

**ỨNG DỤNG SPARK SQL VÀ SPARK MLLIB TRONG PHÂN TÍCH Y TẾ DỮ LIỆU LỚN: NGHIÊN CỨU CÁC YẾU TỐ TÁC ĐỘNG ĐẾN BỆNH TIM MẠCH TẠI TRUNG QUỐC**

Môn học: Dữ liệu lớn và ứng dụng

Lớp học phần: 25D1INF50907903

Giảng viên: TS.Võ Văn Hải

Nhóm sinh viên: 09

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Đóng góp** |
| 1 | Lưu Nhật Hà | 31221021261 | 100% |
| 2 | Trịnh Nguyễn Kim Như | 31221026907 | 100% |
| 3 | Ngô Kim Quý Phúc | 31221024285 | 100% |

**TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ**

**KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH**

**KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH**

**ỨNG DỤNG SPARK SQL, SPARK MLLIB TRONG PHÂN TÍCH Y TẾ DỮ LIỆU LỚN: NGHIÊN CỨU CÁC YẾU TỐ TÁC ĐỘNG ĐẾN BỆNH TIM MẠCH TẠI TRUNG QUỐC**

Môn học: Dữ liệu lớn và ứng dụng

Lớp học phần: 25D1INF50907903

Giảng viên: TS.Võ Văn Hải

*Thành phố Hồ Chí Minh, 31 tháng 03 năm 2025*

|  |  |  |
| --- | --- | --- |
|  | | **TÓM TẮT** |
| **Từ khóa**: Bệnh tim mạch, học máy, phân cụm K-means, Logistic Regression, SVM, Decision Tree, dữ liệu lớn. | Nghiên cứu ứng dụng học máy và xử lý dữ liệu lớn bằng Spark nhằm phân tích các yếu tố nguy cơ liên quan đến bệnh tim mạch tại Trung Quốc. Tập dữ liệu gồm 239.266 đối tượng tham gia được phân tích dựa trên các nhóm đặc điểm như nhân khẩu học, hành vi sức khỏe, bệnh lý nền, môi trường sống và tiếp cận y tế.  Kết quả cho thấy các yếu tố như tuổi cao, khu vực sinh sống nông thôn, căng thẳng, hút thuốc, sử dụng rượu bia và các bệnh nền (tăng huyết áp, tiểu đường, bệnh thận mạn tính) có liên quan mật thiết đến nguy cơ mắc bệnh tim. Trong khi đó, giới tính, tiền sử gia đình và chế độ ăn không cho thấy ảnh hưởng rõ rệt. Môi trường sống ô nhiễm và hạn chế tiếp cận dịch vụ y tế cũng là yếu tố đáng lưu ý.  Về mặt kỹ thuật, nghiên cứu sử dụng mô hình K-means để phân cụm bệnh nhân thành hai nhóm sức khỏe khác biệt. Đồng thời, mô hình Logistic Regression được áp dụng để dự đoán nguy cơ mắc bệnh tim, cho kết quả ở mức khá với độ nhạy cao.  Trên cơ sở kết quả thu được, nghiên cứu đề xuất ưu tiên can thiệp vào nhóm dân số dễ tổn thương như người lớn tuổi, người có bệnh nền, cư dân nông thôn và người gặp khó khăn trong tiếp cận y tế. Việc ứng dụng dữ liệu lớn trong sàng lọc và dự phòng có thể giúp nâng cao hiệu quả và công bằng trong chăm sóc sức khỏe cộng đồng. | |

**MỤC LỤC**

[**TÓM TẮT** i](#_Toc199627050)

[**DANH SÁCH BẢNG** iv](#_Toc199627051)

[**DANH SÁCH HÌNH ẢNH** iv](#_Toc199627052)

[**DANH SÁCH BIỂU ĐỒ** iv](#_Toc199627053)

[**CHƯƠNG I: TỔNG QUAN ĐỀ TÀI** 1](#_Toc199627054)

[**1.1. Lý do chọn đề tài** 1](#_Toc199627055)

[**1.2. Mục tiêu và câu hỏi nghiên cứu** 1](#_Toc199627056)

[***1.2.1. Mục tiêu nghiên cứu*** 1](#_Toc199627057)

[***1.2.2. Câu hỏi nghiên cứu*** 2](#_Toc199627058)

[**1.3. Đối tượng và phạm vi nghiên cứu** 2](#_Toc199627059)

[***1.3.1. Đối tượng nghiên cứu*** 2](#_Toc199627060)

[***1.3.2. Phạm vi nghiên cứu*** 2](#_Toc199627061)

[**1.4. Phương pháp thực hiện** 3](#_Toc199627062)

[**1.5. Bố cục bài nghiên cứu** 3](#_Toc199627063)

[**CHƯƠNG II: CƠ SỞ LÝ THUYẾT** 4](#_Toc199627064)

[**2.1. Giới thiệu Big Data** 4](#_Toc199627065)

[**2.2. Các thư viện sử dụng trong Python** 5](#_Toc199627066)

[***2.2.1. Thư viện Matplotlib*** 5](#_Toc199627067)

[***2.2.2. Thư viện Seaborn*** 5](#_Toc199627068)

[***2.2.3. Thư viện Pandas*** 6](#_Toc199627069)

[***2.2.4. Thư viện PySpark*** 6](#_Toc199627070)

[***2.2.5. Thư viện Numpy*** 6](#_Toc199627071)

[***2.2.6. Thư viện học máy (MLlib)*** 6](#_Toc199627072)

[**2.3. Mô hình sử dụng** 7](#_Toc199627073)

[***2.3.1. Mô hình học có giám sát*** 7](#_Toc199627074)

[***2.3.2. Mô hình học không giám sát*** 9](#_Toc199627075)

[**2.4. Phương pháp đánh giá** 10](#_Toc199627076)

[***2.4.1. Accuracy*** 10](#_Toc199627077)

[***2.4.2. Precision*** 11](#_Toc199627078)

[***2.4.3. Recall*** 11](#_Toc199627079)

[***2.4.4. F-score*** 11](#_Toc199627080)

[**CHƯƠNG III. PHƯƠNG PHÁP CẤU HÌNH, KHÁM PHÁ VÀ XỬ LÝ DỮ LIỆU** 12](#_Toc199627081)

[**3.1. Cài đặt và cấu hình Apache Hadoop** 12](#_Toc199627082)

[**3.2. Cài đặt, cấu hình Apache Spark** 15](#_Toc199627083)

[**3.3. Tổng quan bộ dữ liệu** 17](#_Toc199627084)

[***3.1.1. Nguồn dữ liệu*** 17](#_Toc199627085)

[***3.1.2. Mô tả các biến*** 17](#_Toc199627086)

[***3.1.3. Phân tích khám phá dữ liệu*** 19](#_Toc199627087)

[**3.4. Các bước thực hiện truy vấn sử dụng SQL SPARK** 20](#_Toc199627088)

[***3.4.1. Khai báo các thư viện cần sử dụng*** 20](#_Toc199627089)

[***3.4.2. Đọc dữ liệu*** 20](#_Toc199627090)

[***3.4.3. Khám phá dữ liệu*** 20](#_Toc199627091)

[***3.4.4. Thực hiện các truy vấn dữ liệu*** 20](#_Toc199627092)

[**3.5. Các bước thực hiện mô hình học máy (giám sát và không giám sát) sử dụng SQL MLlib** 31](#_Toc199627093)

[***3.5.1. Phân cụm*** 31](#_Toc199627094)

[***3.5.2. Phân lớp*** 34](#_Toc199627095)

[**CHƯƠNG IV. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ** 44](#_Toc199627096)

[**4.1. Ứng dụng Spark SQL trong phân tích nguy cơ mắc bệnh tim mạch** 44](#_Toc199627097)

[***4.1.1. Số lượng bệnh nhân mắc bệnh tim phân theo nhóm thông tin và nhân khẩu học*** 44](#_Toc199627098)

[***4.1.2. Số lượng bệnh nhân mắc bệnh tim phân theo mức độ stress và thu nhập*** 45](#_Toc199627099)

[***4.1.3. Tỷ lệ bệnh nhân mắc bệnh tim theo nhóm hành vi sức khỏe và lối sống*** 45](#_Toc199627100)

[***4.1.4 Tỷ lệ bệnh nhân mắc bệnh tim theo nhóm yếu tố bệnh lý & lâm sàng*** 46](#_Toc199627101)

[***4.1.5. Tỷ lệ mắc bệnh tim mạch theo yếu tố bệnh lý và lâm sàng với những người chưa bị bệnh tim trước đó*** 48](#_Toc199627102)

[***4.1.6. Số bệnh nhân mắc bệnh tim do di truyền từ cha mẹ và mức độ rủi ro hiện tại*** 50](#_Toc199627103)

[***4.1.7. Nguy cơ mắc bệnh tim theo yếu tố chất lượng không khí*** 51](#_Toc199627104)

[***4.1.8. Mối quan hệ giữa khả năng tiếp nhận y tế và mức độ phân bổ bệnh viện với tỉ lệ mắc bệnh về tim*** 52](#_Toc199627105)

[***4.1.9. Mối tương quan giữa việc sử dụng phương pháp đông y với tỷ lệ mắc bệnh về tim*** 53](#_Toc199627106)

[***4.1.10. Nguy cơ tái phát bệnh tim của những người khảo sát có thói quen và hành vi sống khác nhau*** 54](#_Toc199627107)

[**4.2. Ứng dụng Spark MLlib cho phân tích học máy** 55](#_Toc199627108)

[**4.2.1. Phân cụm** 55](#_Toc199627109)

[**4.2.2. Phân lớp** 57](#_Toc199627110)

[**CHƯƠNG V: KẾT LUẬN** 61](#_Toc199627111)

[**5.1. Thảo luận** 61](#_Toc199627112)

[**5.2. Hạn chế** 62](#_Toc199627113)

[**5.3. Hướng phát triển** 62](#_Toc199627114)

[**TÀI LIỆU THAM KHẢO** 64](#_Toc199627115)

[**PHỤ LỤC** 65](#_Toc199627116)

**DANH SÁCH BẢNG**

[Bảng 3.1: Kết quả thống kê mô tả của các biến định lượng 19](#_Toc199584390)

[Bảng 4.1: Tỷ lệ bệnh nhân mắc bệnh tim với các yếu tố tình trạng hút thuốc, tiêu thụ rượu bia, hoạt động thể chất, điểm đánh giá chế độ ăn và mức độ căng thẳng 21](#_Toc199584391)

[Bảng 4.2: Tỷ lệ bệnh nhân mắc bệnh tim ở nhóm có ít nhất một yếu tố bệnh lý và lâm sàng với những người chưa bị bệnh tim trước đó 25](#_Toc199584392)

[Bảng 4.3: Chỉ số đánh giá hiệu suất mô hình Logistic Regression, Decision Tree và SVM 33](#_Toc199584393)

**DANH SÁCH HÌNH ẢNH**

[Hình 2.1: Đồ thị hàm sigmoid 12](#_Toc199543748)

[Hình 3.1: Cấu hình biến môi trường JAVA\_HOME trong Windows 16](#_Toc199543749)

[Hình 3.2: Cấu hình Path cho JAVA\_HOME 16](#_Toc199543750)

[Hình 3.3: Kiểm tra phiên bản Java trong Command Prompt 17](#_Toc199543751)

[Hình 3.4: Cấu hình biến môi trường HADOOP\_HOME trong Windows 17](#_Toc199543752)

[Hình 3.5: Cấu hình Path cho HADOOP\_HOME 18](#_Toc199543753)

[Hình 3.6: Kiểm tra phiên bản Hadoop trong Command Prompt 18](#_Toc199543754)

[Hình 3.7: Kiểm tra tiến trình Hadoop bằng lệnh jps trong Command Prompt 19](#_Toc199543755)

[Hình 3.8: Cấu hình biến môi trường SPARK\_HOME trong Windows 19](#_Toc199543756)

[Hình 3.9: Cấu hình Path cho HADOOP\_HOME 20](#_Toc199543757)

[Hình 3.10: Kiểm tra phiên bản Spark trong Command Prompt 20](#_Toc199543758)

[Hình 4.1: Bảng phân loại huyết áp chuẩn 27](#_Toc199543759)

**DANH SÁCH BIỂU ĐỒ**

[Biểu đồ 4.1: Số lượng người mắc bệnh tim phân theo nhóm tuổi, khu vực sống và giới tính 23](#_Toc199543927)

[Biểu đồ 4.2: Số lượng bệnh nhân mắc bệnh tim theo mức độ stress và thu nhập 24](#_Toc199543928)

[Biểu đồ 4.3: Mối quan hệ giữa Blood Pressure và Cholesterol Level phân theo tỷ lệ mắc bệnh tim 26](#_Toc199543929)

[Biểu đồ 4.4: Tỷ lệ mắc bệnh tim ở nhóm yếu tố bệnh lý và lâm sàng với những người chưa bị bệnh tim trước đó 27](#_Toc199543930)

[Biểu đồ 4.5: Mối quan hệ giữa tiền sử gia đình mắc bệnh tim và mức độ rủi ro hiện tại với số người mắc bệnh về tim 29](#_Toc199543931)

[Biểu đồ 4.6: Nguy cơ mắc bệnh tim theo yếu tố chất lượng không khí 30](#_Toc199543932)

[Biểu đồ 4.7: Mối quan hệ giữa khả năng tiếp nhận y tế và mức độ phân bổ bệnh viện với tỉ lệ mắc bệnh về tim 31](#_Toc199543933)

[Biểu đồ 4.8: Tỷ lệ mắc bệnh tim theo đặc điểm sử dụng phương pháp đông y 32](#_Toc199543934)

[Biểu đồ 4.9: Tỷ lệ tái phát bệnh tim qua sự khác biệt về thói quen sống 33](#_Toc199543935)

[Biểu đồ 4.10: Chỉ số Silhoutte theo số cụm 35](#_Toc199543936)

[Biểu đồ 4.11: Đường cong ROC ở mô hình Logistic Regression 36](#_Toc199543937)

[Biểu đồ 4.12: Các chỉ số đánh giá hiệu suất ở mô hình Logistic Regression 37](#_Toc199543938)

[Biểu đồ 4.13: Đường cong độ chính xác – độ nhạy ở mô hình Logistic Regression 38](#_Toc199543939)

[Biểu đồ 4.14: Ma trận nhầm lẫn ở mô hình Logistic Regression 39](#_Toc199543940)

**CHƯƠNG I: TỔNG QUAN ĐỀ TÀI**

**1.1. Lý do chọn đề tài**

Trong những năm gần đây, bệnh tim mạch (Cardiovascular Disease – CVD) đã trở thành nguyên nhân gây tử vong hàng đầu trên toàn cầu. Mỗi năm có khoảng 17,9 triệu người chết vì các bệnh tim mạch, chiếm gần 32% tổng số ca tử vong toàn cầu, trong đó khoảng 75% các ca tử vong xảy ra tại các quốc gia có thu nhập thấp và trung bình ([WHO, 2021](#WHO)). Tại Trung Quốc – quốc gia đông dân nhất thế giới với hơn 1,4 tỷ người, gánh nặng bệnh tim mạch ngày càng gia tăng nhanh chóng. Theo ước tính từ [Trung tâm Kiểm soát và Phòng ngừa Dịch bệnh Trung Quốc (2022),](#china) tỷ lệ mắc bệnh tim mạch ở người trưởng thành đã tăng gần 20% trong vòng 10 năm qua do nhiều nguyên nhân khác nhau. Việc khai thác và phân tích dữ liệu bệnh tim tại Trung Quốc không chỉ giúp nhận diện các yếu tố nguy cơ nội tại trong cộng đồng dân cư mà còn cung cấp những hiểu biết quan trọng để xây dựng các chiến lược phòng ngừa và quản lý hiệu quả bệnh tim mạch, đặc biệt tại các quốc gia đang phát triển có những đặc điểm dịch tễ và thói quen sinh hoạt tương đồng ([Zhou và cộng sự, 2019](#ZHOU)).

Vì vậy, nhóm tác giả quyết định chọn đề tài “*Ứng dụng Spark SQL và Spark MLlib trong phân tích y tế dữ liệu lớn: Nghiên cứu các yếu tố ảnh hưởng đến bệnh tim mạch tại Trung Quốc*”. Đề tài này sử dụng bộ dữ liệu bệnh tim thu thập từ Trung Quốc – một trong những kho dữ liệu hiếm hoi chứa đầy đủ thông tin đa chiều về nhân khẩu học, yếu tố sinh lý và lâm sàng, thói quen, hành vi sống và điều kiện tiếp cận y tế, môi trường của người tham gia khảo sát. Việc lựa chọn Trung Quốc làm phạm vi nghiên cứu là có chủ đích, bởi quốc gia này đang đứng trước bài toán chuyển đổi hệ thống y tế từ mô hình “chữa bệnh” sang “phòng bệnh” ([Ủy ban Y tế Quốc gia Trung Quốc, 2023](#ỦYBAN)). Với quy mô dân số lớn cùng sự chênh lệch đáng kể giữa khu vực nông thôn và thành thị, bộ dữ liệu này phản ánh rõ thực trạng bất bình đẳng trong chăm sóc sức khỏe cũng như tác động của các yếu tố xã hội lên nguy cơ bệnh tim mạch. Ngoài ra, việc áp dụng nền tảng Apache Spark với truy vấn SQL cho phép xử lý và phân tích dữ liệu quy mô lớn một cách nhanh chóng và hiệu quả. Đây là xu thế tất yếu trong ứng dụng khoa học dữ liệu y tế cộng đồng hiện đại, giúp phát hiện sớm các nhóm dân cư có nguy cơ cao, từ đó hỗ trợ các cơ quan y tế và nhà hoạch định chính sách đưa ra các quyết định dựa trên bằng chứng khoa học ([Apache Spark Documentation, 2024](#APACHE2024)).

Tóm lại, đề tài này không chỉ mang giá trị học thuật trong lĩnh vực kỹ thuật xử lý dữ liệu lớn với Spark SQL, mà còn có ý nghĩa thực tiễn quan trọng trong việc nâng cao nhận thức và phát triển các chiến lược can thiệp dựa trên dữ liệu tại Trung Quốc, với tiềm năng mở rộng sang các quốc gia có bối cảnh tương đồng như Việt Nam.

**1.2. Mục tiêu và câu hỏi nghiên cứu**

***1.2.1. Mục tiêu nghiên cứu***

Đề tài nhằm khai thác và phân tích dữ liệu bệnh tim mạch tại Trung Quốc để hiểu rõ hơn về sự phân bố và các yếu tố nguy cơ liên quan đến bệnh tim trong cộng đồng dân cư. Trọng tâm nghiên cứu tập trung vào việc đánh giá ảnh hưởng của đặc điểm nhân khẩu học, các yếu tố sinh lý, lâm sàng và hành vi sống, cũng như điều kiện tiếp cận y tế đối với nguy cơ mắc bệnh tim. Đồng thời, nghiên cứu cũng hướng đến việc phân tích tác động của tiền sử gia đình và yếu tố di truyền, đặc biệt trong bối cảnh sự khác biệt về điều kiện kinh tế - xã hội và môi trường sống giữa các vùng nông thôn và thành thị. Bằng cách ứng dụng công nghệ xử lý dữ liệu lớn với nền tảng Apache Spark và SQL, đề tài mong muốn xây dựng các mô hình phân cụm, phân nhóm để xác định những đặc điểm của các nhóm mắc bệnh và không mắc bệnh về tim mạch, từ đó góp phần nâng cao hiệu quả công tác phòng ngừa và quản lý bệnh tim mạch. Kết quả nghiên cứu không chỉ mang ý nghĩa thực tiễn đối với hệ thống y tế Trung Quốc mà còn có thể được áp dụng mở rộng cho các quốc gia có bối cảnh tương tự trong khu vực và trên thế giới.

***1.2.2. Câu hỏi nghiên cứu***

1. Những yếu tố nhân khẩu học nào (tuổi, giới tính, khu vực sinh sống) có ảnh hưởng rõ rệt đến nguy cơ mắc bệnh tim mạch?

2. Mức độ căng thẳng, hành vi hút thuốc, sử dụng rượu bia, chế độ ăn uống và vận động thể chất ảnh hưởng như thế nào đến tỷ lệ mắc và tái phát bệnh tim?

3. Tác động của các bệnh lý nền như tăng huyết áp, tiểu đường, béo phì và bệnh thận mạn tính đến nguy cơ mắc bệnh tim là gì?

4. Tiền sử gia đình và cá nhân có ảnh hưởng như thế nào đến điểm rủi ro tim mạch (CVD Risk Score)?

5. Mức độ ô nhiễm không khí ảnh hưởng như thế nào đến tỷ lệ mắc bệnh tim mạch, và liệu yếu tố môi trường này có thể được xem là một yếu tố nguy cơ độc lập hay không?

6. Có thể chia nhóm bệnh nhân tim mạch thành các cụm (clusters) có đặc điểm sức khỏe tương đồng hay không, và các cụm đó phản ánh điều gì về rủi ro sức khỏe?

7. Mô hình học máy như Logistic Regression, Decision Tree và SVM có thể dự đoán chính xác khả năng mắc bệnh tim ở bệnh nhân dựa trên các đặc trưng sức khỏe hay không? Mô hình nào là lựa chọn tối ưu nhất?

**1.3. Đối tượng và phạm vi nghiên cứu**

***1.3.1. Đối tượng nghiên cứu***

Nghiên cứu được tiến hành trên 239.266 người dân đang sinh sống tại Trung Quốc, nhằm khảo sát nguy cơ mắc bệnh tim mạch dựa trên các yếu tố nhân khẩu học, thói quen sinh hoạt, môi trường sống cũng như tiền sử bệnh lý liên quan đến tim mạch.

***1.3.2. Phạm vi nghiên cứu***

Dữ liệu được sử dụng là dữ liệu giả lập, được xây dựng từ nhiều nhóm thuộc tính khác nhau nhằm mô phỏng các yếu tố có thể liên quan đến nguy cơ đau tim. Cụ thể, dữ liệu bao gồm các nhóm như nhóm thông tin cá nhân và nhân khẩu học, nhóm hành vi sức khỏe và lối sống, nhóm yếu tố bệnh lý và lâm sàng, nhóm yếu tố di truyền và môi trường, nhóm tiếp cận y tế và chăm sóc sức khỏe, nhóm kinh tế xã hội. Biến mục tiêu trong phân tích là Heart\_Attack.

Việc xử lý và truy vấn dữ liệu được thực hiện thông qua Spark SQL, giúp tăng hiệu quả trong việc thao tác với dữ liệu lớn. Cần lưu ý rằng tất cả các phân tích trong tiểu luận này mang tính chất mô phỏng, hoàn toàn không dựa trên dữ liệu thực tế từ bệnh viện hay tổ chức y tế cụ thể, và không phản ánh tình trạng sức khỏe hay nguy cơ bệnh lý của bất kỳ cá nhân nào.

**1.4. Phương pháp thực hiện**

Quá trình chuẩn bị dữ liệu bao gồm các bước làm sạch, kiểm tra dữ liệu thiếu và chuẩn hóa các cột dạng phân loại nhằm đảm bảo tính nhất quán và độ tin cậy của dữ liệu sử dụng trong phân tích.

Các truy vấn được thực hiện dựa trên các câu hỏi nghiên cứu và các thuộc tính có trong bộ dữ liệu, nhằm khai thác thông tin, đồng thời rút ra những ý nghĩa thực tiễn phục vụ cho việc phân tích sâu và nghiên cứu chuyên sâu.

Phân tích kết quả bao gồm thống kê mô tả, phân tích mối quan hệ giữa các biến, xếp hạng và thực hiện dự đoán sơ bộ để hỗ trợ quá trình ra quyết định.

Các công cụ chính được sử dụng trong nghiên cứu gồm Apache Spark, PySpark SQL, PyCharm và Jupyter Notebook, nhằm đảm bảo tính linh hoạt và khả năng mở rộng trong xử lý và trực quan hóa dữ liệu.

**1.5. Bố cục bài nghiên cứu**

Nghiên cứu được cấu trúc thành năm chương. Chương 1 cung cấp tổng quan về đề tài, giới thiệu bối cảnh và làm nổi bật tầm quan trọng của nghiên cứu. Chương 2 trình bày cơ sở lý thuyết, thiết lập nền tảng vững chắc cho các phân tích và phát triển tiếp theo. Tiếp theo, chương 3 mô tả chi tiết phương pháp cấu hình và tiền xử lý dữ liệu, bao gồm các quy trình cấu hình Apache Hadoop và Apache Spark, cũng như các bước chuẩn bị, làm sạch và xử lý dữ liệu phù hợp với mục tiêu nghiên cứu. Chương 4 tập trung vào việc triển khai giải pháp và đánh giá kết quả nhằm xác thực hiệu quả của phương pháp đã áp dụng. Cuối cùng, chương 5 tổng kết những đóng góp chính của nghiên cứu và đề xuất các hướng phát triển tiếp theo dựa trên kết quả đạt được.

**CHƯƠNG II: CƠ SỞ LÝ THUYẾT**

**2.1. Giới thiệu Big Data**

***2.1.1. Khái niệm Big Data***

Big Data là một khái niệm được sử dụng để mô tả các tập dữ liệu có kích thước rất lớn, tốc độ tạo ra nhanh và có tính đa dạng cao về định dạng, nguồn gốc và cấu trúc. Các đặc điểm này vượt quá khả năng lưu trữ, quản lý và xử lý của các công nghệ cơ sở dữ liệu truyền thống. Theo [De Mauro và cộng sự (2016)](#DEMAURO), Big Data là “tài sản thông tin được đặc trưng bởi khối lượng lớn, tốc độ cao và đa dạng, đòi hỏi các công nghệ và phương pháp phân tích cụ thể để chuyển đổi thành giá trị”. Định nghĩa này phản ánh bản chất tổng hợp của Big Data không chỉ là dữ liệu lớn về số lượng mà còn là thách thức trong việc phân tích và khai thác dữ liệu hiệu quả.

Mô hình phổ biến nhất để mô tả đặc điểm của Big Data là “3V”: Volume (khối lượng), Velocity (tốc độ) và Variety (đa dạng). Tuy nhiên, các nhà nghiên cứu hiện đại đã mở rộng mô hình này thành “5V” và thậm chí là “7V”, bổ sung các yếu tố như Veracity (độ tin cậy của dữ liệu), Value (giá trị tiềm năng), Variability (mức độ biến thiên),\ và Visualization (trực quan hóa dữ liệu) ([Gandomi và Haider, 2015)](#GANDOMI). Sự mở rộng này phản ánh thực tế rằng không phải mọi dữ liệu lớn đều có giá trị, và thách thức lớn của Big Data không chỉ là lưu trữ mà còn là khả năng xử lý, phân tích và đưa ra quyết định đúng đắn từ dữ liệu đó.

Tuy nhiên, khái niệm Big Data vẫn còn là chủ đề gây tranh luận trong cộng đồng học thuật. Một số học giả tập trung vào khía cạnh công nghệ, cho rằng Big Data là bất kỳ dạng dữ liệu nào vượt quá khả năng xử lý của các hệ thống cơ sở dữ liệu truyền thống, đòi hỏi phải có các phương pháp tiếp cận mới như phân tán dữ liệu, xử lý song song, hoặc trí tuệ nhân tạo [(Ward và Barker, 2013).](#WARD) Những người khác lại nhấn mạnh vào giá trị tiềm năng của dữ liệu trong việc hỗ trợ ra quyết định, đổi mới công nghệ và cải tiến quy trình kinh doanh.

Từ góc nhìn liên ngành, Big Data không chỉ là một công nghệ, mà còn là một hiện tượng xã hội và kinh tế. Nó phản ánh sự chuyển đổi từ phân tích dữ liệu truyền thống sang các mô hình phân tích dữ liệu tiên tiến, nơi mà khả năng xử lý dữ liệu khổng lồ có thể tạo ra lợi thế cạnh tranh, mở rộng tri thức khoa học và tối ưu hóa các hoạt động trong gần như mọi lĩnh vực – từ y tế, tài chính, giáo dục đến sản xuất và quản lý đô thị [(Lovelace, 2016)](#KETCHIN).

***2.1.2. Đặc trưng của Big Data***

Big Data không chỉ nổi bật bởi quy mô dữ liệu lớn mà còn bởi những đặc trưng phức tạp và đa chiều khiến nó khác biệt so với dữ liệu truyền thống. Các đặc điểm này không tồn tại một cách độc lập, mà thường tương tác và gia tăng tính phức tạp cho quá trình quản lý và phân tích dữ liệu.

Một trong những cách tiếp cận phổ biến để mô tả đặc trưng của Big Data là mô hình “5V” bao gồm các đặc điểm: khối lượng (Volume), tốc độ (Velocity), đa dạng (Variety), độ tin cậy (Veracity) và giá trị (Value). Khối lượng phản ánh kích thước dữ liệu được tạo ra mỗi ngày với tốc độ chóng mặt, vượt xa khả năng lưu trữ của các hệ thống cũ. Trong khi đó, tốc độ đề cập đến việc dữ liệu được sinh ra, truyền tải và yêu cầu xử lý gần như tức thời, một đòi hỏi ngày càng phổ biến trong thời đại Internet vạn vật và các hệ thống thời gian thực. Đặc điểm đa dạng lại thể hiện rõ tính phi cấu trúc và hỗn hợp của dữ liệu, từ văn bản, hình ảnh, âm thanh đến các luồng dữ liệu cảm biến, tất cả đều đến từ những nguồn khác nhau với định dạng khác nhau. Độ tin cậy đề cập đến chất lượng và độ chính xác của dữ liệu, khi mà không phải mọi dữ liệu đều có giá trị, đặc biệt trong môi trường dữ liệu mở và có yếu tố con người. Cuối cùng, giá trị là yếu tố then chốt để phân biệt dữ liệu hữu ích với dữ liệu thừa. hỉ khi được phân tích đúng cách, dữ liệu mới thực sự đóng góp vào quá trình ra quyết định và đổi mới.

Bên cạnh mô hình 5V truyền thống, nhiều học giả đã đề xuất mở rộng thành 7V để phản ánh đầy đủ hơn những khía cạnh ngày càng phức tạp của Big Data. Hai yếu tố được thêm vào là tính biến thiên (Variability) và khả năng trực quan hóa (Visualization). Trong đó, tính biến thiên chỉ ra rằng dữ liệu không những thay đổi theo thời gian mà còn phụ thuộc vào ngữ cảnh và mục tiêu sử dụng, làm tăng độ khó trong việc xây dựng mô hình phân tích ổn định. Cùng với , khả năng trực quan hóa trở thành một yêu cầu thiết yếu trong bối cảnh dữ liệu lớn và phức tạp, giúp con người có thể hiểu được các mẫu ẩn trong dữ liệu thông qua biểu đồ, bản đồ nhiệt hay các công cụ đồ họa tương tác.

Tổng hợp các đặc trưng trên, có thể thấy Big Data không chỉ là thách thức về quy mô mà còn là bài toán về độ phức tạp và tính khả dụng của thông tin. Đặc điểm này đòi hỏi các tổ chức không chỉ đầu tư vào hạ tầng kỹ thuật mà còn phải có chiến lược quản trị dữ liệu toàn diện, nhằm khai thác hiệu quả tiềm năng của dữ liệu lớn trong môi trường kinh doanh và khoa học hiện đại.

**2.2. Các thư viện sử dụng trong Python**

### ***2.2.1. Thư viện Matplotlib***

Matplotlib là một thư viện mã nguồn mở trong Python, chuyên dùng cho việc trực quan hóa dữ liệu ([Geeksforgeeks, 2024](#A)). Thư viện này cho phép người dùng tạo ra các biểu đồ có độ phân giải cao như biểu đồ tròn (pie chart), biểu đồ tần suất (histogram), biểu đồ phân tán (scatter plot), đồ thị đường (line graph) và nhiều dạng đồ họa khác. Nhờ khả năng trực quan hóa mạnh mẽ, Matplotlib giúp người phân tích dữ liệu dễ dàng nhận diện xu hướng, mô hình và các điểm bất thường trong tập dữ liệu. Đây là một công cụ không thể thiếu trong các dự án phân tích dữ liệu, đặc biệt khi cần trình bày kết quả một cách sinh động và dễ hiểu.

***2.2.2. Thư viện Seaborn***

Seaborn là một thư viện trực quan hóa dữ liệu mạnh mẽ trong Python, được xây dựng dựa trên Matplotlib ([WsCube Tech, 2024](#CUBE)). Thư viện này được thiết kế chuyên biệt để hỗ trợ hiển thị các mô hình thống kê và trực quan hóa các mối quan hệ phức tạp trong dữ liệu một cách dễ hiểu và thẩm mỹ hơn. Seaborn cung cấp sẵn các bảng màu đa dạng và các kiểu biểu đồ tinh gọn giúp biểu diễn dữ liệu theo mẫu (pattern) dễ dàng. Thư viện này đặc biệt hữu ích trong việc tạo ra các biểu đồ như heatmap, boxplot, violin plot, pair plot,... cũng như đánh giá và thể hiện các mô hình hồi quy (regression models) một cách trực tiếp và rõ ràng. Nhờ đó, người dùng có thể phát hiện xu hướng và mối tương quan giữa các biến một cách trực quan mà không cần xử lý nhiều bước phức tạp.

Với khả năng tích hợp mượt mà với Pandas và Matplotlib, Seaborn là một công cụ lý tưởng cho các nhà khoa học dữ liệu và nhà phân tích khi cần trình bày kết quả thống kê một cách chuyên nghiệp và trực quan.

### ***2.2.3. Thư viện Pandas***

Pandas là một thư viện mã nguồn mở rất quan trọng trong lĩnh vực khoa học dữ liệu và học máy ([Geeksforgeeks, 2024b](#B)). Thư viện này cung cấp các cấu trúc dữ liệu hiệu quả và linh hoạt như Series và DataFrame, giúp người dùng dễ dàng thao tác với dữ liệu dạng bảng. Pandas hỗ trợ nhiều chức năng hữu ích như sắp xếp (sorting), gán lại chỉ mục (reindexing), duyệt phần tử (iteration), nối dữ liệu (concatenation), chuyển đổi dữ liệu (conversion), trực quan hóa đơn giản, và tổng hợp dữ liệu (aggregation). Bên cạnh đó, Pandas còn hỗ trợ mạnh mẽ trong việc làm sạch dữ liệu, xử lý dữ liệu bị thiếu, và chuẩn bị dữ liệu đầu vào cho các thuật toán học máy. Nhờ các tính năng này, Pandas trở thành công cụ không thể thiếu đối với các nhà phân tích dữ liệu.

***2.2.4. Thư viện PySpark***

PySpark là một thư viện giao diện Python của Apache Spark, được thiết kế để xử lý dữ liệu lớn theo mô hình phân tán ([Oracle, 2024](#ORACLE)). PySpark cung cấp nhiều mô-đun và API hỗ trợ đầy đủ các công việc từ tiền xử lý dữ liệu đến xây dựng mô hình học máy và xử lý dòng dữ liệu thời gian thực. Việc kết hợp linh hoạt các module trên cho phép xây dựng một quy trình học máy hoàn chỉnh trên PySpark, từ tiền xử lý dữ liệu, xây dựng mô hình đến đánh giá hiệu quả, đồng thời xử lý dữ liệu hiệu quả trên hệ thống phân tán. Đây là nền tảng quan trọng giúp triển khai các bài toán lớn với dữ liệu khối lượng lớn trong môi trường Spark.

***2.2.5. Thư viện Numpy***

NumPy, viết tắt của “Numerical Python” là một trong những thư viện phổ biến và được sử dụng rộng rãi nhất trong lĩnh vực khoa học dữ liệu và học máy ([Geeksforgeeks, 2024c](#C)). Thư viện này hỗ trợ xử lý các ma trận lớn và dữ liệu đa chiều một cách hiệu quả. NumPy tích hợp sẵn nhiều hàm toán học, giúp việc tính toán trở nên nhanh chóng và tiện lợi hơn. Đáng chú ý, các thư viện nổi tiếng như TensorFlow cũng sử dụng NumPy làm nền tảng để thực hiện nhiều phép toán trên tensor. Một trong những điểm mạnh quan trọng của NumPy là giao diện mảng (Array Interface), giúp tăng khả năng tương tác và tối ưu hóa hiệu suất xử lý dữ liệu.

***2.2.6. Thư viện học máy (MLlib)***

Thư viện học máy Mlib được thiết kế để đơn giản, có khả năng mở rộng và dễ dàng tích hợp với các công cụ phân tích dữ liệu khác. Được xây dựng trên nền tảng Spark, MLlib tận dụng khả năng xử lý song song và phân tán của Spark để hỗ trợ hiệu quả cho các bài toán học máy quy mô lớn. Với tính năng mở rộng, tốc độ xử lý cao và khả năng tương thích với nhiều ngôn ngữ lập trình như Python (PySpark), Scala và Java, MLlib giúp tập trung vào việc xây dựng mô hình và phân tích dữ liệu, thay vì xử lý các vấn đề hạ tầng phức tạp.

Thư viện MLlib cung cấp một tập hợp các thuật toán và công cụ phục vụ cho các bài toán học máy phổ biến. Cụ thể, MLlib hỗ trợ phân loại để phân nhóm dữ liệu theo các nhãn đã biết, hồi quy để dự đoán giá trị liên tục, và phân cụm nhằm nhóm các điểm dữ liệu chưa gán nhãn dựa trên sự tương đồng. Bên cạnh đó, MLlib cũng hỗ trợ lọc cộng tác (collaborative filtering) – thường được dùng trong hệ thống gợi ý, giảm chiều để rút gọn số lượng đặc trưng đầu vào mà vẫn giữ lại thông tin quan trọng và các hàm tối ưu hóa giúp cải thiện hiệu quả huấn luyện mô hình. Tất cả những công cụ này được thiết kế để hoạt động hiệu quả trên dữ liệu lớn, tận dụng sức mạnh phân tán của hệ sinh thái Apache Spark. Thư viện này cũng tích hợp liền mạch với các thành phần khác của Spark như Spark SQL, Spark Streaming và DataFrames, và thường được cài đặt sẵn trong môi trường như Databricks, giúp triển khai và vận hành mô hình học máy một cách hiệu quả.

**2.3. Mô hình sử dụng**

Nhóm nghiên cứu lựa chọn áp dụng hai mô hình học máy tiêu biểu từ thư viện Spark MLlib, đại diện cho hai hướng tiếp cận chính trong lĩnh vực học máy. Các mô hình như Logistic Regression, Tree Decision và SVM đã được sử dụng và so sánh với nhau như những mô hình học có giám sát, nhằm giải quyết bài toán phân loại dữ liệu có nhãn, với mục tiêu dự đoán nguy cơ mắc bệnh tim. Bên cạnh đó, K-Means Clustering được triển khai như một mô hình học không giám sát, phục vụ cho việc khám phá các cấu trúc ẩn trong dữ liệu thông qua phân cụm bệnh nhân.

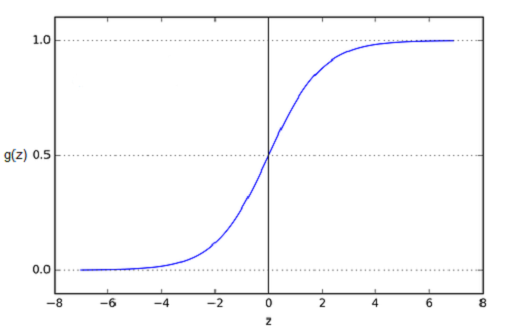
### ***2.3.1. Mô hình học có giám sát***

*2.3.1.1. Logistic Regression*

Hồi quy Logistic (Logistic Regression) là một trong những thuật toán phổ biến trong học máy có giám sát, chủ yếu được ứng dụng trong các bài toán phân loại nhị phân. Mô hình này được xây dựng nhằm dự đoán xác suất một quan sát thuộc về một trong hai lớp dựa trên tập hợp các đặc trưng đầu vào. Khác với hồi quy tuyến tính truyền thống, Logistic Regression sử dụng hàm sigmoid để chuyển đầu ra tuyến tính thành xác suất nằm trong khoảng [0, 1]. Dựa trên giá trị này, mô hình sẽ phân loại điểm dữ liệu vào lớp 0 hoặc lớp 1. Hàm sigmoid có công thức như sau:

​

Hàm sigmoid là một hàm liên tục và trơn, với giá trị đầu ra luôn nằm trong khoảng (0, 1). Hàm có đạo hàm tại mọi điểm trên trục số thực, điều này đảm bảo tính khả vi cần thiết cho các thuật toán tối ưu như gradient descent trong quá trình huấn luyện mô hình. Nhờ đặc tính này, sigmoid phù hợp để ánh xạ đầu ra tuyến tính thành xác suất trong các mô hình phân loại nhị phân.



Hình 2.1: Đồ thị hàm sigmoid

*Nguồn:* *Humphrys, Mark*

Trong thư viện PySpark MLlib, mô hình Logistic Regression được triển khai trong module pyspark.ml.classification. Mô hình cho phép thiết lập các siêu tham số như số vòng lặp tối đa (maxIter), hệ số điều chuẩn (regParam), và lựa chọn thuật toán tối ưu.

*2.3.1.2. Decison Tree*

Decision Tree (Cây quyết định) là một mô hình học có giám sát được sử dụng phổ biến cho cả bài toán phân loại và hồi quy. Mô hình có cấu trúc phân cấp dạng cây, bao gồm nút gốc (root node), các nút quyết định (internal nodes), nhánh (branches) và các nút lá (leaf nodes) – đại diện cho các kết quả đầu ra cuối cùng.

Thuật toán xây dựng cây theo chiến lược “chia để trị” (divide and conquer), bằng cách thực hiện tìm kiếm tham lam (greedy search) để xác định điểm chia tối ưu tại mỗi nút. Quá trình này được lặp lại theo chiều từ trên xuống (top-down), cho đến khi phần lớn dữ liệu được phân loại vào các nhóm thuần nhất. Độ thuần nhất của một nhóm có thể được đo bằng các chỉ số như Entropy, Information Gain hoặc Gini Impurity.

Tuy nhiên, khi cây phát triển quá lớn, mô hình có thể dẫn đến hiện tượng quá khớp (overfitting) do phân mảnh dữ liệu (data fragmentation). Vì vậy, các cây nhỏ thường được ưu tiên hơn để đảm bảo khả năng tổng quát hóa tốt, phù hợp với nguyên lý Occam’s Razor – ưu tiên mô hình đơn giản nhất có thể. Để hạn chế độ phức tạp và cải thiện hiệu suất, kỹ thuật pruning (cắt tỉa) thường được áp dụng nhằm loại bỏ các nhánh không mang lại giá trị phân loại đáng kể. Việc đánh giá hiệu quả mô hình thường được thực hiện thông qua phương pháp cross-validation.

Ngoài ra, để nâng cao độ chính xác và giảm sai số, mô hình cây quyết định có thể được kết hợp thành tập hợp (ensemble) thông qua thuật toán Random Forest, giúp cải thiện hiệu suất phân loại, đặc biệt khi các cây con không tương quan với nhau.

*2.3.1.3. Support Vector Machine (SVM)*

Support Vector Machine (SVM) là một thuật toán học có giám sát (supervised learning) rất phổ biến và hiệu quả trong lĩnh vực học máy, đặc biệt là với các bài toán phân loại. Mục tiêu chính của SVM là tìm ra một siêu phẳng (hyperplane) tối ưu trong không gian đặc trưng N chiều, sao cho có thể phân tách các lớp dữ liệu một cách rõ ràng nhất, đồng thời tối đa hóa khoảng cách (margin) giữa siêu phẳng này và các điểm dữ liệu gần nhất thuộc hai lớp khác nhau. Những điểm gần siêu phẳng này được gọi là support vectors, vì chúng là những điểm đóng vai trò quyết định trong việc xác định vị trí và hướng của siêu phẳng phân tách.

Một trong những ưu điểm quan trọng của SVM là khả năng xử lý tốt cả các bài toán phân loại tuyến tính và phi tuyến. Trong trường hợp dữ liệu không thể phân tách tuyến tính trong không gian ban đầu, SVM sử dụng một kỹ thuật gọi là kernel trick. Kernel trick cho phép ánh xạ dữ liệu từ không gian đầu vào sang một không gian đặc trưng có số chiều cao hơn, nơi các lớp dữ liệu có thể được phân tách bằng một siêu phẳng tuyến tính. Các loại kernel phổ biến bao gồm: tuyến tính, hàm Gauss (RBF - Radial Basis Function), sigmoid và đa thức.

Việc lựa chọn hàm kernel và các tham số đi kèm (chẳng hạn như độ rộng của RBF hay bậc của đa thức) đóng vai trò then chốt trong hiệu suất của mô hình. Quá trình này thường cần được thực hiện thông qua thử nghiệm hoặc các phương pháp tối ưu hóa như cross-validation.

Nhờ khả năng tạo ra ranh giới phân tách có độ chính xác cao và khả năng tổng quát hóa tốt ngay cả với những tập dữ liệu có số chiều lớn (high-dimensional data), SVM trở thành một trong những thuật toán phân loại mạnh mẽ nhất. Nó được ứng dụng rộng rãi trong nhiều lĩnh vực như nhận diện khuôn mặt, phân loại văn bản, sinh học tính toán (ví dụ: phân loại gen) và phát hiện gian lận. Tuy nhiên, điểm yếu của SVM là hiệu suất tính toán có thể giảm khi áp dụng với tập dữ liệu cực lớn, và việc lựa chọn kernel không phù hợp có thể làm giảm độ chính xác của mô hình.

***2.3.2. Mô hình học không giám sát***

Phân cụm K-means là một thuật toán học máy không giám sát phổ biến được sử dụng để phân vùng một tập dữ liệu thành một số cụm được xác định trước. Mục tiêu là nhóm các điểm dữ liệu tương tự lại với nhau và khám phá các mẫu hoặc cấu trúc cơ bản trong dữ liệu.

Tối ưu hóa đóng vai trò then chốt trong thuật toán phân cụm K-Means, với mục tiêu chính là xác định vị trí các tâm cụm sao cho tổng bình phương khoảng cách giữa mỗi điểm dữ liệu và tâm cụm gần nhất được giảm thiểu tối đa. K-Means hướng đến việc phân chia tập dữ liệu thành *k* cụm sao cho các điểm trong cùng một cụm có độ tương đồng cao, trong khi sự khác biệt giữa các cụm là rõ rệt. Mỗi cụm được đại diện bởi một tâm cụm, tính bằng trung bình cộng của tất cả các điểm trong cụm đó. Việc gán nhãn cụm cho từng điểm dựa trên nguyên tắc khoảng cách Euclidean ngắn nhất đến các tâm hiện tại. Quá trình này được lặp lại cho đến khi các tâm cụm hội tụ hoặc không còn sự thay đổi đáng kể trong phân cụm giữa các vòng lặp.

Thuật toán K-Means thực hiện phân cụm dữ liệu thông qua ba bước chính. Đầu tiên, *k* tâm cụm được khởi tạo một cách ngẫu nhiên. Tiếp theo, mỗi điểm dữ liệu được gán vào cụm có tâm gần nhất dựa trên khoảng cách Euclidean. Sau đó, các tâm cụm được cập nhật bằng cách tính trung bình tất cả các điểm dữ liệu thuộc về cụm đó. Quá trình gán và cập nhật này được lặp lại cho đến khi các tâm cụm hội tụ hoặc đạt đến số vòng lặp tối đa được định trước.

Trong PySpark MLlib, thuật toán K-Means thuộc module pyspark.ml.clustering và hỗ trợ các tham số linh hoạt như: k (số cụm cần phân chia), maxIter (số vòng lặp tối đa), và seed (giá trị khởi tạo ngẫu nhiên giúp tái lập kết quả). Những tham số này giúp điều chỉnh quá trình phân cụm hiệu quả, phù hợp với dữ liệu lớn và yêu cầu thực tế.

**2.4. Phương pháp đánh giá**

Trong các bài toán phân loại, việc đánh giá hiệu quả của mô hình đóng vai trò then chốt trong việc xác định khả năng ứng dụng của mô hình vào thực tiễn. Một mô hình phân loại không thể được xem là thành công nếu chỉ đơn thuần hoạt động tốt trên dữ liệu huấn luyện mà không được kiểm chứng nghiêm ngặt bằng các chỉ số đánh giá khách quan. Đặc biệt trong những bài toán có ý nghĩa quan trọng như chẩn đoán bệnh, các chỉ số đánh giá giúp đảm bảo rằng mô hình không gây ra hậu quả nghiêm trọng do sai sót trong dự đoán.

Một trong những công cụ phổ biến nhất để đánh giá mô hình phân loại là ma trận nhầm lẫn – bảng tổng hợp cho thấy mô hình đã dự đoán đúng và sai bao nhiêu trường hợp ở mỗi lớp. Từ ma trận này, ta có thể tính toán nhiều chỉ số quan trọng như độ chính xác tổng thể (Accuracy), độ chính xác của lớp dương (Precision), khả năng bao phủ các trường hợp dương (Recall), và chỉ số cân bằng giữa hai yếu tố này là F-score. Những chỉ số này không chỉ giúp đánh giá toàn diện mô hình mà còn hỗ trợ ra quyết định trong việc lựa chọn mô hình phù hợp với đặc điểm và yêu cầu cụ thể của bài toán.Trong các bài toán phân loại, việc đánh giá hiệu quả của mô hình là một bước thiết yếu nhằm xác định mức độ chính xác và khả năng áp dụng của mô hình vào thực tế. Các phương pháp đánh giá phổ biến thường dựa trên ma trận nhầm lẫn (confusion matrix), từ đó rút ra các chỉ số như Precision, Recall, F-score và Accuracy. Những chỉ số này đặc biệt quan trọng trong các tình huống mà dữ liệu phân bố không cân bằng hoặc khi chi phí của các loại lỗi (false positive, false negative) là khác nhau.

***2.4.1. Accuracy***

Accuracy, hay còn gọi là độ chính xác tổng thể, là tỷ lệ phần trăm giữa số lượng dự đoán đúng và tổng số trường hợp dự đoán trong toàn bộ tập dữ liệu. Đây là chỉ số cơ bản và dễ hiểu nhất, thường được sử dụng như một thước đo ban đầu để xem xét độ chính xác của mô hình. Accuracy có thể cho ta một cái nhìn nhanh về hiệu quả mô hình nếu dữ liệu được phân bố đều giữa các lớp. Tuy nhiên, trong các trường hợp dữ liệu mất cân bằng – chẳng hạn khi số lượng trường hợp dương rất ít so với âm – mô hình có thể chỉ cần dự đoán tất cả là âm và vẫn đạt độ chính xác cao. Điều này dễ tạo ra ảo tưởng rằng mô hình đang hoạt động tốt, trong khi thực tế là nó hoàn toàn không phát hiện được các trường hợp dương. Do đó, Accuracy nên được xem là chỉ số nền tảng và cần được sử dụng cùng với các chỉ số khác để có cái nhìn đầy đủ hơn về hiệu suất phân loại.

***2.4.2. Precision***

Precision phản ánh độ tin cậy trong những dự đoán thuộc lớp dương. Cụ thể, đây là tỷ lệ giữa số lượng các dự đoán đúng thực sự là dương với tổng số lần mô hình đưa ra dự đoán dương. Nếu một mô hình có Precision cao, điều đó có nghĩa là hầu hết các trường hợp mà mô hình cho là dương đều thực sự đúng, và tỷ lệ dự đoán sai dẫn đến dương giả rất thấp. Precision đặc biệt quan trọng trong các ứng dụng mà sai sót trong việc phát hiện dương tính có thể gây ra hậu quả nghiêm trọng, ví dụ như hệ thống phát hiện email rác, nơi việc gắn nhầm một email quan trọng là thư rác có thể gây thiệt hại, hoặc trong tài chính, khi cảnh báo sai về gian lận có thể dẫn đến khóa tài khoản người dùng hợp pháp.

***2.4.3. Recall***

Precision hay còn gọi là độ chính xác dương tính là tỷ lệ giữa số lượng dự đoán đúng ở lớp dương và tổng số dự đoán thuộc lớp dương. Nói cách khác, nó cho biết trong tất cả các trường hợp mà mô hình dự đoán là thuộc lớp dương, có bao nhiêu trường hợp thực sự đúng. Precision đặc biệt quan trọng trong những tình huống mà sai sót trong việc gán nhãn dương tính có thể gây hậu quả nghiêm trọng.

***2.4.4. F-score***

F-score là chỉ số tổng hợp kết hợp giữa Precision và Recall, nhằm đánh giá một cách cân bằng giữa độ chính xác và khả năng bao phủ. F-score đặc biệt hữu ích trong các bài toán có dữ liệu mất cân đối, nơi mà chỉ số Accuracy thường trở nên không đáng tin cậy. F-score được tính bằng trung bình điều hòa của Precision và Recall, nhờ vậy nếu một trong hai chỉ số thấp thì F-score cũng sẽ bị kéo xuống theo. Điều này giúp đảm bảo rằng mô hình không chỉ chính xác mà còn không bỏ sót quá nhiều trường hợp dương. Trong các tình huống thực tế, F-score thường được dùng làm chỉ số chính để so sánh giữa các mô hình khác nhau, nhất là khi mục tiêu là tối ưu cả hai khía cạnh: độ chính xác và độ nhạy của mô hình.

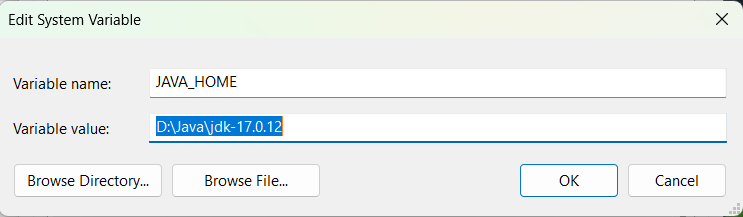
**CHƯƠNG III. PHƯƠNG PHÁP CẤU HÌNH, KHÁM PHÁ VÀ XỬ LÝ DỮ LIỆU**

**3.1. Cài đặt và cấu hình Apache Hadoop**

Trước khi tiến hành cài đặt Apache Hadoop, bước đầu tiên và bắt buộc là cài đặt Java Development Kit (JDK), cụ thể là phiên bản JDK 17 để đảm bảo khả năng tương thích với các phiên bản Hadoop hiện tại. Bộ cài đặt JDK 17 được tải về trực tiếp từ trang chính thức của Oracle tại địa chỉ:

[https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html.](https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html)

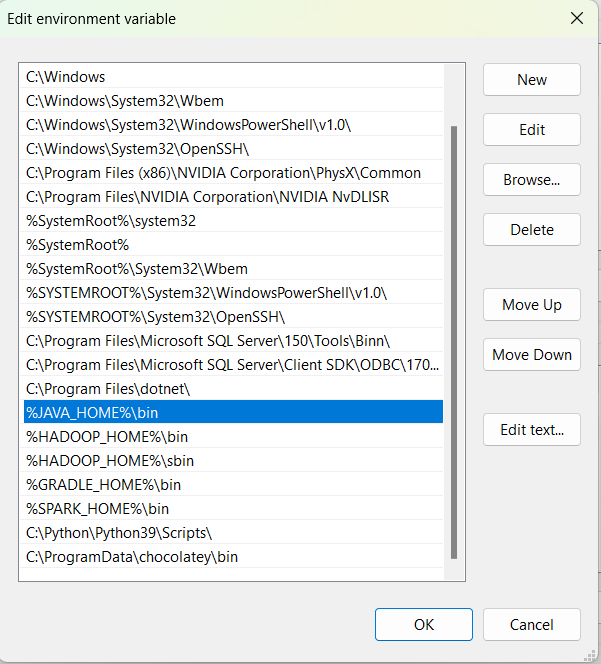
Sau khi cài đặt JDK, cần tiến hành cấu hình biến môi trường để hệ thống có thể nhận diện và sử dụng Java đúng cách. Cụ thể, truy cập vào System Properties bằng cách nhấn chuột phải vào biểu tượng This PC → chọn Properties → Advanced system settings, sau đó nhấn vào nút Environment Variables. Tại đây, trong phần System variables, nhấn New để tạo biến môi trường mới với tên là JAVA\_HOME, trỏ đến thư mục cài đặt JDK.



Hình 3.1: Cấu hình biến môi trường JAVA\_HOME trong Windows

*Nguồn: Tác giả thực hiện chạy chương trình*

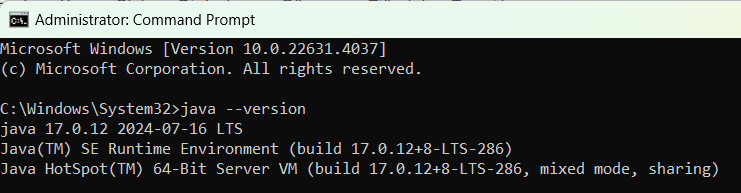
Tiếp theo, chỉnh sửa biến Path và thêm vào dòng %JAVA\_HOME%\bin để Java có thể được truy cập từ bất kỳ đâu trong dòng lệnh.



Hình 3.2: Cấu hình Path cho JAVA\_HOME

*Nguồn: Tác giả thực hiện chạy chương trình*

Sau khi hoàn tất việc cài đặt và cấu hình biến môi trường cho JDK, bước tiếp theo là kiểm tra tính chính xác của cấu hình thông qua Command Prompt (cmd). Nhập lệnh java -version để kiểm tra phiên bản Java hiện tại mà hệ thống đang nhận diện. Nếu cấu hình đúng, màn hình sẽ hiển thị thông tin tương ứng với phiên bản JDK đã cài đặt:

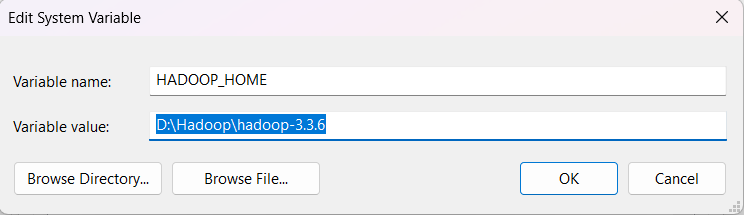


Hình 3.3: Kiểm tra phiên bản Java trong Command Prompt

*Nguồn: Tác giả thực hiện chạy chương trình*

Sau khi hoàn tất cấu hình JDK, bước tiếp theo trong quá trình triển khai là cài đặt Apache Hadoop. Bộ cài chính thức có thể được tải về từ trang chủ của Apache tại địa chỉ: https://hadoop.apache.org/releases.html. Tại đây, lựa chọn phiên bản Hadoop 3.3.6, trong mục Binary download và tiến hành tải về bản nhị phân (binary). Sau khi tải xong, tiến hành giải nén tập tin và bắt đầu tiến hành cấu hình.

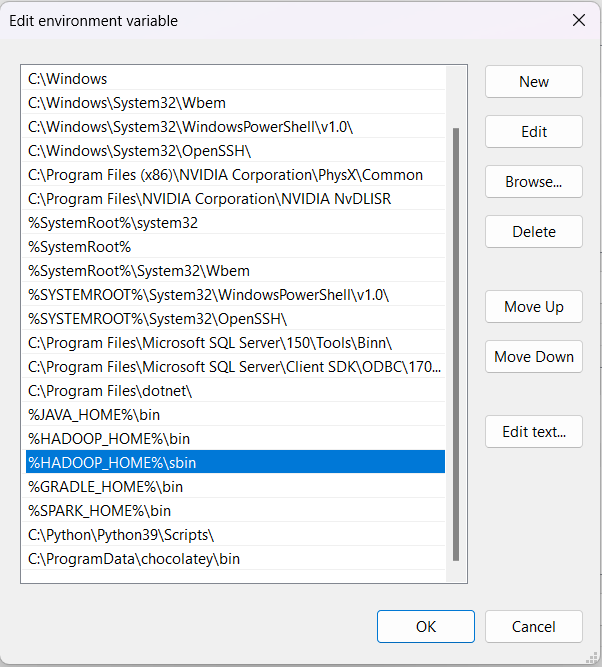
Tương tự như cấu hình JDK, cần thiết lập các biến môi trường hệ thống để đảm bảo khả năng nhận diện Hadoop trong môi trường dòng lệnh. Truy cập System Properties → Advanced system settings → Environment Variables, tại khung System variables, chọn New để tạo biến HADOOP\_HOME, trỏ tới thư mục chứa Hadoop đã giải nén.



Hình 3.4: Cấu hình biến môi trường HADOOP\_HOME trong Windows

*Nguồn: Tác giả thực hiện chạy chương trình*

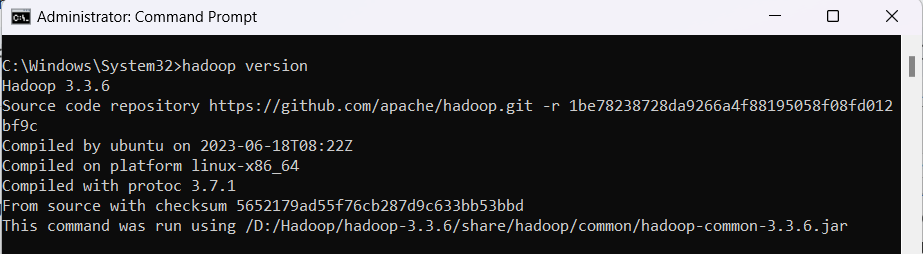
Sau khi khai báo biến môi trường HADOOP\_HOME, cần tiếp tục cấu hình biến hệ thống Path, thực hiện bằng cách chọn biến Path trong mục System variables, sau đó nhấn Edit để mở cửa sổ chỉnh sửa. Tại đây, chọn New và lần lượt thêm hai dòng %HADOOP\_HOME%\bin và %HADOOP\_HOME%\sbin. Các đường dẫn này cho phép thực thi các công cụ và tập lệnh của Hadoop từ bất kỳ vị trí nào trong dòng lệnh. Sau khi thêm xong, nhấn OK để hoàn tất việc cấu hình.



Hình 3.5: Cấu hình Path cho HADOOP\_HOME

*Nguồn: Tác giả thực hiện chạy chương trình*

Khi hoàn tất thiết lập các biến môi trường, có thể tiến hành kiểm tra hệ thống bằng cách mở Command Prompt và chạy lệnh hadoop version. Nếu cấu hình chính xác, thông tin phiên bản Hadoop sẽ được hiển thị trên màn hình.

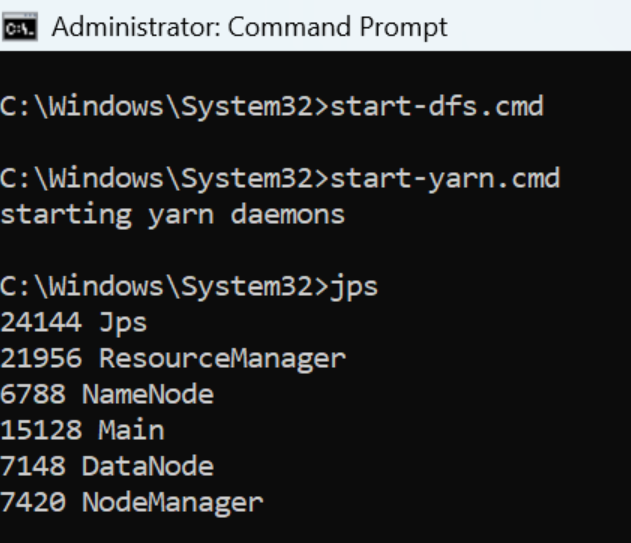


Hình 3.6: Kiểm tra phiên bản Hadoop trong Command Prompt

*Nguồn: Tác giả thực hiện chạy chương trình*

Tiến trình cấu hình hệ thống Hadoop được thực hiện thông qua các tệp XML nằm trong thư mục etc/hadoop, bao gồm bốn tệp cấu hình chính: core-site.xml, hdfs-site.xml, mapred-site.xml, và yarn-site.xml. Để đảm bảo khả năng tương thích trên nền tảng Windows, bộ công cụ hỗ trợ Hadoop Winutils được tải về và sử dụng. Thư mục bin trong bộ công cụ này được sao chép và chép đè lên thư mục bin của thư mục cài đặt hadoop-3.3.6. Tiếp theo, cấu hình môi trường Hadoop được thực hiện bằng cách chỉnh sửa các tham số trong tệp hadoop-env.cmd, nằm trong thư mục etc/hadoop. *[Chi tiết ở Phụ lục 1]*

Tiếp theo, sử dụng lệnh “jps” để kiểm tra các tiến trình đang hoạt động của Hadoop. Trường hợp không xuất hiện đầy đủ các dịch vụ như NameNode, DataNode, ResourceManager hoặc NodeManager, cần khởi động thủ công bằng cách thực thi hai lệnh “start-dfs.cmd” và “start-yarn.cmd”. Sau đó, chạy lại lệnh “jps” để xác nhận rằng các dịch vụ cốt lõi đã được kích hoạt thành công.



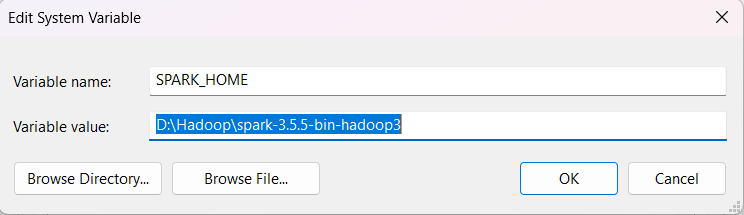
Hình 3.7: Kiểm tra tiến trình Hadoop bằng lệnh jps trong Command Prompt

*Nguồn: Tác giả thực hiện chạy chương trình*

**3.2. Cài đặt, cấu hình Apache Spark**

Để bắt đầu cấu hình Apache Spark, tải gói cài đặt Apache Spark phiên bản tương thích tại đường link [spark-3.5.5-bin-hadoop3.tgz](https://www.apache.org/dyn/closer.lua/spark/spark-3.5.5/spark-3.5.5-bin-hadoop3.tgz) từ trang chính thức. Sau khi tải về, tiến hành giải nén vào một thư mục.

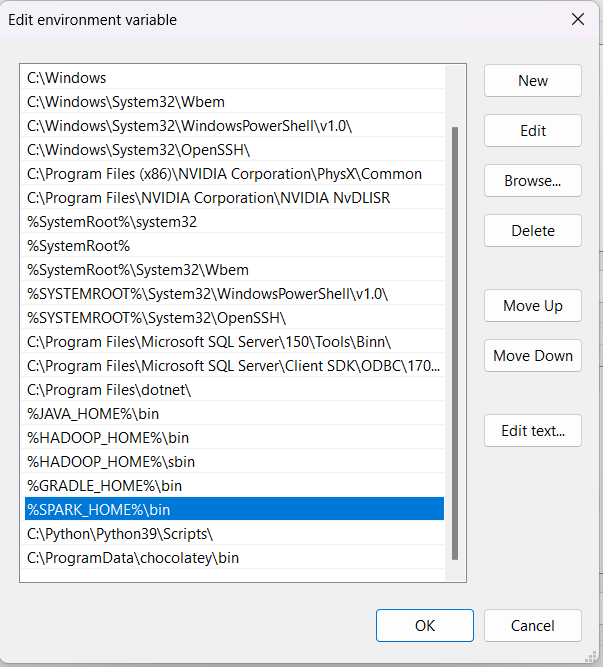
Tiếp theo, cần thiết lập các biến môi trường để hệ điều hành có thể nhận diện Spark. Truy cập mục Advanced system settings, chọn Environment Variables, sau đó trong phần System variables, nhấn New để tạo biến SPARK\_HOME, trỏ đến thư mục chứa Spark vừa giải nén.



Hình 3.8: Cấu hình biến môi trường SPARK\_HOME trong Windows

*Nguồn: Tác giả thực hiện chạy chương trình*

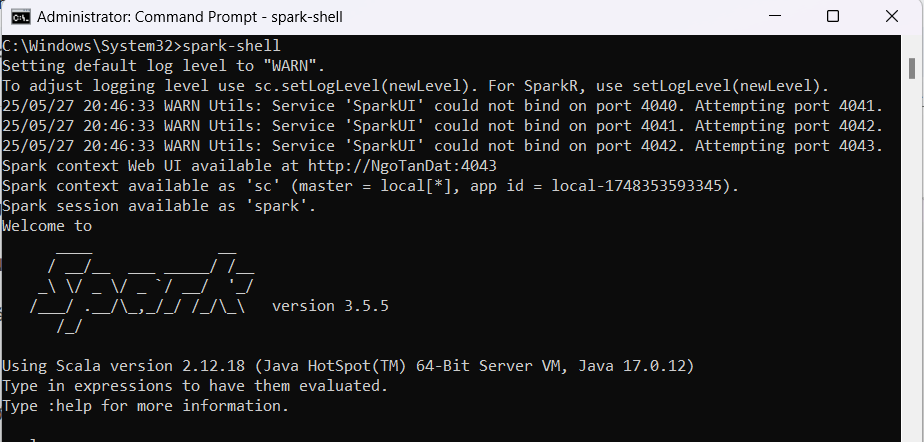
Đồng thời, cập nhật biến Path bằng cách thêm dòng %SPARK\_HOME%\bin nhằm cho phép sử dụng các lệnh Spark trực tiếp từ dòng lệnh. Sau đó bấm OK, thì sẽ thu được kết quả như hình.



Hình 3.9: Cấu hình Path cho HADOOP\_HOME

*Nguồn: Tác giả thực hiện chạy chương trình*

Để xác minh cấu hình đã thực hiện, mở Command Prompt và thực thi lệnh spark-shell. Nếu quá trình cài đặt và thiết lập biến môi trường diễn ra chính xác, giao diện dòng lệnh của Spark sẽ khởi chạy thành công, hiển thị phiên bản Spark cùng một số thông tin khởi động.



Hình 3.10: Kiểm tra phiên bản Spark trong Command Prompt

*Nguồn: Tác giả thực hiện chạy chương trình*

**3.3. Tổng quan bộ dữ liệu**

***3.1.1. Nguồn dữ liệu***

Dữ liệu sử dụng trong nghiên cứu được thu thập từ nền tảng Kaggle, cụ thể là bộ “Heart Attack Risk Dataset of China” do tác giả Ankush Panday chia sẻ. Bộ dữ liệu cung cấp các thông tin sức khỏe, hành vi, môi trường sống và nhân khẩu học của hơn 239.266 người dân tại Trung Quốc, phục vụ cho mục đích phân tích rủi ro mắc bệnh tim mạch.

***3.1.2. Mô tả các biến***

Bộ dữ liệu đã được tổ chức và phân chia thành các nhóm thông tin chính, với mục tiêu hỗ trợ một cách hiệu quả cho các quy trình phân tích dữ liệu chuyên sâu, giúp làm rõ cấu trúc và đặc điểm của dữ liệu, mà còn đóng vai trò quan trọng trong việc lựa chọn đặc trưng phù hợp, từ đó nâng cao hiệu suất và độ chính xác của các mô hình học máy.

Nhóm đầu tiên là thông tin cá nhân và nhân khẩu học thể hiện đặc trưng của đối tượng tham gia khảo sát, giúp xác định đặc điểm dân số trong phân tích.

|  |  |  |
| --- | --- | --- |
| **Biến** | **Kiểu dữ liệu** | **Mô tả** |
| Patient\_ID | Identifier | Mã định danh duy nhất cho từng cá nhân (không dùng để phân tích) |
| Age | Số | Tuổi của bệnh nhân |
| Gender | Phân loại | Giới tính (Male – Nam/Female – Nữ) |
| Rural\_or\_Urban | Phân loại | Khu vực sinh sống (Rural – nông thôn / Urban – thành thị) |
| Region | Phân loại | Vùng địa lý tại Trung Quốc (Eastern, Western, Northern, Southern, Central) |
| Province | Phân loại | Tỉnh/thành cụ thể, ví dụ: Beijing, Shanghai, Gansu,... |

Thứ hai là nhóm hành vi sức khỏe và lối sống, phản ánh các yếu tố liên quan đến thói quen cá nhân.

|  |  |  |
| --- | --- | --- |
| **Biến** | **Kiểu dữ liệu** | **Mô tả** |
| Smoking\_Status | Phân loại | Tình trạng hút thuốc (Smoker / Non-Smoker) |
| Alcohol\_Consumption | Phân loại | Có tiêu thụ rượu bia hay không (Yes / No) |
| Physical\_Activity | Phân loại | Mức độ hoạt động thể chất (Low / Medium / High) |
| Diet\_Score | Phân loại | Chế độ ăn uống (Healthy / Moderate / Poor) |
| Stress\_Level | Phân loại | Mức độ căng thẳng (Low / Medium / High) |

Tiếp theo là nhóm bệnh lý và lâm sàng tập trung vào các yếu tố sức khỏe có thể đo lường hoặc đã được chẩn đoán

|  |  |  |
| --- | --- | --- |
| **Biến** | **Kiểu dữ liệu** | **Mô tả** |
| Hypertension | Phân loại | Có bị tăng huyết áp không (Yes / No) |
| Diabetes | Phân loại | Có bị tiểu đường không (Yes / No) |
| Obesity | Phân loại | Tình trạng béo phì (Yes / No) |
| Cholesterol\_Level | Phân loại | Mức cholesterol trong máu (High / Normal / Low) |
| Blood\_Pressure | Số | Huyết áp |
| Chronic\_Kidney\_Disease | Phân loại | Có bệnh thận mãn tính không (Yes / No) |
| Previous\_Heart\_Attack | Phân loại | Đã từng bị nhồi máu cơ tim chưa (Yes / No) – Cần xem xét mức độ trùng lặp với biến mục tiêu Heart\_Attack |

Ngoài ra, bộ dữ liệu còn chứa nhóm yếu tố di truyền và môi trường giúp xác định ảnh hưởng của bối cảnh sống và yếu tố di truyền đến nguy cơ bệnh tim.

|  |  |  |
| --- | --- | --- |
| **Biến** | **Kiểu dữ liệu** | **Mô tả** |
| Air\_Pollution\_Exposure | Phân loại | Mức tiếp xúc với ô nhiễm không khí (Low / Medium / High) |
| Family\_History\_CVD | Phân loại | Có tiền sử gia đình mắc bệnh tim mạch không (Yes / No) |

Nhóm tiếp cận y tế và chăm sóc sức khỏe phản ánh khả năng tiếp cận và sử dụng các dịch vụ y tế.

|  |  |  |
| --- | --- | --- |
| **Biến** | **Kiểu dữ liệu** | **Mô tả** |
| Healthcare\_Access | Phân loại | Mức độ tiếp cận dịch vụ y tế (Good / Moderate / Poor) |
| Hospital\_Availability | Phân loại | Mức độ sẵn có của bệnh viện gần nơi sinh sống (High / Medium / Low) |
| TCM\_Use | Phân loại | Có sử dụng y học cổ truyền (Traditional Chinese Medicine) không (Yes / No) |

Cuối cùng là nhóm kinh tế – xã hội có thể ảnh hưởng gián tiếp đến sức khỏe thông qua điều kiện sống, chế độ dinh dưỡng và khả năng tiếp cận chăm sóc y tế.

|  |  |  |
| --- | --- | --- |
| **Biến** | **Kiểu dữ liệu** | **Mô tả** |
| Employment\_Status | Phân loại | Tình trạng việc làm (Employed / Unemployed / Retired) |
| Education\_Level | Phân loại | Trình độ học vấn (None / Primary / Secondary / Higher) |
| Income\_Level | Phân loại | Mức thu nhập (Low / Middle / High) |

Biến mục tiêu trong phân tích biểu thị việc cá nhân có từng bị nhồi máu cơ tim hay không, được sử dụng để huấn luyện và đánh giá mô hình dự đoán.

|  |  |  |
| --- | --- | --- |
| **Biến** | **Kiểu dữ liệu** | **Mô tả** |
| Heart\_Attack | Phân loại | Biến mục tiêu (Yes / No) – chỉ ra người đó có từng bị nhồi máu cơ tim |
| CVD\_Risk\_Score | Số | Điểm rủi ro tim mạch tổng quát, dùng để so sánh hoặc xây dựng mô hình hồi quy (0–100) |

[***3.1.3.***](http://3.1.3.th) ***Phân tích khám phá dữ liệu***

Trong bộ dữ liệu, các biến định lượng được thống kê mô tả để hiểu rõ phân bố và đặc điểm chung. Các chỉ số như giá trị trung bình, trung vị, độ lệch chuẩn, giá trị lớn nhất và giá trị nhỏ nhất được tính nhằm xác định xu hướng và mức độ phân tán của dữ liệu.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Biến** | **Mô tả** | **Số lượng quan sát** | **Giá trị trung bình** | **Độ lệch chuẩn** | **Giá trị nhỏ nhất** | **Giá trị lớn nhất** |
| Age | Tuổi của bệnh nhân | 239,266 | 59.47 | 17.35 | 30 | 89 |
| Blood Pressure | Huyết áp của bệnh nhân (mmHg) | 239,266 | 141.23 | 20.32 | 90 | 179 |
| CVD Risk Score | Điểm rủi ro bệnh tim mạch (0–100) | 239,266 | 54.49 | 25.99 | 10 | 99 |

Bảng 3.1: Kết quả thống kê mô tả của các biến định lượng

*Nguồn: Tác giả thực hiện chạy chương trình*

Trong tập dữ liệu gồm 239,266 quan sát, ba biến định lượng được phân tích bao gồm tuổi, huyết áp và điểm rủi ro bệnh tim mạch. Về độ tuổi, bệnh nhân có tuổi trung bình là khoảng 59.47 tuổi với độ lệch chuẩn 17.35, cho thấy dữ liệu trải rộng từ người trung niên đến cao tuổi, trong đó người trẻ nhất là 30 tuổi và lớn tuổi nhất là 89. Huyết áp trung bình của bệnh nhân là 141.23 mmHg, cao hơn mức huyết áp bình thường, với độ lệch chuẩn 20.32 và dao động từ 90 đến 179 mmHg, phản ánh tình trạng cao huyết áp phổ biến trong tập dữ liệu. Điểm rủi ro bệnh tim mạch có giá trị trung bình là 54.49 trên thang điểm 0 đến 100, với độ lệch chuẩn khá lớn là 25.99, cho thấy sự phân tán rủi ro đáng kể giữa các bệnh nhân – từ mức rất thấp (10 điểm) đến mức rất cao (99 điểm).

Kết quả rà soát tính đầy đủ và tính duy nhất của các quan sát cho thấy toàn bộ tập dữ liệu không chứa giá trị thiếu ở bất kỳ biến nào, đồng thời không xuất hiện các dòng dữ liệu trùng lặp. Điều này cho thấy chất lượng dữ liệu đầu vào tương đối tốt, đảm bảo tính toàn vẹn và đáng tin cậy để thực hiện các bước phân tích khám phá cũng như xây dựng mô hình phân cụm, phân nhóm mà không cần thực hiện các bước tiền xử lý phức tạp nhằm làm sạch dữ liệu.

**3.4. Các bước thực hiện truy vấn sử dụng SQL SPARK**

***3.4.1. Khai báo các thư viện cần sử dụng***

|  |
| --- |
| *# Khai báo thư viện*  import pandas as pd  import matplotlib.pyplot as plt  plt.style.use('ggplot')  from matplotlib.pyplot import figure  %matplotlib inline  import seaborn as sns  from pyspark.sql import SparkSession |

***3.4.2. Đọc dữ liệu***

|  |
| --- |
| *# Khởi tạo SparkSession*  spark = SparkSession.builder.appName("Heart Attack Analysis").getOrCreate()  *# Đọc dữ liệu*  df = spark.read.option("inferSchema", True).option("header", True).csv("D:/data/heart\_attack\_china.csv")  *# Tạo TempView*  df.createOrReplaceTempView("heart\_attack") |

***3.4.3. Khám phá dữ liệu***

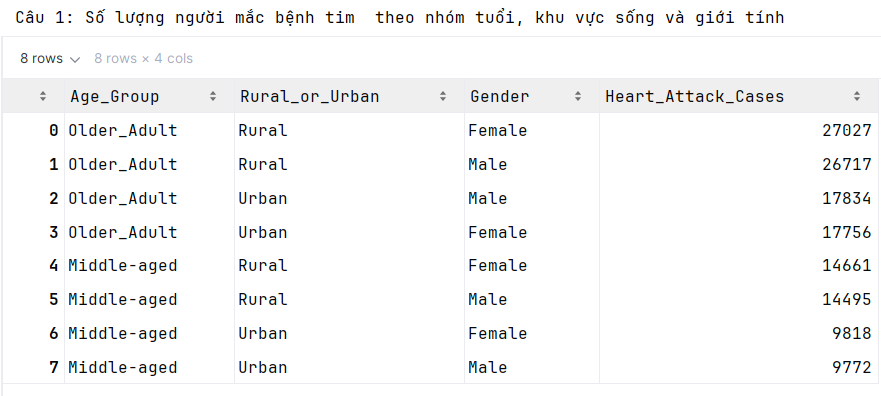
|  |
| --- |
| *# 1. Hiển thị một số dòng đầu tiên*  df.show(10)  *# 2. Thống kê mô tả cho tất cả các cột số*  numeric\_cols = [c for c, t in df.dtypes if t in ['int', 'double', 'float', 'bigint']]  df.select(numeric\_cols).describe().show()  *# 3. Thống kê số lượng giá trị null cho mỗi cột*  df.select([  spark\_sum(col(c).isNull().cast("int")).alias(c + "\_nulls") for c in df.columns  ]).show()  *# 4. Thống kê dòng trùng lặp*  total\_rows = df.count()  distinct\_rows = df.dropDuplicates().count()  duplicate\_rows = total\_rows - distinct\_rows  print(f"Tổng số dòng : {total\_rows}")  print(f"Số dòng duy nhất : {distinct\_rows}")  print(f"Số dòng bị trùng : {duplicate\_rows}") |

***3.4.4. Thực hiện các truy vấn dữ liệu***

Câu 1: Thống kê số lượng người mắc bệnh tim theo nhóm tuổi, khu vực sống và giới tính.

|  |
| --- |
| *#Câu 1:*  print("Câu 1: Số lượng người mắc bệnh tim theo nhóm tuổi, khu vực sống và giới tính")  df1 = spark.sql("""  SELECT  CASE  WHEN Age BETWEEN 30 AND 50 THEN 'Middle-aged'  ELSE 'Older\_Adult'  END AS Age\_Group, Rural\_or\_Urban, Gender,  COUNT(Gender) AS Heart\_Attack\_Cases  FROM heart\_attack  WHERE Heart\_Attack = 'Yes'  GROUP BY Gender, Age\_Group, Rural\_or\_Urban  ORDER BY Heart\_Attack\_Cases DESC  """).toPandas()  df1 |

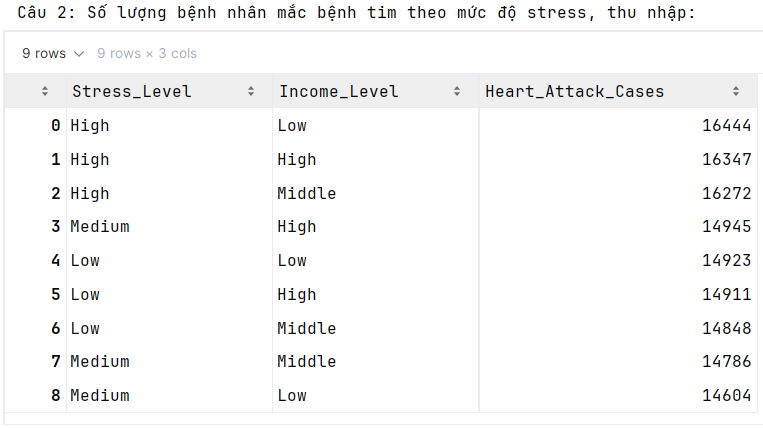
Kết quả đầu ra:



Câu 2: Thống kê số lượng bệnh nhân mắc bệnh tim theo mức độ căng thẳng và nhóm thu nhập.

|  |
| --- |
| *#Câu 2:*  print("Câu 2: Số lượng bệnh nhân mắc bệnh tim theo mức độ stress, thu nhập: ")  df2=spark.sql("""  SELECT  Stress\_Level, Income\_Level,  SUM(CASE WHEN Heart\_Attack = 'Yes' THEN 1 ELSE 0 END) AS Heart\_Attack\_Cases  FROM heart\_attack  GROUP BY Income\_Level,Stress\_Level  ORDER BY Heart\_Attack\_Cases DESC  """).toPandas()  df2 |

Kết quả đầu ra:



Câu 3: Tính tỷ lệ % bệnh nhân bị mắc bệnh tim theo nhóm hành vi sức khỏe và lối sống.

|  |
| --- |
| *# Lấy tổng số người bị bệnh tim*  total\_heart\_attack = spark.sql("""  SELECT COUNT(\*) AS total  FROM heart\_attack  WHERE Heart\_Attack = 'Yes'  """).collect()[0]["total"] |

|  |
| --- |
| *# Câu 3: Tỷ lệ số % bệnh nhân bị mắc bệnh tim theo nhóm hành vi sức khỏe và lối sống*  print("Câu 3: Tỷ lệ % bệnh nhân bị mắc bệnh tim theo nhóm hành vi sức khỏe và lối sống")  df3=spark.sql(f"""  SELECT  Smoking\_Status,  Alcohol\_Consumption,  Physical\_Activity,  Diet\_Score,  Stress\_Level,  COUNT(\*) AS Heart\_Attack\_Cases,  ROUND(100.0 \* COUNT(\*) / {total\_heart\_attack}, 2) AS Heart\_Attack\_Percentage  FROM heart\_attack  WHERE Heart\_Attack = 'Yes'  GROUP BY Smoking\_Status, Alcohol\_Consumption, Physical\_Activity, Diet\_Score, Stress\_Level  ORDER BY Heart\_Attack\_Percentage DESC  """).toPandas()  df3 |

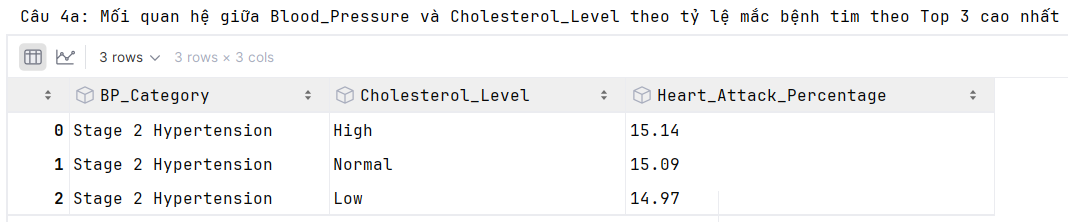
Kết quả đầu ra:



Câu 4: Thống kê đặc điểm mối quan hệ giữa Blood\_Pressure và Heart\_Attack theo Cholesterol\_Level theo top 3 tỷ lệ mắc bệnh tim cao nhất và thấp nhất.

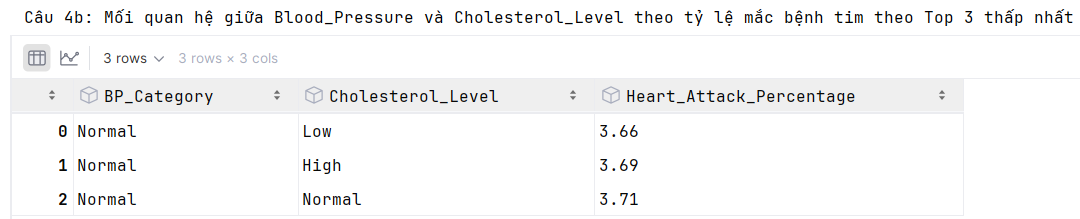
|  |
| --- |
| *#Câu 4: Mối quan hệ giữa Blood\_Pressure và Heart\_Attack theo Cholesterol\_Level*  print("Câu 4a: Mối quan hệ giữa Blood\_Pressure và Cholesterol\_Level theo tỷ lệ mắc bệnh tim theo Top 3 cao nhất")  *# Truy vấn Top 3 cao nhất*  df\_4a = spark.sql(f"""  SELECT  CASE  WHEN Blood\_Pressure < 120 THEN 'Normal'  WHEN Blood\_Pressure BETWEEN 120 AND 129 THEN 'Elevated'  WHEN Blood\_Pressure BETWEEN 130 AND 139 THEN 'Stage 1 Hypertension'  WHEN Blood\_Pressure >= 140 THEN 'Stage 2 Hypertension'  END AS BP\_Category,  Cholesterol\_Level,  ROUND(100.0 \* COUNT(\*) / {total\_heart\_attack}, 2) AS Heart\_Attack\_Percentage  FROM heart\_attack  WHERE Heart\_Attack = 'Yes'  GROUP BY BP\_Category, Cholesterol\_Level  ORDER BY Heart\_Attack\_Percentage DESC  LIMIT 3  """).toPandas()  df\_4a |

Kết quả đầu ra:



|  |
| --- |
| # Truy vấn Top 3 thấp nhất  print("Câu 4b: Mối quan hệ giữa Blood\_Pressure và Cholesterol\_Level theo tỷ lệ mắc bệnh tim theo Top 3 thấp nhất")  df\_4b = spark.sql(f"""  SELECT  CASE  WHEN Blood\_Pressure < 120 THEN 'Normal'  WHEN Blood\_Pressure BETWEEN 120 AND 129 THEN 'Elevated'  WHEN Blood\_Pressure BETWEEN 130 AND 139 THEN 'Stage 1 Hypertension'  WHEN Blood\_Pressure >= 140 THEN 'Stage 2 Hypertension'  END AS BP\_Category,  Cholesterol\_Level,  ROUND(100.0 \* COUNT(\*) / {total\_heart\_attack}, 2) AS Heart\_Attack\_Percentage  FROM heart\_attack  WHERE Heart\_Attack = 'Yes'  GROUP BY BP\_Category, Cholesterol\_Level  ORDER BY Heart\_Attack\_Percentage ASC  LIMIT 3  """).toPandas()  df\_4b |

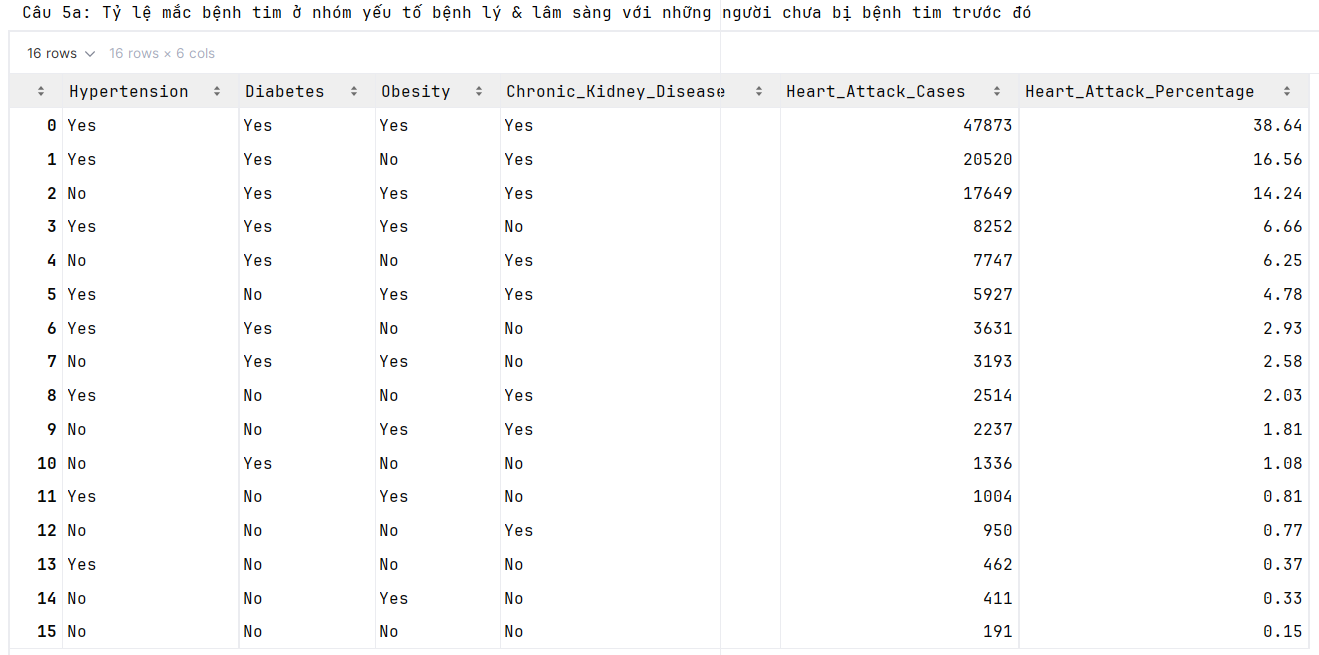
Kết quả đầu ra:



Câu 5a: Thống kê tỷ lệ mắc bệnh tim theo các tổ hợp bệnh nền khác nhau đối với những người chưa bị bệnh tim trước đó.

|  |
| --- |
| *# Tính tổng những người bị nhồi máu tim lần đầu*  total\_first\_heart\_attack = spark.sql("""  SELECT COUNT(\*) as cnt  FROM heart\_attack  WHERE Heart\_Attack = 'Yes' AND Previous\_Heart\_Attack = 'No'  """).collect()[0]['cnt']  *# Phân tích tổ hợp 4 bệnh lý — chỉ người bị nhồi máu cơ tim lần đầu*  print("Câu 5a: Tỷ lệ mắc bệnh tim ở nhóm yếu tố bệnh lý & lâm sàng với những người chưa bị bệnh tim trước đó")  df\_5a=spark.sql(f"""  SELECT  Hypertension,  Diabetes,  Obesity,  Chronic\_Kidney\_Disease,  COUNT(\*) AS Heart\_Attack\_Cases,  ROUND(100.0 \* COUNT(\*) / {total\_first\_heart\_attack}, 2) AS Heart\_Attack\_Percentage  FROM heart\_attack  WHERE Heart\_Attack = 'Yes' AND Previous\_Heart\_Attack = 'No'  GROUP BY Hypertension, Diabetes, Obesity, Chronic\_Kidney\_Disease  ORDER BY Heart\_Attack\_Percentage DESC  """).toPandas()  df\_5a |

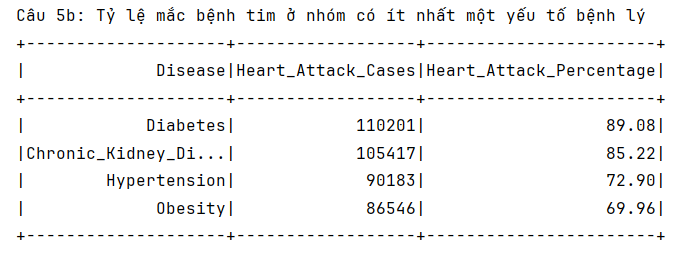
Kết quả đầu ra:



Câu 5b: Thống kê số lượng người bị mắc bệnh tim lần đầu và có ít nhất một bệnh lý.

|  |
| --- |
| *# Tổng số người bị nhồi máu cơ tim lần đầu và có ít nhất một bệnh lý*  total\_with\_condition = spark.sql("""  SELECT COUNT(\*) as cnt  FROM heart\_attack  WHERE Heart\_Attack = 'Yes'  AND Previous\_Heart\_Attack = 'No'  AND (  Hypertension = 'Yes' OR  Diabetes = 'Yes' OR  Obesity = 'Yes' OR  Chronic\_Kidney\_Disease = 'Yes'  )  """).collect()[0]['cnt']  print("Câu 5b: Tỷ lệ mắc bệnh tim ở nhóm có ít nhất một yếu tố bệnh lý")  spark.sql(f"""  SELECT  'Hypertension' AS Disease,  COUNT(\*) AS Heart\_Attack\_Cases,  ROUND(100.0 \* COUNT(\*) / {total\_with\_condition}, 2) AS Heart\_Attack\_Percentage  FROM heart\_attack  WHERE Heart\_Attack = 'Yes' AND Previous\_Heart\_Attack = 'No' AND Hypertension = 'Yes'  UNION ALL  SELECT  'Diabetes' AS Disease,  *COUNT*(*\**) AS Heart\_Attack\_Cases,  *ROUND*(100.0 \* *COUNT*(*\**) / {total\_with\_condition}, 2) AS Heart\_Attack\_Percentage  FROM heart\_attack  WHERE Heart\_Attack = 'Yes' AND Previous\_Heart\_Attack = 'No' AND Diabetes = 'Yes'  UNION ALL  SELECT  'Obesity' AS Disease,  *COUNT*(*\**) AS Heart\_Attack\_Cases,  *ROUND*(100.0 \* *COUNT*(*\**) / {total\_with\_condition}, 2) AS Heart\_Attack\_Percentage  FROM heart\_attack  WHERE Heart\_Attack = 'Yes' AND Previous\_Heart\_Attack = 'No' AND Obesity = 'Yes'  UNION ALL  SELECT  'Chronic\_Kidney\_Disease' AS Disease,  *COUNT*(*\**) AS Heart\_Attack\_Cases,  *ROUND*(100.0 \* *COUNT*(*\**) / {total\_with\_condition}, 2) AS Heart\_Attack\_Percentage  FROM heart\_attack  WHERE Heart\_Attack = 'Yes' AND Previous\_Heart\_Attack = 'No' AND Chronic\_Kidney\_Disease = 'Yes'  ORDER BY Heart\_Attack\_Percentage DESC  """).show() |

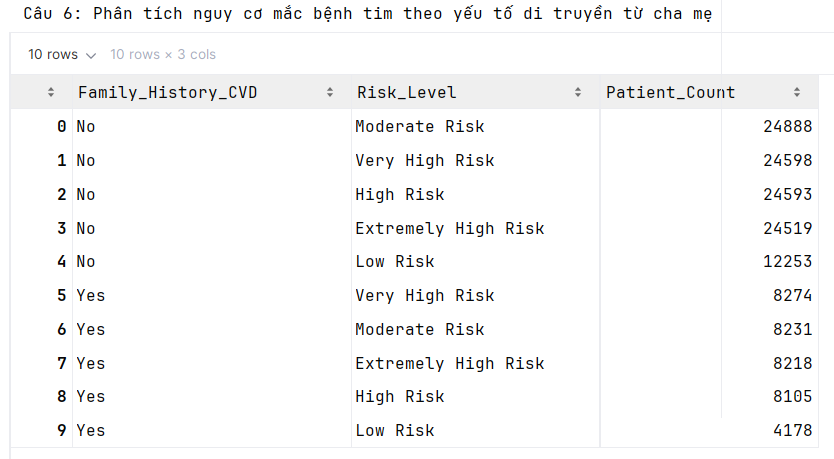
Kết quả đầu ra:



Câu 6: Phân tích nguy cơ mắc bệnh tim dựa trên yếu tố di truyền từ gia đình và chỉ số rủi ro mắc bệnh tim.

|  |
| --- |
| *#Câu 6: Phân tích nguy cơ mắc bệnh tim do di truyền từ gia đình và chỉ số rủi ro*  print("Câu 6: Phân tích nguy cơ mắc bệnh tim theo yếu tố di truyền từ gia đình và chỉ số rủi ro")  df\_6=spark.sql("""  SELECT  Family\_History\_CVD,  Risk\_Level,  COUNT(\*) AS Patient\_Count  FROM (  SELECT  Family\_History\_CVD,  CASE  WHEN CVD\_Risk\_Score BETWEEN 10 AND 19 THEN 'Low Risk'  WHEN CVD\_Risk\_Score BETWEEN 20 AND 39 THEN 'Moderate Risk'  WHEN CVD\_Risk\_Score BETWEEN 40 AND 59 THEN 'High Risk'  WHEN CVD\_Risk\_Score BETWEEN 60 AND 79 THEN 'Very High Risk'  ELSE 'Extremely High Risk'  END AS Risk\_Level  FROM heart\_attack  WHERE Heart\_Attack = 'Yes' OR Previous\_Heart\_Attack = 'Yes'  ) sub  GROUP BY Family\_History\_CVD, Risk\_Level  ORDER BY Patient\_Count DESC  """).toPandas()  df\_6 |

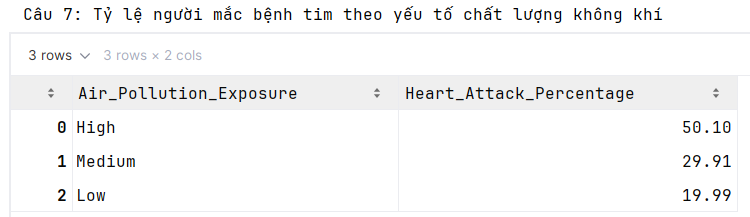
Kết quả đầu ra:



Câu 7: Liệt kê top 10 khu vực và chất lượng không khí có tỷ lệ bệnh nhân bị bệnh tim cao nhất.

|  |
| --- |
| *#Câu 7: Top 10 khu vực và chất lượng không khí có nhiều bệnh nhân bị bệnh tim nhất*  print("Câu 7: Tỷ lệ người mắc bệnh tim theo yếu tố chất lượng không khí")  df\_7=spark.sql("""  WITH heart\_attack\_filtered AS (  SELECT \*  FROM heart\_attack  WHERE Heart\_Attack = 'Yes' AND Air\_Pollution\_Exposure IN ('Low', 'Medium', 'High')  ),  total\_cases AS (  SELECT COUNT(\*) AS total FROM heart\_attack\_filtered  )  SELECT  Air\_Pollution\_Exposure,  ROUND(100.0 \* COUNT(\*) / SUM(COUNT(\*)) OVER (), 2) AS Heart\_Attack\_Percentage  FROM heart\_attack\_filtered  GROUP BY Air\_Pollution\_Exposure  ORDER BY Heart\_Attack\_Percentage DESC  """).toPandas()  df\_7 |

Kết quả đầu ra:



Câu 8: Phân tích tỷ lệ mắc bệnh tim theo nhóm tiếp cận y tế & chăm sóc sức khỏe.

|  |
| --- |
| *#Câu 8:*  print("Câu 8: Phân tích tỷ lệ mắc bệnh tim theo nhóm tiếp cận y tế & chăm sóc sức khỏe")  df\_8=spark.sql(f"""  SELECT  Healthcare\_Access,  Hospital\_Availability,  COUNT(\*) AS Heart\_Attack\_Cases,  ROUND(100.0 \* COUNT(\*) / {total\_heart\_attack}, 2) AS Heart\_Attack\_Percentage  FROM heart\_attack  WHERE Heart\_Attack = 'Yes'  GROUP BY Healthcare\_Access, Hospital\_Availability  ORDER BY Heart\_Attack\_Percentage DESC  """).toPandas()  df\_8 |

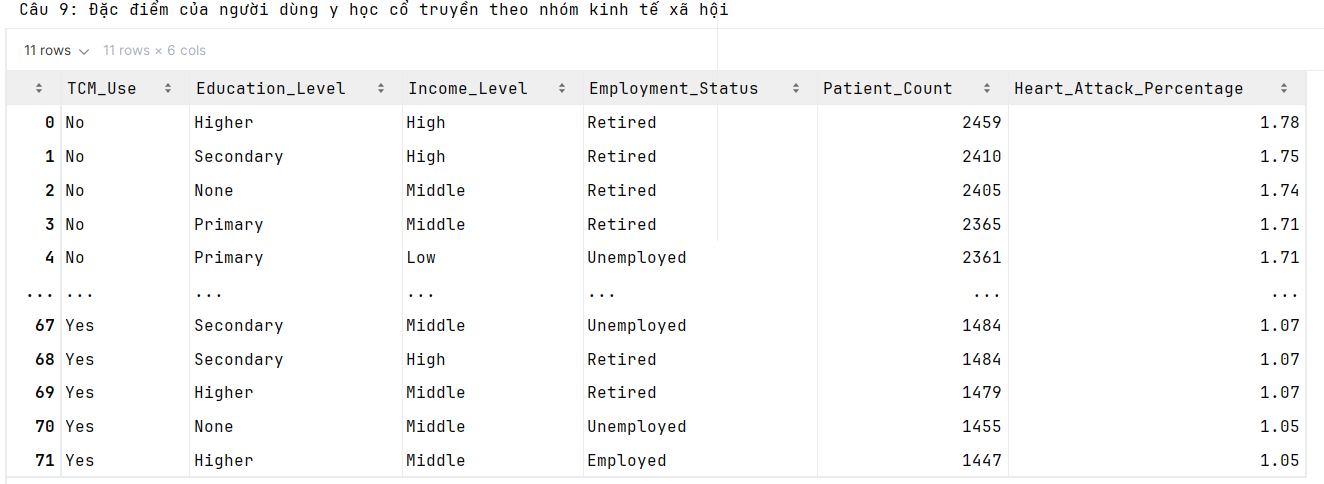
Kết quả đầu ra:



Câu 9: Đặc điểm của người dùng y học cổ truyền và sự tương quan với nguy cơ mắc bệnh tim khi dùng hoặc không dùng y học cổ truyền.

|  |
| --- |
| *#Câu 9*  print("Câu 9: Đặc điểm của người dùng y học cổ truyền theo nhóm kinh tế xã hội")  df\_9=spark.sql(f"""  SELECT  TCM\_Use,  Education\_Level,  Income\_Level,  Employment\_Status,  COUNT(\*) AS Patient\_Count,  ROUND(  100.0 \* COUNT(\*) / {total\_heart\_attack},  2  ) AS Heart\_Attack\_Percentage  FROM heart\_attack  WHERE Heart\_Attack = 'Yes'  GROUP BY TCM\_Use, Education\_Level, Income\_Level, Employment\_Status  ORDER BY Heart\_Attack\_Percentage DESC  """).toPandas()  df\_9 |

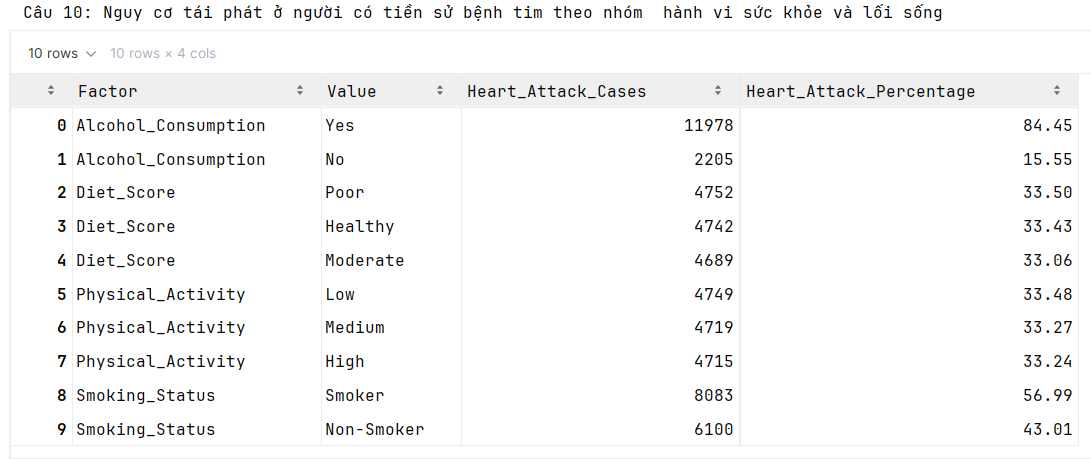
Kết quả đầu ra:



Câu 10: Nguy cơ tái phát ở người có tiền sử bệnh tim nếu họ sử dụng bia, rượu và có thói quen xấu.

|  |
| --- |
| *#Câu 10: Nguy cơ tái phát ở người có tiền sử bệnh tim nếu họ sử dụng bia, rượu và có thói quen xấu*  *# Tính tổng số người có Heart\_Attack = 'Yes'*  print("Câu 10: Nguy cơ tái phát ở người có tiền sử bệnh tim theo nhóm hành vi sức khỏe và lối sống")  *# Truy vấn số ca theo từng nhóm (tử số)*  df\_10 = spark.sql(f"""  SELECT \*  FROM (  SELECT 'Smoking\_Status' AS Factor,  Smoking\_Status AS Value,  COUNT(\*) AS Heart\_Attack\_Cases,  ROUND(100.0 \* COUNT(\*) / SUM(COUNT(\*)) OVER (PARTITION BY 'Smoking\_Status'), 2) AS Heart\_Attack\_Percentage  FROM heart\_attack  WHERE Heart\_Attack = 'Yes' and Previous\_Heart\_Attack = 'Yes'  GROUP BY Smoking\_Status  UNION ALL  SELECT 'Alcohol\_Consumption' AS Factor,  Alcohol\_Consumption AS Value,  COUNT(\*) AS Heart\_Attack\_Cases,  ROUND(100.0 \* COUNT(\*) / SUM(COUNT(\*)) OVER (PARTITION BY 'Alcohol\_Consumption'), 2) AS Heart\_Attack\_Percentage  FROM heart\_attack  WHERE Heart\_Attack = 'Yes' and Previous\_Heart\_Attack = 'Yes'  GROUP BY Alcohol\_Consumption  UNION ALL  SELECT 'Physical\_Activity' AS Factor,  Physical\_Activity AS Value,  COUNT(\*) AS Heart\_Attack\_Cases,  ROUND(100.0 \* COUNT(\*) / SUM(COUNT(\*)) OVER (PARTITION BY 'Physical\_Activity'), 2) AS Heart\_Attack\_Percentage  FROM heart\_attack  WHERE Heart\_Attack = 'Yes' and Previous\_Heart\_Attack = 'Yes'  GROUP BY Physical\_Activity  UNION ALL  SELECT 'Diet\_Score' AS Factor,  Diet\_Score AS Value,  COUNT(\*) AS Heart\_Attack\_Cases,  ROUND(100.0 \* COUNT(\*) / SUM(COUNT(\*)) OVER (PARTITION BY 'Diet\_Score'), 2) AS Heart\_Attack\_Percentage  FROM heart\_attack  WHERE Heart\_Attack = 'Yes' and Previous\_Heart\_Attack = 'Yes'  GROUP BY Diet\_Score  ) AS union\_data  ORDER BY Factor ASC, Heart\_Attack\_Percentage DESC  """).toPandas()  df\_10 |

Kết quả đầu ra:



**3.5. Các bước thực hiện mô hình học máy (giám sát và không giám sát) sử dụng SQL MLlib**

***3.5.1. Phân cụm***

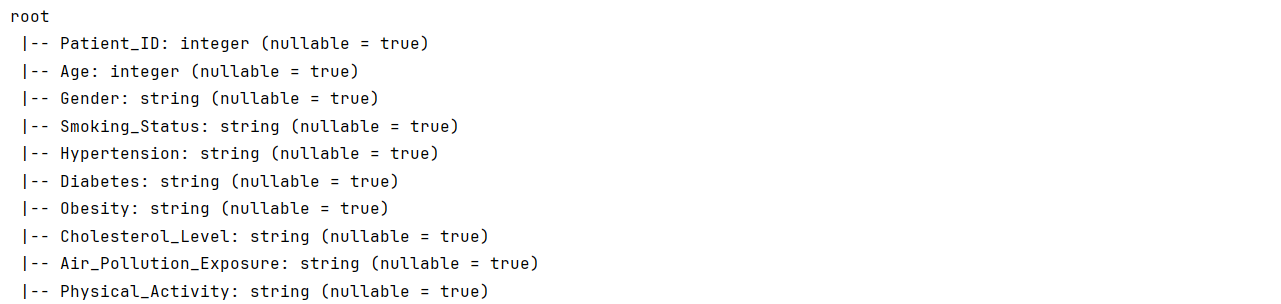
*3.5.1.1. Khởi tạo môi trường Spark*

|  |
| --- |
| *from pyspark.sql import SparkSession*  *import os*  *# Tạo thư mục tạm nếu chưa có*  *temp\_dir = "D:/spark\_temp"*  *os.makedirs(temp\_dir, exist\_ok=True)*  *# Khởi tạo SparkSession*  *spark = SparkSession.builder \*  *.appName("Heart Attack KMeans Clustering") \*  *.config("spark.local.dir", temp\_dir) \*  *.getOrCreate()* |

*3.5.1.2. Đọc và phân tích sơ bộ bộ dữ liệu*

|  |
| --- |
| df = spark.read.csv("D:/data/heart\_attack\_china.csv", header=True, inferSchema=True)  df.printSchema()  df.show(5)  *# Loại bỏ cột không cần thiết*  target\_col = "Heart\_Attack"  drop\_cols = ["Patient\_ID", target\_col]  feature\_cols = [c for c in df.columns if c not in drop\_cols]  *# Phân loại cột*  categorical\_cols = [c for c, t in df.dtypes if t == 'string' and c in feature\_cols]  numeric\_cols = [c for c in feature\_cols if c not in categorical\_cols]  print(f"Categorical columns: {categorical\_cols}")  print(f"Numeric columns: {numeric\_cols}") |

Kết quả đầu ra

**

*3.5.1.3. Xử lý đặc trưng*

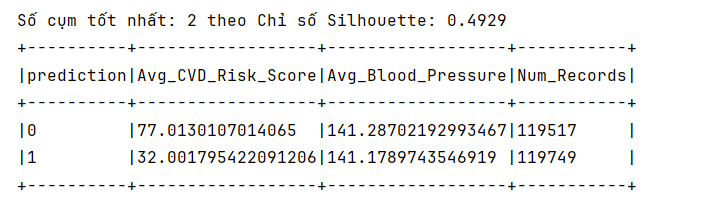
|  |
| --- |
| from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler, VectorIndexer  from pyspark.ml import Pipeline  *# Index & OneHotEncoder cho cột phân loại*  indexers = [StringIndexer(inputCol=c, outputCol=c+"\_idx", handleInvalid="skip") for c in categorical\_cols]  encoders = [OneHotEncoder(inputCols=[c+"\_idx" for c in categorical\_cols],  outputCols=[c+"\_vec" for c in categorical\_cols])]  *# Assemble toàn bộ features*  assembler\_inputs = [c + "\_vec" for c in categorical\_cols] + numeric\_cols  assembler = VectorAssembler(inputCols=assembler\_inputs, outputCol="features")  *# Index cho vector đặc trưng*  featureIndexer = VectorIndexer(inputCol="features", outputCol="indexedFeatures", maxCategories=4)  *# Pipeline xử lý đặc trưng*  stages = indexers + encoders + [assembler, featureIndexer]  pipeline = Pipeline(stages=stages)  *# Biến đổi dữ liệu*  pipeline\_model = pipeline.fit(df)  data = pipeline\_model.transform(df).select("indexedFeatures") |

*3.5.1.4. Chọn số cụm tối ưu cho thuật toán KMeans*

|  |
| --- |
| from pyspark.ml.evaluation import ClusteringEvaluator  from pyspark.ml.clustering import KMeans  import numpy as np  import matplotlib.pyplot as plt  from matplotlib.ticker import MaxNLocator  def find\_optimal\_k(data, feature\_col="indexedFeatures", k\_min=2, k\_max=10, runs=3):  sil\_scores = []  k\_values = list(range(k\_min, k\_max+1))  evaluator = ClusteringEvaluator(featuresCol=feature\_col, metricName="silhouette", distanceMeasure="squaredEuclidean")  for k in k\_values:  run\_scores = []  for \_ in range(runs):  kmeans = KMeans(featuresCol=feature\_col, k=k, seed=np.random.randint(10000))  model = kmeans.fit(data)  preds = model.transform(data)  score = evaluator.evaluate(preds)  run\_scores.append(score)  sil\_scores.append(np.mean(run\_scores))  *# Vẽ biểu đồ Silhouette*  plt.figure(figsize=(8,6))  plt.plot(k\_values, sil\_scores, marker='o', color='red')  plt.xlabel("Số cụm")  plt.ylabel("Chỉ số Silhouette")  plt.title("Chỉ số Silhouette theo số cụm")  plt.gca().xaxis.set\_major\_locator(MaxNLocator(integer=True))  plt.show()  best\_k = k\_values[np.argmax(sil\_scores)]  print(f"Số cụm tốt nhất: {best\_k} theo Chỉ số Silhouette: {max(sil\_scores):.4f}")  return best\_k  best\_k = find\_optimal\_k(data, k\_min=2, k\_max=10, runs=3) |

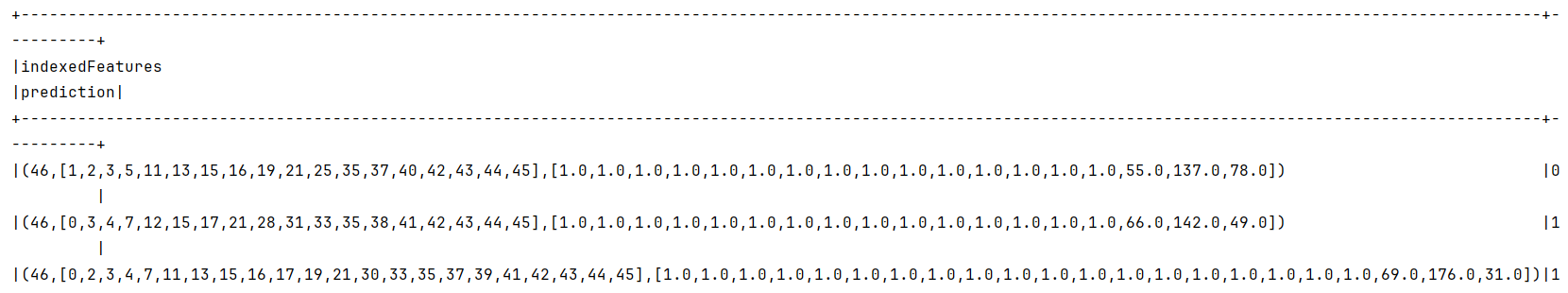
*3.5.1.5. Hiển thị thông tin sau khi đã phân cụm bằng KMeans.*

|  |
| --- |
| best\_k = find\_optimal\_k(data, k\_min=2, k\_max=10, runs=3)  from pyspark.sql.functions import avg, count  *# Huấn luyện mô hình với số cụm tốt nhất*  kmeans = KMeans(featuresCol="indexedFeatures", k=best\_k, seed=42)  model = kmeans.fit(data)  *# Gán cụm vào dữ liệu đầy đủ*  full\_data = pipeline\_model.transform(df)  predictions = model.transform(full\_data)  *# Phân tích cụm theo CVD\_Risk\_Score và Blood\_Pressure*  analyzed = predictions.select("prediction", "CVD\_Risk\_Score", "Blood\_Pressure")  cluster\_summary = analyzed.groupBy("prediction") \  .agg(  avg("CVD\_Risk\_Score").alias("Avg\_CVD\_Risk\_Score"),  avg("Blood\_Pressure").alias("Avg\_Blood\_Pressure"),  count("\*").alias("Num\_Records")  ) \  .orderBy("prediction")  *# Hiển thị kết quả*  cluster\_summary.show(truncate=False) |

Kết quả đầu ra: **

*3.5.1.6. Hiển thị kết quả phân cụm*

|  |
| --- |
| from pyspark.ml.clustering import KMeans  kmeans = KMeans(featuresCol="indexedFeatures", k=best\_k, seed=42)  model = kmeans.fit(data)  result = model.transform(data)  result.select("indexedFeatures", "prediction").show(10, truncate=False)  *# Hiển thị tâm cụm*  print("Cluster centers:")  for center in model.clusterCenters():  print(center) |

Kết quả đầu ra: **

***3.5.2. Phân lớp***

* *Logistic Regression*

*3.5.2.1. Khởi tạo pyspark và nhập các thư viện cần thiết*

|  |
| --- |
| from pyspark.sql import SparkSession  from pyspark.sql.functions import when, col  from pyspark.ml.feature import StringIndexer, VectorAssembler  from pyspark.ml import Pipeline  from pyspark.ml.classification import LogisticRegression  from pyspark.ml.evaluation import BinaryClassificationEvaluator  from pyspark.sql.types import IntegerType |

*3.5.2.2. Tải dữ liệu*

|  |
| --- |
| from pyspark.sql import SparkSession  from pyspark.sql.functions import when, col  *# 1. Tạo SparkSession, dùng thư mục tạm đã tạo ở D:/spark\_temp*  spark = SparkSession.builder \  .appName("HeartAttackPrediction") \  .config("spark.local.dir", "D:/spark\_temp") \  .getOrCreate()  *# 2. Đọc dữ liệu*  data\_path = 'D:\\data\\heart\_attack\_china.csv'  df = spark.read \  .option("header", "True") \  .option("inferSchema", "True") \  .option("sep", ",") \  .csv(data\_path)  print("There are", df.count(), "rows and", len(df.columns), "columns in the data.")  *# 3. Chuyển cột label Heart\_Attack sang số (0 hoặc 1)*  df = df.withColumn("Heart\_Attack", when(col("Heart\_Attack") == "Yes", 1).otherwise(0)) |

Kết quả đầu ra:

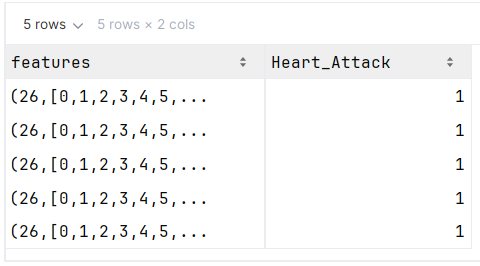


*3.5.2.3. Chuyển đổi dữ liệu*

|  |
| --- |
| *# Lọc các cột dạng số (int, double, float, long)*  numeric\_cols = [col for col, dtype in df.dtypes if dtype in ['int', 'double', 'float', 'bigint']]  *# Bỏ 1 cột đầu*  numeric\_cols\_trimmed = numeric\_cols[1:]  *# Describe*  df.select(numeric\_cols\_trimmed).describe().show() |

|  |
| --- |
| from pyspark.ml.feature import StringIndexer  from pyspark.ml.feature import VectorAssembler  from pyspark.ml import Pipeline  *# 3. Danh sách đặc trưng đầu vào (không bao gồm label)*  indep\_features = [  "Age", "Gender", "Smoking\_Status", "Hypertension", "Diabetes", "Obesity",  "Cholesterol\_Level", "Air\_Pollution\_Exposure", "Physical\_Activity", "Diet\_Score",  "Stress\_Level", "Alcohol\_Consumption", "Family\_History\_CVD", "Healthcare\_Access",  "Rural\_or\_Urban", "Region", "Province", "Hospital\_Availability", "TCM\_Use",  "Employment\_Status", "Education\_Level", "Income\_Level", "Blood\_Pressure",  "Chronic\_Kidney\_Disease", "Previous\_Heart\_Attack", "CVD\_Risk\_Score"  ]  *# 4. Xác định các cột kiểu chuỗi cần encode*  categorical\_cols = [c for c, t in df.dtypes if t == "string" and c in indep\_features]  *# 5. Tạo StringIndexer cho các cột phân loại*  indexers = [  StringIndexer(inputCol=feature, outputCol=feature + "\_idx", handleInvalid="keep")  for feature in categorical\_cols  ]  *# 6. Tạo danh sách cột đầu vào cho VectorAssembler*  final\_features = [  feature + "\_idx" if feature in categorical\_cols else feature  for feature in indep\_features  ]  *# 7. VectorAssembler kết hợp đặc trưng*  assembler = VectorAssembler(inputCols=final\_features, outputCol="features")  *# 8. Tạo pipeline*  pipeline = Pipeline(stages=indexers + [assembler])  *# 9. Fit & transform pipeline*  pipeline\_model = pipeline.fit(df)  df\_transformed = pipeline\_model.transform(df)  *# 10. Chọn cột đầu ra*  df\_final = df\_transformed.select("features", "Heart\_Attack")  *# 11. Chia train/test*  train\_data, test\_data = df\_final.randomSplit([0.8, 0.2], seed=42)  *# 12. Xem vài dòng kết quả*  train\_data.show(5) |

Kết quả đầu ra:



*3.5.2.4. Mô hình dự đoán Logistic Regression*

|  |
| --- |
| from pyspark.ml.classification import LogisticRegression  *# Đặt tên biến label*  target\_feature = "Heart\_Attack"  *# Khởi tạo mô hình Logistic Regression*  lr = LogisticRegression(  featuresCol="features",  labelCol=target\_feature,  predictionCol="prediction",  rawPredictionCol="rawPrediction",  probabilityCol="probability",  maxIter=10  )  *# Huấn luyện mô hình và dự đoán*  lr\_model = lr.fit(train\_data)  lr\_predictions = lr\_model.transform(test\_data) |

*3.5.2.5. ROC và biểu đồ ROC curve*

|  |
| --- |
| from pyspark.ml.evaluation import BinaryClassificationEvaluator  evaluator = BinaryClassificationEvaluator(  labelCol=target\_feature,  rawPredictionCol="rawPrediction",  metricName="areaUnderROC"  )  *# Evaluate the model*  evaluator = BinaryClassificationEvaluator(labelCol=target\_feature)  area\_under\_curve = evaluator.evaluate(lr\_predictions)  print(f"Area under ROC curve: {area\_under\_curve}") |

|  |
| --- |
| from matplotlib import pyplot as plt  *# Lấy training summary từ mô hình đã huấn luyện*  trainingSummary = lr\_model.summary  *# Chuyển dữ liệu ROC về pandas để vẽ*  lrROC = trainingSummary.roc.toPandas()  *# Vẽ biểu đồ ROC*  plt.figure(figsize=(8, 6))  plt.plot(lrROC['FPR'], lrROC['TPR'], label='ROC Curve', color='red')  plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guessing')  plt.xlabel('False Positive Rate')  plt.ylabel('True Positive Rate')  plt.title('ROC Curve - Training Set')  plt.grid(True)  plt.legend()  plt.show()  *# In chỉ số AUC trên tập huấn luyện*  print('Training set areaUnderROC: {:.4f}'.format(trainingSummary.areaUnderROC)) |

*3.5.2.6. Biểu đồ Recall-Precision*

|  |
| --- |
| pr = trainingSummary.pr.toPandas()  plt.plot(pr['recall'], pr['precision'], color='red')  plt.ylabel('Precision')  plt.xlabel('Recall')  plt.grid(True)  plt.title("Precision-Recall Curve")  plt.show() |

*3.5.2.7. Các số liệu đánh giá*

|  |
| --- |
| from pyspark.sql.functions import column as col  *# Calculate true positives, true negatives, false positives, false negatives*  tp = lr\_predictions.filter((col(target\_feature) == 1) & (col('prediction') == 1)).count()  tn = lr\_predictions.filter((col(target\_feature) == 0) & (col('prediction') == 0)).count()  fp = lr\_predictions.filter((col(target\_feature) == 0) & (col('prediction') == 1)).count()  fn = lr\_predictions.filter((col(target\_feature) == 1) & (col('prediction') == 0)).count()  *# Calculate accuracy*  accuracy = (tp + tn) / (tp + tn + fp + fn)  print(f"Accuracy: {accuracy}")  *# Calculate precision*  precision = tp / (tp + fp) if (tp + fp) != 0 else 0  print(f"Precision: {precision}")  *# Calculate recall*  recall = tp / (tp + fn) if (tp + fn) != 0 else 0.0  print(f"Recall: {recall}")  *# Calculate F1 measure*  f1\_measure = 2 \* (precision \* recall) / (precision + recall) if (precision + recall) != 0 else 0.0  print(f"F1 measure: {f1\_measure}") |

*3.5.2.8. Ma trận nhầm lẫn*

|  |
| --- |
| import pandas as pd  import seaborn as sns  import matplotlib.pyplot as plt  *# Convert kết quả thành pandas DataFrame*  preds\_df = lr\_predictions.select("prediction", "Heart\_Attack").toPandas()  *# Confusion matrix*  conf\_matrix = pd.crosstab(preds\_df["Heart\_Attack"], preds\_df["prediction"], rownames=["Actual"], colnames=["Predicted"])  *# Vẽ*  plt.figure(figsize=(6, 4))  sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Reds')  plt.title("Confusion Matrix - Logistic Regression")  plt.show() |

*3.5.2.9. Thực hiện tương tự với mô hình Decision Tree và SVM*

* *Decision Tree*

|  |
| --- |
| from pyspark.sql import SparkSession  from pyspark.sql.functions import when, col  from pyspark.ml.feature import StringIndexer, VectorAssembler  from pyspark.ml import Pipeline  from pyspark.ml.classification import LogisticRegression  from pyspark.ml.evaluation import BinaryClassificationEvaluator  from pyspark.sql.types import IntegerType  from pyspark.sql import SparkSession  from pyspark.sql.functions import when, col  *# 1. Tạo SparkSession đã tạo ở D:/spark\_temp*  spark = SparkSession.builder \  .appName("HeartAttackPrediction") \  .config("spark.local.dir", "D:/spark\_temp") \  .getOrCreate()  *# 2. Đọc dữ liệu*  data\_path = 'D:\\data\\heart\_attack\_china.csv'  df = spark.read \  .option("header", "True") \  .option("inferSchema", "True") \  .option("sep", ",") \  .csv(data\_path)  print("There are", df.count(), "rows and", len(df.columns), "columns in the data.")  *# 3. Chuyển cột label Heart\_Attack sang số (0 hoặc 1)*  df = df.withColumn("Heart\_Attack", when(col("Heart\_Attack") == "Yes", 1).otherwise(0))  from pyspark.ml.feature import StringIndexer  from pyspark.ml.feature import VectorAssembler  from pyspark.ml import Pipeline  *# 3. Danh sách đặc trưng đầu vào*  indep\_features = [  "Age", "Gender", "Smoking\_Status", "Hypertension", "Diabetes", "Obesity",  "Cholesterol\_Level", "Air\_Pollution\_Exposure", "Physical\_Activity", "Diet\_Score",  "Stress\_Level", "Alcohol\_Consumption", "Family\_History\_CVD", "Healthcare\_Access",  "Rural\_or\_Urban", "Region", "Province", "Hospital\_Availability", "TCM\_Use",  "Employment\_Status", "Education\_Level", "Income\_Level", "Blood\_Pressure",  "Chronic\_Kidney\_Disease", "Previous\_Heart\_Attack", "CVD\_Risk\_Score"  ]  *# 4. Xác định các cột kiểu chuỗi cần encode*  categorical\_cols = [c for c, t in df.dtypes if t == "string" and c in indep\_features]  *# 5. Tạo StringIndexer cho các cột phân loại*  indexers = [  StringIndexer(inputCol=feature, outputCol=feature + "\_idx", handleInvalid="keep")  for feature in categorical\_cols  ]  *# 6. Tạo danh sách cột đầu vào cho VectorAssembler*  final\_features = [  feature + "\_idx" if feature in categorical\_cols else feature  for feature in indep\_features  ]  *# 7. VectorAssembler kết hợp đặc trưng*  assembler = VectorAssembler(inputCols=final\_features, outputCol="features")  *# 8. Tạo pipeline*  pipeline = Pipeline(stages=indexers + [assembler])  *# 9. Fit & transform pipeline*  pipeline\_model = pipeline.fit(df)  df\_transformed = pipeline\_model.transform(df)  *# 10. Chọn cột đầu ra*  df\_final = df\_transformed.select("features", "Heart\_Attack")  *# 11. Chia train/test*  train\_data, test\_data = df\_final.randomSplit([0.8, 0.2], seed=42)  *# 12. Xem vài dòng kết quả*  train\_data.show(5)  from pyspark.ml.classification import DecisionTreeClassifier  *# Khởi tạo mô hình Decision Tree*  dt = DecisionTreeClassifier(featuresCol="features", labelCol="Heart\_Attack", maxDepth=5)  *# Huấn luyện mô hình trên tập train*  dt\_model = dt.fit(train\_data)  *# Dự đoán trên tập test*  dt\_predictions = dt\_model.transform(test\_data)  from pyspark.ml.evaluation import BinaryClassificationEvaluator  *# Đánh giá mô hình Decision Tree*  evaluator = BinaryClassificationEvaluator(labelCol="Heart\_Attack")  area\_under\_curve = evaluator.evaluate(dt\_predictions)  print(f"Area under ROC curve: {area\_under\_curve}")  from pyspark.sql.functions import col  tp = dt\_predictions.filter((col("Heart\_Attack") == 1) & (col('prediction') == 1)).count()  tn = dt\_predictions.filter((col("Heart\_Attack") == 0) & (col('prediction') == 0)).count()  fp = dt\_predictions.filter((col("Heart\_Attack") == 0) & (col('prediction') == 1)).count()  fn = dt\_predictions.filter((col("Heart\_Attack") == 1) & (col('prediction') == 0)).count()  accuracy = (tp + tn) / (tp + tn + fp + fn)  print(f"Accuracy: {accuracy}")  precision = tp / (tp + fp) if (tp + fp) != 0 else 0  print(f"Precision: {precision}")  recall = tp / (tp + fn) if (tp + fn) != 0 else 0.0  print(f"Recall: {recall}")  f1\_measure = 2 \* (precision \* recall) / (precision + recall) if (precision + recall) != 0 else 0.0  print(f"F1 measure: {f1\_measure}") |

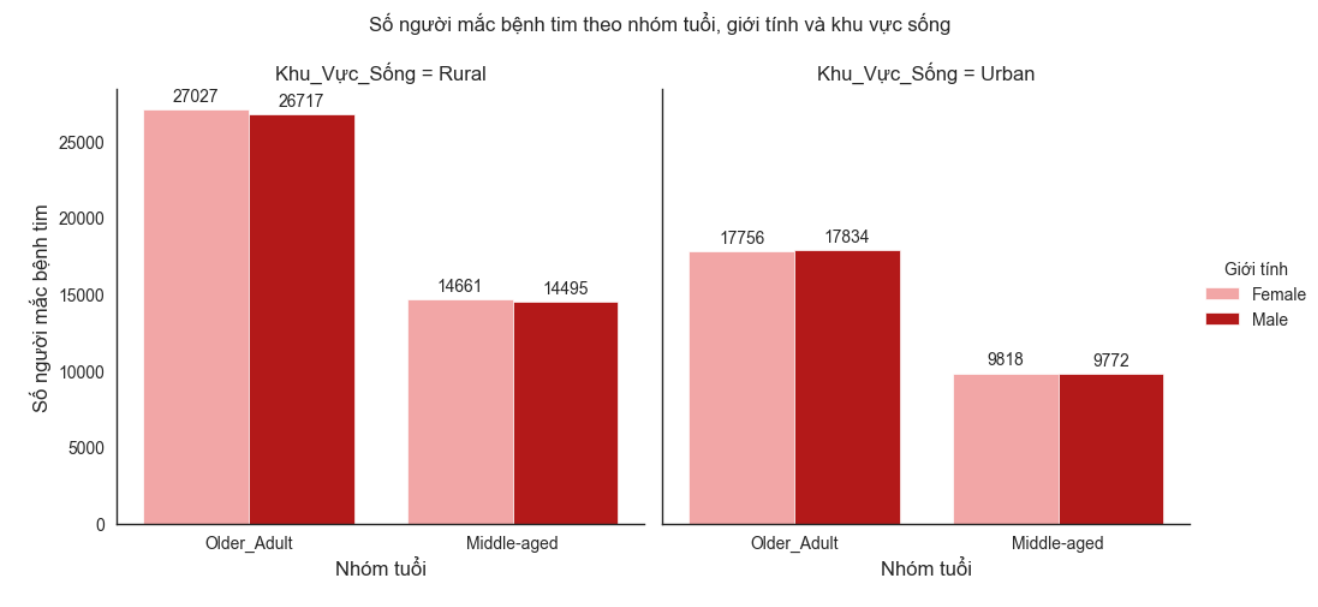
* *SVM*

|  |
| --- |
| from pyspark.sql import SparkSession  from pyspark.sql.functions import when, col  from pyspark.ml.feature import StringIndexer, VectorAssembler  from pyspark.ml import Pipeline  from pyspark.ml.classification import LinearSVC  from pyspark.ml.evaluation import BinaryClassificationEvaluator  *# 1. Tạo SparkSession*  spark = SparkSession.builder \  .appName("HeartAttackPrediction\_SVM") \  .config("spark.local.dir", "D:/spark\_temp") \  .getOrCreate()  *# 2. Đọc dữ liệu*  data\_path = 'D:\\data\\heart\_attack\_china.csv'  df = spark.read.option("header", "True") \  .option("inferSchema", "True") \  .option("sep", ",") \  .csv(data\_path)  print(f"Data rows: {df.count()}, columns: {len(df.columns)}")  *# 3. Chuyển cột label Heart\_Attack sang số (0 hoặc 1)*  df = df.withColumn("Heart\_Attack", when(col("Heart\_Attack") == "Yes", 1).otherwise(0))  *# 4. Danh sách các đặc trưng input (thay thế bằng cột thực tế trong dataset)*  indep\_features = [  "Age", "Gender", "Smoking\_Status", "Hypertension", "Diabetes", "Obesity",  "Cholesterol\_Level", "Air\_Pollution\_Exposure", "Physical\_Activity", "Diet\_Score",  "Stress\_Level", "Alcohol\_Consumption", "Family\_History\_CVD", "Healthcare\_Access",  "Rural\_or\_Urban", "Region", "Province", "Hospital\_Availability", "TCM\_Use",  "Employment\_Status", "Education\_Level", "Income\_Level", "Blood\_Pressure",  "Chronic\_Kidney\_Disease", "Previous\_Heart\_Attack", "CVD\_Risk\_Score"  ]  *# 5. Xác định các cột categorical cần encode*  categorical\_cols = [c for c, t in df.dtypes if t == "string" and c in indep\_features]  *# 6. Tạo StringIndexer cho các cột categorical*  indexers = [StringIndexer(inputCol=c, outputCol=c + "\_idx", handleInvalid="keep") for c in categorical\_cols]  *# 7. Tạo danh sách features cuối cùng, thay thế categorical bằng cột index*  final\_features = [c + "\_idx" if c in categorical\_cols else c for c in indep\_features]  *# 8. VectorAssembler để gom tất cả features thành 1 vector*  assembler = VectorAssembler(inputCols=final\_features, outputCol="features")  *# 9. Tạo pipeline gồm indexers và assembler*  pipeline = Pipeline(stages=indexers + [assembler])  *# 10. Fit & transform dữ liệu*  pipeline\_model = pipeline.fit(df)  df\_transformed = pipeline\_model.transform(df)  *# 11. Chọn cột features và label*  df\_final = df\_transformed.select("features", "Heart\_Attack")  *# 12. Chia dữ liệu train/test*  train\_data, test\_data = df\_final.randomSplit([0.8, 0.2], seed=42)  *# 13. Khởi tạo mô hình Linear SVC*  svm = LinearSVC(featuresCol="features", labelCol="Heart\_Attack", maxIter=10, regParam=0.1)  *# 14. Huấn luyện mô hình trên train*  svm\_model = svm.fit(train\_data)  *# 15. Dự đoán trên test*  svm\_predictions = svm\_model.transform(test\_data)  *# 16. Đánh giá với AUC*  evaluator = BinaryClassificationEvaluator(labelCol="Heart\_Attack", rawPredictionCol="rawPrediction")  auc = evaluator.evaluate(svm\_predictions)  print(f"Area under ROC curve (SVM): {auc:.4f}")  *# 17. Tính các chỉ số TP, TN, FP, FN*  tp = svm\_predictions.filter((col("Heart\_Attack") == 1) & (col("prediction") == 1)).count()  tn = svm\_predictions.filter((col("Heart\_Attack") == 0) & (col("prediction") == 0)).count()  fp = svm\_predictions.filter((col("Heart\_Attack") == 0) & (col("prediction") == 1)).count()  fn = svm\_predictions.filter((col("Heart\_Attack") == 1) & (col("prediction") == 0)).count()  *# 18. Tính Accuracy, Precision, Recall, F1 Score*  accuracy = (tp + tn) / (tp + tn + fp + fn) if (tp + tn + fp + fn) != 0 else 0  precision = tp / (tp + fp) if (tp + fp) != 0 else 0  recall = tp / (tp + fn) if (tp + fn) != 0 else 0  f1 = 2 \* (precision \* recall) / (precision + recall) if (precision + recall) != 0 else 0  print(f"Accuracy (SVM): {accuracy:.4f}")  print(f"Precision (SVM): {precision:.4f}")  print(f"Recall (SVM): {recall:.4f}")  print(f"F1 Score (SVM): {f1:.4f}")  *# 19. Dừng SparkSession*  spark.stop() |

**CHƯƠNG IV. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ**

**4.1. Ứng dụng Spark SQL trong phân tích nguy cơ mắc bệnh tim mạch**

***4.1.1. Số lượng bệnh nhân mắc bệnh tim phân theo nhóm thông tin và nhân khẩu học***

******

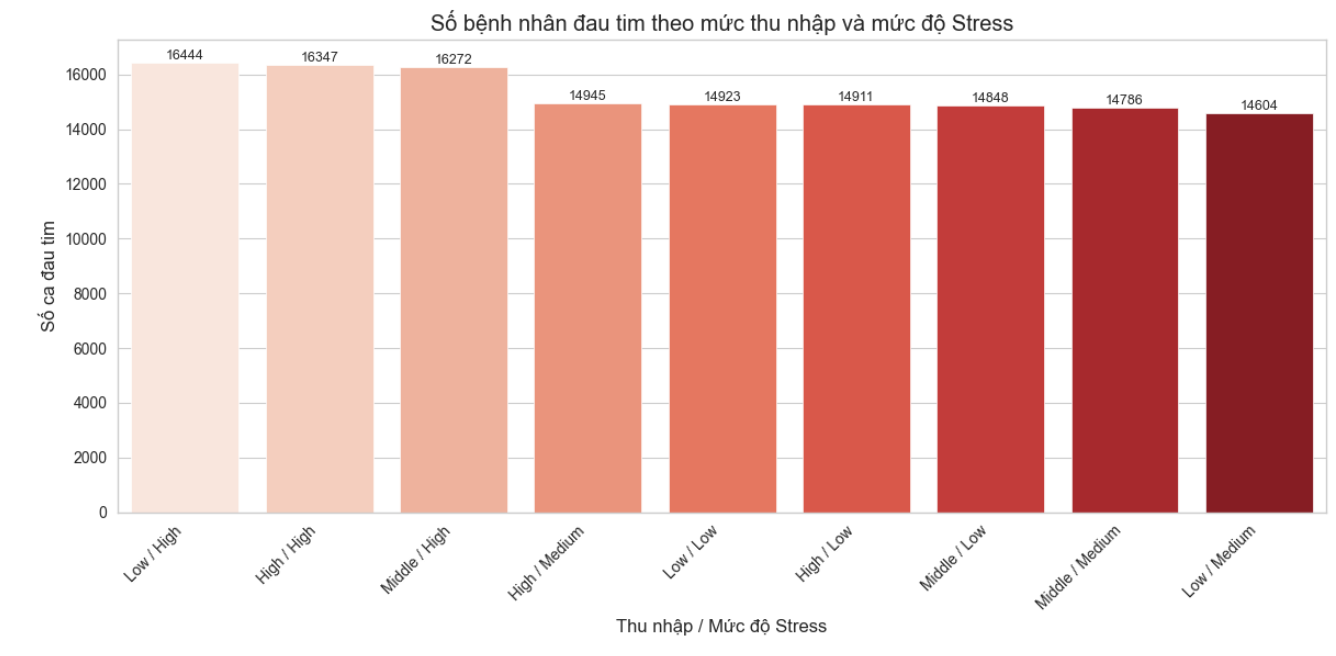
Biểu đồ 4.1: Số lượng người mắc bệnh tim phân theo nhóm tuổi, khu vực sống và giới tính

*Nguồn: Tác giả thực hiện chạy chương trình*

Biểu đồ cho thấy số ca đau tim tập trung nhiều nhất ở nhóm người cao tuổi, đặc biệt là những người sống tại khu vực nông thôn. Đáng chú ý, người cao tuổi ở nông thôn có tỷ lệ mắc bệnh tim cao nhất, với số ca lên tới 27.027 ca ở nữ và 26.717 ca ở nam. So với nhóm trung niên, người cao tuổi ở cả hai khu vực nông thôn và thành thị đều có số ca đau tim vượt trội. Xu hướng chung cũng cho thấy người sống ở nông thôn có nguy cơ mắc bệnh cao hơn người sống ở thành thị, bất kể giới tính. Điều này cho thấy ảnh hưởng kép giữa tuổi tác và điều kiện sống trong việc hình thành nguy cơ tim mạch. Môi trường sống kém thuận lợi, thiếu dịch vụ y tế và thói quen sinh hoạt truyền thống có thể là yếu tố làm gia tăng tỷ lệ mắc bệnh. Đặc biệt, nhiều người cao tuổi ở nông thôn thường sống một mình hoặc thiếu sự chăm sóc y tế thường xuyên, làm trầm trọng thêm nguy cơ. Thêm vào đó, việc chậm phát hiện và điều trị khiến bệnh tiến triển âm thầm cho đến khi xảy ra biến cố nghiêm trọng như đau tim.

Qua đó, ta thấy được rằng sức khỏe tim mạch bị chi phối đáng kể bởi độ tuổi, môi trường sống, bất kể giới tính. Nhóm người cao tuổi, đặc biệt là người sống ở khu vực nông thôn, có nguy cơ mắc bệnh tim cao nhất. Điều này có thể xuất phát từ việc tiếp cận dịch vụ y tế còn hạn chế, thiếu kiến thức phòng bệnh, hoặc những thói quen sinh hoạt không lành mạnh phổ biến ở nông thôn. Từ đó, các chương trình y tế cộng đồng cần tập trung ưu tiên tầm soát, truyền thông và chăm sóc sức khỏe tim mạch cho người cao tuổi ở vùng nông thôn. Việc triển khai các hình thức như khám bệnh lưu động, giáo dục sức khỏe tại chỗ và hỗ trợ y tế định kỳ sẽ góp phần giảm thiểu rủi ro và nâng cao chất lượng cuộc sống cho nhóm dân số dễ bị tổn thương này.

***4.1.2. Số lượng bệnh nhân mắc bệnh tim phân theo mức độ stress và thu nhập***



Biểu đồ 4.2: Số lượng bệnh nhân mắc bệnh tim theo mức độ stress và thu nhập

*Nguồn: Tác giả thực hiện chạy chương trình*

Bảng thống kê số ca đau tim theo mức độ căng thẳng và mức thu nhập cho thấy căng thẳng là yếu tố ảnh hưởng lớn nhất đến nguy cơ mắc bệnh tim. Cụ thể, những người có mức căng thẳng cao ghi nhận số ca đau tim cao nhất, bất kể họ thuộc nhóm thu nhập thấp (16.444 ca), trung bình (16.272 ca) hay cao (16.347 ca) . Trong khi đó, các nhóm có mức căng thẳng trung bình hoặc thấp đều có số ca đau tim thấp hơn rõ rệt, dao động từ 14.604 đến 14.945 ca.

Dữ liệu chỉ ra rằng mức độ căng thẳng có tác động mạnh mẽ hơn thu nhập trong việc quyết định nguy cơ đau tim. Những người có mức căng thẳng cao, dù giàu hay nghèo, đều có số ca đau tim cao hơn hẳn so với những người ít căng thẳng. Trong khi đó, những người có mức thu nhập thấp nhưng căng thẳng ít lại có nguy cơ thấp hơn. Điều này nhấn mạnh rằng việc kiểm soát căng thẳng và chăm sóc sức khỏe tâm thần là yếu tố then chốt trong việc phòng ngừa bệnh tim mạch, đôi khi còn quan trọng hơn cả điều kiện kinh tế. Theo một số nghiên cứu, stress gây ra ba giai đoạn phản ứng sinh lý: báo động, thích nghi, và kiệt quệ. Khi gặp stress, cơ thể tăng nhịp tim, huyết áp, hô hấp và trương lực cơ, ảnh hưởng đến hệ thần kinh và tim mạch.

***4.1.3. Tỷ lệ bệnh nhân mắc bệnh tim theo nhóm hành vi sức khỏe và lối sống***

Bảng 4.1: Tỷ lệ bệnh nhân mắc bệnh tim với các yếu tố tình trạng hút thuốc, tiêu thụ rượu bia, hoạt động thể chất, điểm đánh giá chế độ ăn và mức độ căng thẳng

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Smoking Status** | **Alcohol Consumption** | **Physical Activity** | **Diet Score** | **Stress Level** | **Heart Attack Cases** | **Heart Attack Percentage** |
| Smoker | Yes | Medium | Moderate | High | 2764 | 2.23 |
| Smoker | Yes | Low | Moderate | High | 2710 | 2.19 |
| Smoker | Yes | High | Moderate | High | 2700 | 2.18 |
| Smoker | Yes | Medium | Poor | High | 2686 | 2.17 |
| Smoker | Yes | Low | Healthy | High | 2683 | 2.17 |
| Smoker | Yes | High | Healthy | High | 2641 | 2.13 |
| Smoker | Yes | High | Poor | High | 2645 | 2.13 |
| Smoker | Yes | Low | Poor | High | 2610 | 2.11 |
| Smoker | Yes | Medium | Healthy | High | 2592 | 2.90 |
| Smoker | Yes | Low | Healthy | Low | 2433 | 1.96 |

*Nguồn: Tác giả thực hiện chạy chương trình*

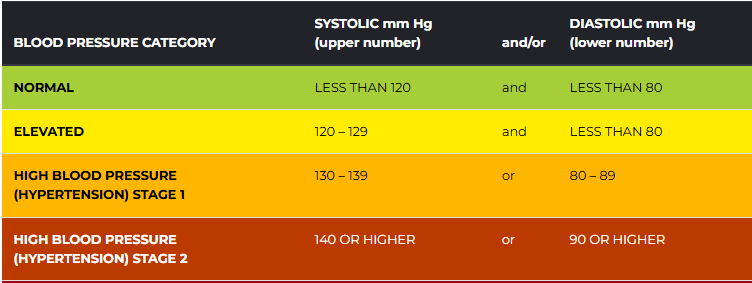
Dựa trên bảng dữ liệu, có thể nhận thấy rằng nhóm các hành vi sức khỏe và lối sống đều ảnh hưởng đáng kể đến tỷ lệ mắc bệnh tim. Đặc biệt tất cả các cá nhân trong bảng đều là người hút thuốc và có tiêu thụ rượu, đây vốn là hai yếu tố nguy cơ cao, đã dẫn đến sự chênh lệch rõ rệt trong tỷ lệ đau tim. Rượu bia và hút thuốc đều ảnh hưởng mạnh đến bệnh tim mạch nhưng theo cách khác nhau. Uống rượu vừa phải (3-14 ly/tuần) có thể giảm nguy cơ nhồi máu cơ tim và suy tim sung huyết, trong khi uống nhiều rượu (3 ly/ngày trở lên) và hút thuốc làm tăng nguy cơ cao huyết áp, đột quỵ, tăng triglyceride và các bệnh tim mạch khác. Tránh hút thuốc và hạn chế rượu sẽ giúp giảm đáng kể nguy cơ bệnh tim mạch.

Một lần nữa chỉ số căng thẳng chính là yếu tố đặc biệt quan trọng, ảnh hưởng mạnh mẽ đến nguy cơ đau tim. Đồng thời, chế độ ăn uống hợp lý và hoạt động thể chất đều đặn cũng đóng vai trò hỗ trợ quan trọng trong việc làm giảm nguy cơ này.

Đáng chú ý, trường hợp duy nhất có mức độ căng thẳng thấp, kết hợp với chế độ ăn lành mạnh, ghi nhận tỷ lệ đau tim thấp nhất là 1.96%. Diet Score cũng cho thấy vai trò quan trọng khi những người có chế độ ăn lành mạnh (Healthy) thường có tỷ lệ đau tim thấp hơn so với người ăn uống kém (Poor) hay trung bình (Moderate**)**. Ngoài ra, hoạt động thể chất lại có xu hướng tỷ lệ nghịch với nguy cơ đau tim khi những người vận động cao thường có tỷ lệ thấp hơn so với nhóm vận động thấp hoặc trung bình. Nhiều nghiên cứu đã chỉ ra rằng những người thường xuyên tập thể dục có nguy cơ thấp hơn bị đau tim đột ngột hoặc các sự cố tim mạch nguy hiểm đến tính mạng khác. Mặc dù tập luyện mang lại nhiều lợi ích riêng, cách tốt nhất để phòng ngừa bệnh tim là kết hợp vận động thường xuyên với chế độ ăn uống lành mạnh.

Tóm lại, nhóm hành vi sức khỏe và lối sống có tác động rõ rệt đến tỷ lệ mắc bệnh tim. Việc cải thiện các yếu tố như chế độ ăn, vận động và kiểm soát căng thẳng không chỉ giúp nâng cao chất lượng cuộc sống mà còn góp phần giảm thiểu nguy cơ bệnh tim mạch, cũng như hạn chế rượu bia và hút thuốc.

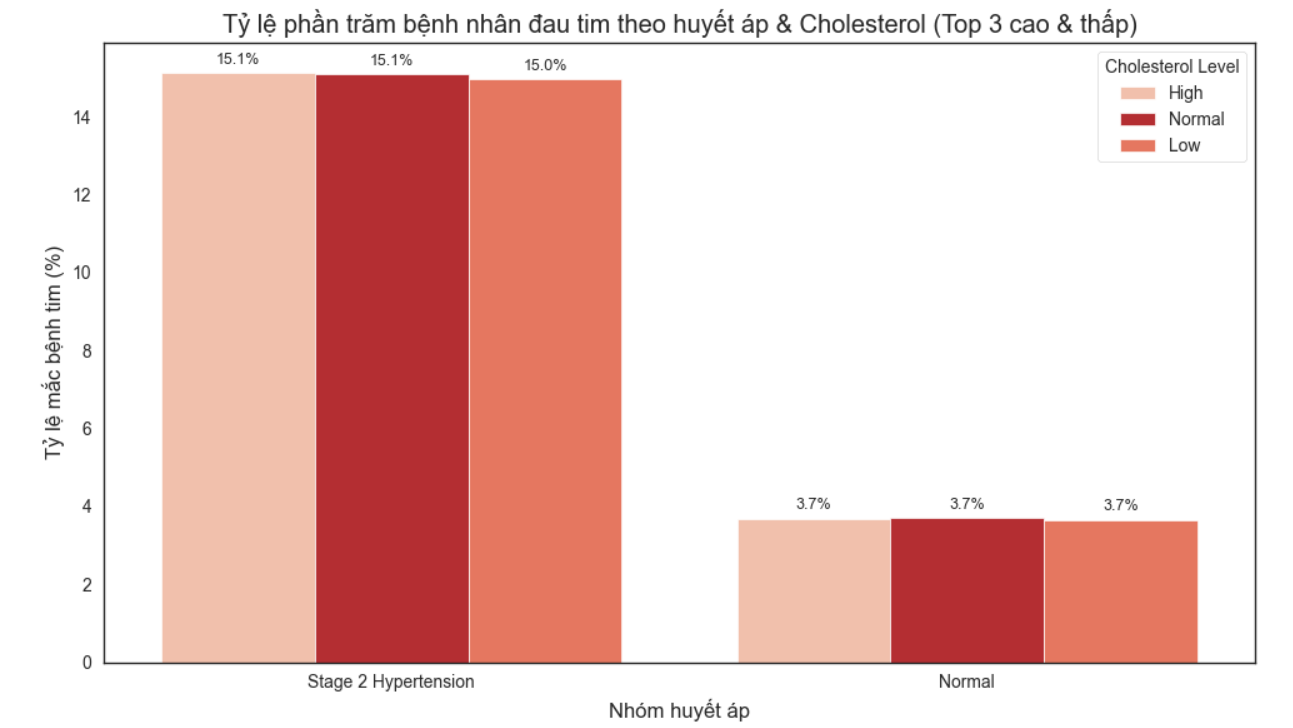
***4.1.4 Tỷ lệ bệnh nhân mắc bệnh tim theo nhóm yếu tố bệnh lý & lâm sàng***

******

Hình 4.1: Bảng phân loại huyết áp chuẩn

*Nguồn: American Heart Association*

Theo hướng dẫn y tế, nếu huyết áp của bệnh nhân nằm trong ngưỡng bình thường, cần tiếp tục duy trì các thói quen có lợi cho tim mạch, bao gồm chế độ ăn uống cân bằng và tập luyện thể dục thường xuyên. Đối với những trường hợp huyết áp tăng nhẹ, người bệnh cần được theo dõi và phối hợp với chuyên gia y tế để kiểm soát tình trạng này kịp thời. Trong giai đoạn Tăng huyết áp độ 1, ưu tiên hàng đầu là điều chỉnh lối sống. Việc sử dụng thuốc có thể được cân nhắc dựa trên mức độ nguy cơ mắc bệnh tim mạch hoặc đột quỵ. Đặc biệt, nếu bệnh nhân có kèm theo các bệnh nền như tiểu đường, suy tim hoặc bệnh thận, bác sĩ nên chỉ định điều trị bằng thuốc. Ở giai đoạn Tăng huyết áp độ 2, cần kết hợp điều trị bằng thuốc hạ huyết áp cùng với thay đổi lối sống nhằm đạt hiệu quả kiểm soát huyết áp tối ưu và giảm thiểu nguy cơ biến chứng tim mạch.



Biểu đồ 4.3: Mối quan hệ giữa Blood Pressure và Cholesterol Level phân theo tỷ lệ mắc bệnh tim

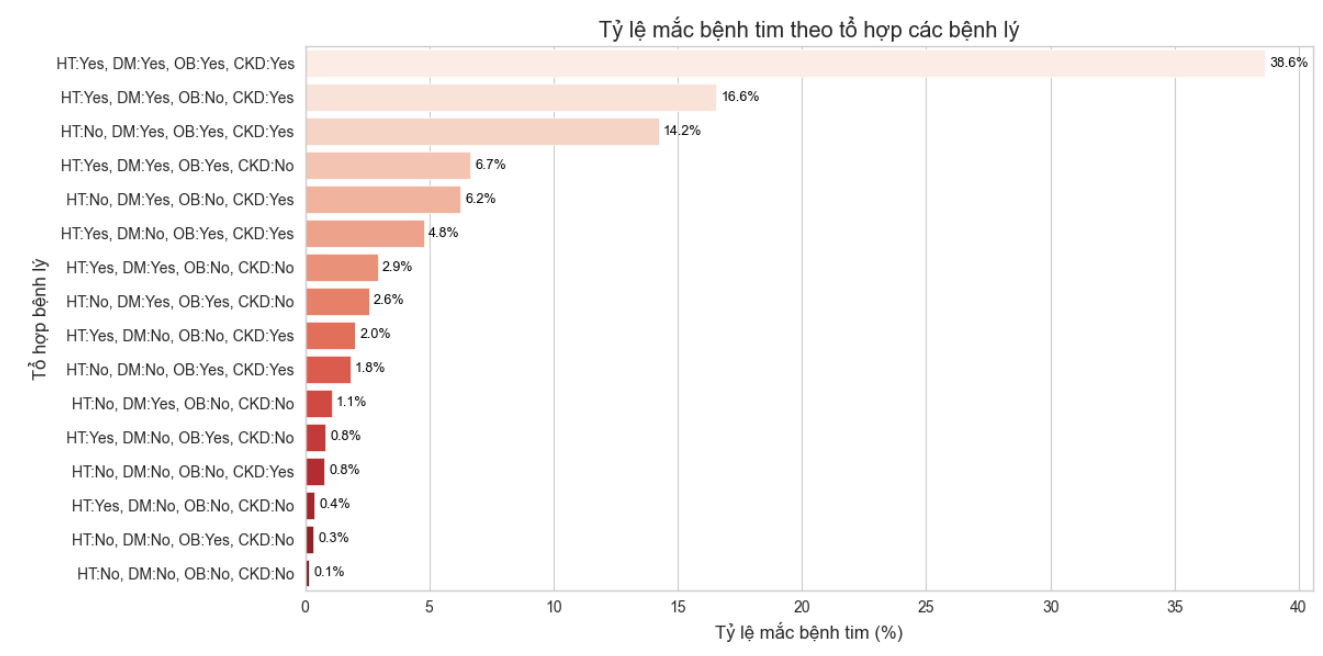
*Nguồn: Tác giả thực hiện chạy chương trình*

Mối quan hệ giữa huyết áp, mức cholesterol và tỷ lệ mắc bệnh tim cho thấy nhóm người thuộc giai đoạn Tăng huyết áp độ 2 (Stage 2 Hypertension) có nguy cơ mắc bệnh tim cao nhất, bất kể mức cholesterol là cao, bình thường hay thấp. Cụ thể, tỷ lệ mắc bệnh tim ở nhóm có cholesterol cao là 15.14%, tiếp theo là nhóm có cholesterol bình thường với 15.09%, và nhóm có cholesterol thấp là 14.97%. Mức chênh lệch giữa các nhóm cholesterol là không đáng kể, cho thấy huyết áp cao đóng vai trò chi phối mạnh trong việc gia tăng nguy cơ bệnh tim, ngay cả khi các yếu tố khác như cholesterol đã được kiểm soát. Do đó, quá trình kiểm soát huyết áp đặc biệt ở giai đoạn nặng cần được xem là ưu tiên hàng đầu trong các chiến lược phòng ngừa bệnh tim.

Ngược lại với nhóm tăng huyết áp, những người có huyết áp bình thường ghi nhận tỷ lệ mắc bệnh tim thấp hơn đáng kể. Trong nhóm này, sự khác biệt giữa các mức cholesterol là rất nhỏ. Cụ thể, tỷ lệ mắc bệnh tim dao động từ 3.66% ở nhóm có cholesterol thấp đến 3.71% ở nhóm cholesterol bình thường – mức cao nhất trong ba nhóm. Điều này cho thấy khi huyết áp được kiểm soát ở mức bình thường, cholesterol không phải là yếu tố quyết định rõ rệt đến nguy cơ mắc bệnh tim. Tuy nhiên, tỷ lệ dù thấp vẫn có xu hướng tăng nhẹ ở nhóm cholesterol bình thường và cao, cho thấy việc duy trì cả huyết áp và cholesterol ở mức lý tưởng vẫn là cần thiết để phòng tránh rủi ro.

***4.1.5. Tỷ lệ mắc bệnh tim mạch theo yếu tố bệnh lý và lâm sàng với những người chưa bị bệnh tim trước đó***

*4.1.5.1. Tỷ lệ mắc bệnh tim mạch theo yếu tố bệnh lý và lâm sàng với những người chưa bị bệnh tim trước đó*



Biểu đồ 4.4: Tỷ lệ mắc bệnh tim ở nhóm yếu tố bệnh lý và lâm sàng với những người chưa bị bệnh tim trước đó

*Nguồn: Tác giả thực hiện chạy chương trình*

Bảng dữ liệu cung cấp cái nhìn toàn diện về mối liên hệ giữa các yếu tố bệnh lý nền và nguy cơ mắc bệnh tim ở những người chưa từng có tiền sử tim mạch. Kết quả cho thấy, sự hiện diện đồng thời của các yếu tố như tăng huyết áp, tiểu đường, béo phì và bệnh thận mãn tính làm gia tăng rõ rệt nguy cơ mắc bệnh tim. Cụ thể, nhóm có đầy đủ cả bốn yếu tố bệnh lý ghi nhận tỷ lệ mắc bệnh cao nhất, đạt 38.64%, trong khi nhóm không có bất kỳ yếu tố nào chỉ ghi nhận tỷ lệ 0.15%, phản ánh mức độ ảnh hưởng sâu sắc của các bệnh lý nền đến nguy cơ tim mạch.

Khi xem xét từng yếu tố một cách riêng biệt, tiểu đường là yếu tố có tác động mạnh nhất. Nhóm chỉ mắc tiểu đường có tỷ lệ mắc bệnh là 1.08%, cao hơn so với nhóm chỉ có bệnh thận mãn tính (0.77%), béo phì (0.33%) hoặc tăng huyết áp (0.37%). Điều này cho thấy tiểu đường, khi xuất hiện độc lập, là yếu tố đơn lẻ có nguy cơ cao nhất trong bốn bệnh lý được khảo sát.

Ngoài ra, khi các yếu tố kết hợp, nguy cơ mắc bệnh tim tăng theo cấp số. Nhóm có ba bệnh lý tăng huyết áp, tiểu đường và bệnh thận mãn tính (không béo phì) có tỷ lệ mắc bệnh là 16.56%, trong khi nhóm có hai bệnh lý là tăng huyết áp và tiểu đường (không có béo phì và bệnh thận) có tỷ lệ 2.93%. Sự gia tăng này phản ánh xu hướng cộng hưởng nguy cơ khi nhiều bệnh lý nền cùng tồn tại.

Kết quả này nhấn mạnh vai trò của việc phát hiện và kiểm soát sớm các bệnh lý mãn tính, đặc biệt là tiểu đường, nhằm hạn chế nguy cơ tiến triển thành bệnh tim mạch. Việc can thiệp y tế kịp thời đối với những người có một hoặc nhiều yếu tố nguy cơ sẽ góp phần nâng cao hiệu quả phòng ngừa trong cộng đồng.

*4.1.5.2. Tỷ lệ bệnh nhân mắc bệnh tim ở nhóm có ít nhất một yếu tố bệnh lý và lâm sàng với những người chưa bị bệnh tim trước đó*

Bảng 4.2: Tỷ lệ bệnh nhân mắc bệnh tim ở nhóm có ít nhất một yếu tố bệnh lý và lâm sàng với những người chưa bị bệnh tim trước đó

|  |  |  |
| --- | --- | --- |
| **Disease** | **Heart Attack Cases** | **Heart Attack Percentage** |
| Diabetes | 110201 | 89.08 |
| Chronic Kidney Disease | 105417 | 85.22 |
| Hypertension | 90183 | 72.90 |
| Obesity | 86546 | 69.96 |

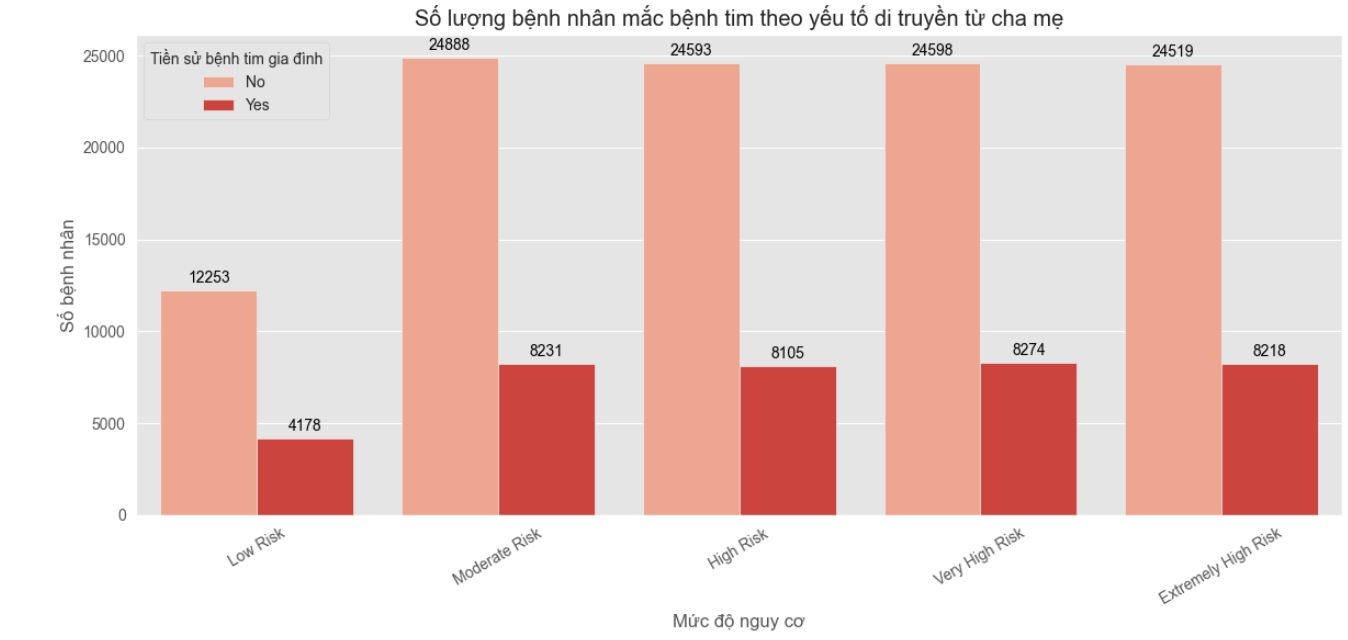
*Nguồn: Tác giả thực hiện chạy chương trình*

Phân tích số liệu cho thấy sự hiện diện của các bệnh lý nền có liên quan mật thiết đến nguy cơ mắc bệnh tim. Trong đó, tiểu đường nổi bật là yếu tố nguy cơ có tác động lớn nhất, với tỷ lệ bệnh nhân bị nhồi máu cơ tim lên đến 89.08% (tương ứng 110,201 trường hợp). Đứng kế tiếp là bệnh thận mạn tính với tỷ lệ 85.22% (105,417 ca), cho thấy mối liên hệ chặt chẽ giữa tổn thương chức năng thận và nguy cơ tim mạch.

Tăng huyết áp cũng thể hiện vai trò quan trọng khi chiếm 72.90% tổng số trường hợp mắc bệnh tim (90,183 ca). Mặc dù béo phì là một yếu tố thường được nhắc đến trong dự phòng tim mạch, tuy nhiên tỷ lệ ghi nhận ở nhóm này thấp hơn so với các yếu tố còn lại, ở mức 69.96% (86,546 ca).

Kết quả này làm rõ mức độ đóng góp của từng yếu tố bệnh lý trong nguy cơ phát triển bệnh tim, qua đó nhấn mạnh sự cần thiết của việc sàng lọc và can thiệp sớm ở những người có bệnh lý chuyển hóa hoặc tổn thương cơ quan đích. Đặc biệt, kiểm soát đường huyết và chức năng thận nên được ưu tiên trong các chương trình phòng chống bệnh tim mạch nhằm giảm gánh nặng lên hệ thống y tế và tăng hiệu quả điều trị trong cộng đồng.

***4.1.6. Số bệnh nhân mắc bệnh tim do di truyền từ cha mẹ và mức độ rủi ro hiện tại***

**

Biểu đồ 4.5: Mối quan hệ giữa tiền sử gia đình mắc bệnh tim và mức độ rủi ro hiện tại với số người mắc bệnh về tim

*Nguồn: Tác giả thực hiện chạy chương trình*

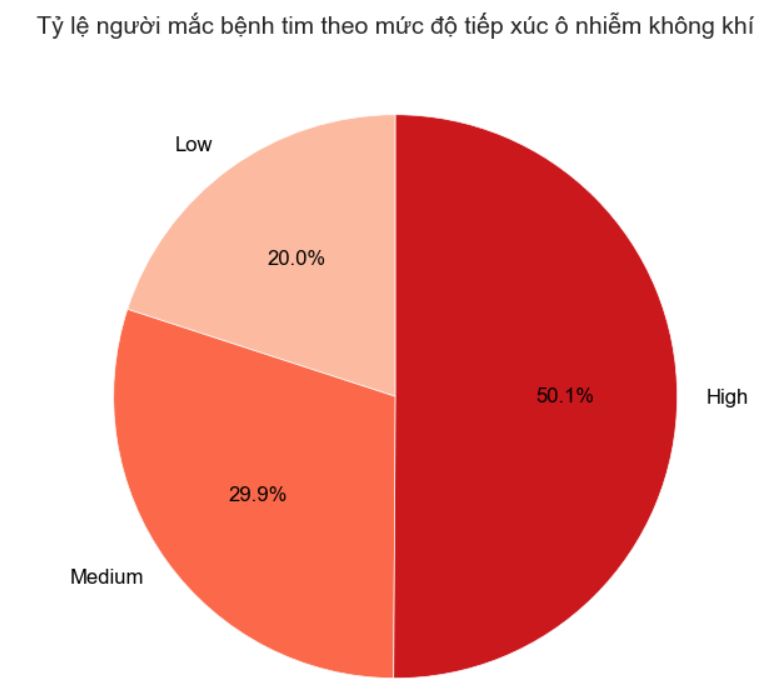
Mối liên hệ giữa tiền sử gia đình mắc bệnh tim và mức độ rủi ro hiện tại cho thấy một kết luận đáng chú ý là mức độ rủi ro lâm sàng, được đo bằng chỉ số CVD Risk Score mới là yếu tố ảnh hưởng đáng kể đến xác suất mắc bệnh tim mạch. Trong khi yếu tố di truyền không tạo ra sự khác biệt rõ nét như giả định phổ biến.

Cụ thể, ở cả hai nhóm bệnh nhân có và không có tiền sử gia đình mắc bệnh tim, số lượng bệnh nhân phân bố rất tương đồng theo từng mức độ rủi ro. Các mức “Moderate Risk”, “High Risk”, “Very High Risk” và “Extremely High Risk” đều ghi nhận số lượng lớn bệnh nhân từ hơn 8.000 người ở nhóm có tiền sử đến hơn 24.000 ở nhóm không có tiền sử. Trong khi đó, nhóm “Low Risk” chỉ chiếm một phần nhỏ tổng số bệnh nhân (khoảng 4.000 đến 12.000 người). Điều này cho thấy rằng sự khác biệt về mặt di truyền không đủ lớn để ảnh hưởng đáng kể đến phân bố bệnh nhân theo mức độ rủi ro.

Ngoài ra, nhóm không có tiền sử gia đình thậm chí ghi nhận số lượng bệnh nhân ở các mức rủi ro cao còn lớn hơn về mặt tuyệt đối so với nhóm có tiền sử. Thực tế này càng củng cố lập luận rằng yếu tố di truyền không phải là điều kiện quyết định và việc phân tầng bệnh nhân theo chỉ số rủi ro hiện tại mang tính phản ánh chính xác hơn về tình trạng sức khỏe tim mạch thực tế.

Từ góc nhìn phân tích dữ liệu, kết quả này gợi mở một định hướng quan trọng cho công tác phòng ngừa và kiểm soát bệnh tim mạch. Cụ thể, các chương trình tầm soát và can thiệp nên tập trung vào việc đánh giá các yếu tố rủi ro hiện tại có thể đo lường và can thiệp được như huyết áp, cholesterol, tình trạng hút thuốc, chế độ ăn uống và mức độ vận động thể chất. Việc ưu tiên những yếu tố này sẽ góp phần xây dựng chiến lược y tế chủ động, chính xác và cá nhân hóa hơn, thay vì quá phụ thuộc vào tiền sử gia đình.

***4.1.7. Nguy cơ mắc bệnh tim theo yếu tố chất lượng không khí***



Biểu đồ 4.6: Nguy cơ mắc bệnh tim theo yếu tố chất lượng không khí

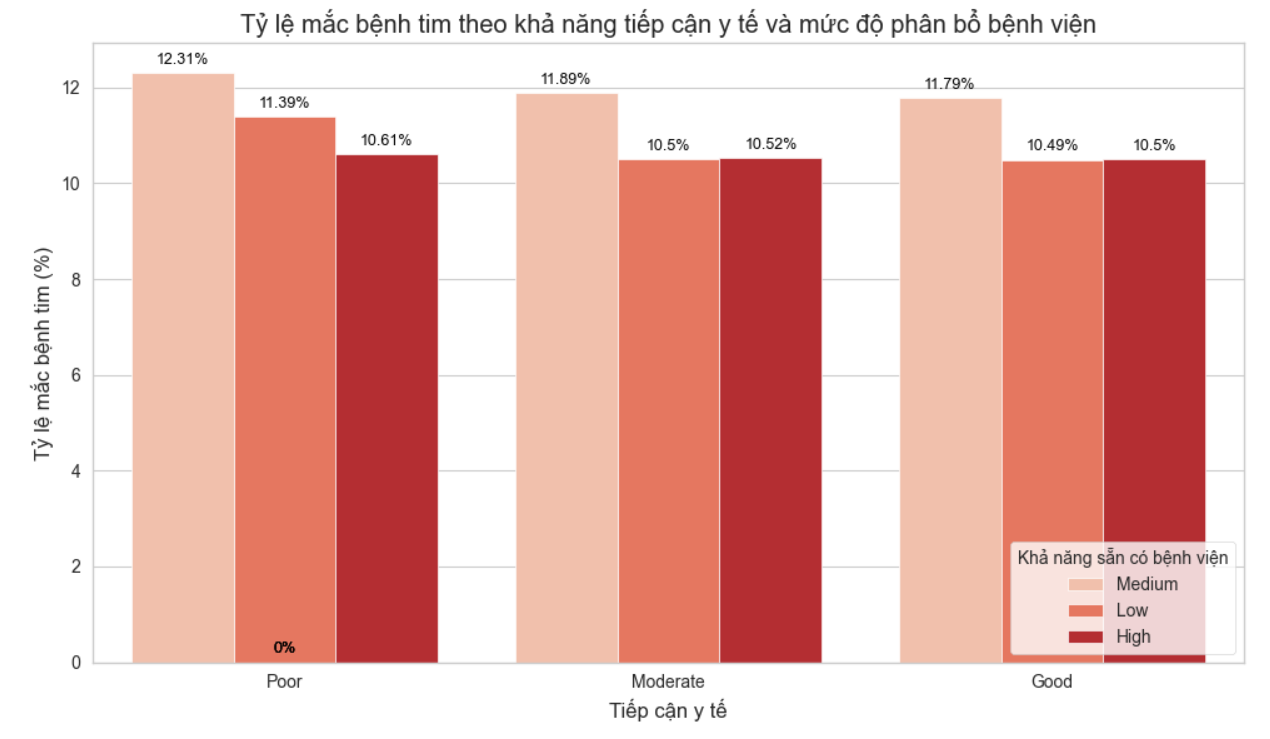
*Nguồn: Tác giả thực hiện chạy chương trình*

Số liệu đã làm nổi bật mối quan hệ đáng kể giữa mức độ phơi nhiễm với ô nhiễm không khí và nguy cơ mắc bệnh tim mạch. Cụ thể, tỷ lệ bệnh nhân mắc bệnh tim trong nhóm tiếp xúc với ô nhiễm không khí ở mức cao lên tới 50,10%, cao gấp khoảng 2,5 lần so với nhóm sống trong môi trường có mức độ ô nhiễm thấp (19,99%). Điều này chỉ ra rằng chất lượng không khí là một yếu tố nguy cơ có ảnh hưởng trực tiếp và rõ rệt đến sức khỏe tim mạch.

Ngoài ra, nhóm tiếp xúc ở mức trung bình cũng ghi nhận tỷ lệ mắc bệnh đáng kể, ở mức 29,91%, nằm giữa hai nhóm còn lại nhưng vẫn cho thấy tác động tiêu cực rõ ràng từ ô nhiễm không khí. Sự khác biệt tỷ lệ giữa ba nhóm cung cấp bằng chứng mạnh mẽ cho thấy ô nhiễm không khí không chỉ đơn thuần là một yếu tố môi trường, mà còn là một tác nhân gây bệnh tim mạch độc lập, với mức độ ảnh hưởng tỷ lệ thuận theo mức phơi nhiễm.

Từ góc độ y tế công cộng, kết quả này cho thấy cần đặt trọng tâm vào các chính sách kiểm soát và cải thiện chất lượng không khí, đặc biệt tại các khu vực đô thị và công nghiệp. Đồng thời, việc kết hợp đánh giá yếu tố môi trường vào các chương trình sàng lọc và phòng ngừa bệnh tim mạch là điều cần thiết nhằm xây dựng một chiến lược y tế toàn diện và bền vững hơn.

***4.1.8. Mối quan hệ giữa khả năng tiếp nhận y tế và mức độ phân bổ bệnh viện với tỉ lệ mắc bệnh về tim***

**

Biểu đồ 4.7: Mối quan hệ giữa khả năng tiếp nhận y tế và mức độ phân bổ bệnh viện với tỉ lệ mắc bệnh về tim

*Nguồn: Tác giả thực hiện chạy chương trình*

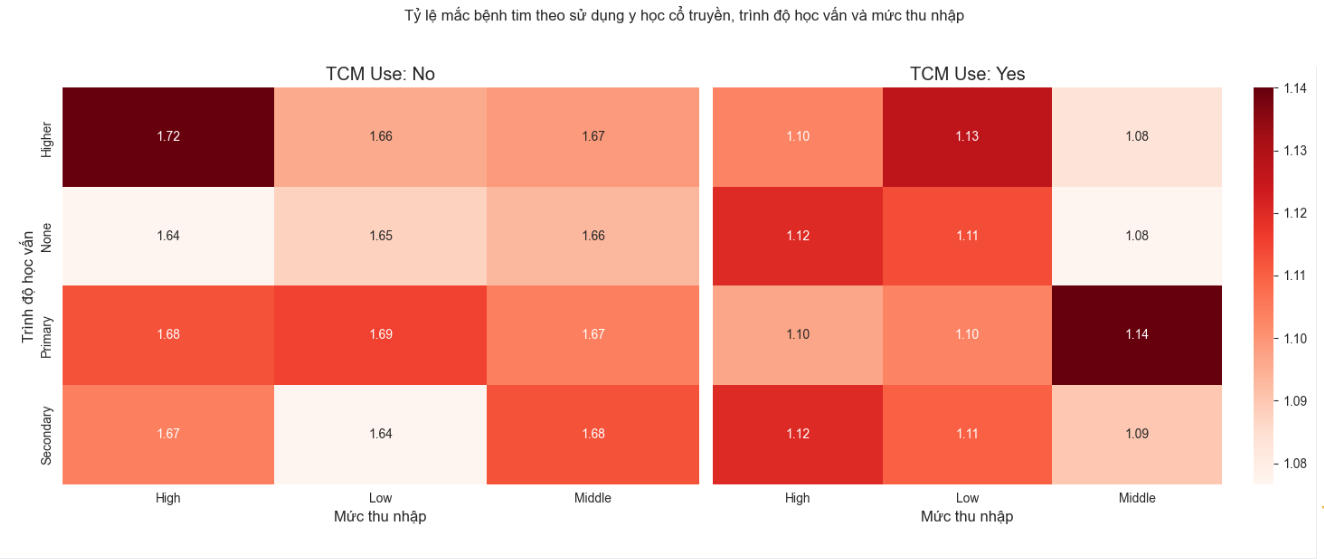
Dữ liệu phân tích các trường hợp mắc bệnh tim cho thấy một bức tranh toàn cảnh rõ nét về mối liên hệ giữa khả năng tiếp cận dịch vụ y tế và mức độ phân bố bệnh viện trong cộng đồng. Khi xem xét toàn bộ các trường hợp mắc bệnh tim, nhóm người có khả năng tiếp cận y tế kém và sống ở khu vực chỉ có mức độ bệnh viện trung bình chiếm tỷ lệ cao nhất, lên đến 12,31%. Đây là dấu hiệu cảnh báo rằng những nhóm dân cư này đang chịu ảnh hưởng nặng nề nhất từ các yếu tố liên quan đến điều kiện y tế, và do đó cần được ưu tiên trong các chính sách can thiệp và hỗ trợ.

Một điểm đáng chú ý khác là khi cộng gộp ba nhóm có khả năng tiếp cận y tế kém, dù khu vực đó có bệnh viện ở mức độ cao, trung hay thấp, thì tổng số ca mắc bệnh tim của họ đều xếp ở các vị trí khá cao và chiếm hơn 34% tổng số ca. Con số này không chỉ thể hiện sự bất cân bằng trong tiếp cận y tế mà còn cho thấy rằng chính khả năng tiếp cận, chứ không đơn thuần là sự hiện diện của cơ sở vật chất, mới là yếu tố ảnh hưởng sâu sắc đến nguy cơ mắc bệnh. Dù bệnh viện có ở gần hay đầy đủ trong khu vực, nhưng nếu người dân không thể tiếp cận được do rào cản tài chính, nhận thức hoặc thiếu hiểu biết, thì hệ thống y tế vẫn chưa thực sự hiệu quả.

Ngược lại, nhóm có khả năng tiếp cận y tế tốt, dù sống trong khu vực có ít bệnh viện nhất, lại chiếm tỷ lệ thấp nhất về số ca mắc bệnh tim, chỉ khoảng 10,49%. Điều này chứng minh rằng khi người dân có thể dễ dàng tiếp cận các dịch vụ chăm sóc sức khỏe cho dù điều kiện xung quanh không lý tưởng, thì nguy cơ mắc bệnh vẫn có thể được kiểm soát và hạn chế một cách đáng kể. Câu chuyện ở đây không đơn giản là có bao nhiêu bệnh viện được xây lên, mà là người dân có thể đến được bệnh viện, hiểu được họ cần chăm sóc sức khỏe thế nào, và có đủ điều kiện để điều trị kịp thời hay không.

Từ góc nhìn của phân tích dữ liệu, điều quan trọng rút ra không phải là xây thêm bệnh viện ở mọi nơi, mà là tập trung cải thiện những yếu tố giúp người dân dễ dàng tiếp cận dịch vụ y tế – như mạng lưới y tế cộng đồng, hỗ trợ tài chính, nâng cao nhận thức, và cải thiện hạ tầng di chuyển. Các nhà hoạch định chính sách cần hiểu rằng hiệu quả y tế không đến từ sự đầu tư dàn trải, mà từ sự can thiệp có trọng điểm. Nhóm dân cư có khả năng tiếp cận y tế thấp chính là nơi cần được ưu tiên nhất, bởi họ là những người đang chịu gánh nặng bệnh tật cao nhất. Nếu không có giải pháp phù hợp, đây sẽ là những “vùng tối” trong bức tranh chăm sóc sức khỏe toàn dân – nơi mà các ca bệnh tim không chỉ là con số, mà là hệ quả của một hệ thống chưa thật sự công bằng.

***4.1.9. Mối tương quan giữa việc sử dụng phương pháp đông y với tỷ lệ mắc bệnh về tim***

****

Biểu đồ 4.8: Tỷ lệ mắc bệnh tim theo đặc điểm sử dụng phương pháp đông y

*Nguồn: Tác giả thực hiện chạy chương trình*

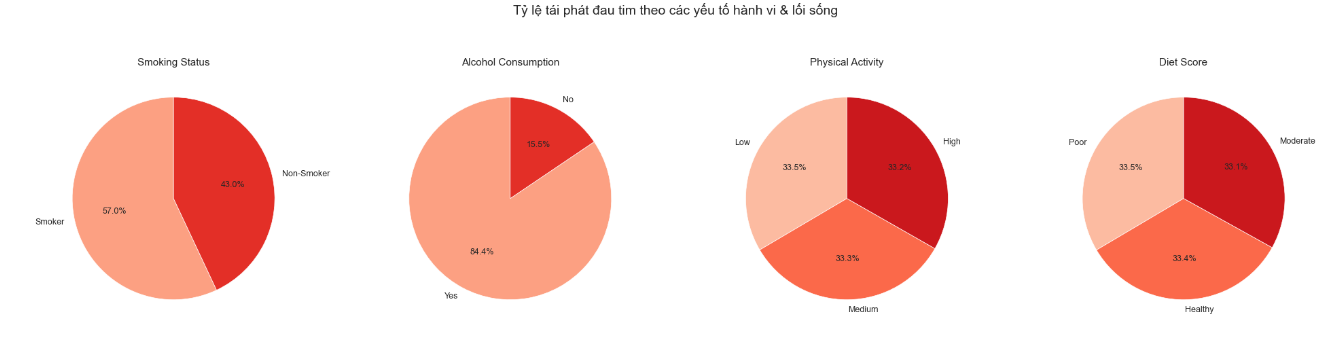
Trong quá trình phân tích hồ sơ các bệnh nhân mắc bệnh tim, một bức tranh thú vị đã hiện ra khi so sánh giữa hai nhóm người có sử dụng đông y và những người không sử dụng.

Nhóm không sử dụng đông y chủ yếu là người đã nghỉ hưu, có trình độ học vấn từ trung học trở lên và thu nhập thuộc loại cao hoặc trung bình. Cụ thể, có người tốt nghiệp đại học, thu nhập cao và hiện đã nghỉ hưu – đây có thể là những người từng làm công việc trí óc hoặc chuyên môn. Tỷ lệ mắc bệnh tim trong nhóm này dao động từ 1.6% đến 1.78%, cho thấy bệnh tim vẫn phổ biến ngay cả ở nhóm dân cư có điều kiện học vấn và kinh tế tốt. Ngược lại, nhóm sử dụng đông y có đặc điểm khá khác biệt. Phần lớn trong số họ vẫn đang đi làm, học vấn thấp hơn (từ không đi học đến chỉ học xong tiểu học hoặc trung học), nhưng thu nhập vẫn đạt mức trung bình đến cao. Đây có thể là những người làm nghề tự do, buôn bán nhỏ hoặc lao động có tay nghề. Tỷ lệ mắc bệnh tim trong nhóm này dao động từ 1.05% đến 1.18%, thấp hơn khoảng 0.6 – 0.7 điểm phần trăm so với nhóm không sử dụng đông y.

Thoạt nhìn, người sử dụng đông y có vẻ ít mắc bệnh tim hơn. Tuy nhiên, sự khác biệt này không đủ lớn để kết luận rằng việc sử dụng đông y giúp làm giảm nguy cơ mắc bệnh tim. Đặc điểm nhân khẩu học giữa hai nhóm có sự khác biệt đáng kể – đặc biệt là yếu tố tuổi tác. Những người đã nghỉ hưu thường lớn tuổi hơn, và tuổi cao vốn là một yếu tố nguy cơ quan trọng đối với bệnh tim. Trong khi đó, nhóm sử dụng đông y phần lớn còn đang trong độ tuổi lao động, khỏe mạnh hơn về mặt thể chất. Vì vậy, không thể đưa ra kết luận rằng đông y có tác dụng ngăn ngừa bệnh tim dựa trên bộ dữ liệu này. Thay vào đó, điều đáng chú ý là thói quen sử dụng phương pháp chữa bệnh truyền thống dường như gắn với học vấn và lối sống nhiều hơn là hiệu quả điều trị thực sự. Người có học vấn cao và đã nghỉ hưu ít có xu hướng sử dụng đông y, trong khi nhóm học vấn thấp, đang đi làm lại sử dụng nhiều hơn. Việc cả hai nhóm đều có thu nhập cao cho thấy yếu tố tài chính không phải là lý do quyết định.

Từ đây, có thể rút ra một điều quan trọng: hành vi tìm kiếm chăm sóc sức khỏe mang tính văn hóa – xã hội, không thể đánh giá đơn thuần dựa trên hiệu quả y tế hay chi phí. Các nhà quản lý y tế nên hướng tới các chính sách truyền thông sức khỏe phù hợp với từng nhóm dân cư, đồng thời kết hợp y học hiện đại và truyền thống một cách thận trọng và có cơ sở khoa học. Để xác định rõ hơn mối liên hệ giữa phương pháp điều trị và tỷ lệ bệnh tim, cần mở rộng phân tích thêm các yếu tố khác như độ tuổi, giới tính, thói quen sinh hoạt và tiền sử bệnh.

***4.1.10. Nguy cơ tái phát bệnh tim của những người khảo sát có thói quen và hành vi sống khác nhau***



Biểu đồ 4.9: Tỷ lệ tái phát bệnh tim qua sự khác biệt về thói quen sống

*Nguồn: Tác giả thực hiện chạy chương trình*

Dựa trên bảng số liệu khảo sát về những người đã từng mắc bệnh tim và bị tái phát, có thể nhận thấy sự khác biệt đáng kể giữa các nhóm hành vi và thói quen sinh hoạt trong cuộc sống hằng ngày. Đây là nhóm người đã có tiền sử bệnh tim, vì vậy việc tìm hiểu các yếu tố liên quan đến nguy cơ tái phát là vô cùng quan trọng để phòng ngừa và cải thiện chất lượng sống. Yếu tố đầu tiên và nổi bật nhất là việc tiêu thụ đồ uống có cồn. Trong tổng số các trường hợp tái phát bệnh tim được ghi nhận, có đến 84.45% là những người có thói quen uống rượu, bia hoặc các loại đồ uống chứa cồn, trong khi chỉ 15.55% là người không sử dụng các loại thức uống này. Tỷ lệ chênh lệch quá lớn này cho thấy rằng thói quen sử dụng đồ uống có cồn có thể là một yếu tố nguy cơ rất lớn dẫn đến việc bệnh tim tái phát. Điều này đặt ra cảnh báo rõ ràng về tầm quan trọng của việc hạn chế hoặc loại bỏ rượu bia trong sinh hoạt hàng ngày đối với những người từng có tiền sử bệnh tim. Bên cạnh đó, thói quen hút thuốc cũng cho thấy mối liên hệ khá rõ ràng với tỷ lệ tái phát bệnh tim. Trong tổng số ca tái phát, 56.99% là người có hút thuốc, trong khi 43.01% là người không hút thuốc. Dù mức độ chênh lệch không quá lớn như trường hợp rượu bia, nhưng vẫn đủ để cho thấy hút thuốc là một yếu tố đáng lo ngại, góp phần làm gia tăng nguy cơ bệnh lý tim mạch trở lại.

Trái ngược với hai yếu tố trên, các yếu tố như chế độ ăn uống và mức độ hoạt động thể chất lại không cho thấy sự khác biệt quá rõ ràng giữa các nhóm. Về điểm số chế độ ăn, ba nhóm gồm ăn uống kém, lành mạnh và trung bình có tỷ lệ tái phát bệnh tim lần lượt là 33.50%, 33.43%, và 33.06%. Sự chênh lệch giữa ba nhóm này là rất nhỏ, cho thấy rằng trong phạm vi khảo sát này, chế độ ăn dù tốt hay chưa tốt thì cũng chưa tạo ra sự khác biệt lớn trong khả năng tái phát bệnh tim. Tương tự, khi xem xét mức độ hoạt động thể chất, ta thấy các nhóm "ít vận động", "vận động trung bình" và "vận động nhiều" cũng có tỷ lệ tái phát khá tương đồng, lần lượt là 33.48%, 33.27% và 33.24%. Điều này cho thấy, trong dữ liệu thu thập được, mức độ vận động thể chất không phải là yếu tố tạo ra sự khác biệt đáng kể trong nguy cơ tái phát bệnh tim, mặc dù trong thực tế, vận động đều đặn vẫn là một khuyến nghị quan trọng cho người bệnh tim.

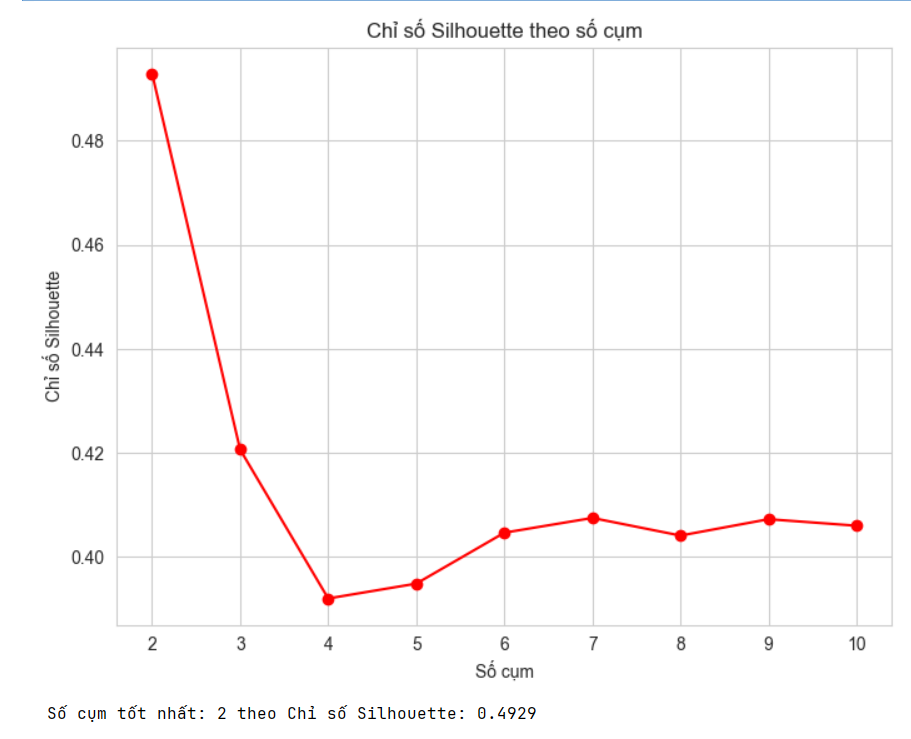
Tổng kết lại, từ góc nhìn phân tích mô tả, có thể khẳng định rằng uống rượu bia và hút thuốc là hai thói quen có ảnh hưởng mạnh mẽ và rõ ràng nhất đến nguy cơ tái phát bệnh tim trong nhóm người đã từng mắc bệnh này. Trong khi đó, chế độ ăn uống và mức độ hoạt động thể chất, tuy vẫn là các yếu tố có vai trò nhất định đối với sức khỏe, nhưng chưa cho thấy tác động rõ rệt trong dữ liệu khảo sát. Kết quả này góp phần nhấn mạnh tầm quan trọng của việc thay đổi hành vi sống lành mạnh – đặc biệt là việc từ bỏ rượu bia và thuốc lá – nhằm giảm thiểu nguy cơ tái phát và cải thiện tiên lượng cho người bệnh tim.

**4.2. Ứng dụng Spark MLlib cho phân tích học máy**

**4.2.1. Phân cụm**

Thuật toán phân cụm K Means đã được áp dụng sau khi toàn bộ dữ liệu được xử lý và chuẩn hóa thông qua một Pipeline tích hợp. Quá trình bao gồm mã hóa các biến phân loại, kết hợp các đặc trưng thành một vector duy nhất và chuẩn hóa đầu vào để đảm bảo tính nhất quán cho mô hình học máy. Mục tiêu của phân cụm là tìm ra các nhóm bệnh nhân có đặc điểm sức khỏe tương đồng, từ đó hỗ trợ cho việc phân tích rủi ro và can thiệp y tế một cách có hệ thống.

Để xác định số lượng cụm tối ưu, mô hình KMeans được đánh giá bằng chỉ số Silhouette, một thước đo cho thấy mức độ phân tách giữa các cụm và mức độ gắn kết nội tại của các điểm dữ liệu trong từng cụm. Giá trị Silhouette trung bình được tính cho các giá trị k từ 2 đến 10. Kết quả chỉ ra rằng mô hình đạt giá trị Silhouette cao nhất là 0.4929 khi k=2, đồng nghĩa với việc dữ liệu được phân chia tốt nhất khi chia thành hai nhóm. Dù giá trị này chưa vượt ngưỡng 0.5 nhưng vẫn đủ mạnh để cho thấy sự hiện diện của hai phân nhóm đặc trưng trong dữ liệu bệnh nhân tim mạch.



Biểu đồ 4.10: Chỉ số Silhoutte theo số cụm

*Nguồn: Tác giả thực hiện chạy chương trình*

Sau khi xác định được số cụm tối ưu, mô hình KMeans với k = 2 được huấn luyện lại và áp dụng trên toàn bộ tập dữ liệu đã chuẩn hóa. Kết quả phân cụm cho thấy mỗi bệnh nhân được gán vào một trong hai cụm, với chỉ số “prediction” thể hiện cụm tương ứng. Điều này cho phép phân nhóm dân số bệnh nhân thành hai tập riêng biệt dựa trên các đặc điểm sức khỏe đo được. Thông tin chi tiết hơn được thể hiện qua các tâm cụm (cluster centers). Các tâm cụm này giúp xác định đặc điểm điển hình của từng nhóm bệnh nhân, ví dụ như mức độ huyết áp, cholesterol hoặc các chỉ số sinh học khác.

Cụ thể, kết quả phân tích cho thấy cụm thứ nhất (prediction = 0) có giá trị trung bình của chỉ số CVD\_Risk\_Score là 77.01, cao hơn rõ rệt so với cụm thứ hai (prediction = 1) với 32.00. Tuy nhiên, cả hai cụm lại có mức huyết áp trung bình tương đương nhau, khoảng 141 mmHg. Điều này cho thấy huyết áp không phải là yếu tố quyết định duy nhất của nguy cơ tim mạch trong tập dữ liệu, và mô hình đã phân nhóm dựa trên tổng thể nhiều yếu tố khác. Nhóm 0 có thể đại diện cho bệnh nhân có nguy cơ cao do hội tụ nhiều đặc điểm bất lợi khác, trong khi nhóm 1 mang đặc điểm của nhóm ít nguy cơ hơn.

Việc phân cụm dữ liệu bệnh nhân tim thành hai nhóm giúp phản ánh rõ ràng hai phân khúc sức khỏe khác biệt trong cộng đồng nghiên cứu. Một nhóm có thể mang đặc điểm của các bệnh nhân có nguy cơ cao, trong khi nhóm còn lại đại diện cho các trường hợp ít nguy cơ hơn. Nhận biết được sự khác biệt giữa các nhóm này là yếu tố then chốt trong việc xây dựng các chiến lược can thiệp y tế, phân bổ nguồn lực và cá nhân hóa phương pháp điều trị.

**4.2.2. Phân lớp**

*4.2.2.1. So sánh các chỉ số đánh giá hiệu suất giữa các mô hình*

Bảng 4.3: Chỉ số đánh giá hiệu suất mô hình Logistic Regression, Decision Tree và SVM

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Mô hình*** | ***Accuracy*** | ***Precision*** | ***Recall*** | ***F1 Score*** | ***AUC*** |
| Logistic Regression | 0.6243 | 0.6250 | 0.8640 | 0.7253 | 0.6409 |
| Decision Tree | 0.6320 | 0.6221 | 0.9144 | 0.7405 | 0.5731 |
| SVM | 0.6189 | 0.6229 | 0.8521 | 0.7197 | 0.6375 |

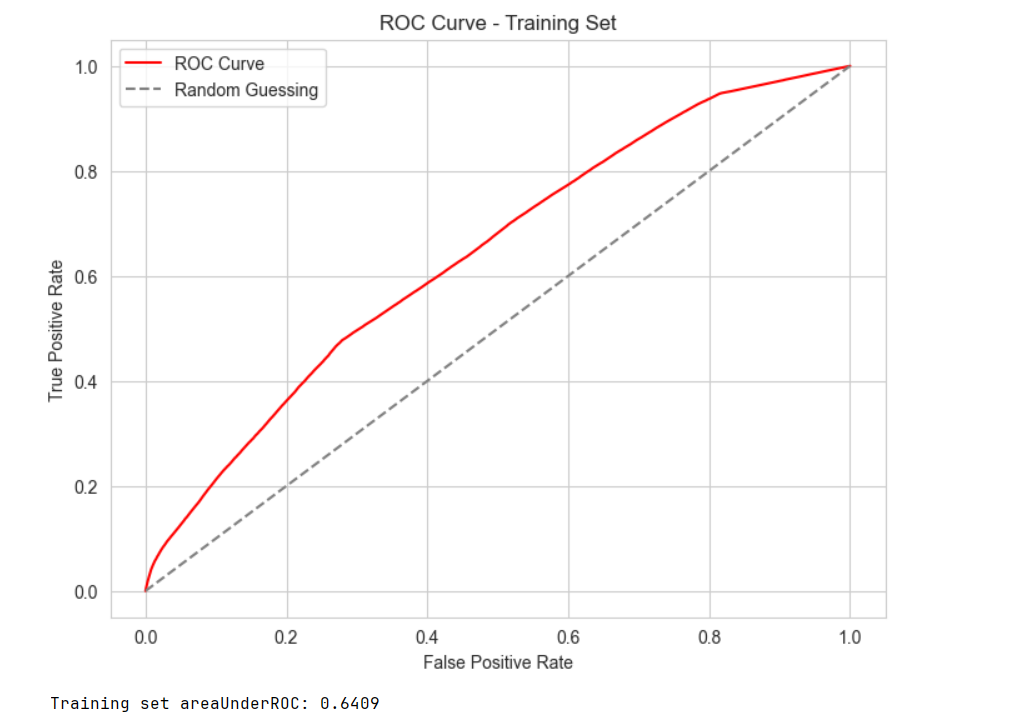
*Nguồn: Tác giả thực hiện chạy chương trình*

Nhóm quyết định tiếp tục thử thêm hai mô hình khác để mở rộng so sánh và lựa chọn mô hình tối ưu hơn. Ở mô hình Decision Tree có Recall và F1 Score cao nhất lần lượt là 0.9144 và 0.7405. Nhưng các kết quả đánh giá lại cho thấy ba mô hình Logistic Regression, Decision Tree và SVM có hiệu năng khá tương đồng, với các chỉ số Accuracy, Precision, Recall, F1 Score và AUC chênh lệch không đáng kể giữa các mô hình. Dù cho Decision Tree có một vài chỉ số tốt hơn, nhưng sự chênh lệch không lớn và các chỉ số còn lại gần như cân bằng giữa các mô hình, nên có thể kết luận rằng hiệu năng tổng thể của ba mô hình là tương đương nhau trong việc dự đoán khả năng bệnh tim ở bệnh nhân. Nếu một mô hình tốt hơn một chút nhưng khó triển khai hoặc dễ overfit (ví dụ: cây quyết định phức tạp), thì mô hình đơn giản và ổn định như Logistic Regression vẫn là lựa chọn hợp lý.

*4.2.2.2. Phân lớp bằng mô hình hồi quy logistic*

Trong nghiên cứu này, nhóm sử dụng mô hình hồi quy logistic (Logistic Regression) được sử dụng để xây dựng hệ thống phân loại nhị phân nhằm dự đoán khả năng một bệnh nhân có bị bệnh tim hay không. Giúp phát hiện sớm các trường hợp có nguy cơ cao để can thiệp kịp thời. Mô hình được huấn luyện trên tập dữ liệu gồm nhiều đặc trưng liên quan đến sức khỏe tim mạch như tuổi tác, huyết áp, cholesterol và các chỉ số lâm sàng khác.

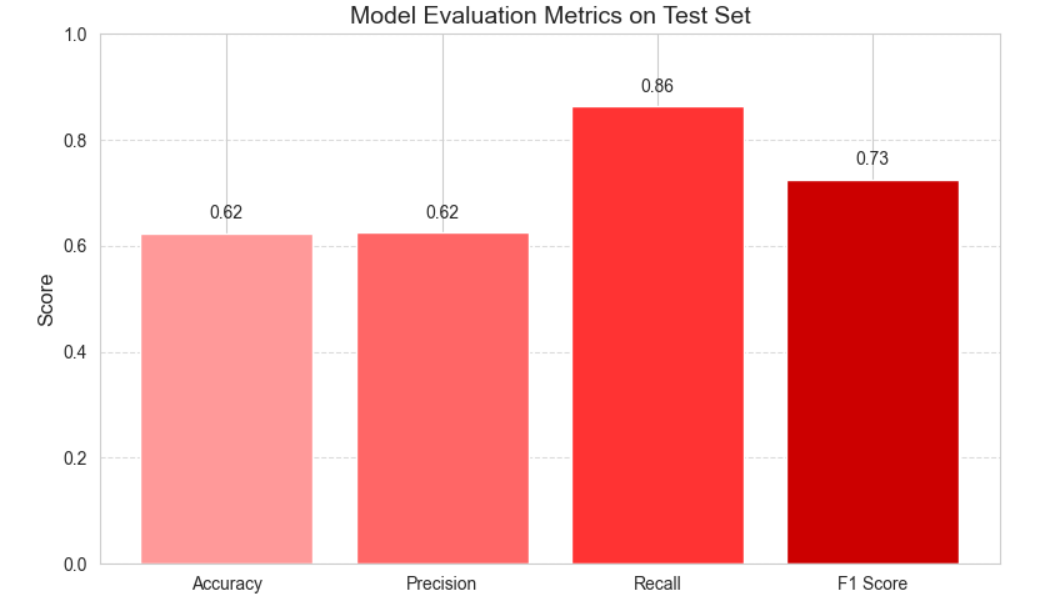
Sau khi huấn luyện mô hình trên tập dữ liệu huấn luyện, hiệu quả phân loại được đánh giá thông qua biểu đồ đường cong ROC (Receiver Operating Characteristic Curve). Trên biểu đồ, đường cong ROC của mô hình được thể hiện bằng đường màu xanh lam, trong khi đường chấm gạch màu xám biểu diễn đường phân loại ngẫu nhiên – tương ứng với mô hình không có khả năng phân biệt hai lớp. Đường cong ROC là công cụ trực quan giúp đánh giá khả năng mô hình phân biệt đúng giữa hai lớp nhị phân trong bài toán phân loại, với trục hoành là tỷ lệ dương tính giả (False Positive Rate) và trục tung là tỷ lệ dương tính thật (True Positive Rate).

****

Biểu đồ 4.11: Đường cong ROC ở mô hình Logistic Regression

*Nguồn: Tác giả thực hiện chạy chương trình*

Kết quả cho thấy diện tích dưới đường cong ROC (AUC - Area Under the Curve) đạt giá trị 0.6409. Chỉ số này phản ánh rằng mô hình có khả năng phân biệt giữa hai lớp tốt hơn mô hình đoán ngẫu nhiên (với AUC = 0.5), tuy nhiên vẫn còn thấp nếu so với một mô hình phân loại mạnh (thường kỳ vọng AUC ≥ 0.80 trong ứng dụng thực tế). Hiểu một cách cụ thể, AUC = 0.6409 có nghĩa là mô hình có xác suất khoảng 64% xếp xác suất cao hơn cho một bệnh nhân thực sự có bệnh so với một người không mắc bệnh. Mặc dù đây là một dấu hiệu tích cực, nhưng hiệu quả này chưa đủ mạnh để đưa vào sử dụng lâm sàng mà không qua cải tiến.



Biểu đồ 4.12: Các chỉ số đánh giá hiệu suất ở mô hình Logistic Regression

*Nguồn: Tác giả thực hiện chạy chương trình*

Sau khi huấn luyện mô hình Logistic Regression trên bộ dữ liệu bệnh tim, chúng tôi tiến hành đánh giá hiệu suất mô hình thông qua các chỉ số phân loại tiêu chuẩn. Kết quả cho thấy độ chính xác (accuracy) đạt khoảng 62.4%, nghĩa là mô hình dự đoán đúng 62 trong số 100 trường hợp. Tuy nhiên, trong bối cảnh y tế – đặc biệt là các bệnh lý nghiêm trọng như bệnh tim – accuracy không phải là chỉ số phản ánh toàn diện chất lượng mô hình. Nguyên nhân là do dữ liệu thường bị mất cân bằng giữa hai lớp (bệnh và không bệnh), khiến cho chỉ số này có thể gây hiểu nhầm.

Một hạn chế cần lưu ý của mô hình là độ chính xác (precision) tương đối thấp, chỉ đạt khoảng 62.5%. Điều này đồng nghĩa với việc trong số những ca được dự đoán mắc bệnh, có gần 37.5% là dự đoán sai — tức là mô hình đã “báo động giả” cho những người thực sự khỏe mạnh. Trong thực tế, đặc biệt là trong lĩnh vực y tế, việc có tỷ lệ dương tính giả cao có thể dẫn đến những hậu quả không mong muốn như gây ra sự lo lắng, căng thẳng cho bệnh nhân, cũng như làm tăng chi phí và công sức do phải tiến hành các xét nghiệm bổ sung không cần thiết.

Do đó, mặc dù mô hình có khả năng phát hiện bệnh nhân tốt nhờ recall cao khoảng 86%, nhưng cần cân nhắc kỹ lưỡng để cải thiện precision nhằm giảm thiểu các báo động sai, đảm bảo tính hiệu quả và tối ưu trong quy trình sàng lọc bệnh.

****

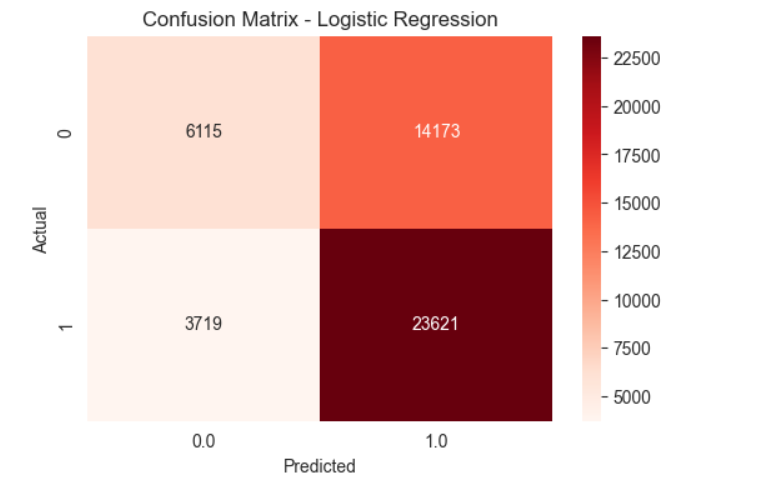
Biểu đồ 4.13: Đường cong độ chính xác – độ nhạy ở mô hình Logistic Regression

*Nguồn: Tác giả thực hiện chạy chương trình*

Từ biểu đồ Precision-Recall cho thấy rõ sự đánh đổi giữa hai chỉ số quan trọng: Precision (độ chính xác) và Recall (khả năng phát hiện đúng người bệnh). Khi Recall tăng lên, nghĩa là mô hình đang cố gắng nhận diện càng nhiều người mắc bệnh tim càng tốt. Tuy nhiên, điều này lại khiến Precision giảm, tức là trong số những người được dự đoán là có bệnh, tỷ lệ dự đoán sai cũng tăng theo.

Nói cách khác, mô hình đang ưu tiên “không bỏ sót người bệnh”, nên có xu hướng dự đoán nhiều người là mắc bệnh hơn, kể cả khi một số trong đó thực tế không bị. Đây là điều thường xảy ra trong các hệ thống y tế, đặc biệt là các mô hình cảnh báo sớm, vì việc dự đoán nhầm người khỏe còn tốt hơn là bỏ sót người bệnh thật. Tuy vậy, khi áp dụng vào thực tế, chúng ta vẫn cần cân nhắc kỹ để tránh gây lo lắng không cần thiết cho người không bệnh và không làm tăng chi phí kiểm tra y tế không cần thiết.

Cuối cùng, chỉ số F1-score đạt 0.725, là trung bình điều hòa giữa precision và recall, phản ánh mức độ cân bằng giữa hai yếu tố: khả năng phát hiện đúng bệnh nhân và độ chính xác trong từng dự đoán dương tính. F1-score ở mức khá cho thấy mô hình có hiệu quả tương đối, đặc biệt là trong bối cảnh đánh đổi giữa việc phát hiện bệnh và kiểm soát sai số.



Biểu đồ 4.14: Ma trận nhầm lẫn ở mô hình Logistic Regression

*Nguồn: Tác giả thực hiện chạy chương trình*

Biểu đồ trên minh họa ma trận nhầm lẫn của mô hình Logistic Regression. Dữ liệu từ ma trận nhầm lẫn cho thấy mô hình Logistic Regression có 6.115 trường hợp được dự đoán chính xác là không mắc bệnh, tương ứng với True Negative (TN). Tuy nhiên, có tới 14.173 trường hợp không mắc bệnh nhưng lại bị dự đoán nhầm là mắc bệnh, gọi là False Positive (FP). Ở chiều ngược lại, 3.719 người thực sự mắc bệnh nhưng bị bỏ sót do mô hình dự đoán sai là không mắc bệnh – đây là False Negative (FN). Trong khi đó, mô hình đã xác định đúng 23.621 trường hợp mắc bệnh, tức là True Positive (TP). Những con số này phản ánh hiệu suất của mô hình và là cơ sở để tính toán các chỉ số quan trọng như precision (độ chính xác) và recall (độ nhạy), từ đó đánh giá mức độ phù hợp của mô hình trong bối cảnh ứng dụng thực tế, đặc biệt là trong lĩnh vực y tế.

**CHƯƠNG V: KẾT LUẬN**

**5.1. Thảo luận**

Nghiên cứu đã phác họa một bức tranh toàn diện và sâu sắc về các yếu tố nguy cơ dẫn đến bệnh tim mạch thông qua phân tích dữ liệu bằng Spark SQL và Spark MLlib. Kết quả cho thấy các yếu tố nhân khẩu học, đặc biệt là độ tuổi và khu vực sinh sống – có ảnh hưởng rõ rệt đến nguy cơ mắc bệnh, trong đó người cao tuổi ở vùng nông thôn là nhóm dễ bị tổn thương nhất. Dữ liệu cho thấy không có sự khác biệt đáng kể giữa nam và nữ về tỷ lệ mắc bệnh, từ đó khẳng định bệnh tim không phân biệt giới tính mà phụ thuộc chủ yếu vào môi trường sống, tình trạng sức khỏe và hành vi cá nhân.

Xét về hành vi và thói quen sống, mức độ căng thẳng là yếu tố có ảnh hưởng mạnh mẽ nhất đến nguy cơ mắc bệnh tim, bất kể mức thu nhập cao, trung bình hay thấp. Bên cạnh đó, hút thuốc và sử dụng rượu bia cũng là hai tác nhân góp phần đáng kể vào cả nguy cơ mắc bệnh lần đầu lẫn tái phát. Ngược lại, chế độ ăn uống và mức độ vận động thể chất, tuy vẫn quan trọng về mặt y học nhưng không cho thấy sự khác biệt rõ rệt trong dữ liệu phân tích. Về mặt bệnh lý nền, những người mắc tăng huyết áp, béo phì, tiểu đường hoặc bệnh thận mạn tính có tỷ lệ mắc bệnh tim cao hơn đáng kể so với người không có bệnh nền. Tuy nhiên, yếu tố di truyền lại không cho thấy ảnh hưởng rõ rệt, cho thấy tiền sử gia đình không phải là yếu tố quyết định hàng đầu. Môi trường sống, cụ thể là chất lượng không khí, cũng đóng vai trò quan trọng: không khí ô nhiễm càng nặng thì tỷ lệ mắc bệnh tim càng cao. Đặc biệt, khả năng tiếp cận dịch vụ y tế còn quan trọng hơn cả số lượng bệnh viện hiện diện trong khu vực, nhấn mạnh vai trò của công bằng y tế và cơ sở hạ tầng hỗ trợ.

Về khía cạnh học máy, nghiên cứu đã ứng dụng thành công thuật toán phân cụm K-means để chia toàn bộ dân số bệnh nhân thành hai nhóm chính. Giá trị Silhouette đạt 0.4929 cho thấy mức độ phân tách tương đối tốt, phản ánh hai phân khúc sức khỏe khác biệt – trong đó một nhóm có đặc điểm của bệnh nhân nguy cơ cao, nhóm còn lại ít rủi ro hơn. Điều này có ý nghĩa quan trọng trong việc xây dựng các chương trình can thiệp y tế có mục tiêu, phân bổ nguồn lực hợp lý và cá nhân hóa phác đồ điều trị. Bên cạnh đó, mô hình phân loại Logistic Regression cũng đã được huấn luyện nhằm dự đoán nguy cơ mắc bệnh tim. Tuy chưa đạt mức độ chính xác cao (AUC = 0.6409, accuracy = 62.4%), mô hình vẫn thể hiện năng lực phát hiện bệnh nhân tương đối tốt nhờ recall đạt tới 86.4%. Tuy nhiên, precision còn thấp (62.5%) dẫn đến tỷ lệ báo động giả cao, cần được cải thiện để tránh gây lo lắng không cần thiết và tiết kiệm chi phí y tế. Chỉ số F1-score đạt 0.725 cho thấy mô hình có hiệu quả tương đối, phù hợp cho các bước sàng lọc ban đầu, nhưng chưa đủ mạnh để ứng dụng lâm sàng mà không qua tối ưu thêm.

Từ các phân tích trên, có thể thấy rằng chiến lược phòng ngừa bệnh tim mạch cần được thiết kế toàn diện, không chỉ tập trung vào giới tính hay thu nhập, mà cần ưu tiên các nhóm có yếu tố nguy cơ cao như người lớn tuổi, cư dân nông thôn, người có bệnh nền, sống trong môi trường ô nhiễm, gặp khó khăn trong tiếp cận y tế hoặc có lối sống thiếu lành mạnh. Cách tiếp cận này không chỉ nâng cao hiệu quả y tế cộng đồng mà còn góp phần xây dựng hệ thống chăm sóc sức khỏe công bằng và bền vững hơn.

**5.2. Hạn chế**

Mặc dù nghiên cứu đã ứng dụng hiệu quả các công cụ phân tích dữ liệu hiện đại như Spark SQL và Spark MLlib để xử lý tập dữ liệu lớn, vẫn tồn tại một số hạn chế đáng lưu ý.

Trước hết, dữ liệu nghiên cứu chỉ bao gồm 239.266 mẫu – một con số tương đối nhỏ so với quy mô dân số hơn 1,4 tỷ người tại Trung Quốc. Do đó, kết quả phân tích có thể chưa phản ánh đầy đủ tính đa dạng về nhân khẩu học, địa lý, kinh tế – xã hội và hành vi sức khỏe trong thực tế, đặc biệt là tại các vùng sâu, vùng xa hoặc khu vực đặc thù.

Thứ hai, nhiều yếu tố tiềm ẩn ảnh hưởng đến bệnh tim mạch như tiền sử điều trị, mức độ tuân thủ y tế, gen di truyền cụ thể hoặc các yếu tố môi trường dài hạn chưa được đưa vào phân tích hoặc chỉ được tiếp cận ở mức khái quát.

Thứ ba, mô hình học máy được sử dụng trong nghiên cứu còn tương đối đơn giản, chủ yếu là hồi quy logistic và KMeans, chưa khai thác các thuật toán tiên tiến hơn như Random Forest, XGBoost hay Deep Learning, có thể mang lại độ chính xác cao hơn.

Ngoài ra, mô hình phân loại hiện tại chỉ đạt AUC ở mức 0.6409 và độ chính xác khoảng 62.4%, chưa đủ mạnh để triển khai trong thực tiễn lâm sàng mà không qua cải tiến. Cuối cùng, dữ liệu sử dụng mang tính cắt ngang tại một thời điểm nhất định, chưa thể đánh giá được các yếu tố tiến triển theo thời gian hay mối quan hệ nhân – quả giữa các biến.

**5.3. Hướng phát triển**

Dựa trên những kết quả và hạn chế đã phân tích, nghiên cứu trong tương lai có thể được mở rộng theo một số hướng sau nhằm nâng cao độ chính xác và tính ứng dụng thực tiễn:

Thứ nhất, cần mở rộng quy mô và tính đại diện của bộ dữ liệu bằng cách thu thập thêm dữ liệu từ nhiều khu vực địa lý, tầng lớp dân cư và nhóm tuổi khác nhau, đặc biệt là các vùng nông thôn, miền núi hoặc nhóm dân cư thiểu số. Việc tăng số lượng mẫu lên mức hàng triệu sẽ giúp phản ánh chính xác hơn bức tranh sức khỏe tim mạch của dân số quy mô lớn như Trung Quốc.

Thứ hai, nên áp dụng các mô hình học máy tiên tiến hơn như Random Forest, Gradient Boosting (XGBoost, LightGBM) hoặc mạng nơ-ron sâu (Deep Learning) để cải thiện hiệu quả dự đoán và giảm thiểu sai số. Đồng thời, tích hợp thêm các kỹ thuật như xử lý mất cân bằng dữ liệu (imbalanced data), phân tích chuỗi thời gian (time-series analysis) hay học tăng cường (reinforcement learning) sẽ giúp mô hình thích nghi tốt hơn với dữ liệu y tế phức tạp.

Thứ ba, nghiên cứu nên chuyển hướng sang dữ liệu dọc (longitudinal data) để theo dõi tiến triển sức khỏe tim mạch theo thời gian, từ đó xác định được các yếu tố nguy cơ dài hạn và mối liên hệ nhân – quả rõ ràng hơn giữa hành vi, môi trường và bệnh lý. Việc kết hợp với dữ liệu thiết bị đeo (wearable devices), hồ sơ y tế điện tử (EHR) và dữ liệu môi trường thực tế cũng là một hướng phát triển tiềm năng.

Cuối cùng, cần nghiên cứu thêm về các giải pháp hỗ trợ ra quyết định y tế (Clinical Decision Support Systems – CDSS) dựa trên kết quả phân tích dữ liệu lớn, từ đó ứng dụng vào hệ thống khám chữa bệnh thực tế. Nếu được triển khai hiệu quả, các mô hình phân tích dữ liệu này có thể trở thành công cụ hỗ trợ bác sĩ trong việc chẩn đoán sớm, quản lý bệnh mạn tính và xây dựng chiến lược phòng ngừa chủ động cho cộng đồng.

**TÀI LIỆU THAM KHẢO**

[1] World Health Organization: WHO. (2021). *Cardiovascular diseases (CVDs)*. <https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)>

[2] China, WCOROCH a. DI (2023). Report on Cardiovascular Health and Diseases in China 2022: an Updated Summary. *PubMed*, *36*(8), 669–701. https://doi.org/10.3967/bes2023.106

[3] Du, X., Patel, A., Anderson, C. S., Dong, J., & Ma, C. (2019). Epidemiology of cardiovascular disease in China and opportunities for improvement. *Journal of the American College of Cardiology*, *73*(24), 3135–3147. https://doi.org/10.1016/j.jacc.2019.04.036

[4] National Health Commission of China. (2023). *Health system reform and prevention strategies.*

[5] Apache Spark Documentation. (2024). https://spark.apache.org/docs/latest/

[6] De Mauro, A., Greco, M., & Grimaldi, M. (2016). *A formal definition of Big Data based on its essential features. Library Review, 65*(3), 122–135. https://doi.org/10.1108/lr-06-2015-0061

[7] Gandomi, A., & Haider, M. (2014). *Beyond the hype: Big data concepts, methods, and analytics. International Journal of Information Management, 35*(2), 137–144. https://doi.org/10.1016/j.ijinfomgt.2014.10.007

[8] Ward, J. S., & Barker, A. (2013). *Undefined by Data: A Survey of Big Data Definitions.*

[9] Lovelace, R. (2016). The Data Revolution: Big Data, Open Data, Data Infrastructures and Their Consequences, by Rob Kitchin. 2014. Thousand Oaks, California: Sage Publications. ISBN: 978‐1446287484. Journal of Regional Science, 56(4), 722–723. https://doi.org/10.1111/jors.12293

[10] GeeksforGeeks. (2024). Libraries in Python. GeeksforGeeks. <https://www.geeksforgeeks.org/libraries-in-python/>

[11] Python Libraries: top lists, uses, how to choose. WsCube Tech. https://www.wscubetech.com/resources/python/libraries

[12] GeeksforGeeks. (2024b). Libraries in Python. GeeksforGeeks. https://www.geeksforgeeks.org/libraries-in-python/

[13] PySpark Overview - PySpark 4.0.0 documentation. https://spark.apache.org/docs/latest/api/python/index.html

[14] GeeksforGeeks. (2024c,). Libraries in Python. GeeksforGeeks. <https://www-geeksforgeeks-org.translate.goog/libraries-in-python/?_x_tr_sl=en&_x_tr_tl=vi&_x_tr_hl=vi&_x_tr_pto=tc>

[15] Humphrys, Mark (2020). *"Continuous output, the sigmoid function".* Properties of the sigmoid, including how it can shift along axes and how its domain may be transformed.

**PHỤ LỤC**

***Phụ lục 1***

nội dung cấu hình mẫu của bốn tệp XML chính trong thư mục etc/hadoop 4 thư mục: core-site.xml,  hdfs-site.xml, mapred-site.xml, yarn-site.xml.

(1) Thư mục core-site.xml:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>  <!--    Licensed under the Apache License, Version 2.0 (the "License");    you may not use this file except in compliance with the License.    You may obtain a copy of the License at      http://www.apache.org/licenses/LICENSE-2.0    Unless required by applicable law or agreed to in writing, software    distributed under the License is distributed on an "AS IS" BASIS,    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.    See the License for the specific language governing permissions and    limitations under the License. See accompanying LICENSE file.  -->  <!-- Put site-specific property overrides in this file. -->  <configuration>  <property>  <name>fs.defaultFS</name>  <value>hdfs://localhost:9000</value>  </property>  </configuration> |

(2) Thư mục hdfs-site.xml:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>  <!--    Licensed under the Apache License, Version 2.0 (the "License");    you may not use this file except in compliance with the License.    You may obtain a copy of the License at      http://www.apache.org/licenses/LICENSE-2.0    Unless required by applicable law or agreed to in writing, software    distributed under the License is distributed on an "AS IS" BASIS,    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.    See the License for the specific language governing permissions and    limitations under the License. See accompanying LICENSE file.  -->  <!-- Put site-specific property overrides in this file. -->  <configuration>  <property>  <name>dfs.replication</name>  <value>1</value>  </property>  <property>  <name>dfs.namenode.name.dir</name>  <value>C:\hadoop\hadoop-3.3.6\data\namenode</value>  </property>  <property>  <name>dfs.datanode.data.dir</name>  <value>C:\hadoop\hadoop-3.3.6\data\datanode</value>  </property>  </configuration> |

(3) Thư mục mapred-site.xml:

|  |
| --- |
| <?xml version="1.0"?>  <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>  <!--    Licensed under the Apache License, Version 2.0 (the "License");    you may not use this file except in compliance with the License.    You may obtain a copy of the License at      http://www.apache.org/licenses/LICENSE-2.0    Unless required by applicable law or agreed to in writing, software    distributed under the License is distributed on an "AS IS" BASIS,    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.    See the License for the specific language governing permissions and    limitations under the License. See accompanying LICENSE file.  -->  <!-- Put site-specific property overrides in this file. -->  <configuration>  <property>  <name>mapreduce.framework.name</name>  <value>yarn</value>  </property>  </configuration> |

(4) Thư mục yarn-site.xml

|  |
| --- |
| <?xml version="1.0"?>  <!--    Licensed under the Apache License, Version 2.0 (the "License");    you may not use this file except in compliance with the License.    You may obtain a copy of the License at      http://www.apache.org/licenses/LICENSE-2.0    Unless required by applicable law or agreed to in writing, software    distributed under the License is distributed on an "AS IS" BASIS,    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.    See the License for the specific language governing permissions and    limitations under the License. See accompanying LICENSE file.  -->  <configuration>      <property>          <name>yarn.nodemanager.aux-services</name>          <value>mapreduce\_shuffle</value>      </property>      <property>          <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>          <value>org.apache.hadoop.mapred.ShuffleHandler</value>      </property>  </configuration> |