tức là điểm mà sau đó độ méo/quán tính bắt đầu giảm theo tuyến tính (GeeksforGeeks, 2019a).

2.4. Bài toán luật kết hợp trong Khai phá dữ liệu

2.4.1. Khái niệm về luật kết hợp

Luật kết hợp là mối quan hệ kết hợp giữa các tập thuộc tính trong cơ sở dữ liệu.

Một luật kết hợp là một mệnh đề kéo theo có dạng X → Y, trong đó X, Y ⊆ I, thỏa mãn điều kiện X ∩ Y = ϕ. Các tập hợp X và Y được gọi là các tập hợp thuộc tính (itemset). Tập X gọi là nguyên nhân, tập Y gọi là hệ quả. Có 2 độ đo quan trọng đối với luật kết hợp: Độ hỗ trợ (support) và độ tin cậy (confidence), được định nghĩa như sau:

- **Độ hỗ trợ**

- Độ hỗ trợ (support) của luật X → Y, ký hiệu là sup(X → Y, D) trong cơ sở dữ liệu D, là số giao dịch chứa cả X và Y.

$$sup(X \rightarrow Y, \mathbb{D}) = sup(X \cup Y, \mathbb{D})$$

- Độ hỗ trợ tương đối (relative support) của luật X → Y trong cơ sở dữ liệu D kí hiệu là rsup(X → Y, D) là số phần trăm các giao dịch trong D chứa cả X và Y.

$$rsup(X \rightarrow Y, \mathbb{D}) = \frac{sup(X \cup Y, \mathbb{D})}{|\mathbb{D}|}$$

- **Độ tin cậy**

- Độ tin cậy (confidence) của luật X → Y trong D, ký hiệu conf (X → Y , D), là tỉ lệ giữa số giao dịch chứa cả X và Y trên số giao dịch chỉ chứa X.

$$conf(X \rightarrow Y, \mathbb{D}) = \frac{sup(X \cup Y, \mathbb{D})}{sup(X, \mathbb{D})}$$

- Ký hiệu độ tin cậy của một luật r là conf(r). Ta có 0 <= conf(r) <= 1 (VIBLO, 2020).

2.4.2. Quá trình luật kết hợp dữ liệu

Quá trình luật kết hợp dữ liệu bao gồm các bước sau:

Bước 1: Mining frequent itemsets/ patterns: Khai phá tất cả các tập phổ biến từ cơ sở dữ liệu D với ngưỡng hỗ trợ tối thiểu *minsup.*

Bước 2: Generating strong rules from mined frequent itemsets/ patterns: Sinh tất cả các luật mạnh từ các tập phổ biến được khai phá ở bước trước với ngưỡng tin cậy tối thiểu *mincof.*

2.4.3. Một số thuật toán luật kết hợp

2.4.3.1. Thuật toán Apriori

a. Khái niệm

Thuật toán Apriori được công bố bởi R. Agrawal và R. Srikant vào năm 1994 vì để tìm các tập phổ biến trong một bộ dữ liệu lớn. Tên của thuật toán là Apriori vì nó sử dụng kiến thức đã có từ trước (prior) về các thuộc tính, item thường xuyên xuất hiện trong cơ sở dữ liệu.

Phương pháp Apriori dựa trên hai tính chất trên để cải thiện hiệu quả của phương pháp vét cạn bằng cách cắt tỉa các nhánh không cần thiết trên giàn tập mục.

- **Tính chất 1:** Nếu một tập mục X là tập con của một tập mục Y phổ biến, thì X cũng phải là tập phổ biến. Tính chất này dựa trên nhận xét rằng nếu một tập mục không phổ biến, thì nó sẽ xuất hiện trong một số giao dịch nhỏ hơn so với tập mục cha của nó.
- **Tính chất 2:** Nếu một tập mục X không phổ biến, thì tất cả các tập cha của X cũng không phổ biến. Tính chất này dựa trên nhận xét rằng nếu một tập mục phổ biến, thì nó sẽ xuất hiện trong một số giao dịch lớn hơn so với tất cả các tập con của nó.

b. Các thành phần

- **Min_support:** được sử dụng để xác định xem một luật kết hợp có phổ biến hay không. Nếu độ hỗ trợ của một luật kết hợp nhỏ hơn độ hỗ trợ tối thiểu, thì luật kết hợp đó sẽ bị loại bỏ.
- **Min_confidence:** được sử dụng để xác định xem một luật kết hợp có đáng tin cậy hay không. Nếu độ tin cậy của một luật kết hợp nhỏ hơn độ tin cậy tối thiểu, thì luật kết hợp đó sẽ bị loại bỏ.
- **Min_lift:** được sử dụng để xác định xem một luật kết hợp có mạnh mẽ hay không. Nếu độ nâng của một luật kết hợp nhỏ hơn độ nâng tối thiểu, thì luật kết hợp đó sẽ bị loại bỏ.

- **Min_length:** được sử dụng để xác định xem một luật kết hợp có ngắn hay không. Nếu độ dài của một luật kết hợp lớn hơn độ dài tối thiểu, thì luật kết hợp đó sẽ bị loại bỏ.

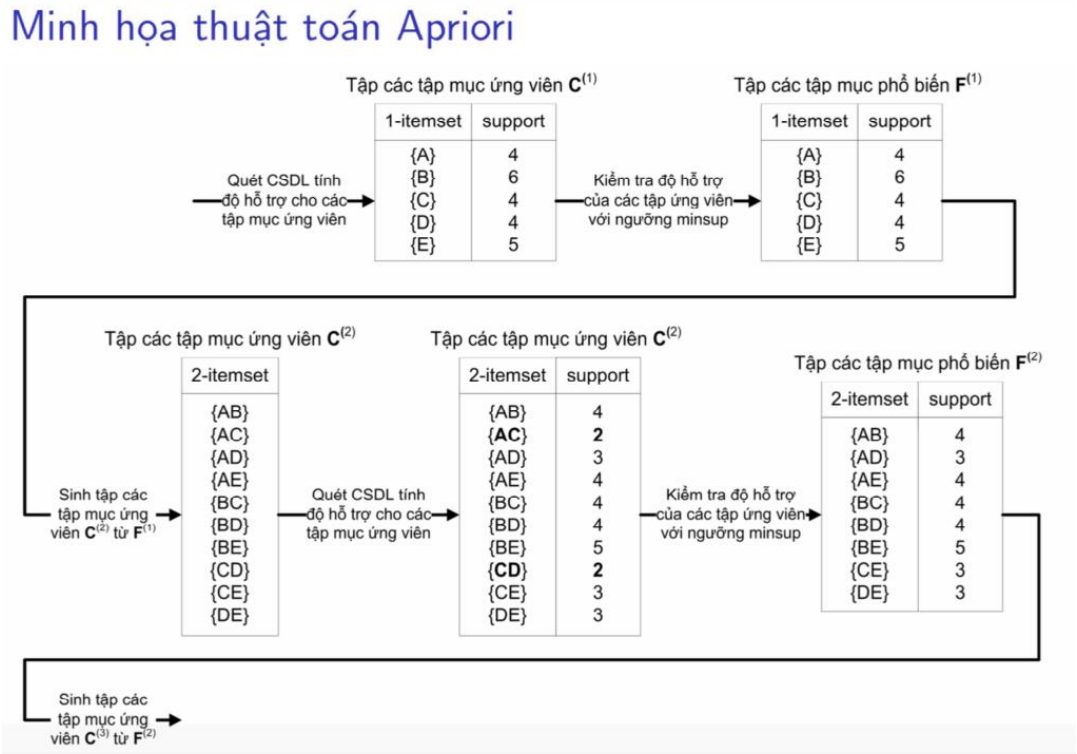
c. Cách thức hoạt động

Thuật toán Apriori hoạt động như sau:

- Khởi tạo giàn tập mục với tất cả các tập mục đơn.
- Lặp lại các bước sau cho đến khi không còn tập mục nào cần xem xét:

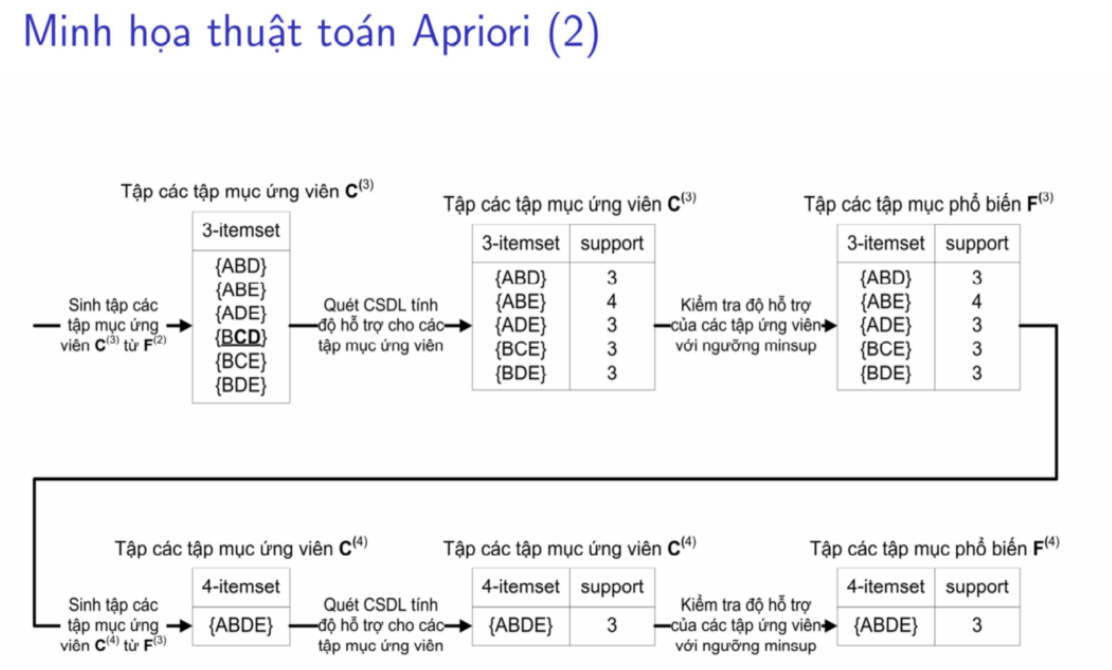
- Tìm tất cả các tập mục phổ biến mới.
- Cắt tỉa tất cả các tập mục không phổ biến.

Các tập mục phổ biến được tìm thấy bằng cách duyệt theo bề rộng trên giàn tập mục. Một tập mục được coi là phổ biến nếu nó xuất hiện trong ít nhất một số lượng giao dịch tối thiểu nhất định, được gọi là độ hỗ trợ tối thiểu.



Hình 2-7: Minh hoạ thuật toán Apriori (1)

Nguồn: (Khoa Hệ thống thông tin quản lý, Học viện Ngân hàng, 2023)



Hình 2-8: Minh hoạ thuật toán Apriori (2)

Nguồn: (Khoa Hệ thống thông tin quản lý, Học viện Ngân hàng, 2023)

d. Ưu - Nhược điểm

- **Ưu điểm:**

- Đơn giản, dễ hiểu và dễ cài đặt: Thuật toán Apriori sử dụng một khái niệm đơn giản là thuộc tính Apriori, giúp giảm đáng kể số lượng các tập ứng viên cần kiểm tra. Do đó, thuật toán Apriori có thể được triển khai một cách dễ dàng và hiệu quả.
- Tìm tập mục phổ biến thực hiện tốt bởi rút gọn kích thước các tập ứng viên nhờ kỹ thuật "tỉa": Thuật toán Apriori sử dụng kỹ thuật "tỉa" để loại bỏ các tập ứng viên không thể là tập mục phổ biến. Kỹ thuật này giúp giảm đáng kể số lượng các tập ứng viên cần kiểm tra, từ đó cải thiện hiệu quả của thuật toán.
- Có thể áp dụng cho nhiều loại tập dữ liệu khác nhau: Thuật toán Apriori có thể được áp dụng cho nhiều loại tập dữ liệu khác nhau, bao gồm tập dữ liệu giao dịch, tập dữ liệu liên tục và tập dữ liệu văn bản.

- **Nhược điểm:**

- Phải duyệt CSDL nhiều lần: Thuật toán Apriori cần duyệt CSDL nhiều lần để tìm ra các tập mục phổ biến. Điều này có thể gây ra sự chậm trễ đáng kể khi xử

lý các tập dữ liệu lớn.

- Số lượng lớn tập ứng viên được tạo ra làm gia tăng sự phức tạp không gian: Thuật toán Apriori tạo ra một số lượng lớn các tập ứng viên. Điều này có thể gây ra sự phức tạp không gian đáng kể, đặc biệt là khi xử lý các tập dữ liệu lớn.
- Để xác định độ support của các tập ứng viên, thuật toán luôn phải quyét lại toàn bộ CSDL: Thuật toán Apriori sử dụng phương pháp đếm trực tiếp để xác định độ support của các tập ứng viên. Phương pháp này có thể gây ra sự chậm trễ đáng kể khi xử lý các tập dữ liệu lớn.

2.4.3.2. Thuật toán ECLAT

a. Khái niệm

Một thuật toán phổ biến khác có tên là Eclat được đề xuất bởi Zaki vào năm 2001. Đây là một phiên bản hiệu quả và mở rộng hơn của thuật toán Apriori. Trong khi phương pháp Apriori sử dụng duyệt chiều rộng (BFS) thì phương pháp Eclat sử dụng duyệt chiều sâu (DFS) giúp nó nhanh hơn Apriori.

b. Cách hoạt động

Thuật toán Eclat hoạt động như sau:

- Khởi tạo: Khởi tạo một tập các tập mục phổ biến 1-itemset.
- Phát triển: Lặp qua các tập mục phổ biến ở mức hiện tại và tạo ra các tập mục phổ biến ở mức tiếp theo bằng cách kết hợp các tập mục tương đương.
- Duyệt: Duyệt qua các tập mục phổ biến để tìm ra các tập mục thỏa mãn ngưỡng hỗ trợ.

Hãy xem xét bản ghi giao dịch sau:

Transaction Id	Bread	Butter	Milk	Coke	Jam
T1	1	1	0	0	1
T2	0	1	0	1	0
T3	0	1	1	0	0
T4	1	1	0	1	0
T5	1	0	1	0	0
T6	0	1	1	0	0
T7	1	0	1	0	0
T8	1	1	1	0	1
T9	1	1	1	0	0

Hình 2-9: Dữ liệu minh hoạ thuật toán ECLAT

Nguồn: (GeeksforGeeks, 2019b)

Ta gọi hàm lần đầu tiên và sắp xếp từng mục với tập hợp nhỏ của nó theo kiểu bảng

- **Với k = 1, min_support = 2**

Item	Tidset
Bread	{T1, T4, T5, T7, T8, T9}
Butter	{T1, T2, T3, T4, T6, T8, T9}
Milk	{T3, T5, T6, T7, T8, T9}
Coke	{T2, T4}
Jam	{T1, T8}

Hình 2-10: Kết quả thuật toán ECLAT (1)

Nguồn: (GeeksforGeeks, 2019b)

- **Với k = 2**

Item	Tidset
{Bread, Butter}	{T1, T4, T8, T9}
{Bread, Milk}	{T5, T7, T8, T9}
{Bread, Coke}	{T4}
{Bread, Jam}	{T1, T8}
{Butter, Milk}	{T3, T6, T8, T9}
{Butter, Coke}	{T2, T4}
{Butter, Jam}	{T1, T8}
{Milk, Jam}	{T8}

Hình 2-11: Kết quả thuật toán ECLAT (2)

Nguồn: (GeeksforGeeks, 2019b)

- **Với k=3**

Item	Tidset
{Bread, Butter, Milk}	{T8, T9}
{Bread, Butter, Jam}	{T1, T8}

Hình 2-12: Kết quả thuật toán ECLAT (3)

Nguồn: (GeeksforGeeks, 2019b)

- **Với k=4**

Item	Tidset
{Bread, Butter, Milk, Jam}	{T8}

Hình 2-13: Kết quả thuật toán ECLAT (4)

Nguồn: (GeeksforGeeks, 2019b)

Chúng ta dừng lại ở k=4 vì không còn cặp item-tidset nào để kết hợp nữa.Vì min_support = 2, ta kết luận các quy tắc sau từ tập dữ liệu đã cho:

Items Bought	Recommended Products
Bread	Butter
Bread	Milk
Bread	Jam
Butter	Milk
Butter	Coke
Butter	Jam
Bread and Butter	Milk
Bread and Butter	Jam

Hình 2-14: Kết quả thuật toán ECLAT (5)

Nguồn: (GeeksforGeeks, 2019b)

c. Ưu - Nhược điểm

- **Ưu điểm**

- Nhanh chóng: Thuật toán Eclat chỉ cần quét tập dữ liệu một lần để tìm ra các tập mục phổ biến.
- Hiệu quả: Thuật toán Eclat có thể tìm ra các tập mục phổ biến có kích thước lớn.

- **Nhược điểm**

- Sử dụng nhiều bộ nhớ: Thuật toán Eclat cần lưu trữ tất cả các tập mục phổ biến ở mỗi mức.
- Khi có nhiều giao dịch, Eclat tốn nhiều bộ nhớ và thời gian tính toán cho phần giao của các tập.

2.4.3.3. Các thuật toán khác

Ngoài các thuật toán Apriori và ECLAT còn có nhiều thuật toán khai phá luật kết hợp khác, chẳng hạn như:

- FP-Growth: Thuật toán FP-Growth dựa trên khái niệm cây FP-Tree, trong đó các giao dịch được biểu diễn dưới dạng một cây nhị phân
- ARM: Thuật toán ARM là một thuật toán khai phá luật kết hợp dựa trên ý tưởng của các ma trận tương quan.
- CBA: Thuật toán CBA là một thuật toán khai phá luật kết hợp dựa trên ý tưởng của các biến ngẫu nhiên.
- TwinSpan: Thuật toán TwinSpan là một thuật toán khai phá luật kết hợp dựa trên phương pháp vét cạn. Tuy nhiên, thuật toán TwinSpan sử dụng một số kỹ thuật như cây nhị phân để cải thiện hiệu suất của phương pháp vét cạn

Lựa chọn thuật toán khai phá luật kết hợp phù hợp phụ thuộc vào nhiều yếu tố, chẳng hạn như:

- Kích thước của tập dữ liệu: Các thuật toán như Apriori và FP-Growth không hiệu quả đối với các tập dữ liệu lớn.
- Độ phức tạp của các luật kết hợp: Các thuật toán như ECLAT và TWINSPAN hiệu quả hơn đối với các luật kết hợp phức tạp.
- Kiến thức về dữ liệu: Các thuật toán như ARM và CBA yêu cầu kiến thức về dữ liệu để hoạt động hiệu quả.

CHƯƠNG 3: XÂY DỰNG CÁC MÔ HÌNH DỰ BÁO

3.1. Cơ sở dữ liệu xây dựng mô hình

3.1.1. Giới thiệu về dữ liệu

Về nguồn gốc và thời gian: Bộ dữ liệu ban đầu được lấy từ CDC và là một phần chính của Hệ thống giám sát yếu tố rủi ro hành vi (BRFSS), thực hiện khảo sát qua điện thoại hàng năm để thu thập dữ liệu về tình trạng sức khỏe của cư dân Hoa Kỳ. Như mô tả của CDC: "Được thành lập vào năm 1984 với 15 tiểu bang, BRFSS hiện thu thập dữ liệu ở tất cả 50 tiểu bang, Quận Columbia và ba bang của Hoa Kỳ. lãnh thổ. BRFSS hoàn thành hơn 400.000 cuộc phỏng vấn người lớn mỗi năm, trở thành hệ thống khảo sát sức khỏe được tiến hành liên tục lớn nhất trên thế giới. Trong bài nghiên cứu này, nhóm sử dụng bộ dữ liệu thu thập năm 2022 (hiện bộ dữ liệu 2022 đầy đủ về mặt thời gian - phù hợp nhất cho bài nghiên cứu).

Về định dạng bộ dữ liệu: .csv

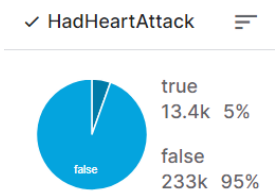
Về kích thước bộ dữ liệu: 46.1 MB

Về quy mô bộ dữ liệu: gồm 445.132 dòng và 18 cột.

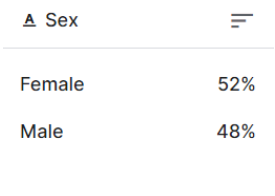
```
[ ]  print("Số dòng và số cột lần lượt là",Heart_2022_df.shape)

     Số dòng và số cột lần lượt là (445132, 18)
```

Hình 3-1: Quy mô bộ dữ liệu

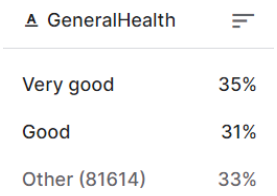
Về tổng quan các trường dữ liệu: bao gồm 18 trường dữ liệu tương ứng 18 thuộc tính, với tỷ lệ, số lượng như sau.



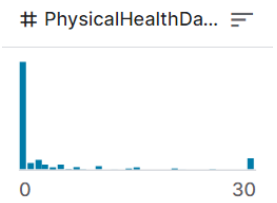
Thuộc tính mắc bệnh tim



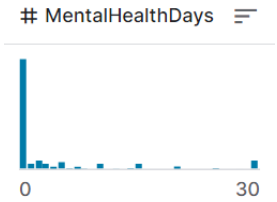
Thuộc tính giới tính



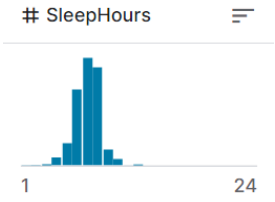
Thuộc tính tình trạng sức khỏe chung



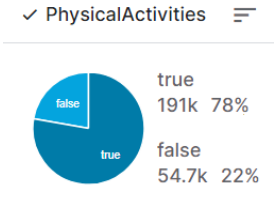
Số ngày tình trạng sức khỏe thể chất không ổn



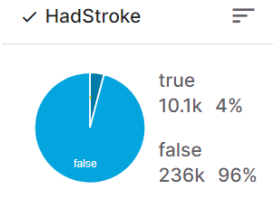
Số ngày tình trạng sức khỏe tinh thần không ổn



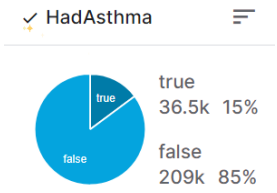
Số giờ ngủ/ngày



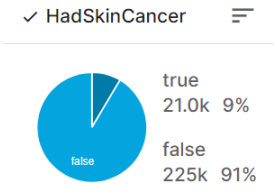
Thuộc tính tình trạng thường xuyên hoạt động thể chất hay không



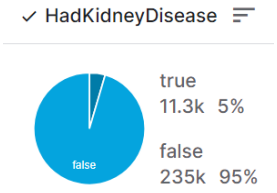
Thuộc tính mắc đột quỵ



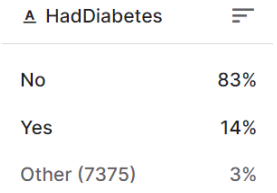
Thuộc tính mắc hen suyễn



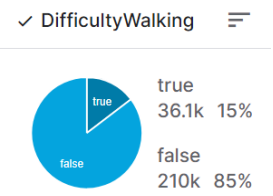
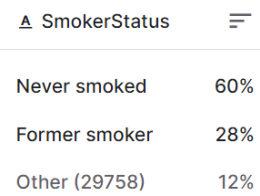
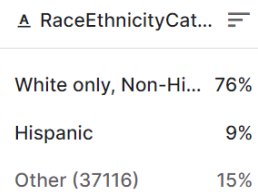
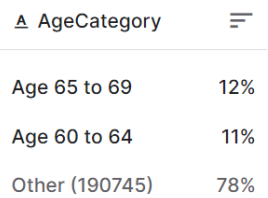
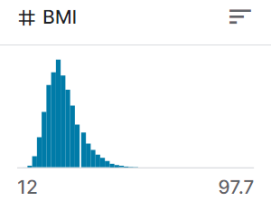
Thuộc tính mắc ung thư da



Thuộc tính mắc bệnh thận



Thuộc tính mắc bệnh tiểu đường

			
Thuộc tính gặp khó khăn trong việc đi lại	Thuộc tính hút thuốc hay không	Thuộc tính chủng tộc	Thuộc tính độ tuổi tham gia khảo sát
			
Thuộc tính chỉ số BMI (tỉ lệ giữa cân nặng và chiều cao bình phương)	Thuộc tính uống rượu hay không		

Bảng 3-1: Bảng mô tả 18 trường dữ liệu

3.1.2. Tiền xử lý dữ liệu

Bước 1: Xác định các thư viện và mục đích sử dụng của mỗi thư viện

```
import pandas as pd
```

- Đọc file dữ liệu

```
from sklearn.preprocessing import LabelEncoder
```

- Chuyển đổi kiểu dữ liệu từ dạng object sang kiểu số (Phục vụ thuật toán phân lớp)

```
from sklearn.tree import DecisionTreeClassifier
```

- Khai báo thư viện cây quyết định

```
from sklearn.model_selection import train_test_split
```

- Chia bộ dữ liệu thành 2 tệp dữ liệu học (train) và tệp dữ liệu kiểm thử (test)

```
from sklearn.metrics import confusion_matrix
```

- Khai báo thư viện xác định ma trận nhầm lẫn

```
from sklearn.metrics import accuracy_score
```

- Khai báo thư viện xác định độ chính xác của mô hình

```
from sklearn.ensemble import RandomForestClassifier
```

- Khai báo thư viện Random Forest

```
import numpy as np
```

- Phục vụ các phép tính

```
import matplotlib.pyplot as plt
```

- Trực quan hóa dữ liệu bằng biểu đồ bánh tròn

```
import seaborn as sns
```

```
from sklearn.cluster import KMeans
```

Khai báo thư viện phân cụm KMeans

```
from apyori import apriori
```

Khai báo thư viện luật kết hợp

```
from sklearn.metrics import classification_report #Hiệu suất mô hình
```

Khai báo thư viện xác định hiệu suất của mô hình

Bước 2: Data Clean

Kiểm tra số lượng bản ghi có dữ liệu trống của mỗi trường dữ liệu

```
missing_values = Heart_2022_df.isnull().sum()
print("Số lượng giá trị khuyết thiếu trong mỗi cột:\n", missing_values)
```

```
Số lượng giá trị khuyết thiếu trong mỗi cột:
 HadHeartAttack            3065
Sex                           0
GeneralHealth              1198
PhysicalHealthDays        10927
MentalHealthDays           9067
PhysicalActivities         1093
SleepHours                 5453
HadStroke                  1557
HadAsthma                  1773
HadSkinCancer              3143
HadKidneyDisease           1926
HadDiabetes                1087
DifficultyWalking         24012
SmokerStatus              35462
RaceEthnicityCategory     14057
AgeCategory                9079
BMI                       48806
AlcoholDrinkers           46574
dtype: int64
```

Hình 3-2: Kiểm tra số lượng bản ghi có dữ liệu bị trống trên mỗi trường dữ liệu

- Thay thế giá trị trống giá trị trung bình của trường dữ liệu

Thay thế giá trị khuyết thiếu bằng giá trị trung bình đối với cột BMI và cột SleepHours

```
[6] mean_bmi = Heart_2022_df['BMI'].mean()
    Heart_2022_df['BMI'].fillna(round(mean_bmi,2),inplace=True)
    print("Số lượng giá trị khuyết thiếu trong cột BMI là: ",Heart_2022_df['BMI'].isnull().sum())

    Số lượng giá trị khuyết thiếu trong cột BMI là:  0

[7] mean_SleepHours = Heart_2022_df['SleepHours'].mean()
    Heart_2022_df['SleepHours'].fillna(round(mean_SleepHours,0),inplace=True)
    print("Số lượng giá trị khuyết thiếu trong cột SleepHours là: ",Heart_2022_df['SleepHours'].isnull().sum())

    Số lượng giá trị khuyết thiếu trong cột SleepHours là:  0
```

Hình 3-3: Code thay thế giá trị NULL trong mỗi trường dữ liệu bằng giá trị trung bình

Thay thế các giá trị khuyết thiếu trong cột BMI và SleepHours bằng giá trị trung bình của mỗi cột. Đối với cột BMI, các giá trị được làm tròn đến chữ số thứ hai sau dấu phẩy. Còn đối với cột SleepHours, ta lấy giá trị nguyên.

- Thay thế bằng giá trị xuất hiện nhiều nhất của trường dữ liệu

```
[8] print("Giá trị xuất hiện nhiều nhất trong cột GeneralHealth:", Heart_2022_df['GeneralHealth'].mode())
    Heart_2022_df['GeneralHealth'].fillna('Very Good',inplace=True)
    print("Số lượng giá trị khuyết thiếu trong cột GeneralHealth: ",Heart_2022_df['GeneralHealth'].isnull().sum())

    Giá trị xuất hiện nhiều nhất trong cột GeneralHealth: 0    Very good
    Name: GeneralHealth, dtype: object
    Số lượng giá trị khuyết thiếu trong cột GeneralHealth:  0
```

Hình 3-4: Thay thế bằng giá trị xuất hiện nhiều nhất của trường dữ liệu

- Thay thế giá trị nhiều nhất trong cột GeneralHealth bằng giá trị xuất hiện nhiều

nhất là “Very Good”, tương tự với các cột còn lại. Ta có kết quả như sau:

```
missing_values = Heart_2022_df.isnull().sum()
print("Số lượng giá trị khuyết thiếu trong mỗi cột:\n", missing_values)
```

```
Số lượng giá trị khuyết thiếu trong mỗi cột:
 HadHeartAttack            0
Sex                        0
GeneralHealth              0
PhysicalHealthDays         0
MentalHealthDays           0
PhysicalActivities         0
SleepHours                 0
HadStroke                  0
HadAsthma                  0
HadSkinCancer              0
HadKidneyDisease           0
HadDiabetes                0
DifficultyWalking          0
SmokerStatus               0
RaceEthnicityCategory      0
AgeCategory                0
BMI                        0
AlcoholDrinkers            0
dtype: int64
```

Hình 3-5: Số lượng giá trị trống trong mỗi cột

- Xuất file csv sau khi đã tiền xử lý dữ liệu:

```
Heart_2022_df.to_csv('Heart_2022_processed.csv', index=False)
```

Hình 3-6: Xuất file sau khi tiền xử lý dữ liệu

3.2. Xây dựng mô hình

3.2.1. Xây dựng mô hình theo thuật toán phân lớp

Hai thuật toán phân lớp nhóm sử dụng thuộc nhóm thuật toán phân lớp dựa trên cây: thuật toán cây quyết định và thuật toán Random Forest. Trong khi cây quyết định là một mô hình đơn giản và dễ hiểu, Random Forest là một mô hình mạnh mẽ hơn, có khả năng giảm overfitting và cung cấp tính ổn định và độ chính xác cao hơn đặc biệt khi xử lý các tập dữ liệu lớn và phức tạp.

Trong hai thuật toán phân lớp dưới đây, nhóm quy ước biến đổi kiểu dữ liệu kiểu object sang kiểu số theo quy ước như sau:

- Đối với các trường dữ liệu nhận giá trị No/Yes: No - 0, Yes – 1.
- Đối với trường dữ liệu Sex: Female - 0, Male -1 (Sex).

- Đối với trường dữ liệu GeneralHealth: Excellent - 0, Fair -1, Good - 2, Poor - 3, Very good – 4.
- Đối với trường dữ liệu AgeCategory: Age 35 to 39 - 0, Age 40 to 44 - 1, Age 45 to 49 - 2, Age 50 to 54 - 3, Age 55 to 59 - 4, Age 60 to 64 - 5, Age 65 to 69 - 6,Age 70 to 74 - 7, Age 75 to 79 - 8, Age 80 or older - 9 .
- Đối với trường dữ liệu RaceEthnicityCategory: Black only, Non-Hispanic - 0, Hispanic - 1, Multiracial, Non-Hispanic - 2, Other race only, Non-Hispanic - 3, White only, Non-Hispanic – 4.
- Đối với trường dữ liệu SmokerStatus: Current smoker - now smokes every day - 0, Current smoker - now smokes some days - 1, Former smoker- 2, Never smoked - 3 (Never smoked).
- Đối với trường dữ liệu HadDiabetes: No - 0, No, pre-diabetes or borderline diabetes- 1, Yes - 2, Yes, but only during pregnancy (female) – 3.

Sau khi đã quy ước, nhóm tiến hành chuyển đổi kiểu dữ liệu như sau:

```
[71] lb_make = LabelEncoder()
    Heart_2022_df['HadHeartattack'] = lb_make.fit_transform(Heart_2022_df['HadHeartAttack'])
    Heart_2022_df['Sex'] = lb_make.fit_transform(Heart_2022_df['Sex'])
    Heart_2022_df['GeneralHealth'] = lb_make.fit_transform(Heart_2022_df['GeneralHealth'])
    Heart_2022_df['PhysicalActivities'] = lb_make.fit_transform(Heart_2022_df['PhysicalActivities'])
    Heart_2022_df['PhysicalHealthDays'] = lb_make.fit_transform(Heart_2022_df['PhysicalHealthDays'])
    Heart_2022_df['MentalHealthDays'] = lb_make.fit_transform(Heart_2022_df['MentalHealthDays'])
    Heart_2022_df['SleepHours'] = lb_make.fit_transform(Heart_2022_df['SleepHours'])
    Heart_2022_df['HadStroke'] = lb_make.fit_transform(Heart_2022_df['HadStroke'])
    Heart_2022_df['HadAsthma'] = lb_make.fit_transform(Heart_2022_df['HadAsthma'])
    Heart_2022_df['HadSkinCancer'] = lb_make.fit_transform(Heart_2022_df['HadSkinCancer'])
    Heart_2022_df['HadKidneyDisease'] = lb_make.fit_transform(Heart_2022_df['HadKidneyDisease'])
    Heart_2022_df['HadDiabetes'] = lb_make.fit_transform(Heart_2022_df['HadDiabetes'])
    Heart_2022_df['DifficultyWalking'] = lb_make.fit_transform(Heart_2022_df['DifficultyWalking'])
    Heart_2022_df['SmokerStatus'] = lb_make.fit_transform(Heart_2022_df['SmokerStatus'])
    Heart_2022_df['RaceEthnicityCategory'] = lb_make.fit_transform(Heart_2022_df['RaceEthnicityCategory'])
    Heart_2022_df['AgeCategory'] = lb_make.fit_transform(Heart_2022_df['AgeCategory'])
    Heart_2022_df['AlcoholDrinkers'] = lb_make.fit_transform(Heart_2022_df['AlcoholDrinkers'])
    Heart_2022_df.head(5)
```

	HadHeartattack	Sex	GeneralHealth	PhysicalHealthDays	MentalHealthDays	PhysicalActivities	SleepHours	HadStroke	HadAsthma	HadSkinCancer	HadKidneyDisease	HadDiabetes	DifficultyWalking	SmokerStatus	RaceEthnicityCategory	AgeCategory	BMI	AlcoholDrinkers
0	0	0	5	0	0	0	7	0	0	0	0	2	0	3	4	12	28.53	0
1	0	0	0	0	0	0	5	0	0	1	0	0	0	3	4	12	26.57	0
2	0	0	5	2	3	1	4	0	0	1	0	0	0	3	4	7	25.61	0
3	0	0	0	0	0	1	6	0	1	0	0	0	0	1	4	9	23.30	0
4	0	0	1	2	0	1	8	0	0	0	0	0	0	3	4	4	21.77	1

Hình 3-7: Code chuyển đổi kiểu dữ liệu

Kết quả thu được:

```
print('Kiểu dữ liệu của từng cột trong tệp dữ liệu là: \n',Heart_2022_df.dtypes)

Kiểu dữ liệu của từng cột trong tệp dữ liệu là:
 HadHeartAttack            object
Sex                        object
GeneralHealth              object
PhysicalHealthDays        float64
MentalHealthDays          float64
PhysicalActivities         object
SleepHours                float64
HadStroke                  object
HadAsthma                  object
HadSkinCancer              object
HadKidneyDisease           object
HadDiabetes                object
DifficultyWalking          object
SmokerStatus               object
RaceEthnicityCategory      object
AgeCategory                object
BMI                       float64
AlcoholDrinkers            object
dtype: object
```

→

```
print('Kiểu dữ liệu của từng cột trong tệp dữ liệu là: \n',Heart_2022_df.dtypes)

Kiểu dữ liệu của từng cột trong tệp dữ liệu là:
 HadHeartAttack             int64
Sex                         int64
GeneralHealth               int64
PhysicalHealthDays          int64
MentalHealthDays            int64
PhysicalActivities          int64
SleepHours                  int64
HadStroke                   int64
HadAsthma                   int64
HadSkinCancer               int64
HadKidneyDisease            int64
HadDiabetes                 int64
DifficultyWalking           int64
SmokerStatus                int64
RaceEthnicityCategory       int64
AgeCategory                 int64
BMI                       float64
AlcoholDrinkers             int64
dtype: object
```

Bảng 3-2: Bảng so sánh kiểu dữ liệu của các trường dữ liệu trước và sau khi

chuyển đổi

Ta xác định được dữ liệu *features* và *target* như sau:

- *features:* tập thuộc tính đặc trưng
- *target:* thuộc tính mục tiêu. Trong bài nghiên cứu này, nhóm xác định thuộc tính mục tiêu là *HadHeartAttack*, dự báo khả năng bị bệnh tim của một người.

```
# Chọn features và target
features = ['Sex', 'GeneralHealth', 'PhysicalActivities', 'PhysicalHealthDays', 'MentalHealthDays', 'SleepHours',
            'HadStroke', 'HadAsthma', 'HadSkinCancer', 'HadKidneyDisease', 'HadDiabetes', 'DifficultyWalking',
            'SmokerStatus', 'RaceEthnicityCategory', 'AgeCategory', 'BMI', 'AlcoholDrinkers']
target = ['HadHeartAttack']
# Tạo X và y
X = Heart_2022_df[features]
y = Heart_2022_df[target]
```

Hình 3-8: Xác định các thuộc tính features và target

Sau khi đã có 2 tệp thuộc tính, ta tiếp tục chia nhỏ chúng thành các tệp dữ liệu huấn luyện, và tệp dữ liệu kiểm thử, với tỷ lệ tương ứng 80:20 như sau:

```
[75] X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2)
     print(X_train)
     print(X_test)
     print(y_train)
     print(y_test)
```

Hình 3-9: Phân chia bộ dữ liệu

Sau khi đã có 4 tệp dữ liệu mong muốn, ta sử dụng chúng vào mô hình cây quyết định và mô hình Random Forest như 3.2.1.1 và 3.2.1.2.

Áp dụng mô hình dự đoán khả năng bị bệnh tim của 5 người khác nhau vào 2 mô hình với:

Người thứ nhất

- Giới tính nữ (Female - 0)
- Tình trạng sức khỏe (Excellent - 0, Fair -1, Good - 2, Poor - 3, Very good - 4// Excellent - 0)
- Thể dục thể thao (PhysicalActivities: Yes - 1)
- Số ngày tình trạng sức khỏe thể chất không ổn trong 30 ngày gần đây là 3 ngày (PhysicalHealthDays: 3)
- Số ngày tình trạng sức khỏe tinh thần không ổn (MentalHealthDays: 5)

- Thời gian ngủ (SleepHours - 8)
- Tiển sử đột quỵ (HadStroke: Yes - 1)
- Tiển sử bị hen xuyễn (HadAsthma: No - 0)
- Bị ung thư da (HadSkinCancer: No - 0)
- Tiền sử bệnh thận (HadKidneyDisease: No - 0)
- Tiền sử tiểu đường (HadDiabetes: Yes, but only during pregnancy (female) - 3
- Đi lại khó khăn (DifficultyWalking: Yes - 1)
- Tình trạng hút thuốc (SmokerStatus: 3)
- Dân tộc (RaceEthnicityCategory: 4)
- Tuổi (AgeCategory: 3)
- BMI (27)
- Uống rượu (Yes: 1)

Người thứ hai:

- Giới tính nam (Male - 1)
- Tình trạng sức khỏe (Excellent - 0, Fair -1, Good - 2, Poor - 3, Very good - 4// Poor - 3)
- Thể dục thể thao (PhysicalActivities: No - 0)
- Số ngày tình trạng sức khỏe thể chất không ổn trong 30 ngày gần đây là 3 ngày (PhysicalHealthDays: 25)
- Số ngày tình trạng sức khỏe tinh thần không ổn (MentalHealthDays: 25)
- Thời gian ngủ (SleepHours - 2)
- Tiển sử đột quỵ (HadStroke: Yes - 1)
- Tiển sử bị hen xuyễn (HadAsthma: Yes - 1)
- Bị ung thư da (HadSkinCancer: Yes - 1)
- Tiền sử bệnh thận (HadKidneyDisease: yes - 1)
- Tiền sử tiểu đường (HadDiabetes: Yes, but only during pregnancy - 1
- Đi lại khó khăn (DifficultyWalking: Yes - 1)
- Tình trạng hút thuốc (SmokerStatus: 2)
- Dân tộc (RaceEthnicityCategory: 1)

- Tuổi (AgeCategory: 4)
- BMI (26,3)
- Uống rượu (Yes: 1)

Người thứ ba:

- Giới tính nữ (Female - 0),
- Tình trạng sức khỏe (Excellent - 0, Fair -1, Good - 2, Poor - 3, Very good - 4//Poor - 3),
- Thể dục thể thao (PhysicalActivities: Yes - 1)
- Số ngày tình trạng sức khỏe thể chất không ổn trong 30 ngày gần đây là 3 ngày (PhysicalHealthDays: 0).
- Số ngày tình trạng sức khỏe tinh thần không ổn (MentalHealthDays: 0)
- Thời gian ngủ (SleepHours - 8)
- Tiển sử đột quỵ (HadStroke: No - 0)
- Tiển sử bị hen xuyễn (HadAsthma: Yes - 0)
- Bị ung thư da (HadSkinCancer: No - 0)
- Tiền sử bệnh thận (HadKidneyDisease: Yes - 1)
- Tiền sử tiểu đường (HadDiabetes: Yes, but only during pregnancy (female) - 3
- Đi lại khó khăn (DifficultyWalking: No - 1)
- Tình trạng hút thuốc (SmokerStatus: 1)
- Dân tộc (RaceEthnicityCategory: 1)
- Tuổi (AgeCategory: 1)
- BMI (28)
- Uống rượu (No-0)

Người thứ tư:

- Giới tính nữ (Female - 0),
- Tình trạng sức khỏe (Excellent - 0, Fair -1, Good - 2, Poor - 3, Very good - 4// Excellent - 0),
- Thể dục thể thao (PhysicalActivities: Yes - 1)

- Số ngày tình trạng sức khỏe thể chất không ổn trong 30 ngày gần đây là 3 ngày (PhysicalHealthDays: 3).
- Số ngày tình trạng sức khỏe tinh thần không ổn (MentalHealthDays: 5)
- Thời gian ngủ (SleepHours - 12)
- Tiển sử đột quỵ (HadStroke: No - 0)
- Tiển sử bị hen xuyễn (HadAsthma: Yes - 1)
- Bị ung thư da (HadSkinCancer: Yes - 1)
- Tiền sử bệnh thận (HadKidneyDisease: Yes - 1)
- Tiền sử tiểu đường (HadDiabetes: Yes, but only during pregnancy (female) - 3
- Đi lại khó khăn (DifficultyWalking: Yes - 1)
- Tình trạng hút thuốc (SmokerStatus: 1)
- Dân tộc (RaceEthnicityCategory: 1)
- Tuổi (AgeCategory: 1)
- BMI (22)
- Uống rượu (Yes: 1)

Người thứ năm:

- Giới tính nam (male - 1)
- Tình trạng sức khỏe (Excellent - 0, Fair -1, Good - 2, Poor - 3, Very good - 4//Poor - 3)
- Thể dục thể thao (PhysicalActivities: No-0)
- Số ngày tình trạng sức khỏe thể chất không ổn trong 30 ngày gần đây là 3 ngày (PhysicalHealthDays: 30).
- Số ngày tình trạng sức khỏe tinh thần không ổn (MentalHealthDays: 30)
- Thời gian ngủ (SleepHours - 4)
- Tiển sử đột quỵ (HadStroke: Yes - 1)
- Tiển sử bị hen xuyễn (HadAsthma: Yes - 1)
- Bị ung thư da (HadSkinCancer: Yes - 1)
- Tiền sử bệnh thận (HadKidneyDisease: Yes - 1)
- Tiền sử tiểu đường (HadDiabetes: Yes, but only during pregnancy (female) - 3

- Đi lại khó khăn (DifficultyWalking: Yes - 1)
- Tình trạng hút thuốc (SmokerStatus: 1)
- Dân tộc (RaceEthnicityCategory: 1)
- Tuổi (AgeCategory: 1)
- BMI (25.7)
- Uống rượu (Yes-1)

```
# Tạo DataFrame với thông tin của 5 người
data = {
    'Sex': [0, 1, 0, 0, 1],
    'GeneralHealth': [0, 0, 0, 0, 3],
    'PhysicalActivities': [1, 0, 1, 1, 0],
    'PhysicalHealthDays': [3, 25, 0, 3, 30],
    'MentalHealthDays': [5, 25, 0, 5, 30],
    'SleepHours': [8, 2, 8, 12, 4],
    'HadStroke': [1, 1, 0, 0, 1],
    'HadAsthma': [0, 1, 0, 1, 1],
    'HadSkinCancer': [0, 0, 0, 1, 1],
    'HadKidneyDisease': [0, 1, 1, 1, 1],
    'HadDiabetes': [3, 1, 3, 3, 3],
    'DifficultyWalking': [1, 1, 1, 1, 1],
    'SmokerStatus': [3, 2, 1, 1, 1],
    'RaceEthnicityCategory': [4, 1, 1, 1, 1],
    'AgeCategory': [3, 4, 1, 1, 1],
    'BMI': [27, 26.3, 28, 22, 25.7],
    'AlcoholDrinkers': [1, 1, 0, 1, 1]
}

# Chuyển đổi dữ liệu thành DataFrame
new_data = pd.DataFrame(data)
```

Hình 3-10: Chuẩn bị dữ liệu kiểm thử

3.2.1.1. Xây dựng mô hình theo thuật toán cây quyết định

Khai báo mô hình cây quyết định

```
[76] model = DecisionTreeClassifier(criterion="entropy", random_state=100, max_depth=3, min_samples_leaf=5)
```

Hình 3-11: Code khai báo mô hình cây quyết định

Huấn luyện mô hình trên tập huấn luyện

```
[77] model = model.fit(X_train,y_train)
```

Hình 3-12: Code huấn luyện mô hình cây quyết định

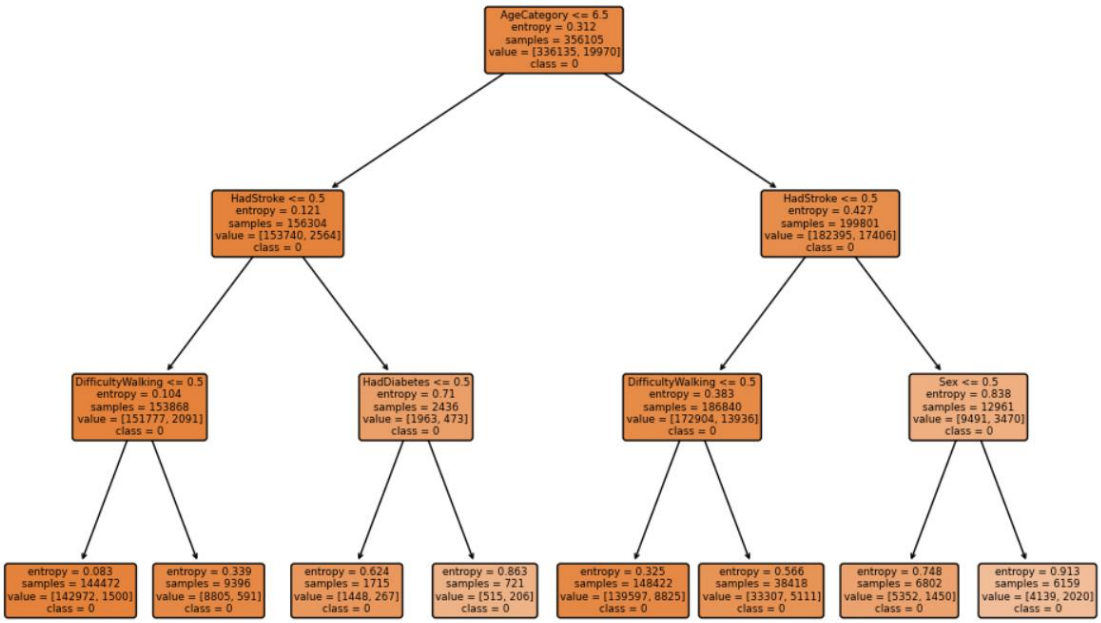
Xây dựng cây quyết định:

```
import matplotlib.pyplot as plt #Thư viện dùng vẽ biểu đồ
from sklearn.tree import DecisionTreeClassifier, export_text #xuất biểu  đồ
# Xuất biểu đồ cây quyết định dưới dạng text
tree_rules = export_text(model, feature_names=features)
print(tree_rules)

# Hiển thị biểu đồ cây quyết định bằng matplotlib
fig, ax = plt.subplots(figsize=(12, 8)) # để tạo ra một subplot để vẽ biểu đồ cây quyết định
tree.plot_tree(model, feature_names=features, filled=True, rounded=True, class_names=["0", "1"], ax=ax)
plt.show()
```

Hình 3-13: Code xây dựng biểu đồ cây quyết định

```
|--- AgeCategory <= 6.50
|    |--- HadStroke <= 0.50
|    |    |--- DifficultyWalking <= 0.50
|    |    |    |--- class: 0
|    |    |--- DifficultyWalking >  0.50
|    |    |    |--- class: 0
|    |--- HadStroke >  0.50
|    |    |--- HadDiabetes <= 0.50
|    |    |    |--- class: 0
|    |    |--- HadDiabetes >  0.50
|    |    |    |--- class: 0
|--- AgeCategory >  6.50
|    |--- HadStroke <= 0.50
|    |    |--- DifficultyWalking <= 0.50
|    |    |    |--- class: 0
|    |    |--- DifficultyWalking >  0.50
|    |    |    |--- class: 0
|    |--- HadStroke >  0.50
|    |    |--- Sex <= 0.50
|    |    |    |--- class: 0
|    |    |--- Sex >  0.50
|    |    |    |--- class: 0
```

Hình 3-14: Minh hoạ cây quyết định



Hình 3-15: Biểu đồ cây quyết định

Ta có:

- Entropy là một đo lường mức độ không chắc chắn. Giá trị entropy càng thấp, mức độ không chắc chắn càng giảm
- Class: Là lớp dự đoán cho nút đó (0 or 1)
- Samples: Là số lượng mẫu huấn luyện tại nút đó.
- Value: Là số lượng mẫu của mỗi lớp tại nút đó. Ví dụ, nếu nút dự đoán 10 mẫu thuộc lớp 0 và 5 mẫu thuộc lớp 1, giá trị của Value sẽ là [10, 5].

Chạy mô hình dự đoán kết quả dự đoán dựa trên bộ dữ liệu features test và in ra kết quả như sau:

```
[78] y_pred = model.predict(X_test)
     print("Giá trị target mô hình dự đoán được:\n",y_pred)

     Giá trị target mô hình dự đoán được:
      [0 0 0 ... 0 0 0]
```

Hình 3-16: Code dự đoán các giá trị trên tập test

Xác định ma trận nhầm lẫn.

```
[79] print("Confusion Matrix:", confusion_matrix(y_test,y_pred))

     Confusion Matrix: [[83916     0]
      [ 5111     0]]
```

Hình 3-17: Code xây dựng ma trận nhầm lẫn

Giải thích kết quả như sau. Ta có ma trận nhầm lẫn:

Thực tế	Dự đoán	
	Không mắc bệnh tim	Mắc bệnh tim
Không mắc bệnh tim	83.916	0
Mắc bệnh tim	5.111	0

Bảng 3-3: Ma trận nhầm lẫn - Mô hình cây quyết định

Xác định độ chính xác của mô hình:

```
[80] print("Accuracy: ",accuracy_score(y_test,y_pred)*100)

     Accuracy:  94.25904500881755
```

Hình 3-18: Code lấy ra độ chính xác của mô hình cây quyết định

Ứng dụng mô hình, dự đoán cho 5 người và có kết quả như sau:

```
## Dự đoán khả năng bị bệnh tim cho 5 người sử dụng mô hình cây quyết định
new_data_predictions_decision_tree = model.predict(new_data)

# In kết quả
for i, prediction in enumerate(new_data_predictions_decision_tree):
    print(f"Person {i + 1} - Predicted HadHeartAttack: {prediction}")
```

```
Person 1 - Predicted HadHeartAttack: 0
Person 2 - Predicted HadHeartAttack: 0
Person 3 - Predicted HadHeartAttack: 0
Person 4 - Predicted HadHeartAttack: 0
Person 5 - Predicted HadHeartAttack: 0
```

Hình 3-19: Code dự đoán giá trị kiểm thử

Kết quả cả 5 người được dự đoán đều có kết quả chẩn đoán không mắc bệnh tim.

3.2.1.2. Xây dựng mô hình theo thuật toán Random Forest

Khởi tạo một mô hình Random Forest:

```
# Khởi tạo mô hình Random Forest
random_forest_model = RandomForestClassifier(n_estimators=100, random_state=42)
```

Hình 3-20: Code khởi tạo mô hình Random Forest

Huấn luyện mô hình Random Forest:

```
# Huấn luyện mô hình trên tập huấn luyện
random_forest_model.fit(X_train, y_train.values.ravel())
```

Hình 3-21: Code huấn luyện mô hình Random Forest

Chạy mô hình dự đoán kết quả dự đoán dựa trên bộ dữ liệu features test và in ra kết quả như sau:

```
# Dự đoán trên tập kiểm tra
y_pred = random_forest_model.predict(X_test)
print("Giá trị target mô hình dự đoán được:\n",y_pred)
```

```
Giá trị target mô hình dự đoán được:
 [0 0 0 ... 0 1 0]
```

Hình 3-22: Code dự đoán các giá trị trên tập test

Đánh giá hiệu suất và độ chính xác của mô hình:

```
# Đánh giá hiệu suất
accuracy1 = accuracy_score(y_test, y_pred)
classification_report_result = classification_report(y_test, y_pred)
# In kết quả
print("Accuracy: ",accuracy_score(y_test,y_pred)*100)
print("Classification Report:")
print(classification_report_result)
```

```
Accuracy:  93.87713839621688
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.99      0.97     83889
           1       0.26      0.03      0.06      5138

    accuracy                           0.94     89027
   macro avg       0.60      0.51      0.51     89027
weighted avg       0.90      0.94      0.92     89027
```

Hình 3-23: Đánh giá hiệu suất mô hình Random Forest

Giải thích kết quả:

- Accuracy: Mô hình có độ chính xác là: 93.87%
- Precision, Recall, F1-score cho Lớp 0 (Không mắc bệnh tim)
+ Precision: 94% - Trong số các trường hợp được dự đoán là không mắc bệnh tim, có 94% là đúng.
+ Recall: 99% - Trong số tất cả các trường hợp thực tế không mắc bệnh tim, mô hình dự đoán đúng 99%.
+ F1-score: 97% - Kết hợp giữa precision và recall cho lớp 0.
- Precision, Recall, F1-score cho Lớp 1 (Mắc bệnh tim):
+ Precision: 26% - Trong số các trường hợp được dự đoán là mắc bệnh tim, chỉ có 26% là đúng.
+ Recall: 3% - Chỉ có 3% trong số tất cả các trường hợp thực tế mắc bệnh tim được dự đoán đúng.
+ F1-score: 6% - Kết hợp giữa precision và recall cho lớp 1.
- Support: Số lượng trường hợp thực tế trong mỗi lớp.
- Macro Avg: Đơn giản là lấy trung bình của các chỉ số (precision, recall, f1-score) cho cả hai lớp mà không quan tâm đến kích thước của từng lớp. = (Precision_Lớp0 +

Precision_Lớp1) / 2

- Weighted Avg: Lấy trung bình có trọng số dựa trên kích thước của từng lớp.

*Công thức: Weighted Avg = (Precision_Lớp0 * Số_lượng_Lớp0 + Precision_Lớp1 * Số_lượng_Lớp1) / Tổng_số_lượng*

Xác định ma trận nhầm lẫn:

```
: # Xác định độ nhầm lẫn
  print("Confusion Matrix:", confusion_matrix(y_test,y_pred))

Confusion Matrix: [[83413   476]
 [ 4975   163]]
```

Hình 3-24: Code xây dựng ma trận nhầm lẫn

Giải thích kết quả:

Thực tế	Dự đoán	
	Không mắc bệnh tim	Mắc bệnh tim
Không mắc bệnh tim	83.413	476
Mắc bệnh tim	4975	163

Bảng 3-4: Ma trận nhầm lẫn - Mô hình Random Forest

Ứng dụng mô hình, dự đoán cho 5 người và có kết quả như sau:

```
# Dự đoán khả năng bị bệnh tim sử dụng mô hình random forest
predictions = random_forest_model.predict(new_data)

# In kết quả
for i, prediction in enumerate(predictions):
    print(f"Person {i + 1} - Predicted HadHeartAttack: {prediction}")
```

```
Person 1 - Predicted HadHeartAttack: 0
Person 2 - Predicted HadHeartAttack: 1
Person 3 - Predicted HadHeartAttack: 0
Person 4 - Predicted HadHeartAttack: 0
Person 5 - Predicted HadHeartAttack: 1
```

Hình 3-25: Dự đoán giá trị kiểm thử bằng mô hình Random Forest

Giải thích kết quả: Có 2 trong số 5 người có khả năng mắc bệnh tim

3.2.2. Xây dựng mô hình phân cụm bằng thuật toán K-Means

Thuật toán K-Means được sử dụng để phân chia các bệnh nhân theo từng nhóm với các đặc điểm gần tương tự nhau. Từ đó có thể dễ dàng nhận biết bệnh và cán bộ y tế có thể đưa ra phương pháp điều trị cho phù hợp.

Các bước tiến hành

Bước 1: Khai báo thư viện

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.cluster import KMeans
```

Hình 3-26: Khai báo thư viện để phân cụm dữ liệu

- Thư viện pandas: thư viện xử lý và phân tích dữ liệu.
- Thư viện matplotlib: thư viện tạo ra các biểu đồ và đồ thị.
- Thư viện seaborn: thư viện trực quan hóa dữ liệu dựa trên matplotlib.
- Thư viện numpy: thư viện tính toán
- Thư viện sklearn.cluster: Thư viện cung cấp các công cụ và thuật toán để thực hiện phân cụm.
- Lớp Kmeans: sử dụng để thực hiện thuật toán gom cụm K-means

Bước 2: Đọc dữ liệu

```
[ ] df = pd.read_csv('/content/drive/MyDrive/Khai phá dữ liệu/BTL/Heart_2022_processed.csv')
```

```
[ ] df.head()
```

	HadHeartAttack	Sex	GeneralHealth	PhysicalHealthDays	MentalHealthDays	PhysicalActivities	SleepHours	HadStroke
0	0	0	5	0	0	0	8	0
1	0	0	0	0	0	0	5	0
2	0	0	5	2	3	1	4	0
3	0	0	0	0	0	1	6	0
4	0	0	1	2	0	1	9	0

Hình 3-27: Đọc dữ liệu cần phân cụm

Bước 3: Hiểu dữ liệu

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 445132 entries, 0 to 445131
Data columns (total 18 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   HadHeartAttack         445132 non-null  int64
 1   Sex                    445132 non-null  int64
 2   GeneralHealth          445132 non-null  int64
 3   PhysicalHealthDays     445132 non-null  int64
 4   MentalHealthDays       445132 non-null  int64
 5   PhysicalActivities     445132 non-null  int64
 6   SleepHours             445132 non-null  int64
 7   HadStroke              445132 non-null  int64
 8   HadAsthma              445132 non-null  int64
 9   HadSkinCancer          445132 non-null  int64
 10  HadKidneyDisease       445132 non-null  int64
 11  HadDiabetes            445132 non-null  int64
 12  DifficultyWalking      445132 non-null  int64
 13  SmokerStatus           445132 non-null  int64
 14  RaceEthnicityCategory  445132 non-null  int64
 15  AgeCategory            445132 non-null  int64
 16  BMI                    445132 non-null  float64
 17  AlcoholDrinkers        445132 non-null  int64
dtypes: float64(1), int64(17)
memory usage: 61.1 MB
```

Hình 3-28: Thông tin bộ dữ liệu cần phân cụm

Bước 4: Tạo df_1 là danh sách những người bị bệnh tim

```
[ ] df_1 = df[df['HadHeartAttack']==1]
```

Hình 3-29: Lấy ra thông tin của những người bị bệnh tim

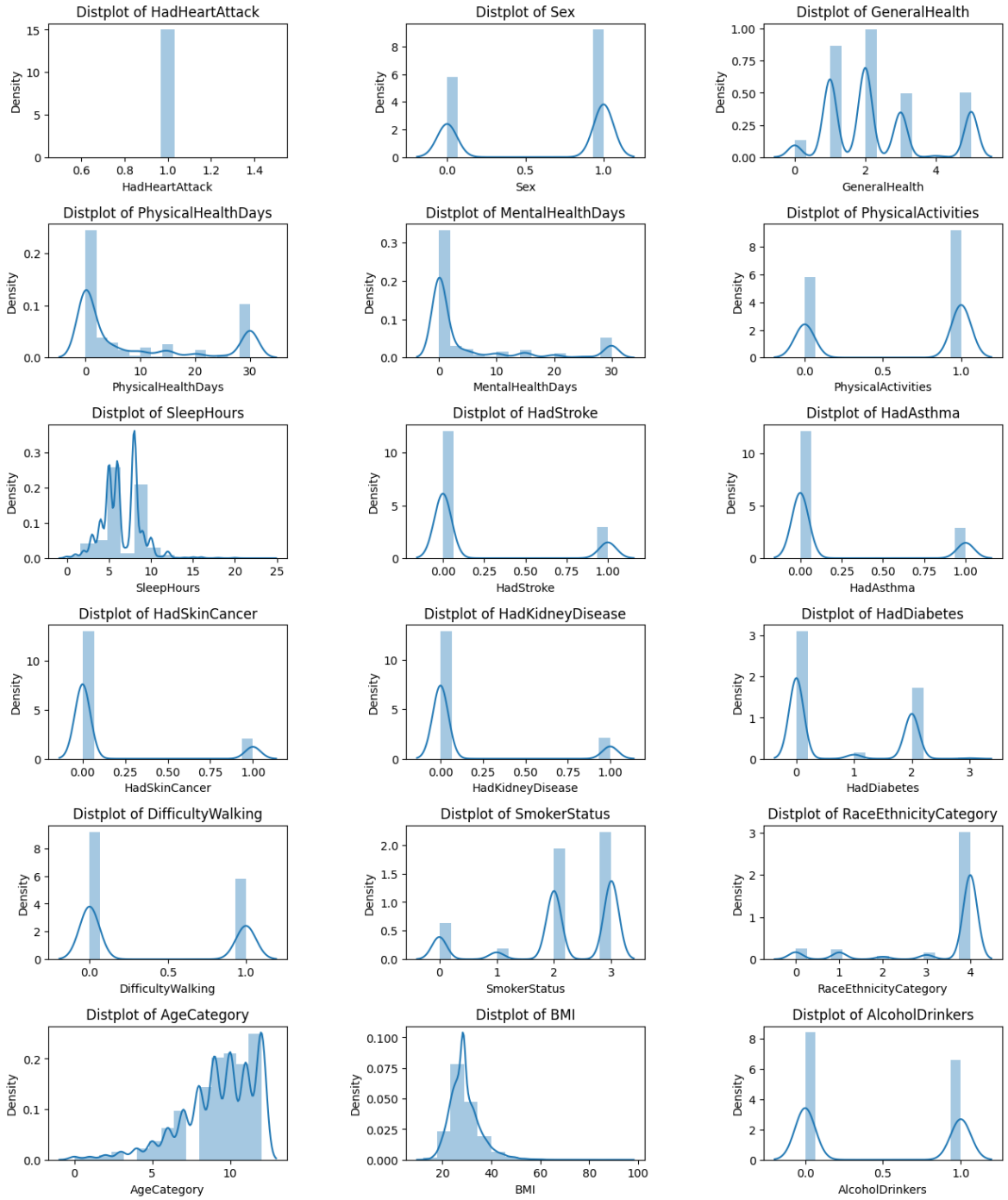
Bước 5: Lựa chọn thuộc tính phân cụm

Vẽ biểu đồ phân phối của các thuộc tính trong ***df_1***:

```
plt.figure(1, figsize=(15, 18)) #Tạo một hình vẽ mới với số thứ tự 1 và kích thước (chiều rộng, chiều cao) là (15, 18)
n = 0 #Khởi tạo biến đếm n với giá trị ban đầu là 0
for x in df.columns:
    n += 1
    plt.subplot(6, 3, n) #Tăng giá trị của biến đếm n lên 1 sau mỗi lần lặp
    plt.subplots_adjust(hspace=0.5, wspace=0.5) #Tạo một ô đồ thị con trên hình vẽ lớn có kích thước 6 hàng,
    # 3 cột và đặt vị trí của đồ thị con bằng giá trị hiện tại của biến đếm n
    sns.distplot(df_1[x], bins=15) #Điều chỉnh khoảng cách theo chiều dọc (hspace) và chiều ngang (wspace) giữa các đồ thị con trong hình vẽ lớn
    plt.title('Distplot of {}'.format(x)) #Đặt tiêu đề cho đồ thị con với nội dung là "Distplot of" kèm theo tên của cột x
plt.show() #Vẽ biểu đồ
```

Hình 3-30: Code vẽ biểu đồ phân phối của các thuộc tính

Biểu đồ phân phối của các thuộc tính:



Hình 3-31: Biểu đồ phân phối của các thuộc tính trong df_1

Dựa vào các biểu đồ trên, nhận thấy các thuộc tính *'PhysicalHealthDays'*, *'MentalHealthDays* và *'BMI'* có khoảng giá trị rộng và phân bố đều hơn các thuộc tính khác. Chọn những thuộc tính này để phân cụm dữ liệu.

Tiến hành phân cụm dựa trên các cặp thuộc tính:

- Phân cụm dựa trên *'PhysicalHealthDays'* và *'BMI'*

- Phân cụm dựa trên *'PhysicalHealthDays'* và *'MentalHealthDays*

3.2.2.1. Phân cụm dựa trên 'PhysicalHealthDays' và 'BMI'

Tạo DataFrame có tên ***df_cluster1*** gồm thuộc tính *'PhysicalHealthDays'* và *'BMI'* của những người bị bệnh tim:

```
df_cluster1 = df_1[['PhysicalHealthDays', 'BMI']]
```

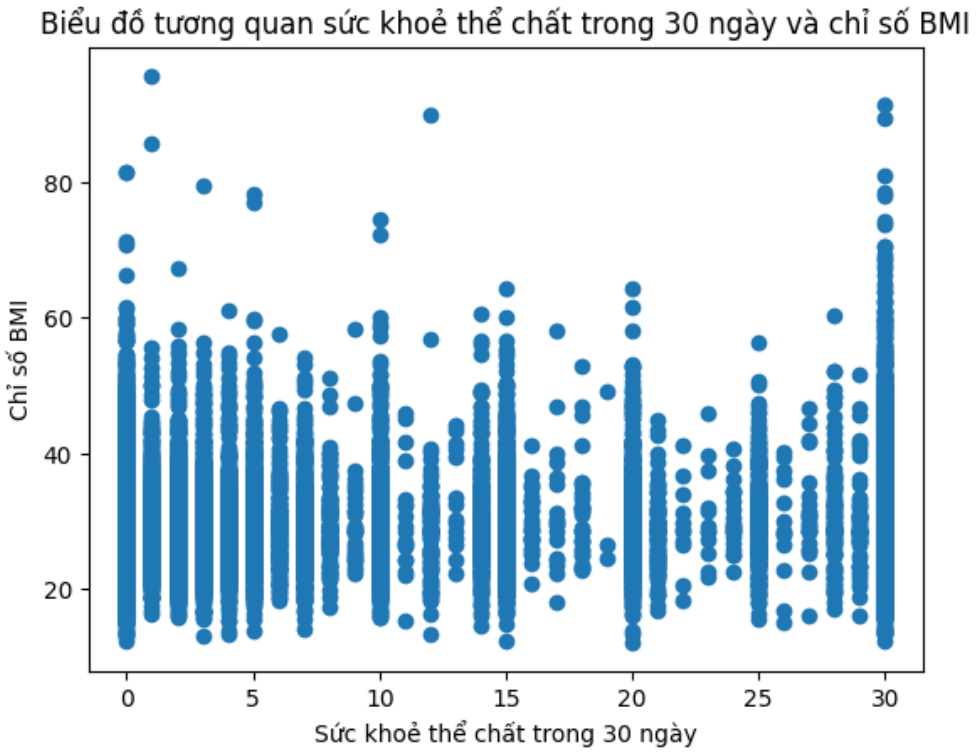
Hình 3-32: Tạo df_cluster1

Vẽ biểu đồ tương quan của 2 thuộc tính trên:

```
plt.scatter(df_cluster1['PhysicalHealthDays'], df_cluster1['BMI']) #Biểu đồ tương quan giữa hai giá trị
plt.xlabel('Sức khoẻ thể chất trong 30 ngày') #Tiêu đề trục hoành
plt.ylabel('Chỉ số BMI') #Tiêu đề trục tung
plt.title('Biểu đồ tương quan sức khoẻ thể chất trong 30 ngày và chỉ số BMI') #Tiêu đề biểu đồ
plt.show()
```

Hình 3-33: Biểu đồ tương quan của 'PhysicalHealthDays' và 'BMI' của những người bị bệnh tim

Kết quả thu được:



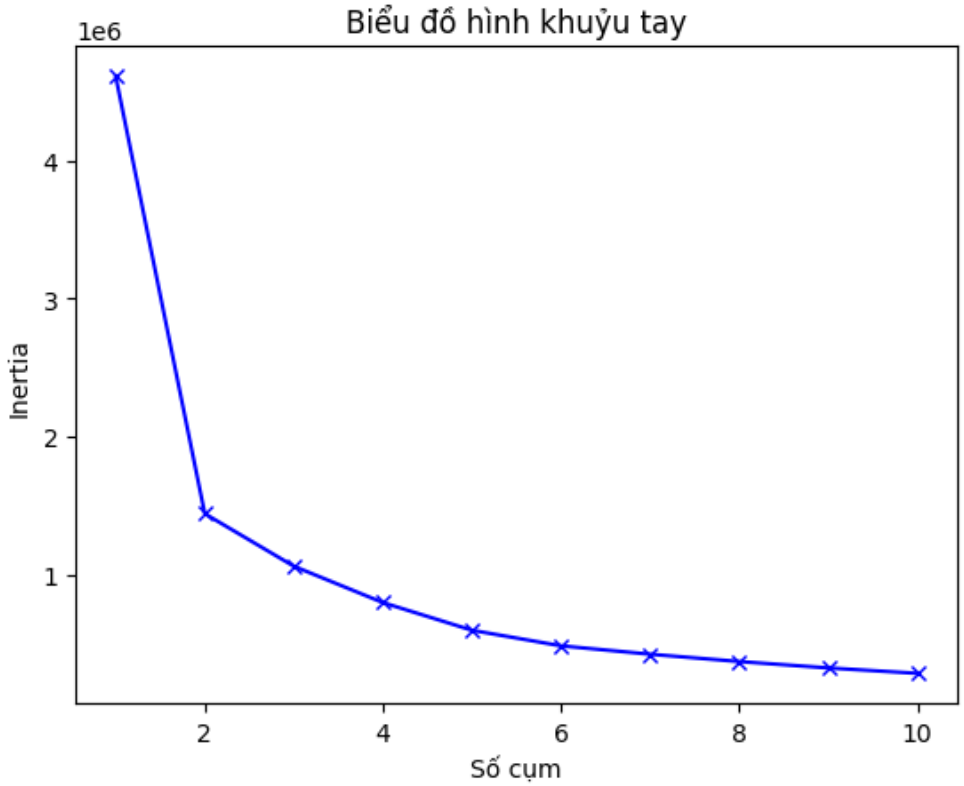
Hình 3-34: Biểu đồ tương quan sức khoẻ thể chất trong 30 ngày và chỉ số BMI

Tìm số cụm tối ưu bằng phương pháp Elbow:

```
inertia = [] #Khởi tạo danh sách inertia để lưu trữ giá trị cho biết mức độ biến động của các điểm dữ liệu trong cùng một cụm.
k_range = range(1, 11) #Tạo dải giá trị từ 1 đến 10
for k in k_range: #tạo vòng lặp từ 1 đến 10 cho giá trị k
    kmeans = KMeans(n_clusters=k, random_state=42) #Tạo KMeans với số cụm là k và khởi tạo ngẫu nhiên của centroid được bằng random_state=42
    kmeans.fit(df_cluster1) #Huấn luyện mô hình KMeans trên dữ liệu df_cluster1 để tìm centroid và phân loại các điểm vào cụm tương ứng
    inertia.append(kmeans.inertia_) #Thêm giá trị inertia của mô hình KMeans hiện tại vào danh sách inertia

# Vẽ đồ thị Elbow
plt.plot(k_range, inertia, 'bx-') #Vẽ đồ thị đường với các điểm được đánh dấu bằng ký hiệu dấu x.
plt.xlabel('Số cụm') #tiêu đề trục hoành
plt.ylabel('Inertia') #tiêu đề trục tung
plt.title('Biểu đồ hình khuỷu tay') #tiêu đề biểu đồ
plt.show()
```

Hình 3-35: Code vẽ biểu đồ hình khuỷu tay

Biểu đồ hình khuỷu tay:



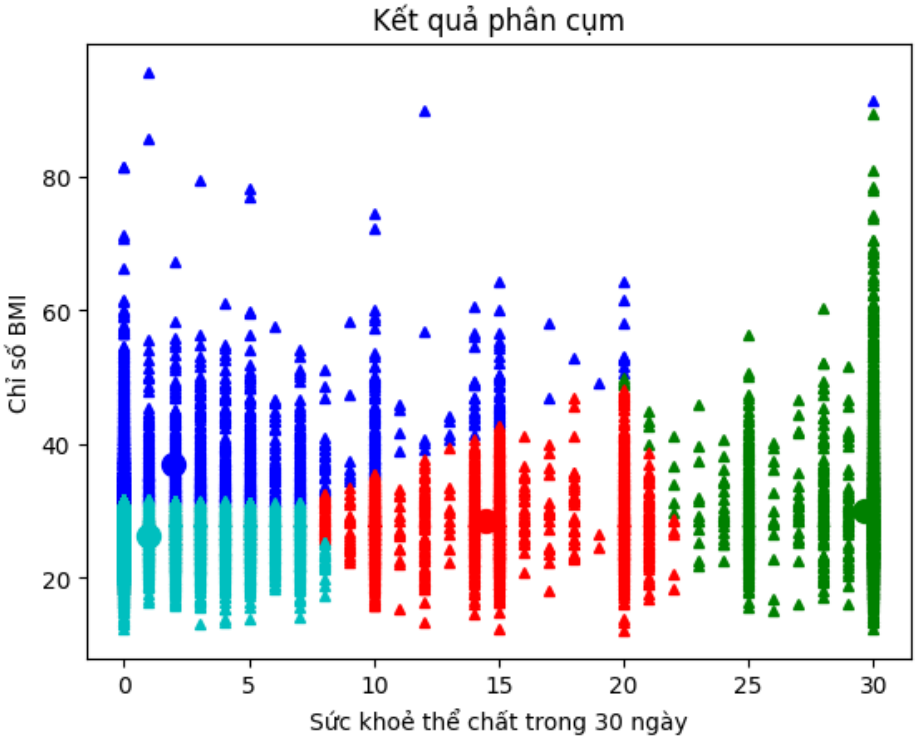
Hình 3-36: Biểu đồ hình khuỷu tay của 'PhysicalHealthDays' và 'BMI'

Dựa vào biểu đồ trên, ta nhận thấy inertia giảm nhanh chóng với số lượng cụm từ 2 đến 4. Sau đó, inertia bắt đầu tăng dần với số lượng cụm từ 5 trở lên. Do đó, số lượng cụm tối ưu là 4. Ta thấy, inertia giảm nhanh chóng từ 2 đến 4. Điều này cho thấy rằng các điểm dữ liệu được phân thành 4 cụm là phù hợp nhất. Sau 4 cụm, inertia bắt đầu tăng lên, nghĩa là việc có thêm cụm sẽ không cải thiện khả năng phân cụm của dữ liệu. Vậy nên, ta phân cụm dữ liệu của 2 thuộc tính *'PhysicalHealthDays'* và *'BMI'* thành 4 cụm:

```
clus1 = 4  # Xác định số cụm (sửa thành 4)
kmeans = KMeans(n_clusters=clus1).fit(X1) # mô hình K-means được huấn luyện trên dữ liệu X1 đã được phân cụm
pred_label1 = kmeans.predict(X1) # dự đoán nhãn của các điểm dữ liệu trong X2 dựa trên mô hình K-means đã huấn luyện

plt.xlabel('Sức khoẻ thể chất trong 30 ngày')
plt.ylabel('Chỉ số BMI')
plt.title('Kết quả phân cụm')

plt_colors = ['b', 'g', 'r', 'c'] # danh sách các màu hỗ trợ cho 4 cụm (sửa thành 4)

for i in range(4): # lặp qua 4 cụm (sửa thành 4)
    data1 = X1[pred_label1 == i]
    plt.plot(data1[:, 0], data1[:, 1], plt_colors[i] + '^', markersize=4)
    plt.plot(kmeans.cluster_centers_[i][0], kmeans.cluster_centers_[i][1], plt_colors[i] + 'o', markersize=10)
plt.show()
```

Hình 3-37: Code phân cụm dữ liệu dựa trên 'PhysicalHealthDays' và 'BMI'

Kết quả phân cụm:



Hình 3-38: Kết quả phân cụm dữ liệu dựa trên 'PhysicalHealthDays' và 'BMI'

Dựa vào kết quả phân cụm, có thể thấy rằng các bệnh nhân bị bệnh tim được phân thành 5 nhóm dựa trên chỉ số BMI và sức khoẻ thể chất trong 30 ngày:

- Nhóm 1: Nhóm người ổn định về sức khoẻ thể chất và có chỉ số BMI trong khoảng từ 12 đến 30 (màu xanh lơ).
- Nhóm 2: Nhóm người có khoảng 7 đến 22 ngày trên tháng không được đảm bảo về sức khoẻ thể chất và có chỉ số BMI rơi vào khoảng 12 đến 50 (màu đỏ).
- Nhóm 3: Nhóm người có chỉ số BMI ở mức cao, từ 35 trở lên (màu xanh tím

than).

- Nhóm 4: Nhóm người không ổn định về sức khoẻ thể chất (màu xanh lá cây).

3.2.2.2. Phân cụm dựa trên 'PhysicalHealthDays' và 'MentalHealthDays'

Tương tự các bước và kỹ thuật sử dụng trong phần trước, ta tạo DataFrame ***df_cluster2*** gồm các thuộc tính *'PhysicalHealthDays' và 'MentalHealthDays'* của những người mắc bệnh tim:

```
df_cluster2 = df_1[['MentalHealthDays', 'PhysicalHealthDays']]
```
0.0s

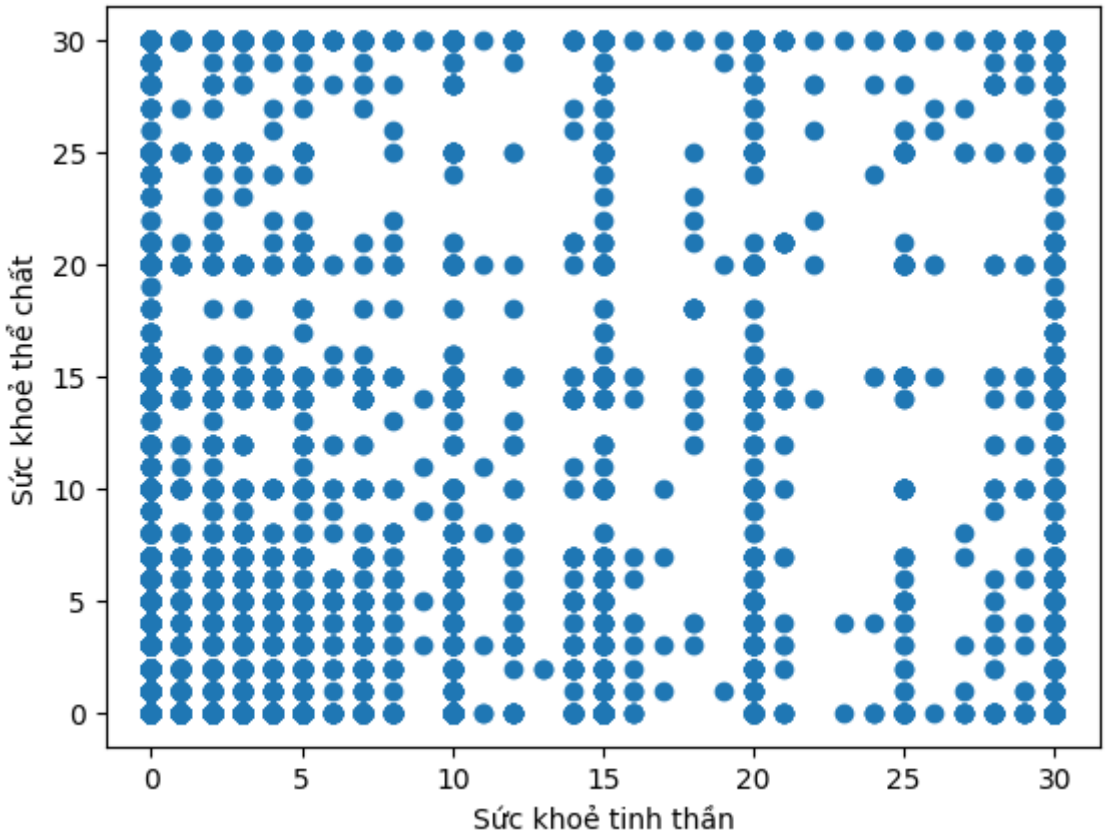
Hình 3-39: Tạo df_cluster2

Biểu đồ tương quan của các thuộc tính *PhysicalHealthDays' và 'MentalHealthDays'*

```
plt.scatter(df_cluster2['MentalHealthDays'], df_cluster2['PhysicalHealthDays'])
plt.xlabel('Sức khoẻ tinh thần')
plt.ylabel('Sức khoẻ thể chất')
plt.show()
```
✓ 0.2s

Hình 3-40: Code vẽ biểu đồ tương quan của các thuộc tính 'PhysicalHealthDays' và 'MentalHealthDays'

Biểu đồ tương quan của các thuộc tính *'PhysicalHealthDays' và 'MentalHealthDays'*:



Hình 3-41: Biểu đồ tương quan của các thuộc tính 'PhysicalHealthDays' và 'MentalHealthDays'

Chuyển các thuộc tính từ dạng DataFrame về dạng mảng:

```
X2 = df_cluster2.values[:, 0:df_cluster2.shape[1]]
```

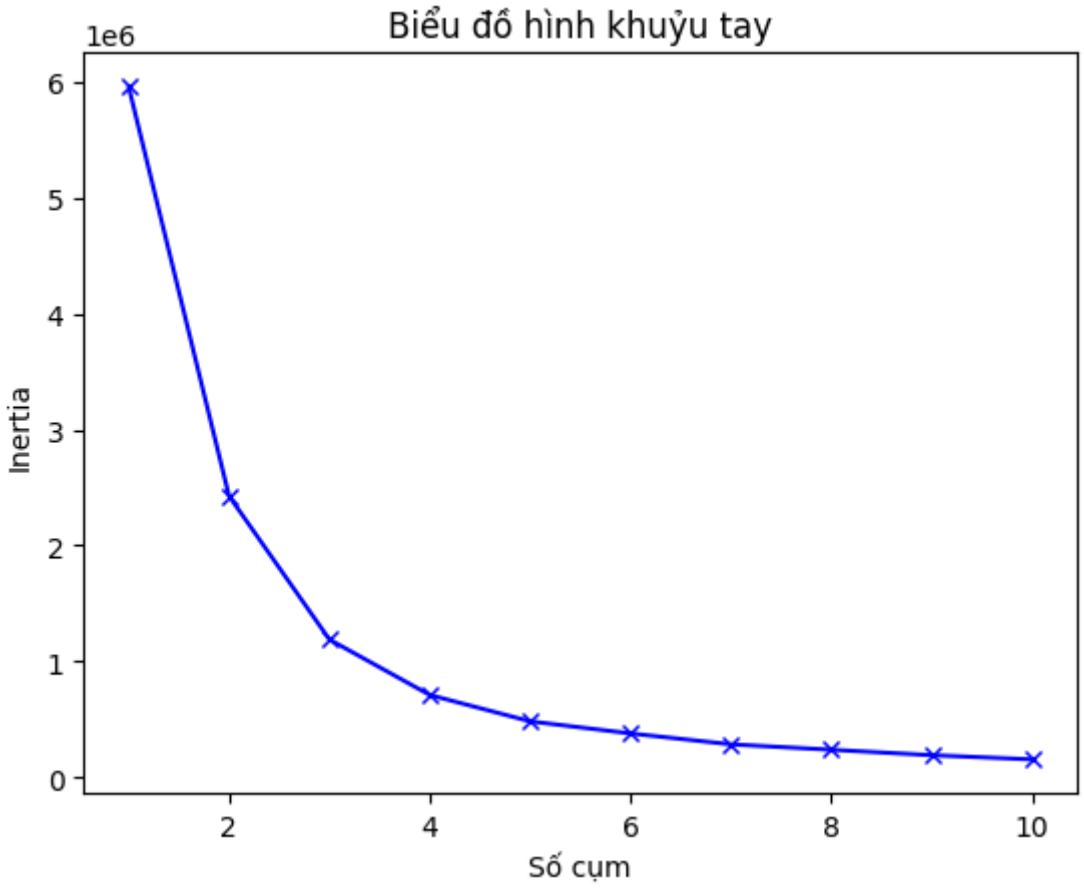
Hình 3-42: Chuyển các thuộc tính từ dạng df_cluster2 về dạng mảng

Vẽ biểu đồ hình khuỷu tay để tìm số cụm tối ưu nhất:

```
inertia = []
k_range = range(1, 11)
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df_cluster2)
    inertia.append(kmeans.inertia_)

# Vẽ đồ thị Elbow
plt.plot(k_range, inertia, 'bx-')
plt.xlabel('Số cụm')
plt.ylabel('Inertia')
plt.title('Biểu đồ hình khuỷu tay')
plt.show()
```

Hình 3-43: Vẽ biểu đồ hình khuỷu tay cho 'PhysicalHealthDays' và 'MentalHealthDays'

Biểu đồ hình khuỷu tay:



Hình 3-44: Vẽ biểu đồ hình khuỷu tay của 'PhysicalHealthDays' và 'MentalHealthDays'

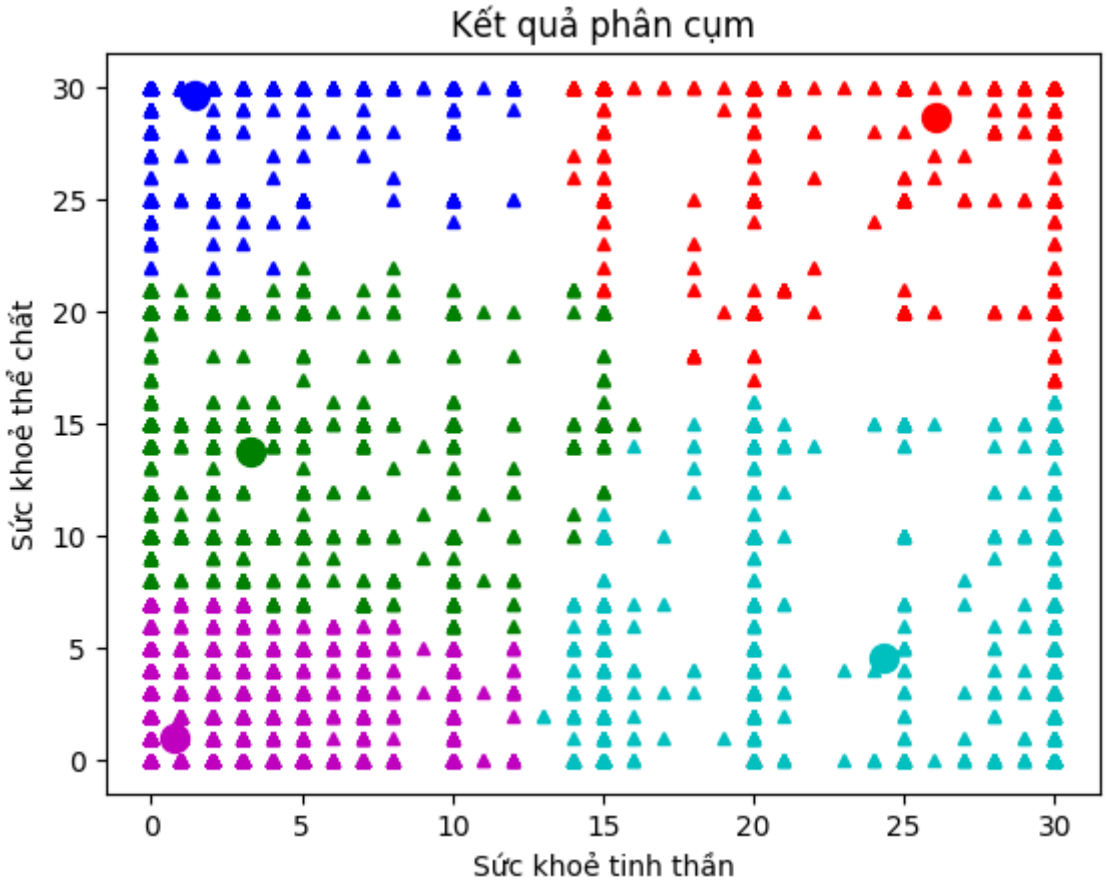
Dựa vào biểu đồ hình khuỷu tay, dễ thấy số cụm tối ưu là 5 cụm. Vậy nên, ta phân

cụm hai thuộc tính là *'PhysicalHealthDays' và 'MentalHealthDays'* thành 5 cụm.

```
clus2 = 5  # Xác định số cụm (sửa thành 5)
kmeans2 = KMeans(n_clusters=clus2).fit(X2)  # mô hình K-means được huấn luyện trên dữ liệu X2 đã được phân cụm
pred_label2 = kmeans2.predict(X2)  # dự đoán nhãn của các điểm dữ liệu trong X2 dựa trên mô hình K-means đã huấn luyện

# Vẽ biểu đồ
plt.xlabel('Sức khoẻ tinh thần')
plt.ylabel('Sức khoẻ thể chất')
plt.title('Kết quả phân cụm')

plt_colors = ['b', 'g', 'r', 'c', 'm']  # danh sách các màu hỗ trợ cho 5 cụm (sửa thành 5)

for i in range(5):  # Phân thành 5 cụm (sửa thành 5)
    data2 = X2[pred_label2 == i]
    plt.plot(data2[:, 0], data2[:, 1], plt_colors[i] + '^', markersize=4)
    plt.plot(kmeans2.cluster_centers_[i][0], kmeans2.cluster_centers_[i][1], plt_colors[i] + 'o', markersize=10)
plt.show()
```

Hình 3-45: Phân cụm dựa trên 'PhysicalHealthDays' và 'MentalHealthDays'

Kết quả phân cụm:



Hình 3-46: Kết quả phân cụm dựa trên 'PhysicalHealthDays' và 'MentalHealthDays'

Dựa vào kết quả phân cụm, có thể thấy rằng các bệnh nhân bị bệnh tim được phân thành 5 nhóm dựa trên sức khoẻ thể chất và tinh thần trong 30 ngày:

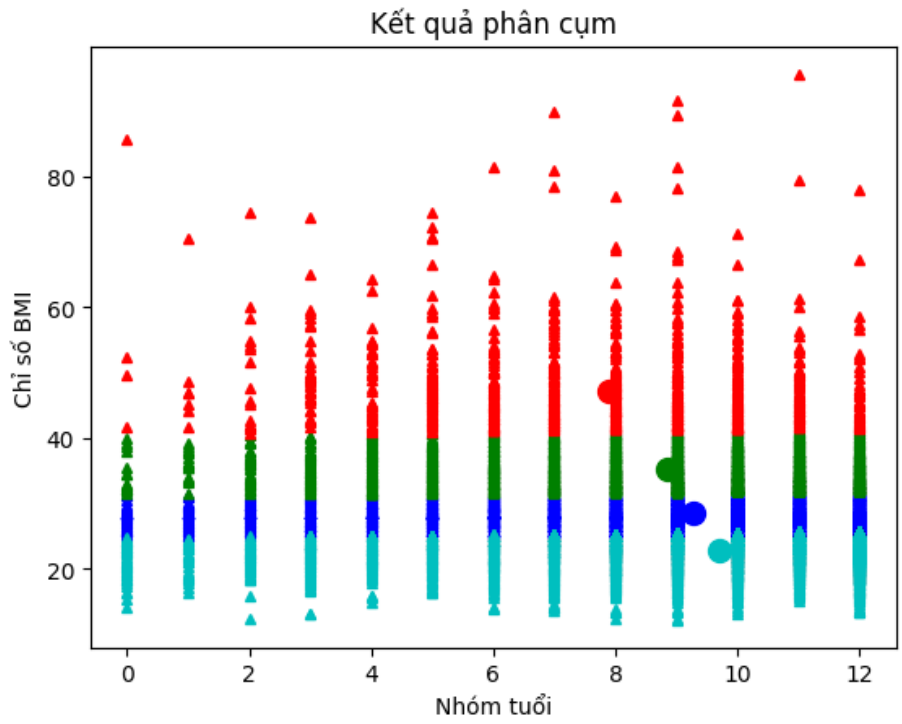
- Nhóm 1: Nhóm người khoẻ mạnh cả về thể chất và tinh thần (màu tím).
- Nhóm 2: Nhóm người khoẻ mạnh về tinh thần và có sức khoẻ thể chất ở mức trung bình (màu xanh lá cây).
- Nhóm 3: Nhóm người khoẻ mạnh về tinh thần nhưng không tốt về thể chất (màu xanh tím than).
- Nhóm 4: Nhóm người có sức khoẻ tốt về thể chất nhưng không khoẻ về tinh thần (màu xanh lơ).
- Nhóm 5: Nhóm người không khoẻ về cả thể chất và tinh thần (màu đỏ).

3.2.2.3. Đánh giá về kỹ thuật phân cụm

Do bài toán hướng đến việc dự đoán nguy cơ mắc bệnh tim mạch dựa những triệu chứng của người mắc nên việc sử dụng kỹ thuật phân cụm sẽ nhằm mục đích tìm ra những khoảng giá trị thường xuất hiện cùng nhau của các thuộc tính (đại diện cho các triệu chứng). Kỹ thuật phân cụm trong hai ví dụ nhóm chúng em đưa ra đã cho ra kết quả là những nhóm bệnh nhân được phân chia dựa trên thang đo của từng cặp triệu chứng.

Nếu phân cụm trên những bộ dữ liệu dựa trên những tập dữ liệu mà giá trị của các thuộc tính được phân bố rời rạc trên một khoảng giá trị không đủ dài, kỹ thuật phân cụm bằng K-Means sẽ cho kết quả không thực sự có ý nghĩa. Phân cụm theo ‘AgeCategory’ và ‘BMI’ là một ví dụ.

Như hình bên dưới, việc phân cụm theo chỉ số BMI và nhóm tuổi chỉ được phân chia theo chỉ số BMI. Các cụm trải dài trên mọi nhóm tuổi. Sở dĩ có điều này là do tuổi của bệnh nhân trong bộ dữ liệu được phân chia theo nhóm, không có tuổi cụ thể. Vậy nên, một giá trị biểu thị nhóm tuổi sẽ xuất hiện nhiều lần trong toàn bộ tập dữ liệu, dẫn đến việc phân cụm như **Hình 3-44**.



Hình 3-47: Kết quả phân cụm theo 'AgeCategory' và 'BMI'

Cả hai ví dụ về phân cụm đều cho thấy kỹ thuật phân cụm hoạt động tốt trên những tập thuộc tính mà giá trị của các thuộc tính đó có được phân bố trên một khoảng giá trị rộng. Điều này làm cho những phần tử có khoảng cách xa nhau sẽ không có những tính chất tương tự nhau. Khi đó, toàn bộ những thuộc tính đưa vào sẽ được phân chia thành các cụm mà các phần tử gần nhau có tính chất tương tự nhau.

3.2.3. Xây dựng mô hình theo thuật toán luật kết hợp

3.2.3.1. Thuật toán Apriori

- Cài đặt các thư viện liên quan:

```
[ ]  !pip install apyori

     Requirement already satisfied: apyori in /usr/local/lib/python3.10/dist-packages (1.1.2)
```

```
[ ]  import numpy as np
     import pandas as pd
     from apyori import apriori
```

Hình 3-48: Khai báo các thư viện để khai phá luật kết hợp bằng thuật toán Apriori

- Đọc bộ dữ liệu

	HadHeartAttack	Sex	GeneralHealth	PhysicalHealthDays	MentalHealthDays	PhysicalActivities	SleepHours	HadStroke	HadAsthma	HadSkinCancer
0	No	Female	Very good	0.0	0.0	No	8.0	No	No	No
1	No	Female	Excellent	0.0	0.0	No	6.0	No	No	Yes
2	No	Female	Very good	2.0	3.0	Yes	5.0	No	No	Yes
3	No	Female	Excellent	0.0	0.0	Yes	7.0	No	Yes	No
4	No	Female	Fair	2.0	0.0	Yes	9.0	No	No	No
5	Yes	Male	Poor	1.0	0.0	No	7.0	Yes	No	No
6	No	Female	Very good	0.0	0.0	Yes	7.0	No	No	No
7	No	Female	Good	0.0	0.0	No	8.0	No	No	No
8	No	Female	Good	0.0	0.0	Yes	6.0	No	No	Yes

Hình 3-49: Bộ dữ liệu để khai phá luật kết hợp

```
print("Số dòng:",df.shape[0])
print("Số cột:", df.shape[1])
```

```
Số dòng: 445132
Số cột: 18
```

Hình 3-50: Kích thước bộ dữ liệu khai phá luật kết hợp

- Mô tả bộ dữ liệu

```
df['AgeCategory'].value_counts()
```

```
Age 65 to 69       56178
Age 60 to 64       44511
Age 70 to 74       43472
Age 55 to 59       36821
Age 80 or older    36251
Age 50 to 54       33644
Age 75 to 79       32518
Age 40 to 44       29942
Age 45 to 49       28531
Age 35 to 39       28526
Age 18 to 24       26941
Age 30 to 34       25807
Age 25 to 29       21990
Name: AgeCategory, dtype: int64
```

Hình 3-51: Số lượng mỗi giá trị của thuộc tính 'AgeCategory'

```
df['GeneralHealth'].value_counts()
```

```
Very good    148444
Good         143598
Excellent     71878
Fair          60273
Poor          19741
Very Good      1198
Name: GeneralHealth, dtype: int64
```

Hình 3-52: Số lượng mỗi giá trị của thuộc tính GeneralHealth

- Biến đổi một số cột của dữ liệu để phù hợp với thuật toán

```
#DO BẢNG TRÊN CÓ GIÁ TRỊ CÁC CỘT LÀ Yes, No; giữ nguyên thì kết quả cuối cùng chỉ hiện Yes và No ko rõ là của cột nào
#Vậy ta đổi lại tên Yes + tên cột, No + tên cột để phân biệt
df["HadHeartAttack"].replace("Yes","Yes_HadHeartAttack",inplace = True)
df["HadHeartAttack"].replace("No","No_HadHeartAttack",inplace = True)
df["PhysicalActivities"].replace("Yes","Yes_PhysicalActivities",inplace = True)
df["PhysicalActivities"].replace("No","No_PhysicalActivities",inplace = True)
df["HadStroke"].replace("Yes","Yes_HadStroke",inplace = True)
df["HadStroke"].replace("No","No_HadStroke",inplace = True)
df["HadAsthma"].replace("Yes","Yes_HadAsthma",inplace = True)
df["HadAsthma"].replace("No","No_HadAsthma",inplace = True)
df["HadSkinCancer"].replace("Yes","Yes_HadSkinCancer",inplace = True)
df["HadSkinCancer"].replace("No","No_HadSkinCancer",inplace = True)
df["HadKidneyDisease"].replace("Yes","Yes_HadKidneyDisease",inplace = True)
df["HadKidneyDisease"].replace("No","No_HadKidneyDisease",inplace = True)
df["HadDiabetes"].replace("Yes","Yes_HadDiabetes",inplace = True)
df["HadDiabetes"].replace("No","No_HadDiabetes",inplace = True)
df["DifficultyWalking"].replace("Yes","Yes_DifficultyWalking",inplace = True)
df["DifficultyWalking"].replace("No","No_DifficultyWalking",inplace = True)
df["AlcoholDrinkers"].replace("Yes","Yes_AlcoholDrinkers",inplace = True)
df["AlcoholDrinkers"].replace("No","No_AlcoholDrinkers",inplace = True)
df
```

Hình 3-53: Biến đổi dữ liệu để khai phá luật kết hợp

```
[ ]  # Ghép các giá trị trong cột với tên cột
     df["PhysicalHealthDays"] = df["PhysicalHealthDays"].astype(str) + "_PhysicalHealthDays"
     df["MentalHealthDays"] = df["MentalHealthDays"].astype(str) + "_MentalHealthDays"
     df["SleepHours"] = df["SleepHours"].astype(str) + "_SleepHours"
     df["BMI"] = df["BMI"].astype(str) + "_BMI"
     df
```

Hình 3-54: Ghép giá trị với tên cột

- Kết quả sau biến đổi dữ liệu

	HadHeartAttack	Sex	GeneralHealth	PhysicalHealthDays	MentalHealthDays	PhysicalActivities	SleepHours	HadStroke
0	No_HadHeartAttack	Female	Very good	0.0_PhysicalHealthDays	0.0_MentalHealthDays	No_PhysicalActivities	8.0_SleepHours	No_HadStroke
1	No_HadHeartAttack	Female	Excellent	0.0_PhysicalHealthDays	0.0_MentalHealthDays	No_PhysicalActivities	6.0_SleepHours	No_HadStroke
2	No_HadHeartAttack	Female	Very good	2.0_PhysicalHealthDays	3.0_MentalHealthDays	Yes_PhysicalActivities	5.0_SleepHours	No_HadStroke
3	No_HadHeartAttack	Female	Excellent	0.0_PhysicalHealthDays	0.0_MentalHealthDays	Yes_PhysicalActivities	7.0_SleepHours	No_HadStroke
4	No_HadHeartAttack	Female	Fair	2.0_PhysicalHealthDays	0.0_MentalHealthDays	Yes_PhysicalActivities	9.0_SleepHours	No_HadStroke
...	...	...	...	...	...	...	...	...
445127	No_HadHeartAttack	Female	Good	0.0_PhysicalHealthDays	3.0_MentalHealthDays	Yes_PhysicalActivities	6.0_SleepHours	No_HadStroke

- Vì thuật toán cần tiến hành trên dạng List nên ta chuyển bộ dữ liệu từ dạng DataFrame sang List

```
# Chuyển dữ liệu từ dạng bảng -> dạng Danh sách
records = df.head(400000).to_numpy().reshape(-1, 18).tolist()
```

```
#In 5 danh sách đầu tiên
print(records[0:5])
```

Hình 3-55: Chuyển bộ dữ liệu từ dạng DataFrame sang List

- Sử dụng thuật toán apriori với các thông số min_support = 0.005; min_confidence = 0.003; min_lift = 3; min_length = 3; max_length=3

```
association_rules = apriori(records, min_support=0.005, min_confidence=0.002, min_lift=3, min_length=3, max_length=3)
# Chỉ hiển thị các mục có độ dài là 3
filtered_association_rules = list(filter(lambda rule: len(rule.items) == 3, association_rules))
```

Hình 3-56: Khai báo thuật toán Apriori

- Đổi kết quả các luật về dạng List và đếm số luật hình thành

```
# #Đổi các luật về dạng danh sách
association_results = list(filtered_association_rules )
```

```
#Xem kết quả
print(len(filtered_association_rules ))
```

```
240
```

Hình 3-57: Đổi kết quả các luật về dạng List và đếm số luật hình thành

- Hiển thị kết quả luật kết hợp hình thành

```
#Hiển thị luật, độ hỗ trợ, độ tin cậy và lift cho từng luật theo cách rõ ràng hơn
for item in filtered_association_rules:
  #Dòng đầu
  pair = item[0]
  items = [x for x in pair]
  print("Rule: " + items[0] + " + " + items[1] + " -> " + items[2])

  #Dòng thứ hai chứa độ hỗ trợ
  print("Support: " + str(item[1]))

  #Dòng thứ 3
  print("Confidence: " + str(item[2][0][2]))
  print("Lift: " + str(item[2][0][3]))
  print("==============================")
```

Hình 3-58: Hiển thị kết quả luật kết hợp hình thành

- Vì bài toán hướng tới việc dự báo khả năng bị bệnh tim nên ta loại bỏ các luật kết hợp không liên quan đến bệnh tim thu được một số kết quả như sau:

```
==============================
Rule: Yes_DifficultyWalking + Male -> Yes_HadHeartAttack
Support: 0.0118
Confidence: 0.2078654159510283
Lift: 3.508425097278844
==============================
```

Hình 3-59: Kết quả khai phá luật kết hợp (1)

Giải thích:

- Ý nghĩa của luật này là nếu một người có giới tính Nam và gặp phải vấn đề về việc đi lại vận động thì sẽ có nguy cơ mắc bệnh về tim.
- Độ hỗ trợ của luật là 0.0018 lớn hơn so với độ hỗ trợ tối thiểu là 0.005.
- Độ tin cậy của luật là sấp xỉ 0.207 cao hơn độ tin cậy tối thiểu là 0.002.
- Độ nâng hay hiểu là độ mạnh mẽ của luật này là sấp xỉ 3.5 lớn hơn độ nâng tối thiểu là 3.

Một số kết quả khác:

```
==============================
Rule: Yes_HadDiabetes + Male -> Yes_HadHeartAttack
Support: 0.0122175
Confidence: 0.21521997621878716
Lift: 3.172230469729341
==============================
```

Hình 3-60: Kết quả khai phá luật kết hợp (2)

```
Rule: Yes_DifficultyWalking + No_AlcoholDrinkers -> Yes_HadHeartAttack
Support: 0.01468
Confidence: 0.09672052840506662
Lift: 3.051361413520518
==============================
```

Hình 3-61: Kết quả khai phá luật kết hợp (3)

```
==============================
Rule: No_AlcoholDrinkers + Yes_HadStroke -> Yes_HadHeartAttack
Support: 0.006945
Confidence: 0.1223411282864315
Lift: 4.900505839632745
==============================
```

Hình 3-62: Kết quả khai phá luật kết hợp (4)

```
==============================
Rule: Yes_HadStroke + No_HadAsthma -> Yes_HadHeartAttack
Support: 0.0084425
Confidence: 0.1487206588276743
Lift: 4.354286600136856
==============================
```

Hình 3-63: Kết quả khai phá luật kết hợp (5)

```
------------------------------
Rule: Yes_HadStroke + No_HadKidneyDisease -> Yes_HadHeartAttack
Support: 0.0090125
Confidence: 0.15876161536090191
Lift: 4.210345829368213
==============================
```

Hình 3-64: Kết quả khai phá luật kết hợp (6)

```
==============================
Rule: Yes_HadStroke + White only, Non-Hispanic -> Yes_HadHeartAttack
Support: 0.0085525
Confidence: 0.15065838728145506
Lift: 4.503651065883119
==============================
```

Hình 3-65: Kết quả khai phá luật kết hợp (7)

```
==============================
Rule: Yes_DifficultyWalking + Yes_HadAsthma -> Yes_HadHeartAttack
Support: 0.0055525
Confidence: 0.03658315626492728
Lift: 3.4046678701654054
==============================
```

Hình 3-66: Kết quả khai phá luật kết hợp (8)

```
==============================
Rule: 30.0_PhysicalHealthDays + No_AlcoholDrinkers -> Yes_HadHeartAttack
Support: 0.0073025
Confidence: 0.09919179570768812
Lift: 3.1293255211826834
==============================
```

Hình 3-67: Kết quả khai phá luật kết hợp (9)

3.2.3.2. Thuật toán ECLAT

- Cài đặt các thư viện liên quan

```
!pip install pyECLAT

Collecting pyECLAT
  Downloading pyECLAT-1.0.2-py3-none-any.whl (6.3 kB)
Requirement already satisfied: pandas>=0.25.3 in /usr/local/lib/python3.10/dist-packages (from pyECLAT) (1.5.3)
Requirement already satisfied: numpy>=1.17.4 in /usr/local/lib/python3.10/dist-packages (from pyECLAT) (1.23.5)
Requirement already satisfied: tqdm>=4.41.1 in /usr/local/lib/python3.10/dist-packages (from pyECLAT) (4.66.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.25.3->pyECLAT) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.25.3->pyECLAT) (2023.3.post1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas>=0.25.3->pyECLAT) (1.16
Installing collected packages: pyECLAT
Successfully installed pyECLAT-1.0.2

[ ] import numpy as np
    import pandas as pd
```

Hình 3-68: Các thư viện cần thiết để khai phá luật kết hợp bằng thuật toán ECLAT

- Chạy phương thức ECLAT trong thư viện pyECLAT để khởi tạo thuật toán eclat

```
[14] from pyECLAT import ECLAT
     eclat = ECLAT(data=df, verbose = True)
     #verbose=True -> hiển thị thanh loading

     100%|██████████| 2548/2548 [04:00<00:00, 10.57it/s]
     100%|██████████| 2548/2548 [00:11<00:00, 217.63it/s]
     100%|██████████| 2548/2548 [00:01<00:00, 2135.61it/s]
```

Hình 3-69: Chạy phương thức ECLAT

- Sau khi nhận được ECLAT , một khung dữ liệu nhị phân sẽ được tạo tự động, cùng với các tài nguyên về các giá trị độc lập trong bộ dữ liệu có thể được truy cập:

```
[15] eclat.df_bin #tạo 1 binary dataframe
```

Hình 3-70: Tạo binary dataframe

	16.29_BMI	16.56_BMI	19.19_BMI	50.64_BMI	18.44_BMI	36.68_BMI	37.16_BMI	22.48_BMI	18.61_BMI	21.03_BMI	...	23.57_BMI	53.0_BMI	28.19_BMI	38.12_BM
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
49995	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
49996	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
49997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
49998	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
49999	0	0	0	0	0	0	0	0	0	0	...	0	0	0	

50000 rows × 2548 columns

Hình 3-71: Binary dataframe

```
[16] eclat.uniq_ #danh sách các giá trị độc lập
```

```
['16.29_BMI',
 '16.56_BMI',
 '19.19_BMI',
 '50.64_BMI',
 '18.44_BMI',
 '36.68_BMI',
 '37.16_BMI',
 '22.48_BMI',
 '18.61_BMI',
 '21.03_BMI',
 '54.91_BMI',
 '50.85_BMI',
 '28.51_BMI',
 '33.19_BMI',
 '18.75_BMI',
 '35.67_BMI',
```

Hình 3-72: Danh sách các giá trị độc lập

- Sử dụng thuật toán ECLAT với min_support = 0.05, và min_combination = 3

```
[17] indexes, support = eclat.fit(min_support = 0.05, min_combination = 3, max_combination = 3, verbose=True)

Combination 3 by 3
24804it [1:09:24,  5.96it/s]
```

Hình 3-73: Sử dụng thuật toán Eclat

- Kết quả các luật tạo ra

```
support
```

```
{'Very good & White only, Non-Hispanic & No_HadKidneyDisease': 0.24126,
 'Very good & White only, Non-Hispanic & No_HadDiabetes': 0.22594,
 'Very good & White only, Non-Hispanic & Male': 0.11752,
 'Very good & White only, Non-Hispanic & Former smoker': 0.06898,
 'Very good & White only, Non-Hispanic & No_DifficultyWalking': 0.23368,
 'Very good & White only, Non-Hispanic & No_HadAsthma': 0.21524,
 'Very good & White only, Non-Hispanic & No_HadSkinCancer': 0.22098,
 'Very good & White only, Non-Hispanic & No_HadStroke': 0.2418,
 'Very good & White only, Non-Hispanic & Female': 0.12996,
 'Very good & White only, Non-Hispanic & No_HadHeartAttack': 0.2389,
 'Very good & White only, Non-Hispanic & Never smoked': 0.16016,
 'Very good & White only, Non-Hispanic & Yes_PhysicalActivities': 0.21364,
 'Very good & White only, Non-Hispanic & No_AlcoholDrinkers': 0.08292,
 'Very good & White only, Non-Hispanic & Yes_AlcoholDrinkers': 0.16456,
 'Very good & White only, Non-Hispanic & 7.0_SleepHours': 0.09186,
 'Very good & White only, Non-Hispanic & 0.0_MentalHealthDays': 0.15816,
 'Very good & White only, Non-Hispanic & 8.0_SleepHours': 0.07548,
 'Very good & White only, Non-Hispanic & 0.0_PhysicalHealthDays': 0.17584,
 'Very good & No_HadKidneyDisease & No_HadDiabetes': 0.2902,
 'Very good & No_HadKidneyDisease & Male': 0.15396,
 'Very good & No_HadKidneyDisease & Former smoker': 0.0795,
 'Very good & No_HadKidneyDisease & No_DifficultyWalking': 0.30222,
 'Very good & No_HadKidneyDisease & No_HadAsthma': 0.27752,
 'Very good & No_HadKidneyDisease & No_HadSkinCancer': 0.29284,
 'Very good & No_HadKidneyDisease & No_HadStroke': 0.31208,
 'Very good & No_HadKidneyDisease & 6.0_SleepHours': 0.06576,
 'Very good & No_HadKidneyDisease & Female': 0.16484,
 'Very good & No_HadKidneyDisease & No_HadHeartAttack': 0.30962,
```

Hình 3-74: Danh sách các luật được tạo ra

- Một số luật về bệnh tim: Vì bài toán hướng tới việc dự báo khả năng bị bệnh tim nên ta loại bỏ các luật kết hợp không liên quan đến bệnh tim thu được kết quả như sau:

No_HadKidneyDisease & No_HadStroke & No_HadHeartAttack	0.87948
No_HadKidneyDisease & No_DifficultyWalking & No_HadHeartAttack	0.79048
No_HadDiabetes & No_HadSkinCancer & No_HadHeartAttack	0.73992
White only, Non-Hispanic & No_HadKidneyDisease & No_HadHeartAttack	0.63048
White only, Non-Hispanic & No_HadSkinCancer & No_HadHeartAttack	0.58516

Hình 3-75: Một số luật về bệnh tim

3.2.3.3. Đánh giá kết quả

Mặc dù mất nhiều thời gian trong việc chuyển bộ dữ liệu sang dạng List xong thuật toán Apriori sử dụng kỹ thuật "tỉa" dựa trên độ tin cậy để loại bỏ các tập ứng viên giúp giảm đáng kể số lượng các tập ứng viên cần kiểm tra, từ đó cải thiện hiệu quả của thuật toán. Đồng thời sử dụng apriori mang đến nhiều kết quả có ý nghĩa tốt hơn.

Một trong những kết quả tạo thành từ thuật toán Apriori:

```
==============================
Rule: Male + Yes_HadStroke -> Yes_HadHeartAttack
Support: 0.0065375
Confidence: 0.11516272515083455
Lift: 5.606071566305685
==============================
```

Hình 3-76: Ví dụ về luật tạo ra từ thuật toán Apriori

Thuật toán Eclat có thể tạo ra một số lượng lớn các tập mẫu tiềm năng. Việc cần phải kiểm tra tất cả các tập mẫu tiềm năng này để tìm các tập mẫu có tần suất xuất hiện cao trong trường hợp này tốn kém về mặt thời gian và quan trọng hơn là cần một dung lượng Ram lưu trữ lớn để chạy thuật toán. Do đó, khi chạy tới 400000 bản ghi xuất hiện tình trạng tràn Ram và buộc phải giảm số bản ghi có thể chạy xuống còn 50000 tuy nhiên vẫn mất tới 1 tiếng để chạy ra kết quả.

```
# Lấy các hàng dữ liệu, bỏ qua header
data = df.iloc[0:50000].values

# Xóa header khỏi DataFrame
df = pd.DataFrame(data, columns=None)
df
```

Hình 3-77: Lấy 50000 bản ghi từ bộ dữ liệu

```
✓ 4m [14] from pyECLAT import ECLAT
         eclat = ECLAT(data=df, verbose = True)
         #verbose=True -> hiển thị thanh loading

         100%|██████████| 2548/2548 [04:00<00:00, 10.57it/s]
         100%|██████████| 2548/2548 [00:11<00:00, 217.63it/s]
         100%|██████████| 2548/2548 [00:01<00:00, 2135.61it/s]
```

Hình 3-78: Thời gian để chạy thuật toán Eclat với 50000 bản ghi

Vì giảm xuống còn 50000 bản ghi nên độ lớn của mẫu khảo sát không đủ lớn và đa dạng, 50000 bản ghi đầu tập trung vào nhóm đối tượng không mắc bệnh tim nên kết quả luật kết hợp ra không đúng với mong muốn tìm những trường hợp phổ biến dẫn tới bệnh tim.

```
No_HadKidneyDisease & No_DifficultyWalking & No_HadHeartAttack          0.79048
```

Hình 3-79: Một trong những kết quả tạo thành từ thuật toán ECLAT

Kết quả của việc chạy hai thuật toán cho thấy rằng thuật toán Apriori có thể là một lựa chọn tốt hơn cho việc dự báo bệnh tim. Thuật toán này có thể tạo ra nhiều luật dự báo phù hợp hơn, và chạy ra ít thời gian hơn.

3.3. Đánh giá chung về các mô hình

Sau khi tiến hành triển khai các kỹ thuật khai phá dữ liệu dựa trên bộ dữ liệu sưu tầm được, nhóm chúng em xin đưa ra một số nhận định như sau:

Thứ nhất, kỹ thuật phân lớp bằng thuật toán Rừng ngẫu nhiên (Random Forest) và Cây quyết định (Decision Tree) hoạt động tốt trên bộ dữ liệu, do bộ dữ liệu có nhiều thuộc tính định danh nên việc phân lớp sẽ trở nên dễ dàng hơn.

Thứ hai, kỹ thuật khai phá luật kết hợp với thuật toán Apriori mang lại hiệu quả tốt trong việc đưa ra một số nhóm triệu chứng phổ biến của bệnh tim, do kỹ thuật này hướng tới việc tìm ra những tập mục phổ biến. Kỹ thuật này cũng hoạt động tốt đối với bộ dữ liệu có những thuật toán định danh.

Thứ ba, kỹ thuật phân cụm hướng tới tìm ra những nhóm bệnh nhân mà giá trị của các thuộc tính (hay đặc điểm của các triệu chứng) thường đi liền với nhau. Tuy nhiên, do đặc điểm của bộ dữ liệu không có nhiều thang đo khoảng, chủ yếu là thang đo định danh nên kỹ thuật này không hoạt động tốt và không mang lại ý nghĩa.

Nhìn chung, với bộ dữ liệu của CDC Hoa Kỳ, để dự đoán nguy cơ mắc bệnh tim của người dân, ta có thể sử dụng kỹ thuật phân lớp để dự đoán nguy cơ mắc bệnh và sử dụng kỹ thuật khai phá luật kết hợp để tìm ra những nhóm triệu chứng phổ biến của những người mắc bệnh tim. Kỹ thuật phân cụm dữ liệu với thuật toán K-Means có thể được sử dụng để phân nhóm các bệnh nhân mắc bệnh để có phương pháp điều trị phù hợp cho từng nhóm bệnh nhân.

CHƯƠNG 4: MỘT SỐ ĐỀ XUẤT

4.1. Ý tưởng cải tiến

4.1.1. Cải tiến hiệu suất Thuật toán phân lớp

Một số cách có thể cải tiến hiệu suất:

+ Kỹ thuật xử lý đặc trưng: Tạo ra thông tin mới hoặc tăng cường thông tin từ dữ liệu hiện có bằng cách sử dụng bộ dữ liệu lớn hơn với nhiều đặc trưng hơn. Hoặc có thể kết hợp để tạo ra các đặc trưng khác. Ví dụ như kết hợp GeneralHealth và PhysicalHealthDays.
+ Đối với mô hình random forest: có thể tối ưu hóa tham số mô hình:Thử nghiệm và tối ưu hóa các tham số của mô hình, như số lượng cây (n_estimators), độ sâu tối đa của cây (max_depth), số mẫu tối thiểu ở mỗi lá (min_samples_leaf).
+ Thử nghiệm nhiều Mô hình: Thử nghiệm với các mô hình khác nhau để xem mô hình nào phù hợp tốt nhất với dữ liệu. Cụ thể, thử nghiệm các mô hình khác như Gradient Boosting, Support Vector Machines, hoặc mô hình học máy sâu (deep learning).

4.1.2. Cải tiến kỹ thuật phân cụm

Như đã trình bày trong phần đánh giá kỹ thuật phân cụm, kỹ thuật này sẽ hoạt động tốt và mang lại nhiều ý nghĩa trên những bộ dữ liệu có giá trị của các thuộc tính được phân bố trên một khoảng giá trị đủ lớn và đặc biệt là với dữ liệu định lượng với miền giá trị liên tục. Do bộ dữ liệu của CDC Hoa Kỳ được sử dụng trong bài chỉ có thuộc tính *'BMI'* có miền giá trị liên tục nên việc phân cụm dựa trên thuộc tính *'BMI'* và các thuộc tính khác trở nên khó khăn, các phần tử trong cụm có những tính chất không giống nhau.

Vì thế, việc khảo sát sức khoẻ của người dân nên có thêm một số thuộc tính mà giá trị của nó là những giá trị định lượng và có miền giá trị đủ lớn. Chẳng hạn, có thể đưa ra thang đo về mức cholesterol trong máu, do đây cũng là một yếu tố quan trọng, ảnh hưởng đến nguy cơ mắc bệnh tim mạch của người dân (Báo Sức khoẻ và Đời sống, 2021).

4.1.3. Cải tiến thời gian chạy của kỹ thuật khai phá luật kết hợp

Ý tưởng nhóm chúng em đưa ra là chuyển các bản ghi thành dạng List trong khai phá luật kết hợp. Vì bộ dữ liệu có kích thước lớn (hơn 400000 bản ghi và 18 cột) liên quan nên việc áp dụng lệnh for lặp lại nhiều lần liên tục và lệnh Append để thêm từng bản ghi vào mảng Record đã được khởi tạo là không hiệu quả về mặt thời gian.

Kết quả dùng vòng lặp For kết hợp với lệnh Append mất tới 6 phút chỉ để load 100 dòng dữ liệu:

```
6m
#Thư viện Apriori ycau tập dữ liệu phải ở dạng List, trong đó toàn bộ tập dữ liệu là một danh
#Chuyển dữ liệu từ dạng bảng -> dạng Danh sách
records = []
for i in range(1,100):
    records.append([str(df.values[i,j]) for j in range(0,18)]) # duyệt qua 18 cột
    #append -> đưa dữ liệu vào list
```

Hình 4-1: Vòng lặp For kết hợp với lệnh Append trong khai phá luật kết hợp

Sau khi cải tiến với việc sử dụng phương thức to_numpy() của đối tượng df để chuyển đổi tập dữ liệu thành một mảng NumPy rồi phương thức reshape() của đối tượng mảng NumPy để định dạng lại mảng thành một mảng 2 chiều với 400.000 hàng và 18 cột. Cuối cùng lại dùng phương thức tolist() của đối tượng mảng NumPy để chuyển đổi mảng thành một danh sách tốc độ được cải thiện đáng kể

Kết quả khi sử dụng nhóm lệnh Numpy chỉ mất chưa tới 1s để load 100 bản ghi:

```
0s
[14] #CÁCH 1: numpy


     # # Khai báo tập dữ liệu
     # df = np.loadtxt("data.csv", dtype=np.str, delimiter=",")

     # Chuyển dữ liệu từ dạng bảng -> dạng Danh sách
     records = df.head(100).to_numpy().reshape(-1, 18).tolist()
```

Hình 4-2: Sử dụng nhóm lệnh Numpy trong khai phá luật kết hợp

Ngay cả khi chuyển đổi 400.000 bản ghi sang dạng List tốc độ vẫn rất nhanh:

```
0s
[12] #CÁCH 1: numpy
     # Khai báo tập dữ liệu
     # df = np.loadtxt("data.csv", dtype=np.str, delimiter=",")

     # Chuyển dữ liệu từ dạng bảng -> dạng Danh sách
     records = df.head(400000).to_numpy().reshape(-1, 18).tolist()
```

Hình 4-3: Sử dụng nhóm lệnh Numpy trong khai phá luật kết hợp cho 400.000 bản ghi

4.2. Hạn chế của đề tài

Đề tài tiến hành nghiên cứu trên bộ dữ liệu gồm 18 trường dữ liệu, trong đó có 01 trường dữ liệu là kết quả có hay không mắc bệnh tim. Như vậy, 17 trường dữ liệu còn lại là các thói quen sinh hoạt, triệu chứng của người mắc bệnh. Trong khi đó, còn rất nhiều yếu tố khác có tác động dẫn tới khả năng mắc bệnh tim của người dân thì chưa được bộ dữ liệu đề cập tới. Chẳng hạn như lượng đường trong máu, cholesterol trong máu, điều kiện làm việc, … cũng là những yếu tố tác động rất lớn đến sức khoẻ của con người.

Ngoài ra, những kiến thức và kỹ thuật áp dụng trong bài làm còn khá đơn giản, chưa có nhiều điều chỉnh về thống số các mô hình. Vậy nên, việc tìm ra các quy luật, khám phá ra tri thức trong bộ dữ liệu còn khá hạn chế về độ chính xác và chưa có khả năng ứng dụng trong thực tiễn.

4.3. Hướng phát triển của đề tài

Như đã đề cập ở Chương I, đề tài được nghiên cứu lấy bộ dữ liệu với bối cảnh thời gian là năm 2022, không gian địa lý là nước Mỹ. Để ứng dụng được kết quả nghiên cứu vào thực tiễn, cần xác định thời gian, không gian phù hợp, thu thập dữ liệu phù hợp với thời gian, không gian đó. Cụ thể, cần khảo sát ở nhiều khu vực với điều kiện địa lý, khí hậu khác nhau; khảo sát ở nhiều quốc gia với mức sống, thói quen làm việc và sinh hoạt khác nhau. Từ đó mới có thể đưa ra được những triệu chứng chung nhất về khả năng mắc bệnh tim mạch. Ngoài ra, bộ dữ liệu cần phong phú về các trường, cũng có thể lược bớt ở một mức độ nhất định nhằm phù hợp với yêu cầu ứng dụng nghiên cứu cụ thể.

Vì thế, bài làm của nhóm có thể là nguồn tham khảo để phát triển thêm những mô hình dự báo về khả năng mắc bệnh tim mạch. Nếu phát triển tại Việt Nam, cần có bộ dữ liệu về tình hình mắc bệnh tim mạch của người dân Việt Nam, với những đặc điểm khác biệt so với những người dân Mỹ được khảo sát trong bộ dữ liệu ví dụ trên.

KẾT LUẬN

Bệnh tim mạch đang ngày càng trở nên phổ biến trong cuộc sống hiện đại ngày nay, do điều kiện làm việc, môi trường sống nhiều áp lực. Con người cũng có xu hướng sử dụng nhiều chất kích thích, ăn những đồ ăn nhiều dầu mỡ, vốn là không tốt cho sức khoẻ tim mạch. Việc nghiên cứu ra tác động của các yếu tố này đến khả năng mắc bệnh tim mạch của người dân đang ngày càng trở nên cần thiết để giúp con người sớm phòng tránh căn bệnh nguy hiểm này.

Tuy nhiên, cần thu thập dữ liệu phong phú về cả thói quen sinh hoạt lẫn những triệu chứng về sức khoẻ thì từ đó mới có thể đưa ra kết luận chính xác về tình hình sức khoẻ của người dân. Việc chẩn đoán trên một số ít triệu chứng sẽ không thực sự chính xác và sẽ gây hoang mang cho những người không thực sự mắc bệnh nhưng được chẩn đoán sai. Ngoài ra, cần tinh chỉnh những mô hình khai phá để tương thích với từng bộ dữ liệu. Đây là yêu cầu bắt buộc do những bộ dữ liệu có tính chất không giống nhau và việc áp dụng máy móc các thông số trong nhiều trường hợp có thể dẫn tới sai lệch kết quả.

Có thể nói, nghiên cứu của nhóm xác định cụ thể các cơ sở lý thuyết, cũng như xây dựng các thuật toán với các bước hướng dẫn chi tiết, và kết quả dự đoán rõ ràng sẽ đem lại hiệu quả ứng dụng cao, cũng như có thể linh hoạt biến đổi dựa trên nhu cầu của nhà phân tích dữ liệu.

TÀI LIỆU THAM KHẢO

1. Báo Sức khoẻ và Đời sống. (2021). *Cholesterol cao: Thủ phạm nguy hiểm gây ra những bệnh lý về tim mạch*. https://suckhoedoisong.vn/cholesterol-cao-thu-pham-nguy-hiem-gay-ra-nhung-benh-ly-ve-tim-mach-169211008192919597.htm

2. Bệnh viện Đa khoa Vinmec. (n.d.). *Bệnh tim là gì? Nguyên nhân, triệu chứng và cách điều trị bệnh tim*. Retrieved December 27, 2023, from https://www.vinmec.com/vi/tim-mach/thong-tin-suc-khoe/nguyen-nhan-va-trieu-chung-cua--benh-tim-mach/

3. GeeksforGeeks. (2019a, June 6). Elbow Method for optimal value of k in KMeans. *GeeksforGeeks*. https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/

4. GeeksforGeeks. (2019b, June 11). ML | ECLAT Algorithm. *GeeksforGeeks*. https://www.geeksforgeeks.org/ml-eclat-algorithm/

5. Hà Quang Thuỵ. (2018). *Bài giảng Khai phá dữ liệu*.

6. Hoàng Huyền Trang. (2016). *Phương pháp phân cụm dựa trên tập thô và giải thuật di truyền*.

7. Hoàng Thị Thanh Hiền. (2016). *Ứng dụng khai phá dũ liệu để xây dựng hệ thống chẩn đoán bệnh trầm cảm cho học sinh phổ thông*.

8. Khoa Hệ thống thông tin quản lý, Học viện Ngân hàng. (2023). *Bài giảng Khai phá dữ liệu*.

9. Nguyễn Thị Việt Hà. (2020, March 20). *Tổng quan về khai phá dữ liệu và phương pháp khai phá*. Tạp chí Công Thương. https://tapchicongthuong.vn/bai-viet/tong-quan-ve-khai-pha-du-lieu-va-phuong-phap-khai-pha-luat-ket-hop-trong-co-so-du-lieu-69634.htm

10. Trần Đinh Toàn, & Dương Thị Mộng Thuỳ. (2022). *Ứng dụng kỹ thuật máy học vào phân loại bệnh tim*.

11. VIBLO. (2020, October 30). *Khai phá mẫu phổ biến và luật kết hợp*. https://viblo.asia/p/khai-pha-mau-pho-bien-va-luat-ket-hop-gGJ59QAa5X2