

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  
**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

...📖...



**BÁO CÁO THÍ NGHIỆM/THỰC NGHIỆM**  
**HỌC PHẦN: TRÍ TUỆ NHÂN TẠO**

**XÂY DỰNG ỨNG DỤNG CHẨN ĐOÁN BỆNH VỀ**  
**MÁU THÔNG QUA CHỈ SỐ XÉT NGHIỆM**

Giáo viên hướng dẫn: Ths Trần Thanh Huân

Nhóm : 10

Thành viên nhóm: Phan Trường Giang

Hồ Cảnh Quý

Nguyễn Văn Toàn

Lê Minh Tân

Lớp: 20242IT6094001

**Hà Nội, 2025**

# MỤC LỤC

DANH MỤC TỪ VIẾT TẮT .....	1
DANH MỤC HÌNH ẢNH .....	2
LỜI MỞ ĐẦU .....	3
Chương 1: Tổng quan về học máy .....	4
1.1. Giới thiệu .....	4
1.2. Định nghĩa .....	4
1.3. Hoạt động của học máy .....	5
1.4. Phân loại phương pháp machine learning .....	5
1.4.1. Supervised learning (Học có giám sát) .....	5
1.4.2. Học không giám sát (Unsupervised Learning) .....	8
1.4.3. Semi-Supervised Learning (Học bán giám sát) .....	10
1.4.4. Reinforcement Learning (Học tăng cường) .....	11
1.5. Ứng dụng của học máy trong y tế .....	14
Chương 2: Các kỹ thuật phân lớp .....	16
2.1. K-nearest neighbor .....	16
2.1.1. Giới thiệu .....	16
2.1.2. Nguyên lý hoạt động .....	16
2.1.3. Ưu điểm .....	17
2.1.4. Nhược điểm .....	18
2.1.5. Ứng dụng thuật toán .....	18
2.2. Naïve bayes .....	18
2.2.1. Giới thiệu .....	18
2.2.2. Nguyên lý hoạt động .....	19
2.2.3. Các loại Naive Bayes .....	21
2.2.4. Ưu điểm .....	22
2.2.5. Nhược điểm .....	22
2.2.6. Ứng dụng thuật toán .....	22
2.3. Hiệu năng của mô hình .....	23
Chương 3: Xây dựng ứng dụng .....	26
3.1. Lợi ích học máy với đề tài .....	26
3.2. Công cụ và ngôn ngữ .....	27
3.3. Xây dựng ứng dụng .....	28
3.3.1. Mô tả dữ liệu .....	28
3.3.2. Áp dụng KNN vào bài toán .....	30
3.3.3. Áp dụng Navie Bayes vào bài toán .....	36

3.4. So sánh độ hiệu quả của hai thuật toán.....	42
KẾT LUẬN .....	47
TÀI LIỆU THAM KHẢO.....	48

## DANH MỤC TỪ VIẾT TẮT

<b>MCH</b>	Mean Corpuscular Hemoglobine (Lượng huyết sắc tố trung bình hồng cầu)
<b>MCHC</b>	Mean Corpuscular Hemoglobine Concentration (Nồng độ huyết sắc tố trung bình hồng cầu)
<b>MCV</b>	Mean Corpuscular Volume (Thể tích trung bình hồng cầu)

## DANH MỤC HÌNH ẢNH

Hình 3.1: Hình ảnh file dữ liệu huấn luyện và thử nghiệm .....	28
Hình 3.2: Hình ảnh dữ liệu huấn luyện .....	29
Hình 3.3: Hình ảnh dữ liệu thử nghiệm.....	30
Hình 3.4: Các thư viện gần nhất (KNN).....	30
Hình 3.5: Thông tin chi tiết về Dataframe (KNN).....	31
Hình 3.6: Dữ liệu được đọc từ file (KNN).....	32
Hình 3.7: Dữ liệu sau khi được chuẩn hóa (KNN).....	33
Hình 3.8: Hàm tính khoảng cách Euclidean giữa hai điểm (KNN) .....	33
Hình 3.9: Hàm dự đoán lớp của kiểu dữ liệu kiểm tra (KNN) .....	34
Hình 3.10: Chia dữ liệu và đào tạo mô hình (KNN) .....	34
Hình 3.11: Ma trận nhầm lẫn và độ chính xác của mô hình (KNN) .....	35
Hình 3.12: Dữ liệu thử nghiệm (KNN) .....	36
Hình 3.13: Kết quả thử nghiệm (KNN).....	36
Hình 3.14: Các thư viện gần nhất (Navie Bayes).....	37
Hình 3.15: Đọc dữ liệu từ file (Navie Bayes) .....	38
Hình 3.16: Thông tin chi tiết về DataFrame (Navie Bayes).....	38
Hình 3.17: Dữ liệu sau khi đã được chuẩn hóa (Navie Bayes).....	39
Hình 3.18: Phân tách dữ liệu huấn luyện (Navie Bayes) .....	39
Hình 3.19: Phân tách dữ liệu thử nghiệm (Navie Bayes).....	40
Hình 3.20: Tạo mô hình Navie Bayes và dữ liệu dự đoán.....	40
Hình 3.21: Ma trận nhầm lẫn và độ chính xác của mô hình (Navie Bayes) .....	41
Hình 3.22: Tạo dữ liệu thử nghiệm lại kết quả (Navie Bayes) .....	41
Hình 3.23: Kết quả sau khi đã thử nghiệm (Navie Bayes) .....	42
Hình 3.24: Thư viện vẽ biểu đồ .....	43
Hình 3.25: So sánh Ma trận nhầm lẫn (Confusion Matrix) .....	43
Hình 3.26: Lưu trữ độ chính xác (Accuracy Score) .....	44
Hình 3.27: Đọc dữ liệu Accuracy Score vừa lưu.....	45
Hình 3.28: So sánh độ chính xác (Accuracy Score) .....	46

# LỜI MỞ ĐẦU

Sự phát triển vượt bậc của công nghệ, đặc biệt là trong lĩnh vực trí tuệ nhân tạo (AI) và học máy (Machine Learning), đã mở ra nhiều cơ hội mới trong việc giải quyết các bài toán phức tạp của khoa học dữ liệu. Một trong những ứng dụng quan trọng của học máy là phân tích dữ liệu y khoa, nơi các hệ thống có thể tự động hóa quá trình xử lý và phát hiện các mẫu từ các tập dữ liệu lớn, giúp tối ưu hóa quy trình và cải thiện độ chính xác của các quyết định.

Là sinh viên khoa công nghệ thông tin, chúng tôi nhận thấy tiềm năng lớn của việc ứng dụng AI và học máy trong việc phân tích dữ liệu xét nghiệm máu để hỗ trợ chẩn đoán bệnh. Việc nghiên cứu và phát triển một ứng dụng chẩn đoán bệnh về máu không chỉ mang lại giá trị thực tiễn mà còn giúp chúng tôi áp dụng kiến thức chuyên ngành vào giải quyết các vấn đề trong đời sống.

Bệnh lý về máu, chẳng hạn như thiếu máu, bệnh bạch cầu, hay rối loạn đông máu, thường được chẩn đoán thông qua các chỉ số xét nghiệm máu. Những chỉ số này chứa đựng một lượng lớn thông tin có thể được khai thác bằng các thuật toán học máy để hỗ trợ phát hiện bất thường, phân loại bệnh, hoặc đưa ra dự đoán. Tuy nhiên, thay vì sử dụng các phương pháp chẩn đoán truyền thống dựa vào chuyên môn y học, đề tài này tập trung vào việc áp dụng học máy để phân tích dữ liệu xét nghiệm máu mà không đòi hỏi sự can thiệp của kiến thức chuyên sâu trong y học.

Mục tiêu chính của đề tài “Nghiên cứu xây dựng ứng dụng chẩn đoán bệnh về máu dựa trên chỉ số xét nghiệm” là phát triển một hệ thống tự động, dựa trên thuật toán học máy, có khả năng phân tích các chỉ số xét nghiệm máu để hỗ trợ bác sĩ hoặc chuyên gia y tế trong việc đưa ra quyết định. Thay vì đi sâu vào các khía cạnh sinh học của bệnh lý, nghiên cứu tập trung vào xử lý dữ liệu, xây dựng mô hình, và tối ưu hóa hiệu năng của hệ thống học máy.

Với cách tiếp cận tập trung vào khoa học dữ liệu và công nghệ, đề tài này không chỉ có giá trị trong việc phát triển các ứng dụng học máy cho lĩnh vực y học mà còn góp phần mở rộng khả năng xử lý dữ liệu trong các bài toán tương tự ở nhiều ngành khác. Kết quả nghiên cứu được kỳ vọng sẽ mang lại một công cụ hiệu quả và dễ triển khai, góp phần cải thiện tốc độ và độ chính xác trong phân tích dữ liệu xét nghiệm, từ đó hỗ trợ nâng cao chất lượng chăm sóc sức khỏe.

Chúng tôi xin chân thành cảm ơn!

# Chương 1: Tổng quan về học máy

## 1.1. Giới thiệu

Học máy (Machine Learning) là một lĩnh vực thuộc trí tuệ nhân tạo, tập trung vào việc nghiên cứu và phát triển các kỹ thuật cho phép hệ thống “học” tự động từ dữ liệu để giải quyết các vấn đề cụ thể.

Hiểu đơn giản, thuật ngữ này nói tới việc con người dạy máy tính nâng cao khả năng thực hiện các tác vụ cụ thể. Cụ thể là cung cấp các dữ liệu và thuật toán có sẵn để máy tính đưa ra dự đoán hoặc tự ra các quyết định. Thông thường, con người chỉ cần lập trình phần mềm với các dòng lệnh cụ thể để máy tính hiểu và thực hiện. Với Machine Learning, máy tính sẽ tự “học” cách giải quyết công việc thông qua những dữ liệu đã được thu thập và cung cấp.

Theo Tom Mitchell trong cuốn sách “Machine Learning” xuất bản năm 1997, Machine Learning như 1 chương trình, nhiệm vụ của nó là thực hiện 1 nhiệm vụ  $T$  nào đó, khi thực hiện xong, ta thu được trải nghiệm  $E$ . Nhờ vào việc học hỏi trải nghiệm  $E$ , ta có thể thay đổi (hoặc không) để tiến tới thực hiện task  $T+1$ , và nhằm cải thiện hiệu suất  $P$ .

## 1.2. Định nghĩa

**Trí tuệ nhân tạo (AI)** là công nghệ cho phép máy móc, đặc biệt là máy tính, "học hỏi" và "suy nghĩ" như con người. Trí tuệ nhân tạo khác với việc lập trình logic trong các ngôn ngữ lập trình là ở việc ứng dụng các hệ thống học máy (machine learning) để mô phỏng trí tuệ của con người trong các xử lý mà con người làm tốt hơn máy tính.

Cụ thể, trí tuệ nhân tạo giúp máy tính có được những trí tuệ của con người như: biết suy nghĩ và lập luận để giải quyết vấn đề, biết giao tiếp do hiểu ngôn ngữ, tiếng nói, biết học và tự thích nghi,...

**Machine Learning** là một tập con của AI. Machine Learning là một lĩnh vực nhỏ của Khoa Học Máy Tính, nó có khả năng tự học hỏi dựa trên dữ liệu đưa vào mà không cần phải được lập trình cụ thể. Những năm gần đây, khi mà khả năng tính toán của các máy tính được nâng lên một tầm cao mới và lượng dữ liệu khổng lồ được thu thập bởi các hãng công nghệ lớn, Machine Learning đã tiến thêm một bước dài và một lĩnh vực mới được ra đời gọi là Deep Learning (Học Sâu). Deep Learning đã giúp máy tính thực thi những việc tưởng chừng như không thể vào 10 năm trước: phân loại cả ngàn vật thể khác nhau trong các bức ảnh, tự tạo chú

thích cho ảnh, bắt chước giọng nói và chữ viết của con người, giao tiếp với con người, hay thậm chí cả sáng tác văn hay âm nhạc.

### **1.3. Hoạt động của học máy**

Quy trình triển khai thuật toán học máy thường bao gồm 6 bước như sau:

Bước 1: Thu thập dữ liệu (Gathering data/Data collection)

Bước 2: Tiền xử lý dữ liệu (Data preprocessing)

Trích xuất dữ liệu – Data extraction

Làm sạch dữ liệu – Data cleaning

Chuyển đổi dữ liệu – Data transformation

Chuẩn hóa dữ liệu – Data normalization

Trích xuất đặc trưng – Feature extraction

Bước 3: Phân tích dữ liệu (Data analysis)

Bước 4: Xây dựng mô hình máy học (Model building)

Bước 5: Huấn luyện mô hình (Model training)

Bước 6: Đánh giá mô hình (Model evaluation)

Trong số các bước này, thu thập dữ liệu, tiền xử lý và xây dựng bộ dữ liệu thường chiếm nhiều thời gian và công sức nhất. Đây là những bước cực kỳ quan trọng, quyết định đến hiệu quả của thuật toán máy học. Độ chính xác của kết quả phụ thuộc rất lớn vào lượng dữ liệu đầu vào.

### **1.4. Phân loại phương pháp machine learning**

#### **1.4.1. Supervised learning (Học có giám sát)**

Supervised Learning (Học có giám sát) là một phương pháp trong Machine Learning, trong đó mô hình được huấn luyện bằng cách sử dụng các tập dữ liệu đã được gán nhãn. Thuật toán sẽ học cách nhận diện các mẫu và mối quan hệ giữa dữ liệu đầu vào và đầu ra, từ đó có thể dự đoán chính xác kết quả khi gặp dữ liệu mới trong thực tế.

Trong Supervised Learning, dữ liệu có gán nhãn bao gồm các điểm dữ liệu mẫu cùng với các đầu ra hoặc câu trả lời chính xác. Khi dữ liệu đầu vào được đưa vào thuật toán học máy, nó điều chỉnh các trọng số cho đến khi mô hình được huấn luyện phù hợp. Dữ liệu huấn luyện có gán nhãn dạy rõ ràng cho mô hình cách nhận diện mối quan hệ giữa các đặc trưng và nhãn dữ liệu.



Thuật toán supervised learning còn được tiếp tục chia nhỏ ra thành hai loại chính:

### Classification (Phân loại)

Một bài toán được gọi là classification nếu các label của input data được chia thành một số hữu hạn nhóm. Ví dụ: Gmail xác định xem một email có phải là spam hay không; các hãng tín dụng xác định xem một khách hàng có khả năng thanh toán nợ hay không. Ba ví dụ phía trên được chia vào loại này.

### Regression (Hồi quy)

Nếu label không được chia thành các nhóm mà là một giá trị thực cụ thể. Ví dụ: một căn nhà rộng  $x$  m<sup>2</sup> x  $y$  m<sup>2</sup>, có  $yy$  phòng ngủ và cách trung tâm thành phố  $z$  km sẽ có giá là bao nhiêu?

- Các thuật toán trong học có giám sát là

- Linear Regression: Thuộc bài toán hồi quy, dùng để dự đoán giá trị đầu ra liên tục dựa trên mối quan hệ tuyến tính giữa biến đầu vào (độc lập) và biến đầu ra (phụ thuộc).
- Logistic Regression: Thuộc bài toán phân loại, dùng để phân loại dữ liệu nhị phân hoặc nhiều lớp bằng cách tính xác suất của một đối tượng thuộc một trong các lớp.
- Decision Trees: Thuộc bài toán phân loại hoặc hồi quy, dùng cây phân nhánh để mô hình hóa các quyết định và hậu quả có thể xảy ra. Cây quyết định hoạt động tốt với cả dữ liệu phân loại và dữ liệu liên tục.
- Random Forest: Thuộc bài toán phân loại hoặc hồi quy, là tập hợp của nhiều cây quyết định, tạo ra mô hình mạnh mẽ hơn bằng cách kết hợp nhiều cây để tăng độ chính xác và tránh quá khớp (overfitting - hiểu đơn giản là mô hình hoạt động rất tốt trên dữ liệu huấn luyện nhưng lại hoạt động kém trên dữ liệu mới).
- Support Vector Machine - SVM: Thuộc bài toán phân loại hoặc hồi quy, Tìm một siêu phẳng (hyperplane) tốt nhất để phân chia dữ liệu thành các lớp khác nhau, tối đa hóa khoảng cách giữa các lớp.
- K-Nearest Neighbors - KNN: Thuộc bài toán phân loại hoặc hồi quy, dự đoán giá trị đầu ra dựa trên dữ liệu của những "người hàng xóm gần nhất" trong không gian tính toán. Mô hình so sánh đầu vào mới với các điểm gần nhất để phân loại hoặc hồi quy.

- Naive Bayes: Thuộc bài toán phân loại, mô hình xác suất dựa trên Định lý Bayes, giả định rằng các đặc trưng của dữ liệu đều độc lập với nhau. Thường được sử dụng cho các bài toán phân loại văn bản.
- Ưu điểm
- Độ chính xác và khả năng dự đoán cao: Các mô hình học có giám sát đặc biệt được tôn sùng vì khả năng đưa ra các dự đoán có độ chính xác cao, nhờ vào sự phụ thuộc của chúng vào các tập dữ liệu được gắn nhãn. Ví dụ, các thuật toán học có giám sát trong chăm sóc sức khỏe có thể dự đoán kết quả của bệnh nhân dựa trên hồ sơ bệnh án và kết quả điều trị trước đây.
  - Nâng cao hiệu quả thuật toán với Feature Learning: Học có giám sát không chỉ tập trung vào việc đạt được độ chính xác cao mà còn tăng cường hiệu quả của các thuật toán thông qua học tính năng.
  - Khả năng mở rộng trên nhiều ứng dụng khác nhau: Học có giám sát có khả năng mở rộng cao, giúp nó thích ứng với nhiều ứng dụng khác nhau trong nhiều ngành công nghiệp khác nhau.
  - Hiệu suất mạnh mẽ trong môi trường được kiểm soát: Học có giám sát phát triển mạnh trong môi trường được kiểm soát với dữ liệu đầu vào được xác định rõ ràng và kỳ vọng đầu ra.
  - Cải tiến liên tục thông qua phản hồi: Học có giám sát được thiết kế để cải thiện liên tục thông qua các cơ chế phản hồi. Khi mô hình tiếp xúc với nhiều dữ liệu được gắn nhãn hơn theo thời gian, nó sẽ tinh chỉnh các thuật toán của mình, giảm lỗi và tăng cường khả năng ra quyết định.
- Nhược điểm
- Sự phụ thuộc vào dữ liệu được gắn nhãn: Một nhược điểm lớn của học có giám sát là sự phụ thuộc mạnh mẽ vào dữ liệu được gắn nhãn, việc thu thập dữ liệu này có thể tốn kém và mất thời gian. Ví dụ, Trong việc thu thập các bản quét X-quang hoặc MRI được gắn nhãn chính xác trong hình ảnh y tế đòi hỏi sự tham gia đáng kể từ các bác sĩ chuyên khoa X-quang, hạn chế tốc độ và khả năng mở rộng của việc phát triển các công cụ chẩn đoán mới.
  - Các vấn đề tổng quát: Các mô hình học có giám sát có thể gặp khó khăn với việc khái quát hóa, đặc biệt là khi gặp dữ liệu khác biệt đáng kể so với tập huấn luyện.

- Chi phí tính toán cao: Các mô hình học có giám sát thường đòi hỏi các nguồn tài nguyên tính toán đáng kể, đặc biệt là trong giai đoạn đào tạo, khi lượng lớn dữ liệu được gắn nhãn được xử lý.
- Độ nhạy cảm với các tính năng không liên quan: Thuật toán học có giám sát gán tầm quan trọng không cần thiết cho các tính năng không liên quan đến nhiệm vụ, đặc biệt nếu các tính năng này nổi bật trong dữ liệu đào tạo
- Khó khăn với các vấn đề không chuẩn: Hạn chế này khiến việc áp dụng học có giám sát trở nên khó khăn trong các lĩnh vực đòi hỏi sự đổi mới hoặc sáng tạo hoặc trong đó các vấn đề không thể dễ dàng định lượng và dán nhãn.

### 1.4.2. Học không giám sát (Unsupervised Learning)

Học không giám sát, còn được gọi là học máy không giám sát, sử dụng các thuật toán học máy (ML) để phân tích và nhóm các tập dữ liệu không có nhãn. Các thuật toán này khám phá các mẫu ẩn hoặc nhóm dữ liệu mà không cần sự can thiệp của con người.

Khả năng phát hiện điểm tương đồng và khác biệt trong thông tin của học không giám sát khiến nó trở thành giải pháp lý tưởng cho phân tích dữ liệu khám phá, chiến lược bán chéo, phân khúc khách hàng và nhận dạng hình ảnh.

Các mô hình học không giám sát được sử dụng cho ba nhiệm vụ chính: phân cụm, liên kết và giảm chiều:

- Phân cụm là một kỹ thuật khai thác dữ liệu để nhóm dữ liệu chưa gắn nhãn dựa trên điểm giống hoặc khác nhau của chúng. Ví dụ, thuật toán phân cụm K-means gán các điểm dữ liệu tương tự vào các nhóm, trong đó giá trị K biểu thị kích thước của nhóm và độ chi tiết. Kỹ thuật này hữu ích cho phân khúc thị trường, nén hình ảnh, v.v.
- Association là một loại phương pháp học không giám sát khác sử dụng các quy tắc khác nhau để tìm mối quan hệ giữa các biến trong một tập dữ liệu nhất định. Các phương pháp này thường được sử dụng để phân tích giỏ hàng và công cụ đề xuất, theo hướng khuyến nghị "Khách hàng đã mua mặt hàng này cũng đã mua".
- Giảm chiều là một kỹ thuật học được sử dụng khi số lượng các tính năng (hoặc chiều) trong một tập dữ liệu nhất định quá cao. Nó giảm số lượng dữ

liệu đầu vào xuống một kích thước có thể quản lý được trong khi vẫn bảo toàn tính toàn vẹn của dữ liệu. Kỹ thuật này thường được sử dụng trong giai đoạn tiền xử lý dữ liệu, chẳng hạn như khi bộ mã hóa tự động loại bỏ nhiễu khỏi dữ liệu trực quan để cải thiện chất lượng hình ảnh.

- Ưu điểm:

- Khám phá các mô hình và cấu trúc tự nhiên: Các thuật toán học không giám sát rất giỏi trong việc xác định các cấu trúc và mẫu ẩn trong dữ liệu mà không dễ thấy ngay lập tức, mang lại lợi thế to lớn trong nhiều lĩnh vực
- Xử lý dữ liệu chưa được gắn nhãn: Một trong những điểm mạnh đáng kể của học không giám sát là khả năng làm việc với dữ liệu không có nhãn, chiếm phần lớn dữ liệu có sẵn trong thế giới thực
- Tính linh hoạt trong việc điều chỉnh mô hình: Học không giám sát có tính linh hoạt đáng kể trong việc thích ứng với dữ liệu mới mà không cần đào tạo lại từ đầu.
- Hiệu quả trong Phân tích dữ liệu thăm dò: Học không giám sát đặc biệt hiệu quả đối với phân tích dữ liệu khám phá, trong đó mục tiêu là hiểu và tóm tắt các đặc điểm chính của một tập dữ liệu mà không có bất kỳ giả định nào trước đó.
- Khả năng mở rộng cho các tập dữ liệu lớn: Học không giám sát có khả năng mở rộng cao, khiến nó đặc biệt phù hợp để xử lý các tập dữ liệu lớn đang ngày càng trở nên phổ biến trong thời đại dữ liệu lớn

- Nhược điểm

- Thiếu chính xác trong việc giải thích kết quả: Một nhược điểm đáng kể của học không giám sát là thiếu độ chính xác trong việc diễn giải kết quả do không có dữ liệu được gắn nhãn
- Dễ bị ảnh hưởng bởi tính năng mở rộng và nhiễu: Các thuật toán học không giám sát đặc biệt nhạy cảm với quy mô của các tính năng và sự hiện diện của nhiễu trong dữ liệu.
- Phụ thuộc nhiều vào chất lượng dữ liệu: Hiệu quả của học không giám sát phụ thuộc rất nhiều vào chất lượng của dữ liệu đầu vào.
- Độ phức tạp trong việc lựa chọn thuật toán và điều chỉnh tham số: Học không giám sát liên quan đến quá trình ra quyết định phức tạp liên quan đến việc lựa chọn các thuật toán phù hợp và điều chỉnh các tham số

- Khó khăn trong việc xác thực mô hình: Xác thực hiệu suất của các mô hình học không giám sát đặt ra một thách thức độc đáo, vì không có nhãn hoặc chuẩn mực được xác định trước nào mà đầu ra của mô hình có thể được đo lường trực tiếp

### 1.4.3. Semi-Supervised Learning (Học bán giám sát)

Học bán giám sát là một loại học máy nằm giữa học có giám sát và học không giám sát. Đây là phương pháp sử dụng một lượng nhỏ dữ liệu được gắn nhãn và một lượng lớn dữ liệu không được gắn nhãn để đào tạo mô hình. Mục tiêu của học bán giám sát là học một hàm có thể dự đoán chính xác biến đầu ra dựa trên các biến đầu vào, tương tự như học có giám sát. Tuy nhiên, không giống như học có giám sát, thuật toán được đào tạo trên một tập dữ liệu chứa cả dữ liệu được gắn nhãn và không được gắn nhãn.

Học bán giám sát đặc biệt hữu ích khi có một lượng lớn dữ liệu chưa được gắn nhãn, nhưng việc gắn nhãn cho tất cả lại quá tốn kém hoặc khó khăn.

Ví dụ về học bán giám sát

- Phân loại văn bản : Trong phân loại văn bản, mục tiêu là phân loại một văn bản nhất định thành một hoặc nhiều danh mục được xác định trước. Học bán giám sát có thể được sử dụng để đào tạo mô hình phân loại văn bản bằng cách sử dụng một lượng nhỏ dữ liệu được gắn nhãn và một lượng lớn dữ liệu văn bản không được gắn nhãn.
- Phân loại hình ảnh : Trong phân loại hình ảnh, mục tiêu là phân loại một hình ảnh nhất định thành một hoặc nhiều danh mục được xác định trước. Học bán giám sát có thể được sử dụng để đào tạo mô hình phân loại hình ảnh bằng cách sử dụng một lượng nhỏ dữ liệu được gắn nhãn và một lượng lớn dữ liệu hình ảnh không được gắn nhãn.
- Phát hiện bất thường : Trong phát hiện bất thường, mục tiêu là phát hiện các mẫu hoặc quan sát bất thường hoặc khác với chuẩn mực

Ứng dụng của học bán giám sát

- Phân tích giọng nói: Vì việc dán nhãn cho các tệp âm thanh là một nhiệm vụ rất chuyên sâu nên học bán giám sát là một phương pháp rất tự nhiên để giải quyết vấn đề này.

- Phân loại nội dung Internet: Việc gắn nhãn cho từng trang web là một quá trình không thực tế và không khả thi và do đó sử dụng các thuật toán học bán giám sát. Ngay cả thuật toán tìm kiếm của Google cũng sử dụng một biến thể của học bán giám sát để xếp hạng mức độ liên quan của một trang web cho một truy vấn nhất định.
- Phân loại trình tự protein: Vì các sợi DNA thường có kích thước rất lớn nên sự phát triển của phương pháp học bán giám sát đã trở nên cấp thiết trong lĩnh vực này.

#### Ưu điểm

- Giảm chi phí nhãn: Học bán giám sát làm giảm đáng kể dữ liệu được gắn nhãn cần thiết, cắt giảm chi phí liên quan đến việc gắn nhãn thủ công
- Cải thiện hiệu suất: Bằng cách sử dụng cả dữ liệu có nhãn và không nhãn, các mô hình có thể đạt hiệu suất tốt hơn so với các mô hình chỉ được đào tạo trên dữ liệu có nhãn.
- Khả năng mở rộng: Học bán giám sát cho phép các mô hình chỉ được đào tạo trên dữ liệu cá nhân

#### Thách thức và hạn chế:

- Chất lượng dữ liệu chưa được gắn nhãn: Nếu dữ liệu không được gắn nhãn không đại diện cho phân phối dữ liệu thực hoặc chứa nhiều nhiễu, hiệu suất của mô hình có thể giảm.
- Mô hình tự tin: Việc dựa vào các nhãn giả từ các dự đoán của mô hình đòi hỏi phải có sự tin tưởng vào mô hình ban đầu. Nếu mô hình được đào tạo kém, các nhãn giả có thể không chính xác
- Độ phức tạp tính toán: Một số phương pháp bán giám sát, đặc biệt là phương pháp dựa trên đồ thị hoặc kỹ thuật dựa trên học sâu, có thể tốn kém và mất tính toán.

#### **1.4.4. Reinforcement Learning (Học tăng cường)**

Học tăng cường (Reinforcement Learning) là một nhánh của học máy tập trung vào cách các tác nhân có thể học cách đưa ra quyết định thông qua thử nghiệm và sai sót để tối đa hóa phần thưởng tích lũy. Reinforcement Learning cho phép máy học bằng cách tương tác với môi trường và nhận phản hồi dựa trên hành động của chúng. Phản hồi này ở dạng phần thưởng hoặc hình phạt .

## Các thành phần Reinforcement Learning:

- Chính sách : Chiến lược mà tác nhân sử dụng để xác định hành động tiếp theo dựa trên trạng thái hiện tại.
- Chức năng phần thưởng : Chức năng cung cấp phản hồi về các hành động đã thực hiện, hướng dẫn tác nhân đạt được mục tiêu.
- Hàm giá trị : Ước tính phần thưởng tích lũy trong tương lai mà tác nhân sẽ nhận được từ một trạng thái nhất định.
- Mô hình môi trường : Biểu diễn môi trường dự đoán trạng thái và phần thưởng trong tương lai, hỗ trợ cho việc lập kế hoạch.

## Các loại tăng cường trong Reinforcement Learning

### 1. Tăng cường tích cực

Sự củng cố tích cực được định nghĩa là khi một sự kiện xảy ra do một hành vi cụ thể, làm tăng cường độ và tần suất của hành vi. Nói cách khác, nó có tác động tích cực đến hành vi.

- Ưu điểm : Tối đa hóa hiệu suất, giúp duy trì sự thay đổi theo thời gian.
- Nhược điểm : Sử dụng quá mức có thể dẫn đến tình trạng dư thừa làm giảm hiệu quả.

### 2. Tăng cường tiêu cực

Củng cố tiêu cực được định nghĩa là việc củng cố hành vi vì một điều kiện tiêu cực bị ngăn chặn hoặc tránh đi.

- Ưu điểm : Tăng tần suất hành vi, đảm bảo tiêu chuẩn hiệu suất tối thiểu.
- Nhược điểm : Nó chỉ có thể khuyến khích hành động vừa đủ để tránh bị phạt.

## Ứng dụng của Học tăng cường

- Robot: Reinforcement Learning được sử dụng để tự động hóa các nhiệm vụ trong môi trường có cấu trúc như sản xuất, nơi robot học cách tối ưu hóa chuyển động và cải thiện hiệu quả.

- Chơi trò chơi: Các thuật toán Reinforcement Learning nâng cao đã được sử dụng để phát triển các chiến lược cho các trò chơi phức tạp như cờ vua, cờ vây và trò chơi điện tử, vượt trội hơn người chơi trong nhiều trường hợp.
- Kiểm soát công nghiệp: Reinforcement Learning giúp điều chỉnh và tối ưu hóa thời gian thực các hoạt động công nghiệp, chẳng hạn như quy trình lọc dầu trong ngành dầu khí.
- Hệ thống đào tạo cá nhân: Reinforcement Learning cho phép tùy chỉnh nội dung hướng dẫn dựa trên mô hình học tập của từng cá nhân, cải thiện sự tương tác và hiệu quả.

#### Ưu điểm

- Giải quyết các vấn đề phức tạp: Reinforcement Learning có khả năng giải quyết các vấn đề cực kỳ phức tạp mà các kỹ thuật thông thường không thể giải quyết được.
- Sửa lỗi: Mô hình liên tục học hỏi từ môi trường xung quanh và có thể sửa các lỗi xảy ra trong quá trình đào tạo.
- Tương tác trực tiếp với môi trường: Các tác nhân Reinforcement Learning học hỏi từ các tương tác thời gian thực với môi trường của chúng, cho phép học tập thích ứng.
- Xử lý môi trường không xác định: Reinforcement Learning có hiệu quả trong môi trường mà kết quả không chắc chắn hoặc thay đổi theo thời gian, khiến nó trở nên cực kỳ hữu ích cho các ứng dụng thực tế.

#### Nhược điểm

- Không phù hợp với các vấn đề đơn giản : Reinforcement Learning thường quá mức cần thiết đối với các nhiệm vụ đơn giản trong khi các thuật toán đơn giản hơn sẽ hiệu quả hơn.
- Yêu cầu tính toán cao : Việc đào tạo các mô hình Reinforcement Learning đòi hỏi một lượng lớn dữ liệu và sức mạnh tính toán, khiến việc này tốn nhiều tài nguyên.
- Phụ thuộc vào chức năng khen thưởng : Hiệu quả của Reinforcement Learning phụ thuộc rất nhiều vào thiết kế chức năng khen thưởng. Phần



thường được thiết kế kém có thể dẫn đến hành vi không tối ưu hoặc không mong muốn.

- Khó khăn trong việc gỡ lỗi và giải thích : Việc hiểu lý do tại sao một tác nhân Reinforcement Learning đưa ra một số quyết định nhất định có thể là một thách thức, khiến việc gỡ lỗi và khắc phục sự cố trở nên phức tạp

### **1.5. Ứng dụng của học máy trong y tế**

- Chẩn đoán hình ảnh y tế: ML giúp phân tích hình ảnh như X-quang, CT, MRI để phát hiện sớm các bất thường như khối u, tổn thương mô, hỗ trợ bác sĩ trong việc chẩn đoán chính xác hơn.
- Phát hiện bệnh sớm: ML có thể xử lý dữ liệu lớn từ hồ sơ bệnh án, xét nghiệm để nhận diện các dấu hiệu ban đầu của bệnh, giúp can thiệp kịp thời và nâng cao hiệu quả điều trị.
- Dự đoán nguy cơ bệnh tật: Bằng cách phân tích thông tin di truyền, lối sống và tiền sử bệnh, ML hỗ trợ dự đoán khả năng mắc bệnh trong tương lai, từ đó đề xuất các biện pháp phòng ngừa phù hợp.
- Cá nhân hóa phác đồ điều trị: ML giúp xây dựng kế hoạch điều trị riêng biệt cho từng bệnh nhân dựa trên dữ liệu cá nhân, tối ưu hóa hiệu quả và giảm thiểu tác dụng phụ.
- Phát triển thuốc mới: ML hỗ trợ trong việc phân tích dữ liệu sinh học để tìm ra các hợp chất tiềm năng, rút ngắn thời gian và chi phí trong quá trình nghiên cứu và phát triển thuốc.
- Trợ lý ảo và chatbot y tế: Ứng dụng ML trong các trợ lý ảo giúp cung cấp thông tin y tế, nhắc nhở uống thuốc, theo dõi triệu chứng và hỗ trợ bệnh nhân trong việc quản lý sức khỏe hàng ngày.
- Theo dõi bệnh nhân từ xa: ML kết hợp với thiết bị đeo thông minh để giám sát liên tục các chỉ số sức khỏe như nhịp tim, huyết áp, giúp phát hiện sớm các bất thường và cảnh báo kịp thời.
- Hỗ trợ phẫu thuật bằng robot: ML được tích hợp trong các hệ thống robot phẫu thuật, giúp nâng cao độ chính xác, giảm thiểu rủi ro và hỗ trợ bác sĩ trong các ca phẫu thuật phức tạp.
- Quản lý hồ sơ y tế điện tử: ML giúp tự động hóa việc nhập liệu, phân loại và phân tích hồ sơ bệnh án, nâng cao hiệu quả quản lý và truy xuất thông tin y tế.

- Phân tích dữ liệu y tế lớn: ML hỗ trợ xử lý và phân tích khối lượng lớn dữ liệu y tế để phát hiện xu hướng, hỗ trợ ra quyết định và cải thiện chất lượng dịch vụ chăm sóc sức khỏe.

## Chương 2: Các kỹ thuật phân lớp

### 2.1. K-nearest neighbor

#### 2.1.1 . Giới thiệu

K-Nearest Neighbors (KNN) là một thuật toán học máy có giám sát thường được sử dụng để phân loại nhưng cũng có thể được sử dụng cho các nhiệm vụ hồi quy. Nó hoạt động bằng cách tìm "k" các điểm dữ liệu gần nhất (hàng xóm) với một đầu vào nhất định và đưa ra dự đoán dựa trên lớp đa số (để phân loại) hoặc giá trị trung bình (đối với hồi quy). Vì KNN không đưa ra giả định về phân phối dữ liệu cơ bản, nó làm cho nó trở thành một phương pháp học tập dựa trên thực thể và phi tham số.

Nó được sử dụng rộng rãi trong các tình huống thực tế vì nó không tham số, nghĩa là nó không đưa ra bất kỳ giả định cơ bản nào về phân phối dữ liệu (trái ngược với các thuật toán khác như GMM, giả định phân phối Gaussian của dữ liệu đã cho). Chúng ta được cung cấp một số dữ liệu trước đó (còn gọi là dữ liệu đào tạo), phân loại tọa độ thành các nhóm được xác định bởi một thuộc tính.

#### 2.1.2 . Nguyên lý hoạt động

- Cách thức hoạt động

Thuật toán K-Nearest Neighbors (KNN) hoạt động theo nguyên tắc tương tự, trong đó nó dự đoán nhãn hoặc giá trị của một điểm dữ liệu mới bằng cách xem xét nhãn hoặc giá trị của K hàng xóm gần nhất của nó trong tập dữ liệu đào tạo.

##### **Bước 1: Chọn giá trị tối ưu của K**

- K đại diện cho số lượng hàng xóm gần nhất cần được xem xét trong khi đưa ra dự đoán.

##### **Bước 2: Tính khoảng cách**

- Để đo lường sự tương đồng giữa các điểm dữ liệu mục tiêu và đào tạo, khoảng cách Euclid được sử dụng. Khoảng cách được tính giữa các điểm dữ liệu trong tập dữ liệu và điểm đích.

##### **Bước 3: Tìm hàng xóm gần nhất**

- K điểm dữ liệu có khoảng cách nhỏ nhất đến điểm mục tiêu là hàng xóm gần nhất.

##### **Bước 4: Bỏ phiếu cho phân loại hoặc lấy trung bình cho hồi quy**

- Khi bạn muốn phân loại một điểm dữ liệu thành một danh mục như spam hoặc không phải spam, thuật toán KNN sẽ xem xét K điểm gần nhất trong

tập dữ liệu. Những điểm gần nhất này được gọi là hàng xóm. Sau đó, thuật toán xem xét các hàng xóm thuộc về danh mục nào và chọn loại xuất hiện nhiều nhất. Đây được gọi là bỏ phiếu đa số.

- Trong hồi quy, thuật toán vẫn tìm kiếm K điểm gần nhất. Nhưng thay vì bỏ phiếu cho một lớp trong phân loại, nó lấy giá trị trung bình của những người hàng xóm K đó. Giá trị trung bình này là giá trị dự đoán cho điểm mới cho thuật toán.

- Các loại khoảng cách

- **Khoảng cách Euclidean**

Đây không gì khác ngoài khoảng cách Descartes giữa hai điểm nằm trong mặt phẳng/siêu mặt phẳng. Khoảng cách Euclid cũng có thể được hình dung là độ dài của đường thẳng nối hai điểm đang xét. Hệ mét này giúp chúng ta tính toán độ dịch chuyển rỗng giữa hai trạng thái của một vật thể.

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^d (x_j - X_{ij})^2}$$

- **Khoảng cách Manhattan**

Khoảng cách Manhattan thường được sử dụng khi chúng ta quan tâm đến tổng quãng đường mà vật thể di chuyển thay vì độ dịch chuyển. Khoảng cách này được tính bằng cách cộng tổng chênh lệch tuyệt đối giữa các tọa độ của các điểm trong n chiều.

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

- **Khoảng cách Minkowski**

Ta có thể nói rằng khoảng cách Euclid cũng như khoảng cách Manhattan là những trường hợp đặc biệt của khoảng cách Minkowski.

$$d(x, y) = \left( \sum_{i=1}^n (x_i - y_i)^p \right)^{\frac{1}{p}}$$

### 2.1.3 . Ưu điểm

- Dễ triển khai vì độ phức tạp của thuật toán không quá cao.
- Dễ dàng thích nghi – Theo cách hoạt động của thuật toán KNN, nó sẽ lưu trữ tất cả dữ liệu trong bộ nhớ và do đó bất cứ khi nào một ví dụ hoặc điểm dữ liệu mới được thêm vào thì thuật toán sẽ tự điều chỉnh theo ví dụ mới đó và cũng đóng góp vào các dự đoán trong tương lai.

- Một số siêu tham số – Các tham số duy nhất cần thiết trong quá trình đào tạo thuật toán KNN là giá trị  $k$  và lựa chọn số liệu khoảng cách mà chúng ta muốn chọn từ số liệu đánh giá của mình.

#### 2.1.4 . Nhược điểm

- **Không mở rộng quy mô:** Như chúng ta đã nghe về điều này, thuật toán KNN cũng được coi là Thuật toán lười biếng. Ý nghĩa chính của thuật ngữ này là nó đòi hỏi rất nhiều sức mạnh tính toán cũng như lưu trữ dữ liệu. Điều này làm cho thuật toán này vừa tốn thời gian vừa cạn kiệt tài nguyên.

- **Lời nguyên của đa chiều:** Có một thuật ngữ được gọi là hiện tượng đạt đỉnh, theo đó thuật toán KNN bị ảnh hưởng bởi lời nguyên của đa chiều, ngụ ý rằng thuật toán gặp khó khăn trong việc phân loại các điểm dữ liệu một cách chính xác khi đa chiều quá cao.

- **Dễ bị quá khớp:** Vì thuật toán bị ảnh hưởng do lời nguyên của chiều nên nó cũng dễ bị vấn đề quá khớp. Do đó, nói chung, các kỹ thuật lựa chọn đặc điểm cũng như giảm chiều được áp dụng để giải quyết vấn đề này.

#### 2.1.5 . Ứng dụng thuật toán

**Phân loại bệnh lý trong y học:** KNN được sử dụng trong các hệ thống hỗ trợ chẩn đoán y học để phân loại bệnh dựa trên dữ liệu của bệnh nhân, ví dụ như chẩn đoán bệnh về máu dựa trên chỉ số xét nghiệm, chẩn đoán ung thư hoặc các bệnh khác.

**Phân loại và nhận dạng hình ảnh:** KNN có thể được sử dụng để phân loại và nhận dạng hình ảnh dựa trên các đặc điểm hình ảnh tương tự.

**Phát hiện thư rác (Spam Detection):** KNN có thể phân loại email là spam hay không spam bằng cách so sánh đặc trưng của email mới với các email đã được gắn nhãn trước đó.

**Hệ thống gợi ý sản phẩm:** Trong thương mại điện tử, KNN giúp gợi ý các sản phẩm tương tự mà người dùng có thể quan tâm dựa trên lịch sử mua hàng hoặc xem sản phẩm.

**Phân tích và phát hiện gian lận:** KNN có thể xác định các giao dịch gian lận trong dữ liệu tài chính bằng cách phát hiện các giao dịch khác biệt so với các giao dịch trước đó của người dùng.

## 2.2. Naïve bayes

### 2.2.1. Giới thiệu

Naive Bayes là một thuật toán phân loại sử dụng xác suất để dự đoán một điểm dữ liệu thuộc về loại nào, giả sử rằng tất cả các tính năng đều không liên quan. Bài viết này sẽ cung cấp cho bạn tổng quan cũng như cách sử dụng và triển khai Naive Bayes nâng cao hơn trong học máy.

Ý tưởng chính đằng sau bộ phân loại Naive Bayes là sử dụng Định lý Bayes để phân loại dữ liệu dựa trên xác suất của các lớp khác nhau dựa trên các đặc điểm của dữ liệu. Nó được sử dụng chủ yếu trong phân loại văn bản có chiều cao

- Phân loại Naive Bayes là một phân loại xác suất đơn giản và có rất ít tham số được sử dụng để xây dựng các mô hình ML có thể dự đoán nhanh hơn các thuật toán phân loại khác.
- Đây là một bộ phân loại xác suất vì nó giả định rằng một tính năng trong mô hình độc lập với sự tồn tại của một tính năng khác. Nói cách khác, mỗi tính năng góp phần vào dự đoán mà không có mối quan hệ nào với nhau.

### 2.2.2. Nguyên lý hoạt động

+ Định lý Bayes

$$P(h | D) = \frac{P(D | h)P(h)}{P(D)}$$

$P(h)$ : xác suất trước (tiên nghiệm) của giả thiết phân lớp  $h$ .

$P(D)$ : xác suất trước (tiên nghiệm) của việc quan sát được dữ liệu  $D$ .

$P(D | h)$ : xác suất có điều kiện của việc quan sát được dữ liệu  $D$ , nếu biết giả thiết phân lớp  $h$  là đúng.

$P(h | D)$ : xác suất có điều kiện của giả thiết phân lớp  $h$  là đúng, nếu quan sát được dữ liệu  $D$ .

Với  $H$  là một tập các giả thiết phân lớp, hệ thống học sẽ tìm giả thiết có thể xảy ra nhất để  $h \in H$  đối với các dữ liệu  $D$  quan sát được. Giả thiết  $h$  này được gọi là giả thiết có xác suất hậu nghiệm cực đại (Maximum a posteriori – MAP).

$$h_{\text{MAP}} = \underset{h \in H}{\operatorname{argmax}} P(h | D) = \underset{h \in H}{\operatorname{argmax}} \frac{P(D | h).P(h)}{P(D)}$$

Giả thiết có thể nhất  $h_{\text{MAP}} = h_1$  nếu  $P(h_1 | D) \geq P(h_2 | D)$ ; ngược lại,  $h_{\text{MAP}} = h_2$ . Do  $P(D)$  là như nhau đối với cả hai giả thiết  $h_1$  và  $h_2$ , nên có thể bỏ qua

đại lượng  $P(D)$ . Vì vậy, chỉ cần tính hai biểu thức:  $P(D | h_1).P(h_1)$  và  $P(D | h_2).P(h_2)$ , là ta có thể đưa ra quyết định phân lớp.

+ Phân lớp Navie Bayes (Navie Bayes Classifical)

Bài toán phân lớp (classification problem): một tập dữ liệu huấn luyện  $D_{train}$ , trong đó mỗi mẫu  $x$  được biểu diễn là một vector  $n$  chiều  $(x_1, x_2, \dots, x_n)$  và một tập xác định các nhãn lớp  $C = \{c_1, c_2, \dots, c_m\}$ . Với mẫu mới  $z$ , thì  $z$  sẽ được phân vào lớp nào?

Mục tiêu: xác định phân lớp phù hợp nhất đối với  $z$ .

$$c_{MAP} = \underset{c_i \in C}{\operatorname{argmax}} P(z_1, z_2, \dots, z_n | c_i).P(c_i)$$

Giả sử trong phương pháp phân lớp Naïve Bayes, các thuộc tính là độc lập có điều kiện đối với các lớp:

$$P(z_1, z_2, \dots, z_n | c_i) = \prod_{j=1}^n P(z_j | c_i)$$

Phân lớp Naïve Bayes tìm phân lớp có thể nhất đối với  $z$

$$c_{NB} = \underset{c_i \in C}{\operatorname{argmax}} P(c_i) \cdot \prod_{j=1}^n P(z_j | c_i)$$

Giai đoạn huấn luyện (training phase)

- Đối với mỗi phân lớp có thể (mỗi nhãn lớp)  $c_i \in C$  tính giá trị xác suất trước  $P(c_i)$ .
- Đối với mỗi giá trị thuộc tính  $x_j$ , tính giá trị xác suất xảy ra của giá trị thuộc tính đó đối với một phân lớp  $c_i$ :  $P(x_j | c_i)$ .
- Giai đoạn phân lớp (classification phase): cần gán nhãn cho một mẫu mới, thực hiện:

+ Đối với mỗi phân lớp  $c_i \in C$ , tính giá trị biểu thức:

$$P(c_i) \cdot \prod_{j=1}^n P(x_j | c_i),$$

+ Xác định phân lớp cha của  $z$  là lớp có thể nhất  $c^*$ :

$$c^* = \underset{c_i \in C}{\operatorname{argmax}} P(c_i) \cdot \prod_{j=1}^n P(x_j | c_i)$$

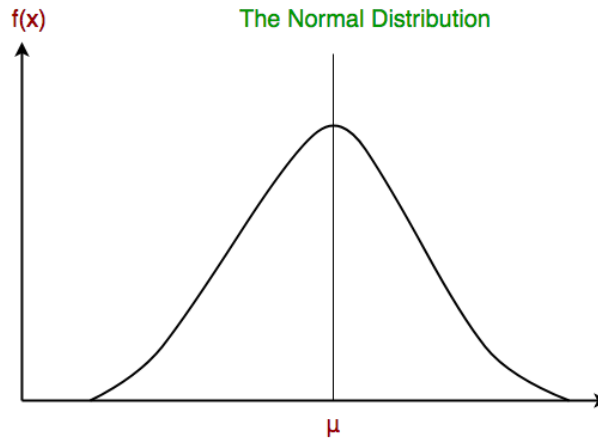
Nghĩa là  $z$  được dự đoán thuộc lớp  $c_i$  nếu và chỉ nếu

$$P(c_i | z) > P(c_j | z) \text{ với } 1 \leq j \leq m, j \neq i.$$

### 2.2.3. Các loại Naive Bayes

- Gaussian Naive Bayes:

Trong Gaussian Naive Bayes, các giá trị liên tục liên quan đến mỗi tính năng được cho là phân phối theo phân phối Gaussian. Phân phối Gaussian cũng được gọi là phân phối chuẩn. Khi được vẽ, nó cung cấp một đường cong hình chuông đối xứng quanh giá trị trung bình của các giá trị tính năng như được hiển thị bên dưới:



Hình 2.1: Bảng mô tả phân phối chuẩn

Bảng cập nhật các xác suất trước đó cho tính năng triển vọng như sau:

Xác suất của các tính năng được coi là theo phân phối chuẩn Gauss, do đó, xác suất có điều kiện được đưa ra bởi:

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

- Multinomial Naive Bayes:

Các vector đặc trưng biểu diễn tần suất mà một số sự kiện nhất định được tạo ra bởi phân phối đa thức. Đây là mô hình sự kiện thường được sử dụng để phân loại tài liệu.

- Bernoulli Naive Bayes:



Trong mô hình sự kiện Bernoulli đa biến, các đặc điểm là các boolean độc lập (biến nhị phân) mô tả các đầu vào. Giống như mô hình đa thức, mô hình này phổ biến cho các tác vụ phân loại tài liệu, trong đó các đặc điểm xuất hiện thuật ngữ nhị phân (tức là một từ xuất hiện trong tài liệu hay không) được sử dụng thay vì tần suất thuật ngữ (tức là tần suất của một từ trong tài liệu).

#### **2.2.4. Ưu điểm**

- Dễ triển khai và hiệu quả về mặt tính toán.
- Có hiệu quả trong trường hợp có nhiều tính năng.
- Hoạt động tốt ngay cả khi dữ liệu đào tạo hạn chế.
- Nó hoạt động tốt khi có các tính năng phân loại.
- Đối với các tính năng số, dữ liệu được cho là xuất phát từ các phân phối chuẩn

#### **2.2.5. Nhược điểm**

- Giả sử các tính năng là độc lập, điều này không phải lúc nào cũng đúng trong dữ liệu thực tế.
- Có thể bị ảnh hưởng bởi những thuộc tính không liên quan.
- Có thể gán xác suất bằng 0 cho các sự kiện không nhìn thấy được, dẫn đến khả năng khái quát kém.

#### **2.2.6. Ứng dụng thuật toán**

- Chẩn đoán y khoa: Giúp dự đoán khả năng mắc bệnh dựa trên các triệu chứng.
- Học từ dữ liệu xét nghiệm máu của nhiều bệnh nhân để xác định xác suất mắc các bệnh như: Thiếu máu (anemia), bạch cầu cấp và mãn tính (leukemia), đa hồng cầu (polycythemia vera), rối loạn đông máu.
- Phân loại bệnh máu tự động: Trong các bệnh viện có nhiều dữ liệu xét nghiệm máu, Naive Bayes có thể giúp tự động phân loại bệnh nhân vào các nhóm nguy cơ để theo dõi và điều trị.
- Phân tích tủy xương và dữ liệu tế bào học: Khi kết hợp với dữ liệu ảnh và dữ liệu phân tích hình thái tế bào máu, thuật toán có thể hỗ trợ bác sĩ trong chẩn đoán chính xác hơn.

### 2.3. Hiệu năng của mô hình

Phần này mô tả các biện pháp hiệu suất được sử dụng trong tài liệu tham khảo. Các chỉ số hiệu suất, bao gồm độ chính xác, độ chính xác, khả năng thu hồi và điểm F1, được sử dụng rộng rãi trong chẩn đoán bệnh. Ví dụ, ung thư phổi có thể được phân loại là dương tính thực sự ( $TP$ ) hoặc âm tính thật ( $TN$ ) nếu cá nhân được chẩn đoán đúng, trong khi nó có thể được phân loại thành dương tính giả ( $FP$ ) hoặc âm tính giả ( $FN$ ) nếu chẩn đoán sai. Các số liệu được sử dụng rộng rãi nhất được mô tả dưới đây.

Ma trận nhầm lẫn (Confusion Matrix)

- Ma trận  $2 \times 2$  (với bài toán nhị phân) hoặc ma trận  $k \times k$  (với nhiều lớp bệnh) giúp đếm số lượng:
  - True Positive (TP): Ca mắc bệnh thực sự và mô hình dự đoán đúng.
  - True Negative (TN): Ca không mắc bệnh và mô hình dự đoán đúng.
  - False Positive (FP): Ca không mắc bệnh nhưng mô hình dự đoán nhầm là bệnh.
  - False Negative (FN): Ca mắc bệnh nhưng mô hình dự đoán nhầm là không bệnh.
- Với bệnh lý về máu, False Negative rất nguy hiểm (bỏ sót ca bệnh), trong khi False Positive chỉ gây tốn kém thêm xét nghiệm xác nhận.

Độ chính xác (Acc) : Độ chính xác biểu thị tổng số trường hợp nhận dạng đúng trong số tất cả các trường hợp. Độ chính xác có thể được tính bằng các công thức sau:

$$ACC = \frac{T_p + T_N}{T_p + T_N + F_p + F_N}$$

Đo lường tổng số trường hợp được dự đoán đúng trên tổng số mẫu

Độ chính xác dương tính ( $Pn$ ): Độ chính xác được đo bằng tỷ lệ giữa dự đoán chính xác và tổng số quan sát tích cực mong đợi.

$$P_n = \frac{T_p}{T_p + F_p}$$

Tỷ lệ trong số các mẫu dự đoán “dương tính” thì có bao nhiêu thật sự dương.

Nhớ lại ( $Rc$ ): Tỷ lệ kết quả có liên quan tổng thể mà thuật toán nhận dạng đúng được gọi là thu hồi.

$$\frac{T_p}{T_n + F_p}$$

Tỷ lệ trong số các ca bệnh thật sự thì có bao nhiêu được mô hình phát hiện (quan trọng để không bỏ sót ca nguy hiểm).

Độ nhạy ( $Sn$ ) : Độ nhạy chỉ biểu thị biện pháp dương tính thực sự khi xét đến tổng số trường hợp và có thể được đo như sau:

$$S_n = \frac{T_p}{T_p + F_N}$$

Độ đặc hiệu ( $Sp$ ): Nó xác định có bao nhiêu kết quả âm tính thực sự được xác định chính xác và tính toán như sau:

$$S_p = \frac{T_N}{T_N + F_P}$$

Tỷ lệ trong số các ca không bệnh thì có bao nhiêu được mô hình nhận đúng (giúp giảm cảnh báo sai).

Đo lường F: Điểm F1 là giá trị trung bình của độ chính xác và độ thu hồi theo cách hài hòa. Điểm F cao nhất là 1, biểu thị độ chính xác và độ thu hồi hoàn hảo.

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Là trung bình điều hòa giữa Precision và Recall; giá trị tối đa là 1 (khi Precision = Recall = 1).

Diện tích dưới đường cong (AUC): Diện tích dưới đường cong biểu diễn hành vi của mô hình trong các tình huống khác nhau. AUC có thể được tính như sau:

$$AUC = \frac{\sum R_i (I_p) - I_p((I_p + 1) / 2)}{I_p + I_n}$$

Ở đâu  $I_p$  Và  $I_n$  biểu thị các mẫu dữ liệu tích cực và tiêu cực và  $R_i$  là xếp hạng của mẫu dương

Giá trị dự đoán âm tính (NPV):

- Ý nghĩa: Tỷ lệ những trường hợp được mô hình dự đoán là âm tính (không bệnh) mà thực sự là âm tính (không bệnh).
- Công thức:  $TN / (TN + FN)$
- Ví dụ tính toán:  $NPV = 850 / (850 + 20) \approx 0.977$

Trong số những ca được mô hình dự đoán là không bệnh, có đến 97.7% thực sự không bệnh.

## **Chương 3: Xây dựng ứng dụng**

### **3.1. Lợi ích học máy với đề tài**

- Phát hiện sớm và chính xác hơn
  - Mô hình ML có thể khám phá mối tương quan phức tạp giữa nhiều chỉ số (Hb, HCT, MCV, RDW, WBC...) mà bác sĩ có thể chưa quan sát hết.
  - Kết hợp thông tin đa chiều giúp gợi ý nguy cơ bệnh lý ở giai đoạn rất sớm (ví dụ: thiếu máu nhẹ giai đoạn I trước khi xuất hiện triệu chứng lâm sàng).
- Hỗ trợ ra quyết định lâm sàng nhanh chóng
  - Bác sĩ thường phải xem hàng chục chỉ số xét nghiệm. Hệ thống học máy cho phép đưa ra thông báo tự động nếu kết quả xét nghiệm thể hiện mô hình “nguy cơ thiếu máu nặng” hoặc “tăng bạch cầu bất thường”.
  - Giảm áp lực cho bác sĩ trong giờ cao điểm, tăng tốc độ ra phác đồ điều trị.
- Giảm chi phí xét nghiệm phụ trợ không cần thiết
  - Qua xác suất dự đoán, mô hình có thể gợi ý chỉ những ca “ngghi ngờ” mới cần làm thêm xét nghiệm tủy xương, xét nghiệm sắt huyết thanh, ferritin.
  - Tránh tình trạng “xét nghiệm lan man” làm tốn kém cho bệnh nhân và hệ thống y tế.
- Cá nhân hoá điều trị
  - Mỗi bệnh nhân có thể có nhiều mẫu xét nghiệm liên tiếp theo thời gian. RNN/LSTM hoặc mô hình học sâu theo chuỗi cho phép dự đoán xu hướng – ví dụ Hb giảm nhanh trong 3 lần xét nghiệm liên → gợi ý phải can thiệp bổ sung sắt ngay.
  - Giúp thiết kế liều lượng thuốc/ bổ sung dinh dưỡng dựa vào tốc độ thay đổi các chỉ số, không chỉ giá trị tĩnh.
- Tiết kiệm nhân lực và mở rộng quy mô
  - Ở bệnh viện huyện hoặc cơ sở y tế vùng xa thiếu bác sĩ chuyên khoa huyết học, ứng dụng ML có thể hỗ trợ kỹ thuật viên xét nghiệm phát hiện ca nghi ngờ.

- Giúp mở rộng dịch vụ “khám từ xa” (telemedicine): gửi kết quả xét nghiệm lên hệ thống, mô hình tự động gợi ý chẩn đoán bước đầu, bác sĩ hội chẩn online có thể tham khảo.
- Liên tục học và cải tiến
  - Mô hình học máy có thể cập nhật theo dữ liệu mới: bệnh nhân mới, biến đổi chủng vi trùng, dịch bệnh (ví dụ xuất hiện bệnh huyết học do virus).
  - Khi dữ liệu mở rộng, hệ thống tự động điều chỉnh tham số để giữ cho Accuracy, Recall luôn ở mức mong muốn.
- Gia tăng độ tin cậy qua giải thích mô hình
  - Bằng SHAP/LIME, mô hình trả lời “Tại sao đoán thiếu máu thiếu sắt?” qua việc chỉ ra HCT và MCV quá thấp, RDW cao bất thường.
  - Giúp bác sĩ kiểm chứng và có tính “minh bạch” hơn so với phép toán thủ công.
- Tự động hoá báo cáo & cảnh báo
  - Khi kết hợp với hệ thống HIS (Hospital Information System), ngay khi phòng xét nghiệm có kết quả CBC, mô hình ML có thể gửi alert cho bác sĩ trực tiếp nếu chỉ số vượt ngưỡng nguy hiểm (ví dụ bạch cầu  $> 50.000/\mu\text{L}$  gợi ý bạch cầu cấp).
  - Rút ngắn thời gian phản ứng y tế, cải thiện kết quả điều trị.

### 3.2. Công cụ và ngôn ngữ

Để phát triển chương trình học máy cần sử dụng ngôn ngữ Python vì ngôn ngữ Python là một trong những ngôn ngữ phổ biến và được ưa chuộng nhất trong phát triển chương trình học máy. Python có rất nhiều thư viện hỗ trợ học máy, giúp giảm bớt sự phức tạp trong việc xây dựng và huấn luyện các mô hình ví dụ như pandas, numpy, scikit-learn,.... Môi trường được sử dụng để thử nghiệm là Google Colab hay Google Colaboratory là một sản phẩm của Google Research. Colab dựa trên Jupyter Notebook, người dùng có thể viết và thực thi đoạn mã python thông qua trình duyệt.



Ưu điểm:

- **Miễn phí GPU và TPU:** Google Colab cung cấp miễn phí tính năng truy cập đến GPU (Đơn vị xử lý đồ họa) và TPU (Đơn vị xử lý tensor), giúp tăng tốc độ xử lý dữ liệu và huấn luyện mô hình học máy, đặc biệt trong các tác vụ phức tạp như deep learning.
- **Không cần cài đặt phần mềm:** Colab hoạt động trực tuyến trên trình duyệt web mà không yêu cầu người dùng phải cài đặt phần mềm.
- **Thư viện tài nguyên được cài đặt sẵn:** Google Colab đi kèm với một loạt các thư viện tài nguyên phổ biến đã được cài sẵn, như NumPy, Pandas, Matplotlib, Scikit-learn,...

### 3.3. Xây dựng ứng dụng

#### 3.3.1. Mô tả dữ liệu

Chương trình sẽ đưa ra kết quả về chẩn đoán bệnh dựa trên chỉ số xét nghiệm được đưa vào. Dữ liệu đưa vào dưới dạng file csv, xlsx, ...

Name	Date modified	Type	Size
 test.csv	2025-06-07 8:33 SA	Microsoft Excel C...	15 KB
 train.csv	2025-06-07 8:33 SA	Microsoft Excel C...	34 KB

Hình 3.1: Hình ảnh file dữ liệu huấn luyện và thử nghiệm

#### Dữ liệu huấn luyện :

- Kích thước: 34KB
- Số dòng: 1422 dòng
- Định dạng: CSV
- Dữ liệu được chia thành các cột: Giới tính, Hemoglobin(Huyết sắc tố), MCH, MCHC, MCV, Kết quả

Gender	Hemoglobin	MCH	MCHC	MCV	Result
1	14.9	22.7	29.1	83.7	0
0	15.9	25.4	28.3	72.0	0
0	9.0	21.5	29.6	71.2	1
0	14.9	16.0	31.4	87.5	0
1	14.7	22.0	28.2	99.5	0
0	11.6	22.3	30.9	74.5	1

*Hình 3.2: Hình ảnh dữ liệu huấn luyện*

**Các cột dữ liệu được định nghĩa như sau:**

- Gender: Giới tính (0: Nữ, 1: Nam).
- Hemoglobin: Lượng hemoglobin trong máu
- MCH: Lượng hemoglobin trung bình trong hồng cầu.
- MCHC: Nồng độ hemoglobin trong hồng cầu.
- MCV: Thể tích trung bình của hồng cầu.
- Result: Kết quả phân loại bệnh về máu (0: không có bệnh về máu, 1: có dấu hiệu của bệnh về máu).

**Dữ liệu thử nghiệm :**

- Kích thước: 15KB
- Số dòng: 607 dòng
- Định dạng: CSV
- Dữ liệu được chia thành các cột: Giới tính, Hemoglobin(Huyết sắc tố), MCH, MCHC, MCV, loại bỏ cột kết quả



Gender	Hemoglobin	MCH	MCHC	MCV	Result
0	15.2	21.9	29.6	69.5	0
1	13.4	17.7	31.6	97.2	1
1	15.9	22.4	28.9	93.0	0
1	12.1	25.9	29.3	82.6	1
0	14.5	24.2	27.9	71.7	0

Hình 3.3: Hình ảnh dữ liệu thử nghiệm

### 3.3.2. Áp dụng KNN vào bài toán.

Chẩn đoán bệnh về máu dựa trên chỉ số xét nghiệm bằng mô hình KNN.

Chương trình thực hiện trên Google Colab đọc dữ liệu từ file CSV và đưa ra chẩn đoán ngay trên Google Colab.

#### Chương trình:

- Import các thư viện cần thiết cho chương trình:

```
from collections import Counter
import numpy as np
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
```

Hình 3.4: Các thư viện cần thiết (KNN)

Trong đó:

`from collections import Counter`: Lớp Counter để đếm tần suất xuất hiện của các phần tử trong một danh sách.

`import numpy as np`: Thư viện numpy dùng để làm việc với ma trận, cụ thể ở đây là dữ liệu ma trận huấn luyện.

`import pandas as pd`: Thư viện pandas dùng để làm việc với dữ liệu, cụ thể đọc dữ liệu và chuyển dữ liệu từ dạng dataframe về ma trận.

`from sklearn import preprocessing`: Thư viện sklearn để chuẩn hóa dữ liệu.

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

Hàm để chia dữ liệu thành tập huấn luyện và kiểm tra.

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

accuracy\_score: Hàm đánh giá hiệu suất của mô hình.

- Đọc và xử lý dữ liệu:

Dữ liệu được đọc dưới dạng file CSV thông qua thư viện pandas và lưu nội dung vào một Dataframe có tên là data.

```
data = pd.read_csv("train.csv")
print("Kiểm tra dữ liệu:")
print(data.info())
```

Kiểm tra dữ liệu:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1421 entries, 0 to 1420
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Gender      1421 non-null   int64
1   Hemoglobin  1421 non-null   float64
2   MCH         1421 non-null   float64
3   MCHC        1421 non-null   float64
4   MCV         1421 non-null   float64
5   Result      1421 non-null   int64
dtypes: float64(4), int64(2)
memory usage: 66.7 KB
None
```

*Hình 3.5: Thông tin chi tiết về Dataframe (KNN)*

Mục đích thông tin chi tiết giúp ta nắm được cấu trúc và tình trạng của dữ liệu trước khi thực hiện các bước xử lý hoặc phân tích.

```

X = data.iloc[1:,-1].values
print("\nDữ liệu của X: \n", X);

y = data.iloc[1:,-1].values
print("\nDữ liệu của y: \n", y)

```

```

Dữ liệu của X:
[[ 0.  15.9 25.4 28.3 72. ]
 [ 0.   9. 21.5 29.6 71.2]
 [ 0.  14.9 16.  31.4 87.5]
 ...
 [ 1.  13.1 17.7 28.1 80.7]
 [ 0.  14.3 16.2 29.5 95.2]
 [ 0.  11.8 21.2 28.4 98.1]]

```

```

Dữ liệu của y:
[0 1 0 ... 1 0 1]

```

*Hình 3.6: Dữ liệu được đọc từ file (KNN)*

Dữ liệu đọc từ file được trích xuất ra 2 file

- File dữ liệu đầu vào X
- File chứa nhãn y
- Tiền xử lý dữ liệu:

Chuẩn hoá dữ liệu bằng phương pháp điều chỉnh tỉ lệ (Rescale Data).

```
#Xử lý dữ liệu
chuyendoimin_max = preprocessing.MinMaxScaler()
X = chuyendoimin_max.fit_transform(X)
print(X)

[[0.          0.90291262 0.67142857 0.10638298 0.08074534]
 [0.          0.23300971 0.39285714 0.38297872 0.05590062]
 [0.          0.80582524 0.          0.76595745 0.5621118 ]
 ...
 [1.          0.63106796 0.12142857 0.06382979 0.35093168]
 [0.          0.74757282 0.01428571 0.36170213 0.80124224]
 [0.          0.50485437 0.37142857 0.12765957 0.89130435]]
```

Hình 3.7: Dữ liệu sau khi được chuẩn hóa (KNN)

Dữ liệu được chuẩn hoá có giá trị nằm trong khoảng  $[0, 1]$  giúp cải thiện tốc độ và tính hiệu quả của thuật toán.

- Xây dựng và huấn luyện mô hình:

Để áp dụng thuật toán KNN cần hàm tính khoảng cách Euclidean giữa hai điểm dữ liệu gồm nhiều cột dữ liệu khác nhau.

```
#Hàm tính khoảng cách Euclidean giữa hai điểm.
def khoang_cach_euclidean(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))
```

Hình 3.8: Hàm tính khoảng cách Euclidean giữa hai điểm (KNN)

Hàm dự đoán lớp của dữ liệu kiểm tra dựa trên k láng giềng gần nhất với công thức  $\text{np.sqrt}(\text{np.sum}((x1 - x2) ** 2))$ .

```

def duDoan(X_train, y_train, X_test, k):
    list_du_doan = []

    for x_test in X_test:
        list_khoang_cach = []

        for i, x_train in enumerate(X_train):
            #Tính khoảng cách từ điểm test đến điểm train
            khoang_cach = khoang_cach_euclidean(x_test, x_train)
            list_khoang_cach.append((khoang_cach, y_train[i]))

        #Sắp xếp các khoảng cách theo thứ tự tăng dần
        list_khoang_cach.sort(key=lambda x: x[0])

        #Lấy k láng giềng
        k_lang_gieng = list_khoang_cach[:k]

        list_nhan = []
        #Lấy nhãn từ k láng giềng
        for lg in k_lang_gieng:
            nhan = lg[1]
            list_nhan.append(nhan)

        #Đếm số lần xuất hiện của nhãn
        dem_nhan = Counter(list_nhan)
        #Lấy ra cặp dữ liệu xuất hiện nhiều nhất
        cap_pb = dem_nhan.most_common(1)[0]
        #Từ cặp dữ liệu trên, lấy ra nhãn phổ biến nhất
        nhan_pb = cap_pb[0]

        list_du_doan.append(nhan_pb)

    return np.array(list_du_doan)

```

Hình 3.9: Hàm dự đoán lớp của kiểu dữ liệu kiểm tra (KNN)

Hàm dự đoán dữ liệu theo thuật toán KNN.

```

k = 10 #Số lượng láng giềng k
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=84)
list_du_doan = duDoan(X_train, y_train, X_test, k)
# print(list_du_doan)
print(list_du_doan)

```

```

[0 1 0 1 1 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 1 0 1 1 1
 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 1 1 0 1 1 0 1
 0 1 1 0 0 0 0 1 1 0 0 1 0 1 0 1 1 0 0 0 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0
 0 0 1 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 0
 1 1 0 1 0 1 0 1 1 0 1 1 1 1 0 1 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0 1 1 1
 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 0 1 0 0
 1 1 1 1 1 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 1 1
 1 0 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 1 0 1 0 1]

```

Hình 3.10: Chia dữ liệu và đào tạo mô hình (KNN)

Chia tập dữ liệu đầu vào X thành 2 bộ dữ liệu riêng biệt. 80% sử dụng để đào tạo mô hình và 20% còn lại làm dữ liệu thử nghiệm

Cho số lượng láng giềng  $k = 10$  sau đó gọi lại hàm dự đoán với tham biến là dữ liệu huấn luyện là `X_train`, `y_train`, dữ liệu thử nghiệm `X_test` và số lượng láng giềng.

- Đánh giá mô hình :

```
cm = confusion_matrix(y_test, list_du_doan)
print("\nĐánh giá mô hình qua Confusion Matrix: \n", cm)
print(f"\nMô hình đã dự đoán đúng {cm[1][1]} trường hợp True Positive.")
print(f"Mô hình đã dự đoán đúng {cm[0][0]} trường hợp True Negative.")
print(f"Mô hình đã dự đoán sai {cm[0][1]} trường hợp False Positive.")
print(f"Mô hình đã dự đoán sai {cm[1][0]} trường hợp False Negative.")

print(f"\nĐộ chính xác: {accuracy_score(y_test, list_du_doan)*100:.2f}%")
```

Đánh giá mô hình qua Confusion Matrix:

```
[[169   3]
 [  2 110]]
```

Mô hình đã dự đoán đúng 110 trường hợp True Positive.

Mô hình đã dự đoán đúng 169 trường hợp True Negative.

Mô hình đã dự đoán sai 3 trường hợp False Positive.

Mô hình đã dự đoán sai 2 trường hợp False Negative.

Độ chính xác: 98.24%

*Hình 3.11: Ma trận nhầm lẫn và độ chính xác của mô hình (KNN)*

Đánh giá mô hình thông qua ma trận nhầm lẫn (Confusion Matrix) và độ chính xác (Accuracy Score).

- Dự đoán:
  - Lấy 5 dữ liệu từ data test để thử nghiệm.
  - Lấy dữ liệu và nhãn để so sánh

```
data_test = pd.read_csv("test.csv")
X_test_data = data_test.iloc[:5, :-1].values
y_test_data = data_test.iloc[:5, -1].values
print(X_test_data)
print(y_test_data)
```

```
[[ 0.  15.2 21.9 29.6 69.5]
 [ 1.  13.4 17.7 31.6 97.2]
 [ 1.  15.9 22.4 28.9 93. ]
 [ 1.  12.1 25.9 29.3 82.6]
 [ 0.  14.5 24.2 27.9 71.7]]
[0 1 0 1 0]
```

Hình 3.12: Dữ liệu thử nghiệm (KNN)

Lấy dữ liệu thử nghiệm cuối cùng data\_test bằng cách đọc dữ liệu từ file test.csv

Lấy 5 dữ liệu thử nghiệm lần cuối để so sánh với kết quả thử nghiệm cũ.

- Kết quả sau khi thử nghiệm:

```
#Chuẩn hoá dữ liệu cho data test
x_test_data = chuyendoimin_max.transform(X_test_data)
list_du_doan_test = duDoan(X_train, y_train, x_test_data, k)

print("\nDự đoán trên dữ liệu test:")
for i, duDoan in enumerate(list_du_doan_test):
    print(f"{X_test_data[i]} - Dự đoán: {duDoan} - Kết quả thực: {y_test_data[i]}")
```

Dự đoán trên dữ liệu test:

```
[ 0.  15.2 21.9 29.6 69.5] - Dự đoán: 0 - Kết quả thực: 0
[ 1.  13.4 17.7 31.6 97.2] - Dự đoán: 1 - Kết quả thực: 1
[ 1.  15.9 22.4 28.9 93. ] - Dự đoán: 0 - Kết quả thực: 0
[ 1.  12.1 25.9 29.3 82.6] - Dự đoán: 1 - Kết quả thực: 1
[ 0.  14.5 24.2 27.9 71.7] - Dự đoán: 0 - Kết quả thực: 0
```

Hình 3.13: Kết quả thử nghiệm (KNN)

Tiền xử lý dữ liệu sao cho dữ liệu có giá trị nằm trong khoảng [0,1] để đối chiếu với kết quả thử nghiệm cũ.

Dự đoán dữ liệu thử nghiệm bằng cách gọi lại hàm dự đoán duDoan ban đầu  
`list_du_doan_test = duDoan(X_train, y_train, x_test_data, k)`.

In ra dữ liệu dự đoán prediction và kết quả thực y\_data\_test bằng cách duyệt qua toàn bộ dữ liệu đã được dự đoán thông qua model.

### 3.3.3. Áp dụng Navie Bayes vào bài toán

- Thêm các thư viện cần thiết

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn import preprocessing
```

*Hình 3.14: Các thư viện gần nhất (Navie Bayes)*

`import numpy as np`: Thư viện numpy dùng để làm việc với ma trận, cụ thể ở đây là dữ liệu ma trận huấn luyện.

`import pandas as pd`: Thư viện pandas dùng để làm việc với dữ liệu, cụ thể đọc dữ liệu và chuyển dữ liệu từ dạng dataframe về ma trận.

`from sklearn import preprocessing`: Thư viện sklearn để chuẩn hóa dữ liệu.

`from sklearn.model_selection import train_test_split`: Hàm để chia dữ liệu thành tập huấn luyện và kiểm tra.

`from sklearn.metrics import accuracy_score`: Hàm đánh giá hiệu suất của mô hình.

`from sklearn.metrics import confusion_matrix`: Hàm đưa ra ma trận nhầm lẫn.

`from sklearn.naive_bayes import GaussianNB`: Thuật toán Navie Bayes với phân phối chuẩn (GaussianNB)



- Đọc và hiển thị thông tin dữ liệu

```
data_frame = pd.read_csv('train.csv')
data_frame.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1421 entries, 0 to 1420
Data columns (total 6 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Gender          1421 non-null   int64
1   Hemoglobin      1421 non-null   float64
2   MCH             1421 non-null   float64
3   MCHC           1421 non-null   float64
4   MCV             1421 non-null   float64
5   Result          1421 non-null   int64
dtypes: float64(4), int64(2)
memory usage: 66.7 KB
```

Hình 3.15: Đọc dữ liệu từ file (Navie Bayes)

Nhằm hiểu rõ cấu trúc và tình trạng của dữ liệu trước khi thực hiện các bước xử lý hoặc phân tích.

```
X = data_frame.iloc[:, :-1].values
y = data_frame.iloc[:, -1].values
print(X)
print(y)

[[ 1.  14.9 22.7 29.1 83.7]
 [ 0.  15.9 25.4 28.3 72. ]
 [ 0.   9. 21.5 29.6 71.2]
 ...
 [ 1.  13.1 17.7 28.1 80.7]
 [ 0.  14.3 16.2 29.5 95.2]
 [ 0.  11.8 21.2 28.4 98.1]]
[0 0 1 ... 1 0 1]
```

Hình 3.16: Thông tin chi tiết về DataFrame (Navie Bayes)

Dữ liệu được tách thành hai phần như sau:

- Dữ liệu đầu vào X (lấy dữ liệu từ hàng đầu tiên đến hàng cuối cùng, lấy dữ liệu từ cột đầu tiên đến hết ngoại trừ cột cuối cùng)

- Dữ liệu chứa nhãn y (lấy dữ liệu từ hàng đầu tiên đến hàng cuối cùng, chỉ lấy dữ liệu của cột cuối cùng)
- Tiền xử lý dữ liệu

```
convert = preprocessing.MinMaxScaler()
X = convert.fit_transform(X)
print(X)
```

```
[[1.         0.80582524 0.47857143 0.27659574 0.44409938]
 [0.         0.90291262 0.67142857 0.10638298 0.08074534]
 [0.         0.23300971 0.39285714 0.38297872 0.05590062]
 ...
 [1.         0.63106796 0.12142857 0.06382979 0.35093168]
 [0.         0.74757282 0.01428571 0.36170213 0.80124224]
 [0.         0.50485437 0.37142857 0.12765957 0.89130435]]
```

Hình 3.17: Dữ liệu sau khi đã được chuẩn hóa (Navie Bayes)

Sử dụng tiền xử lý dữ liệu `preprocessing.MinMaxScaler()` để chuẩn hóa dữ liệu, sau đó gán lại X chính bằng dữ liệu củ đã được chuẩn hóa `X = convert.fit_transform(X)`.

Dữ liệu được chuẩn hóa có giá trị nằm trong khoảng  $[0,1]$  giúp cải thiện tốc độ và tăng tính hiệu quả của thuật toán.

- Xây dựng và huấn luyện mô hình.

Để áp dụng thuật toán Navie Bayes, đầu tiên ta cần tách dữ liệu ban đầu thành hai dữ liệu khác nhau.

```
np.random.seed(42)
xtrain, xtest, ytrain, ytest = train_test_split(X,y,test_size = 0.2)
print('Dữ liệu huấn luyện')
print(xtrain)
print(ytrain)
```

```
Dữ liệu huấn luyện
[[0.         0.66019417 0.73571429 0.14893617 0.47204969]
 [0.         1.         0.05         0.46808511 0.27950311]
 [0.         0.83495146 0.86428571 0.76595745 0.24534161]
 ...
 [0.         0.53398058 0.87857143 0.29787234 0.94099379]
 [1.         0.52427184 0.86428571 0.65957447 0.26397516]
 [0.         0.47572816 0.42857143 0.9787234  0.45962733]]

[0 0 0 ... 0 1 1]
```

Hình 3.18: Phân tách dữ liệu huấn luyện (Navie Bayes)

Dữ liệu đầu tiên là để huấn luyện (chiếm 80% dữ liệu ban đầu) gồm dữ liệu huấn luyện đầu vào `xtrain`, dữ liệu chứa nhãn `ytrain`.

```
print('Dữ liệu thử nghiệm ban đầu')
print(xtest)
print(ytest)
```

```
Dữ liệu thử nghiệm ban đầu
[[1.          0.59223301 0.22142857 0.42553191 0.46273292]
 [0.          0.7961165  0.87142857 0.63829787 0.04037267]
 [1.          0.66990291 0.83571429 0.68085106 0.24534161]
 ...
 [1.          0.53398058 0.20714286 0.72340426 0.20496894]
 [0.          0.6407767  0.3          0.14893617 0.49068323]
 [1.          0.37864078 0.87857143 0.19148936 0.95341615]]
[[1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 0 1 0 1 1 1 0 0 0 1
 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 0 0 0
 0 0 0 1 0 1 1 1 1 0 1 0 0 1 0 1 1 0 1 0 0 0 0 1 0 1 1 1 0 1 0 1 0 0 1 0 1
 1 0 1 0 1 1 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 1 0 0 1 0 0 0 0 1
 1 1 0 1 0 0 1 1 0 1 1 1 0 0 1 0 1 0 1 0 1 1 1 0 0 0 1 0 1 1 0 0 0 0 0
 0 1 1 1 1 1 0 1 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 0
 1 0 1 1 0 1 1 1 1 0 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 0 1
 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 1 1 1 0 1
 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1]
```

Hình 3.19: Phân tách dữ liệu thử nghiệm (Navie Bayes)

Tiếp theo là dữ liệu thử nghiệm ban đầu (chiếm 20% dữ liệu ban đầu) gồm dữ liệu đầu vào thử nghiệm `xtest`, dữ liệu chứa nhãn để thử nghiệm `ytest`.

```
model = GaussianNB()
model.fit(xtrain,ytrain)
ypred = model.predict(xtest)
print(ypred)
```

```
[[1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 0 1 0 1 1 1 0 0 0 1
 0 1 1 0 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0 0
 0 0 0 1 0 1 1 1 1 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 0 1 1 1 0 1 0 1 0 0 1 0 1
 1 0 1 0 1 1 0 1 0 0 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 1 0 0 1 0 0 0 1 1
 1 1 0 1 0 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 1 1 1 0 0 0 1 0 0 1 0 0 1 0 0
 0 1 1 1 1 1 0 1 1 0 0 1 1 1 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 0
 1 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 1 1 1 0 1
 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 1 1 0 1]
```

Hình 3.20: Tạo mô hình Navie Bayes và dữ liệu dự đoán.

Tạo một mô hình Navie Bayes với phân phối chuẩn Gaussian và gán bằng một biến `model = GaussianNB`.

Huấn luyện mô hình với dữ liệu huấn luyện `xtrain`, `ytrain`.

Tạo dữ liệu dự đoán thông qua dữ liệu thử nghiệm và in ra kết quả dự đoán (`ypred = model.predict(xtest)`).

- Đánh giá mô hình.

```

cm = confusion_matrix(ytest,ypred)
print("Confusion Matrix")
print(cm)
print(f'\nTrue Positives (TP): {cm[0,0]}')
print(f'False Positives (FP): {cm[0,1]}')
print(f'False Negatives (FN): {cm[1,0]}')
print(f'True Negatives (TN): {cm[1,1]}\n')
print('Accuracy Score')
print(accuracy_score(ytest,ypred))

```

```

· Confusion Matrix
[[150  7]
 [ 7 121]]

True Positives (TP): 150
False Positives (FP): 7
False Negatives (FN): 7
True Negatives (TN): 121

Accuracy Score
0.9508771929824561

```

Hình 3.21: Ma trận nhầm lẫn và độ chính xác của mô hình (Navie Bayes)

Đánh giá mô hình thông qua ma trận nhầm lẫn `confusion_matrix` và độ chính xác `accuracy_score` với dữ liệu đánh giá là dữ liệu thử nghiệm `ytest` và dữ liệu dự đoán `ypred`.

- Thử nghiệm kết quả chẩn đoán.

```

data_test = pd.read_csv('test.csv')
X_data_test = data_test.iloc[:,5, :-1].values
y_data_test = data_test.iloc[:,5, -1].values
print(X_data_test)
print(y_data_test)

```

```

· [[ 0.  15.2 21.9 29.6 69.5]
   [ 1.  13.4 17.7 31.6 97.2]
   [ 1.  15.9 22.4 28.9 93. ]
   [ 1.  12.1 25.9 29.3 82.6]
   [ 0.  14.5 24.2 27.9 71.7]]
[0 1 0 1 0]

```

Hình 3.22: Tạo dữ liệu thử nghiệm lại kết quả (Navie Bayes)

Lấy dữ liệu thử nghiệm cuối cùng `data_test` bằng cách đọc dữ liệu từ file `test.csv`

Lấy 5 dữ liệu thử nghiệm lần cuối để so sánh với kết quả thử nghiệm cũ.

```
X_data_test = convert.transform(X_data_test)
predictions_test = model.predict(X_data_test)
print('Dự đoán dữ liệu trên test')
for i, prediction in enumerate(predictions_test):
    print(f'{X_data_test[i]} - Dự đoán: {prediction} - Kết quả thực: {y_data_test[i]}')
```

```
· Dự đoán dữ liệu trên test
[0.      0.83495146 0.42142857 0.38297872 0.00310559] - Dự đoán: 0 - Kết quả thực: 0
[1.      0.66019417 0.12142857 0.80851064 0.86335404] - Dự đoán: 0 - Kết quả thực: 1
[1.      0.90291262 0.45714286 0.23404255 0.73291925] - Dự đoán: 0 - Kết quả thực: 0
[1.      0.53398058 0.70714286 0.31914894 0.40993789] - Dự đoán: 1 - Kết quả thực: 1
[0.      0.76699029 0.58571429 0.0212766 0.07142857] - Dự đoán: 0 - Kết quả thực: 0
```

Hình 3.23: Kết quả sau khi đã thử nghiệm (Navie Bayes)

Tiền xử lý dữ liệu sao cho dữ liệu có giá trị nằm trong khoảng [0,1] để đối chiếu với kết quả thử nghiệm cũ.

Dự đoán dữ liệu thử nghiệm bằng cách gọi lại biến `model` ban đầu `prediction_test = model.predict(X_data_test)`.

In ra dữ liệu dự đoán `prediction` và kết quả thực `y_data_test` bằng cách duyệt qua toàn bộ dữ liệu đã được dự đoán thông qua `model`.

### 3.4. So sánh độ hiệu quả của hai thuật toán.

- Giống nhau.

Sử dụng chung dữ liệu chuẩn đoán, tiền xử lý dữ liệu để đưa về dạng `MinMaxScaler` nhằm tăng hiệu quả của bài toán.

Sử dụng ma trận nhầm lẫn (Confusion Matrix) và độ chính xác (Accuracy Score) để đánh giá mô hình thuật toán.

- Khác nhau.

Vì KNN và Navie Bayes cùng là các thuật toán của học máy nhưng sử dụng hàm dự đoán khác nhau dẫn đến ma trận nhầm lẫn và độ chính xác khác nhau.

- Ma trận nhầm lẫn (Confusion Matrix)

Ta cần thêm các thư viện vào chương trình nhằm so sánh ma trận nhầm lẫn của hai thuật toán KNN và Navie Bayes.

```
import matplotlib.pyplot as plt
import seaborn as sns
```

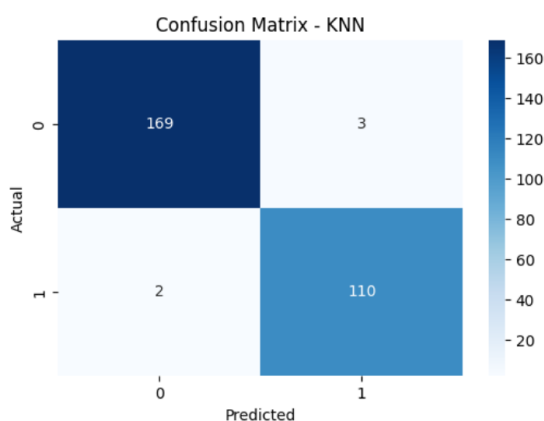
Hình 3.24: Thư viện vẽ biểu đồ

`import matplotlib.pyplot as plt`: đây là thư viện dùng vẽ biểu đồ, đặc biệt là biểu đồ 2D như biểu đồ đường, cột hay heatmap (trong trường hợp sử dụng để hiển thị và so sánh ma trận nhầm lẫn).

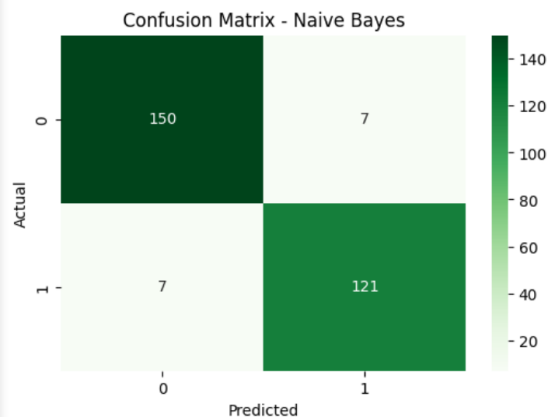
`import seaborn as sns`: đây là thư viện nâng cao hơn `matplotlib`, đặc biệt với những biểu đồ thống kê hoặc trực quan hóa dữ liệu.

Sau khi thêm các thư viện vẽ biểu đồ, thì biểu đồ được thể hiện thông qua các câu lệnh sau.

```
# Biểu đồ heatmap cho Confusion Matrix KNN
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - KNN')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



```
# Biểu đồ heatmap cho Confusion Matrix Naive Bayes
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens')
plt.title('Confusion Matrix - Naive Bayes')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



Hình 3.25: So sánh Ma trận nhầm lẫn (Confusion Matrix)

`plt.figure(figsize = (6, 4))`: Tạo một kích thước với chiều ngang là 6 inch và chiều rộng 4 inch thông qua thư viện `matplotlib` đã thêm trên.

`sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')`: Vẽ biểu đồ heatmap với dữ liệu là `cm` (là ma trận nhầm lẫn của từng thuật toán), `annot = True` dùng để hiển thị số trong các ô khác nhau, `fmt = 'd'` (format) dùng để định dạng số trong các ô là số nguyên và `cmap` để hiển thị màu sắc khác nhau để so sánh.

`plt.title`: dùng để đặt tiêu đề cho biểu đồ.

`plt.xlabel`: dùng để đặt tên cho trục hoành.

`plt.ylabel`: dùng để đặt tên cho trục tung.

`plt.show()`: dùng để hiển thị biểu đồ

#### - Độ chính xác (Accuracy Score)

Độ chính xác của từng thuật toán là duy nhất nên thay vì sử dụng biểu đồ heatmap như ma trận nhầm lẫn (Confusion Matrix) thì ta nên dùng biểu đồ cột đứng barplot để so sánh giá trị độ chính xác của hai thuật toán.

Đầu tiên để sử dụng biểu đồ so sánh độ chính xác, đầu tiên ta cần lưu trữ dữ liệu vào file sau đó đọc dữ liệu từ file đó để so sánh.

```
with open("accuracy_knn.txt", "w") as f:
    f.write(str(accuracy_score(y_test, list_du_doan)))
```

```
from google.colab import files
files.download("accuracy_knn.txt")
```

*Hình 3.26: Lưu trữ độ chính xác (Accuracy Score)*

`with open("accuracy_knn.txt", "w") as f`: dùng để mở file `accuracy_knn.txt` và dùng `"w"` (write) để ghi vào file vừa mở đó.

`f.write(str(accuracy_score()))`: lưu trữ độ chính xác của thuật toán KNN vào file đó với dữ liệu gồm `ytest` (dữ liệu huấn luyện) và `list_du_doan` (dữ liệu dự đoán).

Tương tự ta lưu độ chính xác của thuật toán Navie Bayes vào file `accuracy_nb.txt` tương tự như đoạn mã lệnh trên.

Sau đó trong file so sánh `compare.ipynb` ta sử dụng hai file độ chính xác của từng thuật toán như sau:

```

with open('accuracy_knn.txt', 'r') as f:
    contentknn = f.read()
accuracy_knn = float(contentknn)
print(accuracy_knn)
with open('accuracy_nb.txt', 'r') as f:
    contentnb = f.read()
accuracy_nb = float(contentnb)
print(accuracy_nb)

```

```

0.9823943661971831
0.9508771929824561

```

*Hình 3.27: Đọc dữ liệu Accuracy Score vừa lưu*

with open("accuracy\_knn.txt", "r") as f: dùng để mở file accuracy\_knn.txt và dùng "r" (read) để đọc dữ liệu từ file ban đầu, sau đó gán vào biến contentknn = f.read().

accuracy\_knn = float(contentknn) ép kiểu số thực cho dữ liệu vừa đọc và gán vào biến accuracy\_knn.

Tương tự với Navie Bayes.

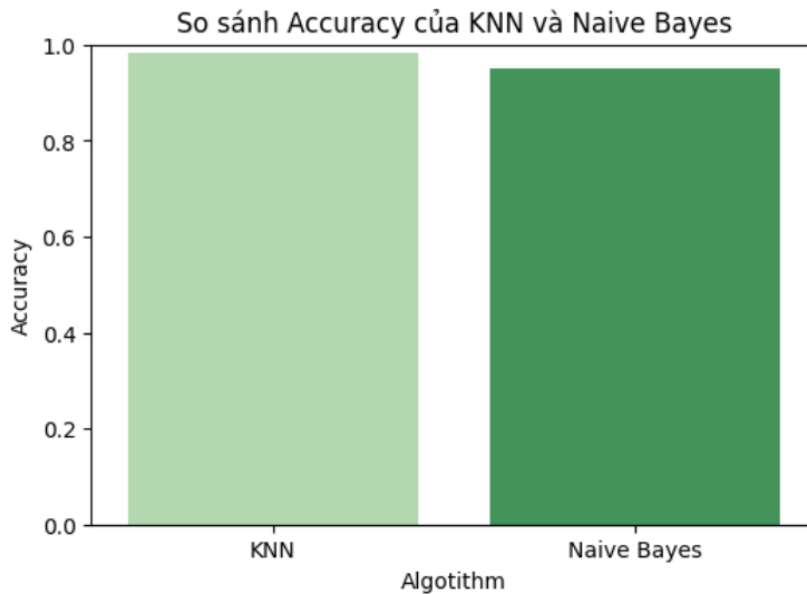
Tiếp theo ta sử dụng hai thư viện matplotlib và seaborn đã thêm ở trên để vẽ biểu đồ so sánh độ chính xác.



```

algorithms = ['KNN', 'Naive Bayes']
accuracies = [accuracy_knn, accuracy_nb]
plt.figure(figsize=(6, 4))
sns.barplot(x=algorithms, y=accuracies, palette='Greens')
plt.title('So sánh Accuracy của KNN và Naive Bayes')
plt.ylim(0, 1)
plt.ylabel('Accuracy')
plt.xlabel('Algotithm')
plt.show()

```



*Hình 3.28: So sánh độ chính xác (Accuracy Score)*

Gọi biến `algorithms = ['KNN', 'Naive Bayes']` để gán các giá trị cho trục hoành và `accuracies = [accuracy_knn, accuracy_nb]` để gán các giá trị cho trục tung.

`sns.barplot(x=algorithms, y=accuracies, palette='Greens')` : tạo biểu đồ cột đứng và gán trục hoành, trục tung là hai biến vừa tạo.

`plt.ylim(0, 1)` : đặt giới hạn cho trục tung với `ymin=0` và `ymax=1`.

## KẾT LUẬN

Nghiên cứu này đã thành công xây dựng một ứng dụng hỗ trợ chuẩn đoán bệnh về máu thông qua chỉ số xét nghiệm. Ứng dụng cho phép nhập liệu kết quả xét nghiệm, phân tích và đưa ra dự đoán về các bệnh có liên quan đến máu.

Các thuật toán học máy được sử dụng đã chứng minh hiệu quả trong việc phân loại và dự đoán bệnh, với độ chính xác đạt [kết quả cụ thể]. Ứng dụng có giao diện thân thiện, dễ sử dụng, phù hợp với cả người dùng phổ thông và cán bộ y tế.

Tuy nhiên, nghiên cứu này vẫn còn một số hạn chế. Ứng dụng mới chỉ hỗ trợ chẩn đoán một số bệnh lý về máu phổ biến, chưa bao gồm đầy đủ các bệnh hiếm gặp. Độ chính xác của ứng dụng cũng cần được tiếp tục cải thiện bằng cách bổ sung dữ liệu huấn luyện và tối ưu thuật toán.

Trong tương lai, nghiên cứu sẽ được tiếp tục phát triển theo các hướng như là mở rộng phạm vi chuẩn đoán, bao gồm thêm nhiều bệnh lý về máu. Nâng cao độ chính xác của ứng dụng bằng cách áp dụng các thuật toán học máy tiên tiến hơn và sử dụng bộ dữ liệu lớn hơn. Kết hợp ứng dụng với các công nghệ khác như trí tuệ nhân tạo (AI), Internet vạn vật (IoT) để tạo ra một hệ thống chẩn đoán bệnh toàn diện và hiệu quả hơn. Triển khai ứng dụng trên quy mô lớn, phục vụ cộng đồng và hỗ trợ công tác khám chữa bệnh.

## TÀI LIỆU THAM KHẢO

- [1] Trần Hùng Cường, Nguyễn Phương Nga, Giáo trình Trí tuệ nhân tạo, 2014.
- [2] <https://vnptai.io/vi/blog/detail/supervised-learning>
- [3] <https://digitaldefynd.com/IQ/supervised-learning-pros-cons/>
- [4] <https://www.ibm.com/think/topics/unsupervised-learning>
- [5] <https://www.geeksforgeeks.org/ml-semi-supervised-learning/>
- [6] <https://theintactone.com/2024/11/13/semi-supervised-learning/>
- [7] <https://www.geeksforgeeks.org/what-is-reinforcement-learning/>
- [8] <https://www.elastic.co/what-is/knn#how-does-knn-work>
- [9] <https://www.geeksforgeeks.org/k-nearest-neighbours/>
- [10] <https://interdata.vn/blog/knn/>
- [11] <https://www.geeksforgeeks.org/naive-bayes-classifiers/>
- [12] <https://basicmachinelearningnaviebayes/2017/08/08/nbc/>
- [13] <https://eithealth.eu/news-article/machine-learning-in-healthcare/>
- [14] <http://bvphuchoichucnanghcm.vn/tri-tue-nhan-cao-ai/5963>
- [15] <https://www.kaggle.com/datasets>
- [16] <https://pmc.ncbi.nlm.nih.gov/articles/PMC8950225/#sec1-healthcare-10-0054>