|  |  |
| --- | --- |
| **bỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

TRƯỜNG ĐẠI HỌC THỦY LỢI

**KHOA CÔNG NGHỆ THÔNG TIN**



BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN PHÂN TÍCH DỮ LIỆU LỚN

ĐỀ TÀI: THUẬT TOÁN NAIVE BAYES VÀ MAP REDUCE HOÁ

TRONG DỰ ĐOÁN KHẢ NĂNG MẮC BỆNH ĐAU TIM Ở NGƯỜI

**Nhóm sinh viên thực hiện:**

Lê Duy Hưng – 2051063561

Trần Việt Phương – 2051063533

Vũ Thanh Sơn – 2051063469

Bùi Minh Hiếu – 2051063767

**Khoa:** Công nghệ thông tin

**Lớp:** 62TH-NB

**Giảng viên hướng dẫn:** TS. Tạ Quang Chiểu

***Hà Nội, Tháng 8 năm 2023***

# **MỞ ĐẦU**

Công nghệ Big Data đang rất bùng nổ, đặc biệt là hiện tại – cách mạng công nghiệp 4.0 và AI trở lên vô cùng mạnh mẽ, thì không thể thiếu sự có mặt của Big Data. Big Data là thuật ngữ được sử dụng để miêu tả các tập dữ liệu lớn và phức tạp mà không thể được xử lý bằng phương pháp truyền thống. Big Data được tạo ra từ nhiều nguồn khác nhau như các hệ thống thông tin, mạng xã hội, cảm biến, máy móc và nhiều nguồn dữ liệu khác. Big Data mang lại tiềm năng lớn trong việc phân tích dữ liệu để khám phá thông tin giá trị, tạo ra cái nhìn sâu sắc về các xu hướng, mô hình và thậm chí dự đoán tương lai.

Y tế là một ngành vô cùng quan trọng trong đời sống và việc thực hiện các xét nghiệm, các nghiên cứu về con người cũng vậy. Việc xác định chính xác khả bị bệnh đau tim ở người giúp có thể xác định sớm và điều trị kịp thời. Tuy nhiên, việc phân loại và xác định thủ công có thể gặp khó khăn và tốn nhiều thời gian. Đây là nơi mà thuật toán Naïve Bayes và Map Reduce có thể được áp dụng để giải quyết vấn đề này.

Thuật toán Naïve Bayes là một thuật toán học có giám sát phổ biến trong lĩnh vực học máy. Nó dựa trên giả định Naïve Bayes rằng các đặc trưng của một mẫu độc lập với nhau khi đã biết lớp của mẫu đó. Thuật toán này sử dụng các xác suất điều kiện và xác suất prior để phân loại các mẫu vào các lớp tương ứng.

Để xử lý dữ liệu lớn và tăng tốc độ tính toán, chúng em áp dụng Map Reduce trong quá trình huấn luyện và dự đoán với thuật toán Naïve Bayes. Map Reduce là một khung làm việc phân tán và song song phổ biến trong xử lý dữ liệu lớn. Nó cho phép chúng ta chia nhỏ công việc và phân phối chúng trên nhiều nút xử lý để tăng tốc độ tính toán và hiệu suất.

Qua bài tập lớn này, chúng em mong muốn hiểu rõ hơn về thuật toán Naïve Bayes và khả năng ứng dụng của nó trong việc phân loại, cùng với đó là Map Reduce và Hadoop trong việc xử lý dữ liệu lớn đê bài toán trở lên tối ưu hơn với đề tài: “Thuật toán Naive Bayes và Map Reduce hoá trong dự đoán khả năng mắc bệnh đau tim ở người”.

Báo cáo gồm 4 chương:

Chương 1: Tổng quan về dữ liệu lớn.

Chương 2: Phân lớp dữ liệu bằng thuật toán Naïve Bayes.

Chương 3: MapReduce thuật toán Naive Bayes trong dự đoán khả năng đau mắc bệnh đau tim ở người.

Chương 4: Kết luận và hướng phát triển.

**MỤC LỤC**

[**MỞ ĐẦU** 2](#_Toc134460367)

[**MỤC LỤC HÌNH ẢNH** 4](#_Toc134460368)

[**CHƯƠNG 1: TỔNG QUAN VỀ DỮ LIỆU LỚN** 5](#_Toc134460369)

[**1.1. Định nghĩa** 5](#_Toc134460370)

[**1.2. Đặc trưng cơ bản của dữ liệu lớn** 5](#_Toc134460371)

[**1.3. Tổng quan về Hadoop** 7](#_Toc134460372)

[1.3.1. Tổng quan 7](#_Toc134460373)

[1.3.2. Kiến trúc của Hadoop 7](#_Toc134460374)

[**CHƯƠNG 2: PHÂN LỚP DỮ LIỆU BẰNG THUẬT TOÁN NAIVE BAYES** 11](#_Toc134460375)

[**2.1. Thuật toán Naive Bayes** 11](#_Toc134460376)

[2.1.1. Định lý Bayes 11](#_Toc134460377)

[2.1.2. Phân lớp Naive Bayes 12](#_Toc134460378)

[**2.2. Ví dụ minh hoạ thuật toán** 13](#_Toc134460379)

[**CHƯƠNG 3: MAP REDUCE THUẬT TOÁN NAIVE BAYES TRONG DỰ ĐOÁN KHẢ NĂNG ĐAU TIM Ở NGƯỜI** 15](#_Toc134460380)

[**3.1. Ý tưởng MapReduce Naive Bayes** 15](#_Toc134460381)

[**3.2. Mô tả bài toán** 16](#_Toc134460382)

[**3.3. Mô tả dữ liệu** 16](#_Toc134460383)

[**3.4. Xây dựng mô hình** 16](#_Toc134460384)

[3.4.1. Xây dựng lớp NaiveBayesTrainMapper 17](#_Toc134460385)

[3.4.2. Xây dựng lớp NaiveBayesTrainReduce 17](#_Toc134460386)

[3.4.3. Xây dựng lớp NaiveBayesTrainJob 18](#_Toc134460387)

[3.4.4. Xây dựng lớp NaiveBayesTestMapper 18](#_Toc134460388)

[3.4.5. Xây dựng lớp NaiveBayesTestReduce 18](#_Toc134460389)

[3.4.6. Xây dựng lớp NaiveBayesTestJob 18](#_Toc134460390)

[**3.5. Demo chương trình cài đặt** 19](#_Toc134460391)

[3.5.1. Demo cài đặt hadoop thành công 19](#_Toc134460392)

[3.5.2. Thử nghiệm 19](#_Toc134460393)

[**CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 22](#_Toc134460394)

[**4.1. Kết luận** 22](#_Toc134460395)

[**4.2. Hướng phát triển** 22](#_Toc134460396)

[**TÀI LIỆU THAM KHẢO** 23](#_Toc134460397)

# **MỤC LỤC HÌNH ẢNH**

[Hình 1. Minh hoạ nguồn gốc của dữ liệu 5](#_Toc134460336)

[Hình 2. Logo biểu tượng của hadoop 7](#_Toc134460337)

[Hình 3. Kiến trúc cảu hadoop 8](#_Toc134460338)

[Hình 4. HDFS (Hadoop Distributed File System) 9](#_Toc134460339)

[Hình 5. Quy trình Map Reduce 9](#_Toc134460340)

[Hình 6. Trình quản lý của YARN 10](#_Toc134460341)

[Hình 7. Hadoop Ecosytem 11](#_Toc134460342)

[Hình 8. Cơ sở dữ liệu khách hàng trong ví dụ 14](#_Toc134460343)

[Hình 9. Lưu đồ thuật toán Naïve Bayes 16](#_Toc134460344)

[Hình 10. Tạo foler data-drybean và thêm vào file train.csv và test.csv ở hadoop 20](#_Toc134460345)

[Hình 11. Cấu trúc thư mục project 20](#_Toc134460346)

[Hình 12. Train Job trên command line 21](#_Toc134460347)

[Hình 13. Test Job trên command line 21](#_Toc134460348)

[Hình 14. Kết quả đầu ra 22](#_Toc134460349)

# **CHƯƠNG 1: TỔNG QUAN VỀ DỮ LIỆU LỚN**

## **1.1. Định nghĩa**

Big Data là các tập dữ liệu có khối lượng lớn và phức tạp. Độ lớn đến mức các phần mềm xử lý dữ liệu truyền thống không có khả năng thu thập, quản lý và xử lý dữ liệu trong một khoảng thời gian hợp lý.

Những tập dữ liệu lớn này có thể bao gồm các dữ liệu có cấu trúc, không có cấu trúc và bán cấu trúc, mỗi tập có thể được khai thác để tìm hiểu insights.

Big Data được tạo ra từ nhiều nguồn khác nhau như các hệ thống thông tin, mạng xã hội, cảm biến, máy móc và nhiều nguồn dữ liệu khác. Đặc điểm quan trọng của Big Data là V3 - Volume (khối lượng dữ liệu lớn), Velocity (tốc độ dữ liệu được tạo ra và xử lý) và Variety (đa dạng về dạng và cấu trúc của dữ liệu).



Hình 1. Minh hoạ nguồn gốc của dữ liệu

Các giải pháp Big Data cung cấp các công cụ, phương pháp và công nghệ được sử dụng để nắm bắt, lưu trữ, tìm kiếm và phân tích dữ liệu trong vài giây để tìm mối quan hệ và hiểu biết về cải tiến và lợi ích cạnh tranh mà trước đây không có.

Các công nghệ và công cụ Big Data như Hadoop, Spark, MapReduce, HDFS, NoSQL và Machine Learning được sử dụng để xử lý và phân tích dữ liệu trên quy mô lớn.

Big Data có ứng dụng rộng rãi trong nhiều lĩnh vực như kinh doanh, y tế, khoa học, xã hội học, marketing, tài chính và nhiều lĩnh vực khác. Nó giúp các tổ chức và doanh nghiệp tận dụng dữ liệu một cách hiệu quả, đưa ra quyết định thông minh và tạo ra giá trị đột phá từ thông tin ẩn chứa trong dữ liệu lớn.

Tóm lại, Big Data đại diện cho những thách thức và cơ hội về xử lý, phân tích và ứng dụng dữ liệu lớn. Nó mở ra một cánh cửa cho khám phá và khai thác tri thức từ nguồn dữ liệu phong phú và mang lại sự tiến bộ và tiềm năng trong nhiều lĩnh vực khác nhau.

## **1.2. Đặc trưng cơ bản của dữ liệu lớn**

Dữ liệu lớn có một số đặc trưng cơ bản sau đây:

1. Khối lượng lớn (Volume): Dữ liệu lớn được đánh giá dựa trên khối lượng rất lớn, thường ở mức tỷ lệ terabytes (TB), petabytes (PB) hoặc thậm chí exabytes (EB). Dữ liệu này có thể được tạo ra từ nhiều nguồn khác nhau như giao dịch khách hàng, dữ liệu cảm biến, thông tin mạng xã hội, dữ liệu máy móc và nhiều nguồn dữ liệu khác.
2. Tốc độ cao (Velocity): Dữ liệu lớn thường được tạo ra với tốc độ nhanh và liên tục. Dữ liệu này có thể được sinh ra từ nhiều nguồn đồng thời và phải được xử lý và phân tích ngay lập tức để đáp ứng yêu cầu kinh doanh và quyết định thời gian thực.
3. Đa dạng (Variety): Dữ liệu lớn có độ đa dạng cao, không chỉ bao gồm các số liệu cấu trúc như bảng dữ liệu SQL mà còn bao gồm dữ liệu phi cấu trúc như văn bản, hình ảnh, âm thanh, video, dữ liệu định vị địa lý và nhiều dạng dữ liệu khác. Đa dạng dữ liệu đòi hỏi các phương pháp và công cụ đặc biệt để lưu trữ, quản lý và phân tích.
4. Tính tin cậy và chính xác (Veracity): Dữ liệu lớn thường mang tính chất không chắc chắn, không chính xác và không đồng nhất. Nó có thể chứa nhiễu, dữ liệu thiếu, trùng lặp hoặc dữ liệu không chính xác. Đối mặt với tính tin cậy và chính xác của dữ liệu lớn đòi hỏi các phương pháp và công cụ để làm sạch và xử lý dữ liệu trước khi phân tích.
5. Value (Giá trị): Value đề cập đến giá trị mà dữ liệu lớn mang lại. Mục tiêu của việc phân tích dữ liệu lớn là tìm ra thông tin giá trị, các mẫu, xu hướng, và tri thức mới từ dữ liệu. Bằng cách phân tích dữ liệu lớn, ta có thể tạo ra thông tin quan trọng và hiểu sâu về ngành kinh doanh, khách hàng, hành vi người dùng, dự đoán xu hướng và đưa ra quyết định thông minh. Giá trị của dữ liệu lớn nằm ở khả năng phát hiện thông tin quan trọng và áp dụng nó để tối ưu hóa quy trình kinh doanh, cải thiện sản phẩm và dịch vụ, tăng cường sự hiểu biết về khách hàng và thúc đẩy sự đổi mới.
6. Khả năng mở rộng (Scalability): Dữ liệu lớn có khả năng mở rộng, tức là có thể xử lý và lưu trữ trên nhiều máy tính hoặc nút xử lý song song. Điều này đòi hỏi sự phân tán, song song và xử lý phân tán để đáp ứng được yêu cầu của dữ liệu lớn.
7. Tính phức tạp (Complexity): Dữ liệu lớn thường có tính phức tạp cao, với cấu trúc phức tạp và mối quan hệ phức tạp giữa các yếu tố dữ liệu. Dữ liệu lớn thường chứa thông tin không rõ ràng, các mẫu tiềm ẩn và mối quan hệ phức tạp giữa các biến. Điều này đòi hỏi các phương pháp và công cụ phân tích dữ liệu tiên tiến để tìm ra cấu trúc ẩn, mô hình dữ liệu và rút ra thông tin quan trọng từ dữ liệu phức tạp này.
8. Tính bất đồng nhất (Heterogeneity): Dữ liệu lớn có thể tồn tại trong nhiều hình thức và định dạng khác nhau. Nó có thể được thu thập từ nhiều nguồn khác nhau với cấu trúc và định dạng khác nhau. Điều này đòi hỏi sự linh hoạt trong việc xử lý và tích hợp dữ liệu từ các nguồn khác nhau.
9. Tính riêng tư và bảo mật (Privacy and Security): Dữ liệu lớn thường chứa thông tin nhạy cảm và cá nhân. Vì vậy, bảo vệ sự riêng tư và đảm bảo an ninh của dữ liệu là một vấn đề quan trọng trong việc xử lý và lưu trữ dữ liệu lớn.

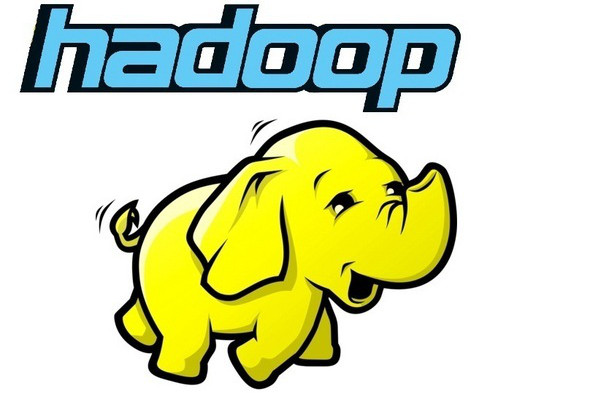
Những đặc trưng trên đòi hỏi các phương pháp và công nghệ đặc biệt để xử lý, lưu trữ, quản lý và phân tích dữ liệu lớn. 5V trong Big Data cung cấp cái nhìn tổng quan về đặc trưng cơ bản của dữ liệu lớn. Sự hiểu biết về những đặc trưng này là quan trọng để định hình và triển khai các phương pháp và công nghệ xử lý dữ liệu lớn hiệu quả, từ việc lưu trữ và quản lý dữ liệu cho đến phân tích và trích xuất thông tin giá trị.

## **1.3. Tổng quan về Hadoop**

### 1.3.1. Tổng quan

Hadoop là một Apache framework nguồn mở viết bằng Java cho phép phát triển các ứng dụng phân tán có cường độ dữ liệu lớn một cách miễn phí. Nó được thiết kế để mở rộng quy mô từ một máy chủ đơn sang hàng ngàn máy tính khác có tính toán và lưu trữ cục bộ (local computation and storage). Hadoop được phát triển dựa trên ý tưởng từ các công bố của Google về mô hình Map-Reduce và hệ thống file phân tán Google File System (GFS). Và có cung cấp cho chúng ta một môi trường song song để thực thi các tác vụ Map-Reduce.

Nhờ có cơ chế streaming mà Hadoop có thể phát triển trên các ứng dụng phân tán bằng cả java lẫn một số ngôn ngữ lập trình khác như C++, Pyhthon, Pearl,...



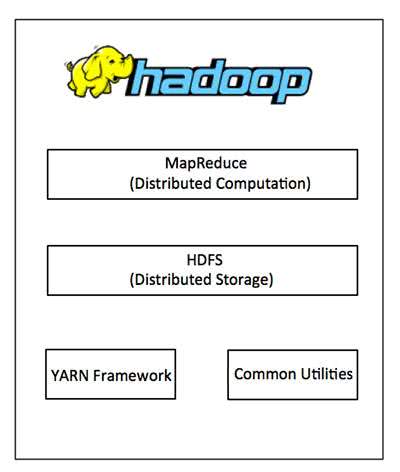
Hình 2. Logo biểu tượng của hadoop

### 1.3.2. Kiến trúc của Hadoop

Hadoop có một cấu trúc liên kết master-slave. Trong cấu trúc này, chúng ta có một node master và nhiều node slave . Chức năng của node master là gán một tác vụ cho các node slave khác nhau và quản lý tài nguyên. Các node slave là máy tính thực tế có thể không mạnh lắm. Các node slave lưu trữ dữ liệu thực trong khi trên master chúng ta có metadata.

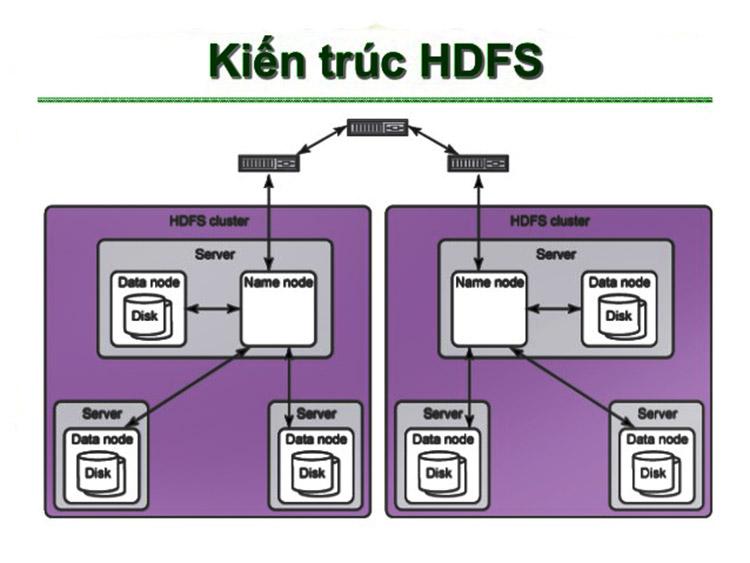
Kiến trúc Hadoop gồm có ba lớp chính đó là

* HDFS (Hadoop Distributed File System)
* Map-Reduce
* Yarn



Hình 3. Kiến trúc cảu hadoop

* Hadoop Distributed File System (HDFS): Đây là hệ thống tệp phân tán được thiết kế để lưu trữ và quản lý dữ liệu lớn trên các nút máy tính trong cụm Hadoop. HDFS chia dữ liệu thành các khối nhỏ và phân phối chúng trên các nút trong cụm để đảm bảo độ tin cậy và hiệu suất cao.

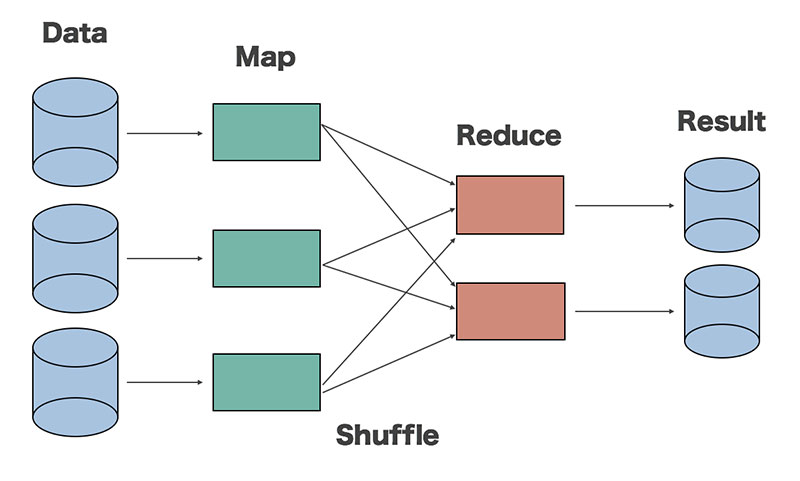


Hình 4. HDFS (Hadoop Distributed File System)

* MapReduce: Đây là mô hình xử lý dữ liệu phân tán trong Hadoop. MapReduce tách dữ liệu thành các phần nhỏ và thực hiện các phép toán xử lý song song trên các nút trong cụm. Các phép toán MapReduce bao gồm giai đoạn "Map" để xử lý dữ liệu và tạo ra các cặp khóa-giá trị, và giai đoạn "Reduce" để tổng hợp kết quả từ các cặp khóa-giá trị.

Map-Reduce thực hiện 2 chức năng chính đó là Map và Reduce

* Map: Sẽ thực hiện đầu tiên, có chức năng tải, phân tích dữ liệu đầu vào và được chuyển đổi thành tập dữ liệu theo cặp key/value
* Reduce: Sẽ nhận kết quả đầu ra từ tác vụ Map, kết hợp dữ liệu lại với nhau thành tập dữ liệu nhỏ hơn

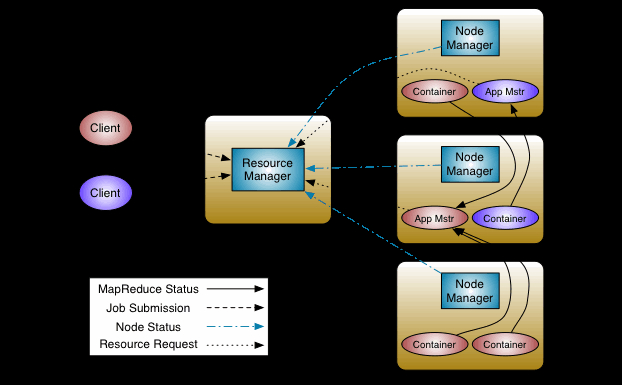


Hình 5. Quy trình Map Reduce

* YARN (Yet Another Resource Negotiator): YARN là một framework quản lý tài nguyên trong Hadoop. Nó giúp quản lý việc phân chia tài nguyên tính toán và lập lịch công việc trên các nút trong cụm. YARN cho phép các ứng dụng khác nhau chạy song song trên cùng một cụm Hadoop.

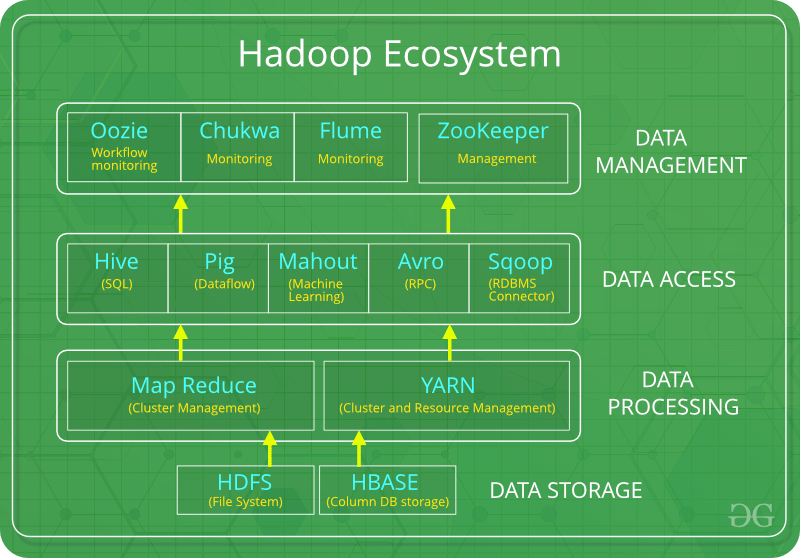
Bên trong YARN, chúng ta có hai trình quản lý ResourceManager và NodeManage

* ResourceManager: Quản lý toàn bộ tài nguyên tính toán của cluster.
* NodeManger: Giám sát việc sử dụng tài nguyên của container và báo cáo với ResourceManger. Các tài nguyên ở đây là CPU, memory, disk, network...



Hình 6. Trình quản lý của YARN

* Hadoop Ecosystem: Hadoop cung cấp một hệ sinh thái phong phú của các công nghệ và công cụ hỗ trợ. Điều này bao gồm các thành phần như Hive (truy vấn dữ liệu), Pig (ngôn ngữ xử lý dữ liệu), HBase (cơ sở dữ liệu cột gia đình), Spark (hệ thống tính toán phân tán), và nhiều công cụ khác để hỗ trợ xử lý và phân tích dữ liệu lớn.



Hình 7. Hadoop Ecosytem

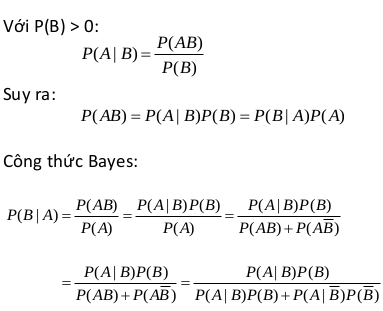
Hadoop đã trở thành một công cụ phổ biến để xử lý và lưu trữ dữ liệu lớn trong các công ty và tổ chức. Nó cho phép xử lý dữ liệu song song trên hàng ngàn nút máy tính và cung cấp khả năng mở rộng linh hoạt để xử lý dữ liệu ngày càng tăng lên.

# **CHƯƠNG 2: PHÂN LỚP DỮ LIỆU BẰNG THUẬT TOÁN NAIVE BAYES**

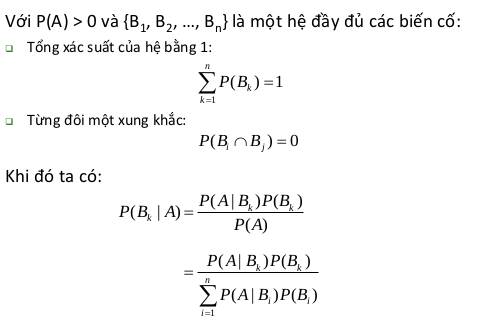
## **2.1. Thuật toán Naive Bayes**

### 2.1.1. Định lý Bayes

* Gọi A, B là hai biến cố:



* Công thức Bayes tổng quát:



* P(A|B)là Xác suất của A với điều kiện B , xác suất của A khi B xảy ra
* P(A)là Xác suất của A
* P(B|A)là Xác suất của B với điều kiện A , xác suất của B khi A xảy ra
* P(B)là xác suất của B

Trong đó ta gọi A là một chứng cứ (evidence) (trong bài toán phân lớp A sẽ là một phần tử dữ liệu), B là một giả thiết nào để cho A thuộc về một lớp C nào đó. Trong bài toán phân lớp chúng ta muốn xác định giá trị P(B/A) là xác suất để giả thiết B là đúng với chứng cứ A thuộc vào lớp C với điều kiện ra đã biết các thông tin mô tả A. P(B|A) là một xác suất hậu nghiệm (posterior probability hay posteriori probability) của B với điều kiện A.

Giả sử tập dữ liệu liệu khách hàng của chúng ta được mô tả bởi các thuộc tính tuổi và thu nhập, và một khách hàng X có tuổi là 25 và thu nhập là 2000$. Giả sử H là giả thiết khách hàng đõ sẽ mua máy tính, thì P(H|X) phản ánh xác xuất người dùng X sẽ mua máy tính với điều kiện ta biết tuổi và thu nhập của người đó.

Ngược lại P(H) là xác suất tiền nghiệm (prior probability hay priori probability) của H. Trong ví dụ trên, nó là xác suất một khách hàng sẽ mua máy tính mà không cần biết các thông tin về tuổi hay thu nhập của họ. Hay nói cách khác, xác suất này không phụ thuộc vào yếu tố X. Tương tự, P(X|H) là xác suất của X với điều kiện H (likelihood), nó là một xác suất hậu nghiệm. VÍ dụ, nó là xác suất người dùng X (có tuổi là 25 và thu nhập là $200) sẽ mua máy tính với điều kiện ta đã biết người đó sẽ mua máy tính. Cuối cùng P(X) là xác suất tiền nghiệm của X. Trong ví dụ trên, nó se là xác xuất một người trong tập dữ liệu sẽ có tuổi 25 và thu nhập $2000.

Posterior = Likelihood \* Prior / Evidence

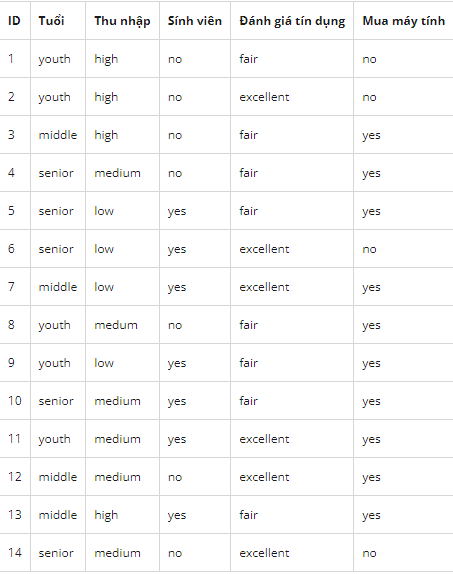
### 2.1.2. Phân lớp Naive Bayes

Bộ phân lớp Naive bayes hay bộ phân lớp Bayes (simple byes classifier) hoạt động như sau:

1. Gọi D là tập dữ liệu huấn luyện, trong đó mỗi phần tử dữ liệu X được biểu diễn bằng một vector chứa n giá trị thuộc tính A1, A2,...,An = {x1,x2,...,xn}
2. Giả sử có m lớp C1, C2,..,Cm. Cho một phần tử dữ liệu X, bộ phân lớp sẽ gán nhãn cho X là lớp có xác suất hậu nghiệm lớn nhất. Cụ thể, bộ phân lớp Bayes sẽ dự đoán X thuộc vào lớp Ci nếu và chỉ nếu: **P(Ci|X) > P(Cj|X) (1<= i, j <=m, i != j)**  
   Giá trị này sẽ tính dựa trên định lý Bayes.
3. Để tìm xác suất lớn nhất, ta nhận thấy các giá trị P(X) là giống nhau với mọi lớp nên không cần tính. Do đó ta chỉ cần tìm giá trị lớn nhất của P(X|Ci) \* P(Ci). Chú ý rằng P(Ci) được ước lượng bằng |Di|/|D|, trong đó Di là tập các phần tử dữ liệu thuộc lớp Ci. Nếu xác suất tiền nghiệm P(Ci) cũng không xác định được thì ta coi chúng bằng nhau P(C1) = P(C2) = ... = P(Cm), khi đó ta chỉ cần tìm giá trị P(X|Ci) lớn nhất.
4. Khi số lượng các thuộc tính mô tả dữ liệu là lớn thì chi phí tính toàn P(X|Ci) là rất lớn, dó đó có thể giảm độ phức tạp của thuật toán Naive Bayes giả thiết các thuộc tính độc lập nhau. Khi đó ta có thể tính:  
   **P(X|Ci) = P(x1|Ci)...P(xn|Ci)**

## **2.2. Ví dụ minh hoạ thuật toán**

* Ví dụ minh hoạ thuật toán:



Hình 8. Cơ sở dữ liệu khách hàng trong ví dụ

Giả sử ta có một khách hàng mới X có các thuộc tính  
X = (age = youth, income = medium, student = yes, credit\_rating = fair)  
Bây giớ cần xác định xem khách hàng X có thuộc lớp Cyes (mua máy tính) hay không, ta tính toán như sau:  
P(Cyes) = 9/14 = 0.357  
Các xác suất thành phần:  
P(age = youth|Cyes) = 2/9 = 0.222  
P(age = youth|Cno) = 3/5 = 0.6  
P(income = medium|Cyes) = 4/9 = 0.444  
P(income = medium|Cno) = 2/5 = 0.4  
P(student = yes|Cyes) = 6/9 = 0.667  
P(student = yes|Cno) = 1/5 = 0.2  
P(credit\_rating = fair|Cyes) = 6/9 = 0.667  
P(credit\_rating = fair|Cno) = 2/5 = 0.2  
Cuối cùng:  
P(X|Cyes) = 0.222 \* 0.444 \* 0.667 \* 0.667 = 0.044  
P(X|Cno) = 0.60.4 \* 0.2 \* 0.4 = 0.019  
P(X|Cyes)\*P(Cyes) = 0.044 \* 0.643  
P(X|Cno)\*P(Cno) =0.019 \* 0.357 = 0.007

 Từ kết quả này ta thấy P(X |Cyes)P(Cyes) có giá trị lớn nhất, do đó thuật toán Bayes sẽ kết luận là khách hàng X sẽ mua máy tính.

* Ưu điểm:
* Giả định độc lập: hoạt động tốt cho nhiều bài toán/miền sữ liệu và ứng dụng. Đơn giản nhưng đủ tốt để giải quyết nhiều bài toán như phân lớp văn bản, lọc spam,..
* Cho phép kết hợp tri thức tiền nghiệm (prior knowledge) và dữ liệu quan sát được (obserwed data).

Tốt khi có sự chệnh lệch số lượng giữa các lớp phân loại.

* Huấn luyện mô hình (ước lượng tham số) dễ và nhanh.
* Nhược điểm:
* Giả định độc lập (ưu điểm cũng chính là nhược điểm)

hầu hết các trường hợp thực tế trong đó có các thuộc tính trong các đối tượng thường phụ thuộc lẫn nhau.

* Vấn đề zero (đã nêu cách giải quyết ở phía trên)
* Mô hình không được huẩn luyện bằng phượng pháp tối ưu mạnh và chặt chẽ.

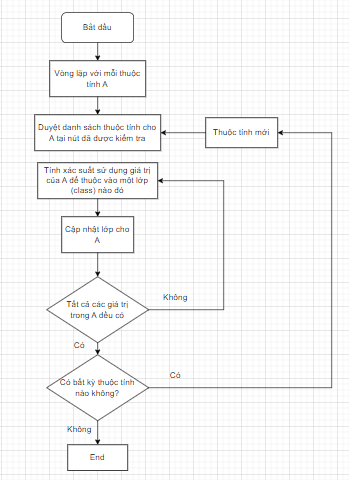
Tham số mủa mô hình là các ước lượng xác suất điều kiện đơn lẻ.

Không tính đến sự tương tác giữa các ước lượng này.

**CHƯƠNG 3: MAP REDUCE THUẬT TOÁN NAIVE BAYES TRONG DỰ ĐOÁN KHẢ NĂNG MẮC BỆNH ĐAU TIM Ở NGƯỜI**

## **3.1. Ý tưởng MapReduce Naive Bayes**

**Lưu đồ của thuật toán Naive Bayes:**



Hình 9. Lưu đồ thuật toán Naïve Bayes

Ý tưởng chính của MapReduce trong thuật toán Naive Bayes là phân tán việc tính toán các xác suất tiền điều kiện và xác suất hậu điều kiện trên các nút xử lý song song, sau đó tổng hợp kết quả để tính toán xác suất dự đoán.

- MapReduce Naive Bayes:

1. Giai đoạn Map:

* Đầu vào: Dữ liệu huấn luyện được chia thành các cặp (key, value), trong đó key là lớp (class) và value là các đặc trưng của mẫu dữ liệu.
* Bước Map: Với mỗi cặp (key, value), thực hiện tính toán xác suất tiền điều kiện và xác suất hậu điều kiện cho từng đặc trưng và lớp.
* Đầu ra: Các cặp (key, value) gom nhóm dựa trên key (lớp) để tạo thành các cặp (key, values).

1. Giai đoạn Reduce:

* Đầu vào: Các cặp (key, values) từ giai đoạn Map.
* Bước Reduce: Tính toán xác suất dự đoán cho mỗi lớp bằng cách kết hợp các xác suất đã tính toán trong giai đoạn Map.
* Đầu ra: Xác suất dự đoán cho các lớp.

Trong quá trình MapReduce, các công việc tính toán và gom nhóm được thực hiện song song trên các nút xử lý riêng biệt. Điều này giúp tăng tốc độ xử lý và khả năng mở rộng của thuật toán Naive Bayes khi áp dụng cho các tập dữ liệu lớn.

Đồng thời, MapReduce cũng cung cấp khả năng xử lý dữ liệu phân tán và độ tin cậy cao thông qua cơ chế sao lưu và khả năng xử lý lỗi tự động. Điều này giúp đảm bảo tính toàn vẹn và hiệu suất của quá trình tính toán Naive Bayes trên dữ liệu lớn.

## **3.2. Mô tả bài toán**

Dữ liệu "HealthCare" chứa thông tin cơ bản về hồ sơ bệnh án của bệnh nhân bao gồm rất nhiều thuộc tính nhưng trong đó có 13 thuộc tính (attribute) nổi bật như tuổi, giới tính, loại đau ngực và các thông số thực nghiệm khác. Mỗi mẫu xét nghiệm được gán một nhãn (label) thể hiện nguy cơ đau tim tương ứng.

Mục tiêu của bài toán là sử dụng thuật toán Naïve Bayes xây dựng một mô hình phân loại (classifier) từ dữ liệu huấn luyện để có thể dự đoán khả năng bị mắc bệnh đau tim của các mẫu (bệnh nhân) chưa biết. Mô hình phân loại sẽ được huấn luyện trên một tập dữ liệu đã được gán nhãn (labeled dataset) gồm các mẫu và nhãn tương ứng.

## **3.3. Mô tả dữ liệu**

Dữ liệu được lấy trên UCI data: https://archive.ics.uci.edu/dataset/45/heart+disease

* Số lượng mẫu: Dữ liệu HealthCare bao gồm tổng cộng 303 mẫu xét nghiệm.
* Thuộc tính (attribute): Dữ liệu chứa 13 thuộc tính (attribute) mô tả các thuộc tính của bệnh nhân. Các thuộc tính bao gồm: age(độ tuổi), sex(giới tính), cp(loại đau ngực), trestbps(huyết áp khi nghỉ ngơi), chol(cholestoral huyết thanh), fbs(đường huyết sau khi ăn), restecg(kết quả điện tâm đồ khi nghỉ ngơi), thalach(nhịp tim tối đa đạt được), exang(đau thắt ngực do vận động), oldpeak(độ giảm st khi vận động so với lúc nghỉ ngơi), slope(góc của đoạn ST tại đỉnh của bài tập vận động), ca(số mạch chính tô bằng tia X), thal(Thalassemia).
* Có 2 kết quả sau khi phân loại dữ liệu là 0(ít nguy cơ đau tim) và 1(nhiều nguy cơ đau tim)

## **3.4. Xây dựng mô hình**

* **Các bước thực hiện**

1. Chuẩn bị dữ liệu:

* Tải dữ liệu "HealCare" từ UCI data hoặc từ nguồn tương tự.
* Tiến hành xử lý dữ liệu bằng cách loại bỏ các dòng hoặc cột không cần thiết, xử lý dữ liệu thiếu, chuyển đổi đặc trưng sang định dạng phù hợp.

1. Chia dữ liệu:

* Chia tập dữ liệu "HealCare" thành hai phần: tập huấn luyện (training set) và tập kiểm tra (test set) chia theo tỷ lệ 70:30.

1. Tiền xử lý dữ liệu (Preprocessing):

* Tiến hành tiền xử lý dữ liệu trên tập huấn luyện và tập kiểm tra, bao gồm chuẩn hóa dữ liệu, mã hóa các đặc trưng dạng văn bản thành dạng số, loại bỏ nhiễu, xử lý dữ liệu không đầy đủ, và các bước tiền xử lý khác.

1. Xây dựng mô hình Naive Bayes:

* Định nghĩa và triển khai mô hình Naive Bayes sử dụng thư viện hoặc code từ đầu. Các bước bao gồm tính toán xác suất tiền điều kiện và xác suất hậu điều kiện, cùng với các phép tính liên quan đến Naive Bayes.

1. MapReduce/Hadoop cho Naive Bayes:
   * Áp dụng MapReduce/Hadoop để tối ưu hóa việc tính toán Naive Bayes trên tập dữ liệu lớn. Định nghĩa và triển khai các công việc Map và Reduce trong Hadoop để tính toán xác suất và dự đoán của mô hình Naive Bayes trên tập dữ liệu "HealCare".
2. Huấn luyện và đánh giá mô hình:

* Sử dụng tập huấn luyện để huấn luyện mô hình Naive Bayes trên MapReduce/Hadoop.
* Sử dụng tập kiểm tra để đánh giá mô hình bằng cách tính toán độ đo độ chính xác (accuracy)

### 3.4.1. Xây dựng lớp NaiveBayesTrainMapper

NaiveBayesTrainMapper: Là một class Java trong MapReduce của thuật toán Naive Bayes, nhiệm vụ của nó là thực hiện phần Map trong giai đoạn huấn luyện (training) của Naive Bayes. Trên mỗi cặp (key, value) đầu vào, mapper sẽ tính toán xác suất tiền điều kiện và xác suất hậu điều kiện cho từng đặc trưng và lớp. Kết quả của mapper sẽ được gửi đến NaiveBayesTrainReduce.

* Input: Các cặp (key, value) trong định dạng key-value, trong đó key là không quan trọng và value chứa dữ liệu huấn luyện.
* Output: Các cặp (key, value) trong định dạng key-value, trong đó key là lớp (class) của dữ liệu và value là thông tin về xác suất tiền điều kiện và xác suất hậu điều kiện cho các đặc trưng.

### 3.4.2. Xây dựng lớp NaiveBayesTrainReduce

NaiveBayesTrainReduce: Là một class Java trong MapReduce của thuật toán Naive Bayes, nhiệm vụ của nó là thực hiện phần Reduce trong giai đoạn huấn luyện (training) của Naive Bayes. Đầu vào của reduce là các cặp (key, values) từ mapper, được gom nhóm dựa trên key (lớp). Trong bước reduce, các xác suất đã tính toán từ mapper sẽ được tổng hợp lại để tính toán xác suất dự đoán cho các lớp. Kết quả cuối cùng sẽ được gửi đến NaiveBayesTrainJob.

* Input: Các cặp (key, values) từ NaiveBayesTrainMapper, trong đó key là lớp (class) và values là danh sách các xác suất tiền điều kiện và xác suất hậu điều kiện cho các đặc trưng.
* Output: Các cặp (key, value) trong định dạng key-value, trong đó key là lớp (class) và value là thông tin về xác suất dự đoán cho lớp đó

### 3.4.3. Xây dựng lớp NaiveBayesTrainJob

NaiveBayesTrainJob: Là một class Java trong MapReduce của thuật toán Naive Bayes, nhiệm vụ của nó là tạo và cấu hình công việc huấn luyện (training job). Nó sẽ định nghĩa cấu trúc và các thông số cần thiết cho công việc MapReduce, bao gồm cài đặt các mapper, reducer, định dạng dữ liệu đầu vào và đầu ra. Khi công việc được thiết lập, NaiveBayesTrainJob sẽ chạy và tiến hành huấn luyện mô hình Naive Bayes.

* Input: Dữ liệu huấn luyện.
* Output: Mô hình Naive Bayes đã được huấn luyện.

### 3.4.4. Xây dựng lớp NaiveBayesTestMapper

NaiveBayesTestMapper: Là một class Java trong MapReduce của thuật toán Naive Bayes, nhiệm vụ của nó là thực hiện phần Map trong giai đoạn kiểm tra (testing) của Naive Bayes. Trên mỗi dữ liệu kiểm tra, mapper sẽ sử dụng mô hình đã được huấn luyện để tính toán xác suất dự đoán cho các lớp. Kết quả của mapper sẽ được gửi đến NaiveBayesTestReduce.

* Input: Các cặp (key, value) trong định dạng key-value, trong đó key là không quan trọng và value chứa dữ liệu kiểm tra.
* Output: Các cặp (key, value) trong định dạng key-value, trong đó key là không quan trọng và value chứa dữ liệu kiểm tra và thông tin về lớp dự đoán.

### 3.4.5. Xây dựng lớp NaiveBayesTestReduce

NaiveBayesTestReduce: Là một class Java trong MapReduce của thuật toán Naive Bayes, nhiệm vụ của nó là thực hiện phần Reduce trong giai đoạn kiểm tra (testing) của Naive Bayes. Đầu vào của reduce là các cặp (key, values) từ mapper, đại diện cho các xác suất dự đoán của các lớp. Trong bước reduce, các xác suất dự đoán sẽ được tổng hợp và đánh giá để xác định lớp dự đoán cuối cùng cho mỗi mẫu dữ liệu kiểm tra. Kết quả cuối cùng của reduce sẽ là lớp dự đoán cho tất cả các mẫu dữ liệu kiểm tra. Kết quả này được gửi đến NaiveBayesTestJob.

* Input: Các cặp (key, values) từ NaiveBayesTestMapper, trong đó key là không quan trọng và values là danh sách các dữ liệu kiểm tra và thông tin về lớp dự đoán.
* Output: Các cặp (key, value) trong định dạng key-value, trong đó key là không quan trọng và value chứa dữ liệu kiểm tra và lớp dự đoán tương ứng.

### 3.4.6. Xây dựng lớp NaiveBayesTestJob

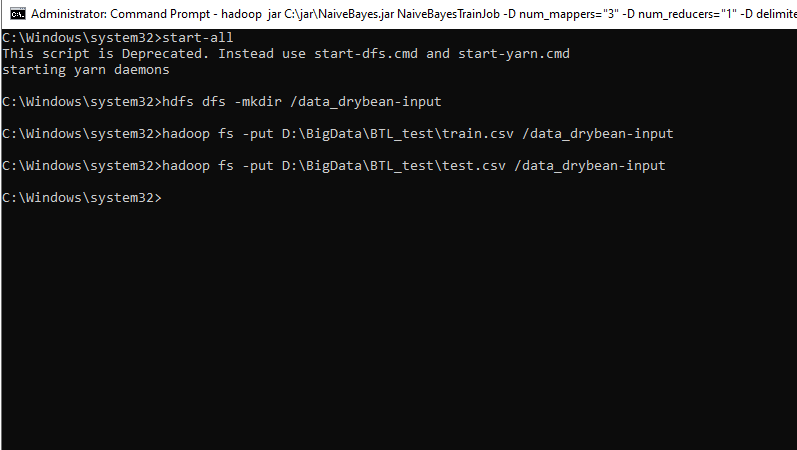
NaiveBayesTestJob: Là một class Java trong MapReduce của thuật toán Naive Bayes, nhiệm vụ của nó là tạo và cấu hình công việc kiểm tra (testing job). Nó sẽ định nghĩa cấu trúc và các thông số cần thiết cho công việc MapReduce, bao gồm cài đặt các mapper, reducer, định dạng dữ liệu đầu vào và đầu ra. Khi công việc được thiết lập, NaiveBayesTestJob sẽ chạy và tiến hành kiểm tra mô hình Naive Bayes trên tập dữ liệu kiểm tra.

* Input: Dữ liệu kiểm tra và mô hình Naive Bayes đã được huấn luyện.
* Output: Kết quả của quá trình phân loại dữ liệu kiểm tra, bao gồm dữ liệu kiểm tra và lớp dự đoán tương ứng.

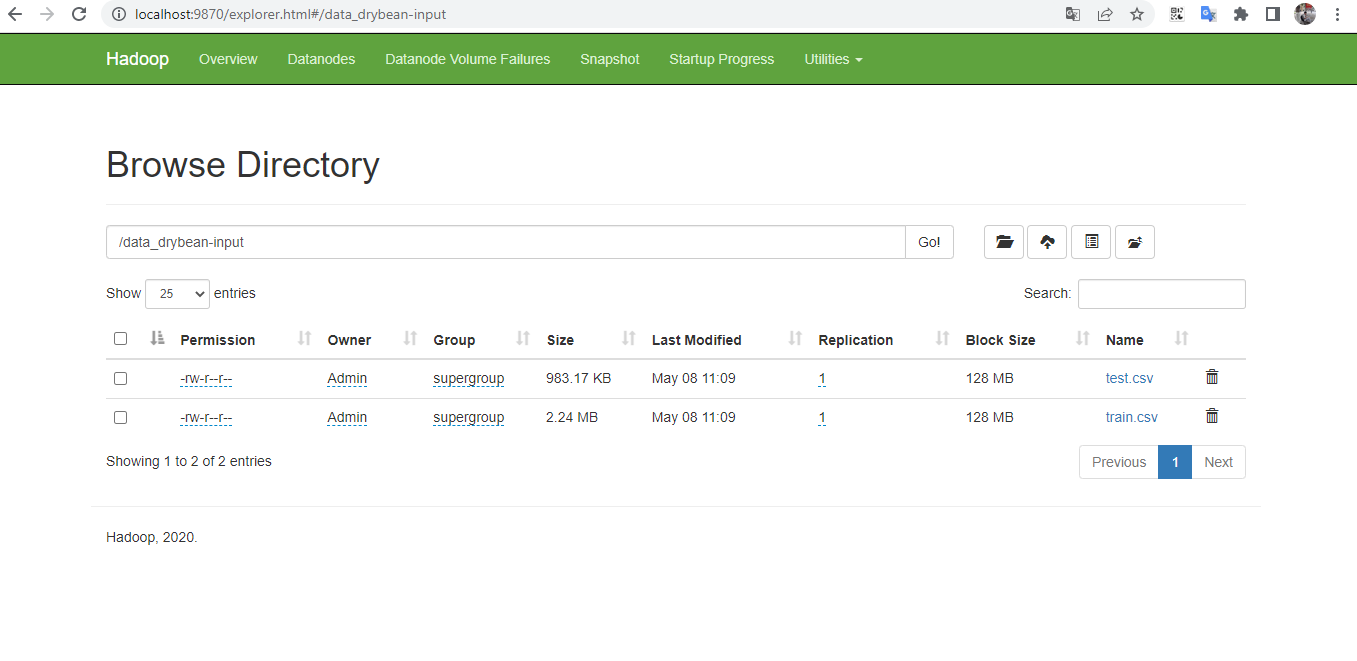
## **3.5. Demo chương trình cài đặt**

### 3.5.1. Demo cài đặt hadoop thành công

Tạo 1 folder trong hadoop để lưu trữ file drybean.csv, train.csv, test.csv:

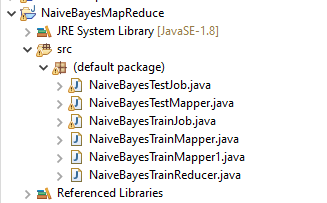


Hình 10. Tạo foler data-drybean và thêm vào file train.csv và test.csv ở hadoop



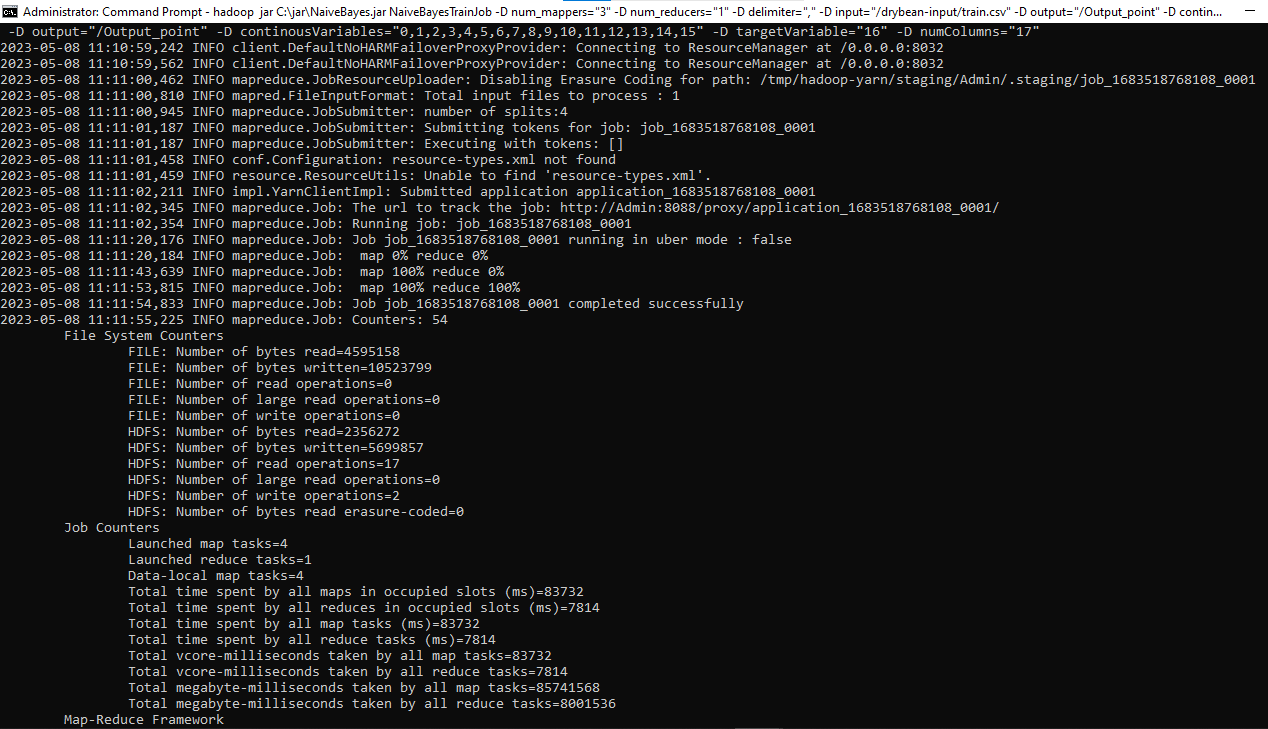
### 3.5.2. Thử nghiệm

* Cấu trúc của thư mục project:



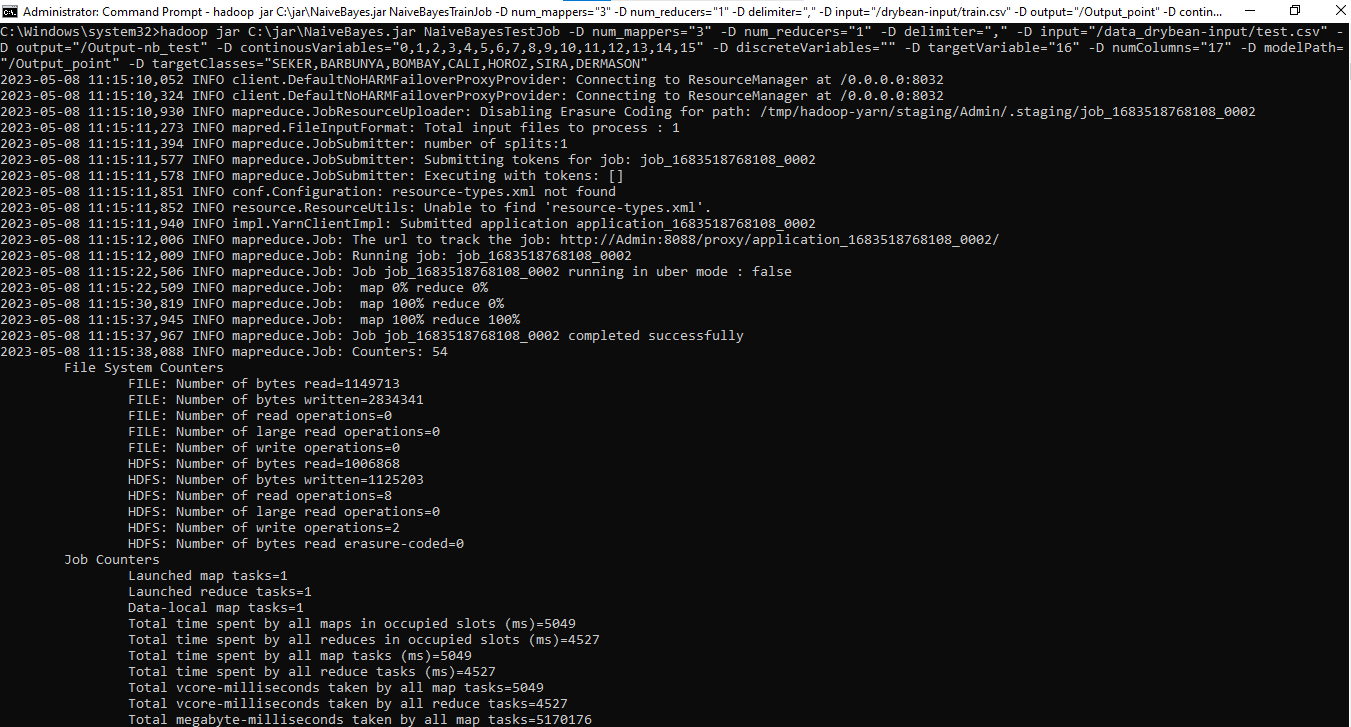
Hình 11. Cấu trúc thư mục project

* Chạy Train Job trên command line



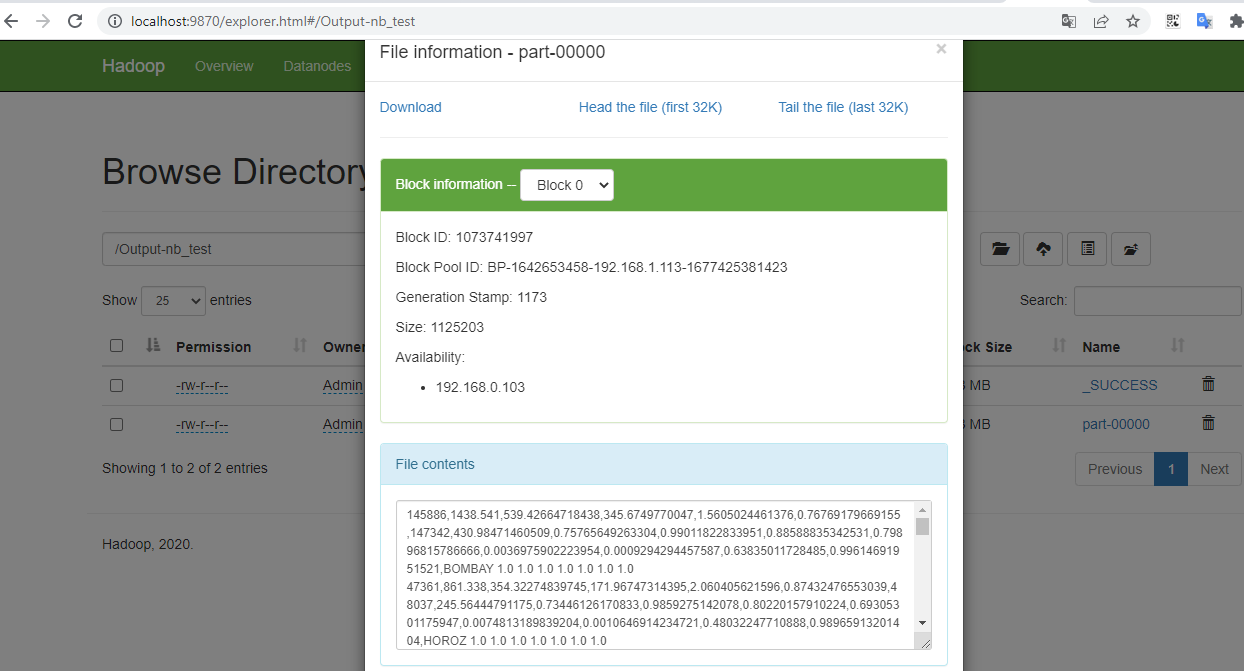
Hình 12. Train Job trên command line

* Chạy Test Job



Hình 13. Test Job trên command line

* File kết quả đầu ra:



Hình 14. Kết quả đầu ra

Đánh giá:

+ Chương trình đã mapreduce hóa thành công thuật toán ID3 đáp ứng yêu cầu đặt ra.

+ Kết quả chạy khớp với kết quả đã thực hiện thủ công trước đó → có triển vọng

mở rộng với tập dữ liệu lớn hơn

# **CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

## **4.1. Kết luận**

Trong bài tập lớn môn Big Data với đề tài "Thuật toán Naive Bayes và Map Reduce hoá trong dự đoán khả năng mắc bệnh đau tim ở người" dưới sự hướng dẫn của thầy Tạ Quang Chiểu, chúng em đã tìm hiểu và áp dụng thuật toán Naive Bayes kết hợp với Map Reduce để phân loại hồ sơ bệnh án của bệnh nhân nhằm xác định sớm bệnh đau tim. Đây là một bài toán phân loại dữ liệu quan trọng trong lĩnh vực xử lý dữ liệu lớn.

Đầu tiên, chúng em đã làm quen với Big Data và các đặc trưng cơ bản của dữ liệu lớn. Sau đó, chúng em đã tìm hiểu về Hadoop - một framework phổ biến được sử dụng để xử lý dữ liệu lớn. Hadoop cung cấp khả năng phân tán tính toán và xử lý song song, giúp chúng ta khai thác tối đa tiềm năng của Big Data.

Tiếp theo, chúng em đã tìm hiểu về thuật toán Naive Bayes - một thuật toán phân loại dựa trên xác suất. Naive Bayes là một thuật toán đơn giản nhưng hiệu quả, phù hợp cho việc phân loại dữ liệu với các đặc trưng độc lập.

Sau đó, chúng em đã áp dụng phương pháp Map Reduce vào thuật toán Naive Bayes. Map Reduce là một mô hình lập trình phân tán giúp chia nhỏ công việc và xử lý song song trên nhiều node trong hệ thống phân tán. Chúng em đã thiết kế và triển khai các class Java như NaiveBayesTrainMapper, NaiveBayesTrainReduce, NaiveBayesTrainJob, NaiveBayesTestMapper, NaiveBayesTestReduce và NaiveBayesTestJob để thực hiện quá trình huấn luyện và kiểm tra mô hình Naive Bayes trên dữ liệu lớn.

Cuối cùng, chúng em đã áp dụng thuật toán Naive Bayes và Map Reduce vào bài toán trong dự đoán khả năng mắc bệnh đau tim ở người sử dụng dữ liệu từ UCI Data. Qua quá trình này, chúng em đã tạo ra mô hình phân loại dựa trên xác suất và triển khai nó trên môi trường Map Reduce để xử lý dữ liệu lớn.

Kết quả của đề tài này là một hệ thống trong dự đoán khả năng mắc bệnh đau hiệu quả, cho phép phân loại dữ liệu lớn một cách nhanh chóng và chính xác. Đồng thời, chúng ta đã có cơ hội tìm hiểu và áp dụng các công nghệ Big Data như Hadoop và Map Reduce để xử lý dữ liệu lớn và tối ưu hóa hiệu suất tính toán.

## **4.2. Hướng phát triển**

Hướng phát triển của đề tài "Thuật toán Naive Bayes và Map Reduce hoá trong trong dự đoán khả năng mắc bệnh đau tim ở người" bao gồm mở rộng phạm vi áp dụng, tối ưu hóa hiệu suất tính toán, xử lý dữ liệu không đồng nhất, kết hợp với các thuật toán phân loại khác, và tích hợp các công cụ và nền tảng Big Data. Qua việc nghiên cứu và áp dụng các hướng phát triển này, đề tài có tiềm năng mang lại giá trị và ứng dụng rộng rãi trong lĩnh vực phân loại dữ liệu lớn.

# **TÀI LIỆU THAM KHẢO**

[1] Tài liệu bài giảng của thầy Tạ Quang Chiểu

[2] MapReduce và hadoop, Link: <https://hieundtlu.github.io/tlu/>

[3] Thuật toán Naive Bayes, Link: <https://www.javatpoint.com/machine-learning-naive-bayes-classifier>

[4] Thuật toán Naive Bayes Classification, Link: <https://machinelearningcoban.com/2017/08/08/nbc/>

[5] Dataset, Link: https://archive.ics.uci.edu/dataset/45/heart+disease

[6] Lý thuyết về hadoop, Link: <https://lanit.com.vn/hadoop-la-gi.html>