**KHOA KỸ THUẬT VÀ CÔNG NGHỆ**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**CHUYÊN ĐỀ ASP.NET**

**HỌC KỲ ……, NĂM HỌC : 2024**

**ĐỀ TÀI : PHÁT TRIỂN ỨNG DỤNG HỌC MÁY CHUẨN ĐOÁN BỆNH DỰA VÀO TRIỆU CHỨNG ĐẦU VÀO**

*Sinh viên thực hiện:*

Họ tên:

MSSV:

Lớp:

*Giáo viên hướng dẫn:*

Họ tên:

*Hà Nội , tháng 07 năm 2024***NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

*Trà Vinh, ngày ….. tháng …… năm ……*

**Giáo viên hướng dẫn**

*(Ký tên và ghi rõ họ tên)*

**LỜI CẢM ƠN**

Em xin chân thành cảm ơn!

**MỤC LỤC**

[**MỞ ĐẦU** 9](#_Toc174484106)

[**A,Tính cấp thiết của đề tài/ Lí do chọn đề tài** 9](#_Toc174484107)

[**B, Mục đích nguyên cứu** 9](#_Toc174484108)

[**C, Phạm vi và đối tượng nguyên cứu** 10](#_Toc174484109)

[**D, Phương pháp nguyên cứu đề tài** 10](#_Toc174484110)

[**CHƯƠNG 1: CƠ SỞ LÝ THUYẾT** 12](#_Toc174484111)

[**1.1. Giới thiệu về học máy** 12](#_Toc174484112)

[**1.1.1. Khái niệm về học máy** 12](#_Toc174484113)

[**1.1.2. Phân loại các thuật toán học máy** 12](#_Toc174484114)

[**1.1.3. Ứng dụng của học máy trong y tế** 13](#_Toc174484115)

[**1.2. Phương pháp Naive Bayes** 13](#_Toc174484116)

[**1.2.1. Định nghĩa và nguyên lý hoạt động** 13](#_Toc174484117)

[**1.2.2. Công thức toán học cơ bản** 13](#_Toc174484118)

[**1.2.3. Các giả định đơn giản trong Naive Bayes** 14](#_Toc174484119)

[**1.2.4. Ưu và nhược điểm của Naive Bayes** 14](#_Toc174484120)

[**1.3. Ứng dụng Naive Bayes trong y tế** 14](#_Toc174484121)

[**1.3.1. Các nghiên cứu đã ứng dụng Naive Bayes** 14](#_Toc174484122)

[**1.3.2. Hiệu quả của Naive Bayes trong chuẩn đoán bệnh** 15](#_Toc174484123)

[**1.3.3. Các yếu tố ảnh hưởng đến hiệu quả của Naive Bayes** 15](#_Toc174484124)

[**1.4. Python Flask** 15](#_Toc174484125)

[**1.4.1. Khái niệm Python Flask** 15](#_Toc174484126)

[**1.4.2. Tại sao cần sử dụng Flask?** 16](#_Toc174484127)

[**1.4.3. Cách sử dụng Flask trong phát triển web** 16](#_Toc174484128)

[**1.4.4. Các khái niệm khác liên quan đến Flask** 18](#_Toc174484129)

[**1.4.5. Các sản phẩm minh họa về python Flask** 18](#_Toc174484130)

[**CHƯƠNG 2: CƠ SỞ THỰC TIỄN** 20](#_Toc174484131)

[**2.1. Các nghiên cứu liên quan** 20](#_Toc174484132)

[**2.1.1. Tổng quan các nghiên cứu về học máy trong y tế** 20](#_Toc174484133)

[**2.1.2. Các nghiên cứu ứng dụng Naive Bayes** 20](#_Toc174484134)

[**2.1.3. So sánh với các phương pháp học máy khác** 21](#_Toc174484135)

[**2.2. Công cụ và dữ liệu sử dụng** 22](#_Toc174484136)

[**2.2.1. Giới thiệu về các công cụ sử dụng** 22](#_Toc174484137)

[**2.2.2. Thu thập và xử lý dữ liệu** 23](#_Toc174484138)

[**CHƯƠNG 3: HIỆN THỰC MÔ HÌNH** 26](#_Toc174484139)

[**3.1. Cơ sở dữ liệu** 26](#_Toc174484140)

[**3.1.1. Cấu trúc dữ liệu ban đầu** 26](#_Toc174484141)

[**3.1.2. Các bước tiền xử lý** 26](#_Toc174484142)

[**3.2. Xây dựng mô hình Naive Bayes** 27](#_Toc174484143)

[**3.2.1. Chia dữ liệu thành tập huấn luyện và kiểm tra** 27](#_Toc174484144)

[**3.2.2. Mã hóa biến phân loại** 27](#_Toc174484145)

[**3.2.3. Huấn luyện mô hình** 27](#_Toc174484146)

[**3.2.4. Dự đoán và đánh giá mô hình** 27](#_Toc174484147)

[**3.3. Đánh giá mô hình** 28](#_Toc174484148)

[**3.3.1. Độ chính xác của mô hình** 28](#_Toc174484149)

[**3.3.2. So sánh với các phương pháp khác** 28](#_Toc174484150)

[**3.4. Xây dựng ứng dụng website áp dụng thực hiện mô hình** 29](#_Toc174484151)

[**3.4.1. Nghiệp vụ của website** 29](#_Toc174484152)

[**3.4.2. Xây dựng bộ dữ liệu của website** 29](#_Toc174484153)

[**3.4.3. Xây dựng các trang giao diện** 31](#_Toc174484154)

[**CHƯƠNG 4: HUẤN LUYỆN VÀ KẾT QUẢ SO SÁNH** 38](#_Toc174484155)

[**4.1. Kết quả huấn luyện mô hình** 38](#_Toc174484156)

[**4.1.1. Kết quả trên tập huấn luyện** 38](#_Toc174484157)

[**4.1.2. Kết quả trên tập kiểm tra** 38](#_Toc174484158)

[**4.2. So sánh với các giải pháp khác** 39](#_Toc174484159)

[**4.2.1. So sánh với Decision Tree** 39](#_Toc174484160)

[**4.2.2. So sánh với Random Forest** 40](#_Toc174484161)

[**4.2.3. So sánh với các phương pháp khác** 41](#_Toc174484162)

[**4.3. Kết quả sản phẩm website chuẩn đoán bệnh python flask** 43](#_Toc174484163)

[**4.3.1. Giao diện website** 43](#_Toc174484164)

[**CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 48](#_Toc174484165)

[**5.1. Kết luận** 48](#_Toc174484166)

[**5.2. Hướng phát triển** 48](#_Toc174484167)

[**TÀI LIỆU THAM KHẢO** 51](#_Toc174484168)

**DANH MỤC HÌNH ẢNH**

[Hình 1 : Hình ảnh banner ngôn ngữ python flask 17](#_Toc174484014)

[Hình 2 : hình ảnh dữ liệu mẫu trong excel 25](#_Toc174484015)

[Hình 3 : Hình ảnh minh họa dữ liệu gốc 31](#_Toc174484016)

[Hình 4 : Hinh ảnh dữ liệu người dung cung cấp 31](#_Toc174484017)

[Hình 5 : Hình ảnh dữ liệu phản hồi từ người dùng 32](#_Toc174484018)

[Hình 6 : Kết quả đạt được từ mô hình 40](#_Toc174484019)

[Hình 7 : Hình ảnh kết quả thuật toán Decision Tree 41](#_Toc174484020)

[Hình 8 : Hình ảnh kết quả thuật toán Random Forest 42](#_Toc174484021)

[Hình 9 : Hình ảnh kết quả thuật toán Support Vector Machine (SVM) 44](#_Toc174484022)

[Hình 10: Hình ảnh giao diện trang chủ wbesite 45](#_Toc174484023)

[Hình 11 : Trang hiển thị kết quả dự đoán 46](#_Toc174484024)

[Hình 12 : Trang thông báo kết quả dự đoán 46](#_Toc174484025)

[Hình 13 : Trang nhận phản hồi của người dùng 47](#_Toc174484026)

[Hình 14 : Trang nhận dữ liệu mới của người dùng 47](#_Toc174484027)

**DANH MỤC BẢNG BIỂU**

[Bảng 1: Bảng các phương pháp nguyên cứu đề tài 10](#_Toc174484091)

[Bảng 2 : Bảng phân loại các thuật toán học máy 12](#_Toc174484092)

[Bảng 3 : Bảng ưu nhược điểm của thuật toán naive bayes 14](#_Toc174484093)

[Bảng 4 : Bảng đặc điểm khác nhau giữa các phương pháp học máy 21](#_Toc174484094)

# **MỞ ĐẦU**

## **A,Tính cấp thiết của đề tài/ Lí do chọn đề tài**

Trong thời đại công nghệ số hiện nay, việc ứng dụng các phương pháp học máy vào lĩnh vực y tế không chỉ mang lại nhiều lợi ích về mặt kỹ thuật mà còn góp phần quan trọng trong việc nâng cao chất lượng chăm sóc sức khỏe. Các thuật toán học máy có khả năng phân tích lượng dữ liệu lớn và phức tạp, từ đó đưa ra các dự đoán và chuẩn đoán bệnh chính xác hơn so với các phương pháp truyền thống.

Bệnh lý, đặc biệt là các bệnh có triệu chứng phức tạp, thường gây khó khăn cho các bác sĩ trong việc chẩn đoán đúng và kịp thời. Với sự hỗ trợ của các mô hình học máy, quá trình này có thể được thực hiện nhanh chóng và chính xác hơn. Phương pháp Naive Bayes, một trong những thuật toán học máy đơn giản nhưng hiệu quả, có thể ứng dụng tốt trong việc phân loại và dự đoán các bệnh dựa trên các triệu chứng đầu vào.

Nhận thấy tính quan trọng của chủ đề này, nhóm em quyết định chọn đề tài để thực hiện nguyên cứu .Việc lựa chọn đề tài "Sử dụng phương pháp học máy Naive Bayes để chuẩn đoán bệnh dựa vào dấu hiệu đầu vào" không chỉ mang tính ứng dụng cao trong thực tiễn mà còn mở ra hướng nghiên cứu sâu rộng, giúp cải thiện và tối ưu hóa các phương pháp chuẩn đoán bệnh trong tương lai, đồng thời mang lại cho em kiến thức cũng như kinh nghiệm trong lĩnh vực máy học cũng như trí tuệ nhân tạo

## **B, Mục đích nguyên cứu**

Mục đích của nghiên cứu này là xây dựng một mô hình học máy sử dụng thuật toán Naive Bayes để chuẩn đoán các bệnh phổ biến dựa trên các triệu chứng đầu vào. Cụ thể:

1. Thiết kế và triển khai mô hình học máy Naive Bayes để phân loại và dự đoán bệnh dựa trên các triệu chứng đầu vào.
2. Đo lường độ chính xác, hiệu suất của mô hình trên tập dữ liệu thực tế và so sánh với các phương pháp khác.
3. Đề xuất phương pháp triển khai mô hình trong các hệ thống y tế thực tế để hỗ trợ các bác sĩ trong việc chuẩn đoán bệnh.

## **C, Phạm vi và đối tượng nguyên cứu**

\* Phạm vi nghiên cứu:

1. Nhóm em nghiên cứu tập trung vào việc sử dụng phương pháp học máy Naive Bayes để chuẩn đoán các bệnh phổ biến có triệu chứng dễ nhận biết.
2. Dữ liệu sử dụng trong nghiên cứu bao gồm các triệu chứng và chẩn đoán bệnh được em thu thập từ các tài liệu y khoa và nguồn dữ liệu thực tế.

\* Đối tượng nghiên cứu:

1. Các bệnh phổ biến như viêm họng, viêm dạ dày, cảm lạnh, viêm bàng quang, viêm khớp, viêm phổi, đau nửa đầu, dị ứng, và một số bệnh khác.
2. Các triệu chứng liên quan đến các bệnh này như đau họng, khó nuốt, ho kéo dài, tiêu chảy, buồn nôn, đau lưng, đau cơ, sổ mũi, ngứa da, phát ban, và nhiều triệu chứng khác.

## **D, Phương pháp nguyên cứu đề tài**

Để đạt được các mục tiêu nghiên cứu, các phương pháp nhóm em đã áp dụng như sau :

Bảng 1: Bảng các phương pháp nguyên cứu đề tài

|  |  |  |
| --- | --- | --- |
| **STT** | **Phương pháp** | **Nội dung chi tiết** |
| 1 | Thu thập dữ liệu | 1. Thu thập dữ liệu từ các tài liệu y khoa và nguồn dữ liệu thực tế. 2. Sử dụng tệp dữ liệu dataNCKH.xlsx chứa các triệu chứng và chẩn đoán bệnh tương ứng mà nhóm em đã thu thập. |
| 2 | Tiền xử lý dữ liệu | 1. Kiểm tra và xử lý các giá trị thiếu trong dữ liệu. 2. Mã hóa các biến phân loại sử dụng phương pháp One-Hot Encoding. |
| 3 | Xây dựng mô hình Naive Bayes | 1. Chia dữ liệu thành tập huấn luyện và tập kiểm tra. 2. Sử dụng thư viện Scikit-learn để xây dựng và huấn luyện mô hình Naive Bayes. |
| 4 | Đánh giá mô hình | 1. Đo lường độ chính xác của mô hình trên tập kiểm tra. 2. So sánh hiệu suất của mô hình Naive Bayes với các mô hình học máy khác như Decision Tree và Random Forest. |
| 5 | Ứng dụng mô hình trong thực tế | 1. Đề xuất phương pháp triển khai mô hình vào các hệ thống y tế để hỗ trợ bác sĩ trong việc chuẩn đoán bệnh. 2. Nhóm em đã xây dựng một website thực tế để đưa vào thực hiện việc chuẩn đoán bệnh |

# **CHƯƠNG 1: CƠ SỞ LÝ THUYẾT**

## **1.1. Giới thiệu về học máy**

### **1.1.1. Khái niệm về học máy**

Học máy (Machine Learning) là một lĩnh vực của trí tuệ nhân tạo (AI) tập trung vào việc phát triển các thuật toán và mô hình cho phép máy tính có khả năng tự học từ dữ liệu và cải thiện hiệu suất dự đoán mà không cần lập trình cụ thể. Máy học giúp máy tính phát hiện các mẫu trong dữ liệu, tự động nhận dạng, và đưa ra các dự đoán hoặc quyết định dựa trên dữ liệu đầu vào.

### **1.1.2. Phân loại các thuật toán học máy**

Các thuật toán học máy được phân loại như sau

Bảng 2 : Bảng phân loại các thuật toán học máy

|  |  |  |
| --- | --- | --- |
| **STT** | **Thuật toán học máy** | **Nội dung** |
| 1 | Học có giám sát (Supervised Learning) | Là học từ một tập dữ liệu đã gán nhãn, trong đó mô hình được huấn luyện để dự đoán nhãn của dữ liệu mới. Các thuật toán phổ biến bao gồm :   1. Hồi quy tuyến tính (Linear Regression) 2. Máy vector hỗ trợ (Support Vector Machines - SVM) 3. Cây quyết định (Decision Trees) 4. Naive Bayes |
| 2 | Học không giám sát (Unsupervised Learning) | Học từ dữ liệu không gán nhãn, nhằm tìm ra cấu trúc ẩn trong dữ liệu. Các thuật toán phổ biến bao gồm:   1. Phân cụm K-means (K-means Clustering) 2. Phân tích thành phần chính (Principal Component Analysis - PCA) 3. Mạng tự tổ chức (Self-Organizing Maps - SOM) |
| 3 | Học bán giám sát (Semi-supervised Learning) | Kết hợp dữ liệu có nhãn và không có nhãn để cải thiện độ chính xác của mô hình. Phương pháp này thường được sử dụng khi dữ liệu có nhãn khan hiếm hoặc tốn kém để thu thập. |
| 4 | Học tăng cường (Reinforcement Learning) | Mô hình học cách hành động thông qua tương tác với môi trường để tối ưu hóa một hàm thưởng cụ thể. Các ứng dụng bao gồm robot học, chơi game và hệ thống khuyến nghị. |

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

Trong thực tiễn cuộc sống thường ngày, nhóm em đã nguyên cứu và tìm hiểu được ứng dụng của học máy trong lĩnh vực y tế như sau :

* 1. Sử dụng các mô hình học máy để phân tích triệu chứng và lịch sử y tế của bệnh nhân nhằm dự đoán bệnh.
  2. Phân tích dữ liệu bệnh nhân để dự đoán phản ứng với các phương pháp điều trị khác nhau.
  3. Sử dụng học máy để theo dõi và dự đoán tình trạng sức khỏe của bệnh nhân theo thời gian.
  4. Phân tích dữ liệu dịch tễ học để phát hiện sớm các đợt bùng phát dịch bệnh.

## **1.2. Phương pháp Naive Bayes**

### **1.2.1. Định nghĩa và nguyên lý hoạt động**

Naive Bayes là một nhóm các thuật toán phân loại dựa trên định lý Bayes, với giả định đơn giản rằng các đặc trưng (feature) trong dữ liệu là độc lập với nhau. Naive Bayes được gọi là "naive" vì giả định tính độc lập này thường không chính xác trong thực tế, nhưng mô hình vẫn hoạt động hiệu quả trong nhiều trường hợp.

### **1.2.2. Công thức toán học cơ bản**

Phương pháp Naive Bayes dựa trên định lý Bayes, được biểu diễn như sau :

P ( C | X ) =

Trong đó :

* + P(C∣X) là xác suất lớp C khi biết đặc trưng X.
  + P(X∣C) là xác suất của đặc trưng X khi biết lớp C.
  + P(C) là xác suất tiên nghiệm của lớp C.
  + P(X) là xác suất của đặc trưng X.

### **1.2.3. Các giả định đơn giản trong Naive Bayes**

\* Giả định độc lập: Các đặc trưng đầu vào được giả định là độc lập với nhau, tức là sự hiện diện của một đặc trưng không ảnh hưởng đến sự hiện diện của đặc trưng khác.

\* Giả định đơn giản: Mỗi đặc trưng đóng góp một phần nhất định vào quyết định cuối cùng của mô hình

### **1.2.4. Ưu và nhược điểm của Naive Bayes**

Bảng 3 : Bảng ưu nhược điểm của thuật toán naive bayes

|  |  |
| --- | --- |
| **Ưu điểm** | **Nhược điểm** |
| 1. Đơn giản và dễ triển khai. 2. Hiệu quả trên các tập dữ liệu lớn. 3. Yêu cầu ít dữ liệu để huấn luyện mô hình. 4. Xử lý tốt các bài toán phân loại đa lớp. | 1. Giả định độc lập giữa các đặc trưng hiếm khi đúng trong thực tế, điều này có thể làm giảm độ chính xác của mô hình. 2. Không phù hợp với các đặc trưng có mối quan hệ phức tạp. 3. Nhạy cảm với dữ liệu hiếm gặp, mô hình có thể bị ảnh hưởng bởi các giá trị ngoại lệ |

## **1.3. Ứng dụng Naive Bayes trong y tế**

### **1.3.1. Các nghiên cứu đã ứng dụng Naive Bayes**

\* Nghiên cứu về chuẩn đoán bệnh tim: Một số nghiên cứu đã sử dụng Naive Bayes để phân tích dữ liệu bệnh nhân và chuẩn đoán bệnh tim với độ chính xác cao.

\* Nghiên cứu về phát hiện bệnh tiểu đường: Naive Bayes đã được áp dụng để dự đoán bệnh tiểu đường dựa trên các yếu tố như tuổi, giới tính, chỉ số khối cơ thể (BMI) và tiền sử gia đình.

\* Nghiên cứu về phân loại bệnh ung thư: Các mô hình Naive Bayes được sử dụng để phân loại các loại ung thư khác nhau dựa trên dữ liệu sinh học và hình ảnh y tế.

### **1.3.2. Hiệu quả của Naive Bayes trong chuẩn đoán bệnh**

\* Độ chính xác cao: Naive Bayes thường đạt độ chính xác cao trong các bài toán chuẩn đoán bệnh, đặc biệt là khi các đặc trưng đầu vào có tính độc lập tương đối.

\* Tốc độ xử lý nhanh: Mô hình Naive Bayes có thể xử lý dữ liệu nhanh chóng và hiệu quả, giúp giảm thời gian chuẩn đoán bệnh.

\* Khả năng mở rộng: Naive Bayes có thể dễ dàng mở rộng và áp dụng cho nhiều loại bệnh khác nhau chỉ với việc thu thập thêm dữ liệu và điều chỉnh mô hình.

### **1.3.3. Các yếu tố ảnh hưởng đến hiệu quả của Naive Bayes**

\* Chất lượng dữ liệu: Dữ liệu chất lượng cao với các đặc trưng rõ ràng và đầy đủ sẽ cải thiện hiệu quả của mô hình.

\* Số lượng dữ liệu: Mô hình Naive Bayes cần đủ dữ liệu để huấn luyện và đánh giá, số lượng dữ liệu lớn sẽ giúp mô hình học tốt hơn và đưa ra dự đoán chính xác hơn.

\* Cách mã hóa dữ liệu: Việc mã hóa các biến phân loại một cách hiệu quả (ví dụ như One-Hot Encoding) sẽ giúp mô hình học tốt hơn và cải thiện độ chính xác.

## **1.4. Python Flask**

### **1.4.1. Khái niệm Python Flask**

Flask là một micro web framework được viết bằng ngôn ngữ lập trình Python. Được tạo ra bởi Armin Ronacher, Flask không giống như các framework web khác, như Django, bởi vì nó không bao gồm các tính năng như ORM (Object-Relational Mapping) hoặc hệ thống xác thực sẵn có. Thay vào đó, Flask chỉ cung cấp các thành phần cơ bản cần thiết để phát triển ứng dụng web, cho phép lập trình viên tự do tích hợp các thư viện hoặc công cụ mà họ cần. Flask sử dụng Jinja2 làm engine cho template và Werkzeug làm bộ công cụ WSGI.



Hình 1 : Hình ảnh banner ngôn ngữ python flask

### **1.4.2. Tại sao cần sử dụng Flask?**

Flask là một framework "micro" bởi vì nó không bao gồm các thành phần không cần thiết. Điều này giúp Flask trở nên linh hoạt và dễ dàng tùy chỉnh cho các nhu cầu cụ thể. Các lập trình viên có thể lựa chọn các công cụ và thư viện mà họ muốn sử dụng thay vì bị ràng buộc bởi một hệ sinh thái có sẵn.

Flask rất dễ hiểu và có cú pháp đơn giản, làm cho nó trở thành lựa chọn tuyệt vời cho những người mới bắt đầu lập trình web. Nó cũng cho phép lập trình viên nhanh chóng triển khai và thử nghiệm các ý tưởng mới.

Mặc dù Flask là một micro-framework, nhưng nó rất dễ dàng để mở rộng bằng cách thêm các extension. Có nhiều extension phổ biến như Flask-SQLAlchemy (ORM), Flask-WTF (form handling), Flask-Login (quản lý người dùng), giúp tăng cường khả năng của Flask mà không làm mất đi sự đơn giản của nó.

Flask có tài liệu chi tiết và dễ hiểu, cộng với một cộng đồng lập trình viên rộng lớn, nên việc học và sử dụng Flask trở nên dễ dàng và có nhiều nguồn hỗ trợ.

### **1.4.3. Cách sử dụng Flask trong phát triển web**

Cài đặt Flask: Flask được cài đặt dễ dàng thông qua pip, công cụ quản lý gói của Python. Chỉ cần mở terminal và gõ lệnh sau:

|  |
| --- |
| pip install Flask |

\* Tạo ứng dụng Flask cơ bản: Một ứng dụng Flask cơ bản có thể được tạo ra chỉ với vài dòng mã

|  |
| --- |
| from flask import Flask  app = Flask(\_\_name\_\_)  @app.route('/')  def home():  return 'Welcome to Flask!'  if \_\_name\_\_ == '\_\_main\_\_':  app.run(debug=True) |

Giải thích mã nguồn:

1. Flask được import từ thư viện flask.
2. app = Flask(\_\_name\_\_) tạo một instance của ứng dụng Flask.
3. @app.route('/') là một decorator để chỉ định URL nào sẽ kích hoạt hàm home.
4. def home(): định nghĩa hàm sẽ chạy khi người dùng truy cập vào URL /.
5. app.run(debug=True) khởi động server Flask với chế độ debug để dễ dàng theo dõi lỗi.

\* Xử lý yêu cầu HTTP: Flask hỗ trợ nhiều phương thức HTTP như GET, POST, PUT, DELETE, giúp ứng dụng có thể xử lý các loại yêu cầu khác nhau

Ví dụ về xử lý yêu cầu POST:

|  |
| --- |
| from flask import request  @app.route('/submit', methods=['POST'])  def submit():  data = request.form['data']  return f'Received: {data}' |

Giải thích:

1. request.form dùng để truy cập dữ liệu được gửi qua phương thức POST từ một form HTML.
2. URL /submit sẽ chấp nhận các yêu cầu POST và xử lý dữ liệu gửi lên.

### **1.4.4. Các khái niệm khác liên quan đến Flask**

WSGI: Flask là một WSGI application, nơi WSGI (Web Server Gateway Interface) là một tiêu chuẩn để triển khai các ứng dụng web bằng Python. Flask dựa trên Werkzeug, một thư viện WSGI, để xử lý các yêu cầu HTTP.

Jinja2: Jinja2 là engine template mặc định của Flask, cho phép bạn viết các trang HTML với các biến động và logic điều kiện.

Middleware: Flask cho phép bạn thêm middleware để xử lý các yêu cầu trước khi chúng được xử lý bởi các hàm chính của ứng dụng. Middleware có thể được sử dụng để ghi log, xác thực, hoặc sửa đổi các yêu cầu và phản hồi.

### **1.4.5. Các sản phẩm minh họa về python Flask**

Xây dựng một blog cá nhân:

* Tạo ứng dụng Flask với các chức năng cơ bản như đăng bài, chỉnh sửa bài viết, và hiển thị danh sách bài viết.
* Sử dụng SQLite để lưu trữ dữ liệu bài viết và Flask-SQLAlchemy để quản lý cơ sở dữ liệu.
* Sử dụng Jinja2 để render các trang HTML với nội dung động từ cơ sở dữ liệu.

API cho ứng dụng di động:

* Xây dựng một RESTful API với Flask để phục vụ các yêu cầu từ ứng dụng di động.
* Sử dụng Flask-RESTful để định nghĩa các route cho các chức năng như đăng ký, đăng nhập, và lấy dữ liệu.
* Tích hợp với JWT (JSON Web Token) để quản lý xác thực và phiên làm việc.

Ứng dụng quản lý công việc:

* Tạo ứng dụng Flask cho phép người dùng thêm, sửa, xóa các công việc cần làm.
* Sử dụng Flask-Login để quản lý người dùng và quyền truy cập.
* Sử dụng AJAX để cập nhật danh sách công việc mà không cần tải lại trang.

# **CHƯƠNG 2: CƠ SỞ THỰC TIỄN**

## **2.1. Các nghiên cứu liên quan**

### **2.1.1. Tổng quan các nghiên cứu về học máy trong y tế**

Học máy đã có nhiều ứng dụng rộng rãi trong lĩnh vực y tế, từ chuẩn đoán bệnh, dự đoán kết quả điều trị, quản lý bệnh nhân, đến phát hiện bệnh dịch. Các nghiên cứu tập trung vào việc sử dụng các thuật toán học máy để phân tích dữ liệu y tế lớn, phát hiện các mẫu và mối quan hệ ẩn trong dữ liệu, từ đó đưa ra các dự đoán chính xác và hỗ trợ các quyết định y tế

Ví dụ:

* Chuẩn đoán hình ảnh y tế: Các mô hình học sâu (deep learning) được sử dụng để phân tích hình ảnh y tế như X-quang, MRI, CT scan nhằm phát hiện sớm các bệnh như ung thư, tổn thương não, và bệnh tim.
* Dự đoán kết quả điều trị: Sử dụng học máy để phân tích dữ liệu bệnh nhân và dự đoán phản ứng với các phương pháp điều trị, giúp cá nhân hóa kế hoạch điều trị.
* Quản lý bệnh nhân: Các hệ thống học máy được áp dụng để theo dõi tình trạng sức khỏe của bệnh nhân theo thời gian, phát hiện sớm các biến chứng và đề xuất can thiệp kịp thời.

### **2.1.2. Các nghiên cứu ứng dụng Naive Bayes**

Naive Bayes là một trong những thuật toán được sử dụng phổ biến trong các nghiên cứu y tế do tính đơn giản và hiệu quả của nó. Các ứng dụng Naive Bayes trong y tế bao gồm :

1. Chuẩn đoán bệnh tim: Nghiên cứu sử dụng Naive Bayes để phân tích các yếu tố nguy cơ như huyết áp, cholesterol, và tiền sử gia đình nhằm chuẩn đoán bệnh tim với độ chính xác cao.
2. Phát hiện bệnh tiểu đường: Naive Bayes đã được áp dụng để dự đoán bệnh tiểu đường dựa trên các đặc điểm nhân khẩu học và y tế như tuổi, giới tính, chỉ số BMI, và tiền sử bệnh.
3. Phân loại ung thư: Sử dụng Naive Bayes để phân loại các loại ung thư khác nhau dựa trên dữ liệu sinh học và hình ảnh y tế, giúp cải thiện khả năng phát hiện sớm và điều trị ung thư.

### **2.1.3. So sánh với các phương pháp học máy khác**

\* Điểm giống nhau giữa các thuật toán máy học Naive Bayes, Decision Tree, Random Forest, và SVM:

1. Mục đích chung: Tất cả đều là các thuật toán học máy được sử dụng cho các bài toán phân loại (classification) và hồi quy (regression).
2. Yêu cầu về dữ liệu: Các thuật toán này đều yêu cầu dữ liệu đã được xử lý và làm sạch trước khi huấn luyện mô hình.
3. Khả năng học từ dữ liệu: Chúng đều có khả năng học từ dữ liệu để tạo ra các mô hình dự đoán và đưa ra các dự đoán cho các mẫu mới.
4. Ứng dụng rộng rãi: Tất cả các thuật toán này đều được áp dụng rộng rãi trong nhiều lĩnh vực khác nhau, bao gồm y tế, tài chính, tiếp thị, và nhiều lĩnh vực khác.
5. Hỗ trợ trong Scikit-learn: Cả bốn thuật toán đều được tích hợp và dễ sử dụng trong thư viện Scikit-learn của Python.

\* Điểm khác nhau giữa Naive Bayes, Decision Tree, Random Forest, và SVM

Bảng 4 : Bảng đặc điểm khác nhau giữa các phương pháp học máy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Đặc điểm** | **Naive Bayes** | **Decision Tree** | **Random Forest** | **SVM** |
| **Nguyên lý hoạt động** | Dựa trên định lý Bayes và giả định độc lập giữa các đặc trưng | Tạo cây quyết định dựa trên việc chia dữ liệu theo các đặc trưng tốt nhất | Kết hợp nhiều cây quyết định bằng phương pháp bagging | Tìm kiếm siêu phẳng phân chia các lớp một cách tối ưu |
| **Giả định** | Giả định các đặc trưng độc lập | Không có giả định cụ thể về dữ liệu | Không có giả định cụ thể về dữ liệu | Giả định dữ liệu có thể phân tách tuyến tính (với kernel tuyến tính) hoặc không tuyến tính (với kernel phi tuyến) |
| **Ưu điểm** | Đơn giản, nhanh, hiệu quả với dữ liệu lớn | Trực quan, dễ hiểu, xử lý tốt dữ liệu không tuyến tính | Độ chính xác cao, giảm overfitting | Hiệu quả cho phân loại với biên độ rộng giữa các lớp |
| **Nhược điểm** | Giả định độc lập thường không đúng trong thực tế | Dễ bị overfitting nếu không cắt tỉa cây (pruning) | Phức tạp, tốn tài nguyên tính toán | Khó triển khai, tốn thời gian huấn luyện với dữ liệu lớn |
| **Độ phức tạp** | Thấp | Trung bình | Cao | Cao |
| **Khả năng mở rộng** | Tốt với dữ liệu lớn | Trung bình | Tốt nhưng yêu cầu tài nguyên tính toán | Khó mở rộng với dữ liệu rất lớn |
| **Tính giải thích** | Cao, dễ hiểu | Rất cao, dễ giải thích kết quả | Trung bình, khó giải thích với nhiều cây | Thấp, khó giải thích do tính phức tạp của mô hình |

## **2.2. Công cụ và dữ liệu sử dụng**

### **2.2.1. Giới thiệu về các công cụ sử dụng**

#### **2.2.1.1. Python**

Python là một ngôn ngữ lập trình mạnh mẽ và linh hoạt, được sử dụng rộng rãi trong khoa học dữ liệu và học máy nhờ vào cú pháp đơn giản và thư viện phong phú

#### **2.2.1.2. Pandas**

Pandas là thư viện mạnh mẽ để xử lý và phân tích dữ liệu, cung cấp các cấu trúc dữ liệu và công cụ cần thiết để thao tác với dữ liệu dạng bảng, giúp việc tiền xử lý và phân tích dữ liệu trở nên dễ dàng.

#### **2.2.1.3. Scikit-learn**

Scikit-learn là một thư viện học máy trong Python, cung cấp các công cụ hiệu quả cho mô hình hóa và phân tích dữ liệu. Thư viện này hỗ trợ các thuật toán học máy phổ biến, bao gồm Naive Bayes, Decision Tree, Random Forest, và SVM.

#### **2.2.1.4. Category\_encoders**

Category\_encoders là một thư viện trong Python giúp mã hóa các biến phân loại, hỗ trợ nhiều phương pháp mã hóa khác nhau như One-Hot Encoding, Target Encoding, và Binary Encoding.

#### **2.2.1.5. Matplotlib và Seaborn**

Matplotlib và Seaborn là các thư viện trực quan hóa dữ liệu trong Python. Matplotlib cung cấp các công cụ cơ bản để tạo biểu đồ, trong khi Seaborn xây dựng trên Matplotlib và cung cấp giao diện thân thiện hơn để tạo ra các biểu đồ thống kê phức tạp.

### **2.2.2. Thu thập và xử lý dữ liệu**

#### **2.2.2.1. Thu thập dữ liệu từ các nguồn y khoa**

Dữ liệu được thu thập từ các nguồn y khoa uy tín, bao gồm:

1. Hồ sơ bệnh án của bệnh nhân từ các bệnh viện và phòng khám.
2. Các tài liệu y khoa và nghiên cứu đã công bố.
3. Các cơ sở dữ liệu y tế trực tuyến.

#### **2.2.2.2. Mô tả và phân tích dữ liệu**

Dữ liệu mẫu bao gồm các triệu chứng bệnh và chẩn đoán bệnh tương ứng. Một mẫu dữ liệu bao gồm 4 thuộc tính bệnh và 1 cột kết quả chuẩn đoán bệnh :

1. symptom\_1: Triệu chứng thứ nhất (ví dụ: đau họng).
2. symptom\_2: Triệu chứng thứ hai (ví dụ: khó nuốt).
3. symptom\_3: Triệu chứng thứ ba (ví dụ: ho kéo dài).
4. symptom\_4: Triệu chứng thứ tư (ví dụ: khạc đàm dai dẳng).
5. diagnose: Chẩn đoán bệnh (ví dụ: viêm họng).

A screenshot of a computer

Description automatically generated

Hình 2 : hình ảnh dữ liệu mẫu trong excel

#### **2.2.2.3. Tiền xử lý dữ liệu**

Tiền xử lý dữ liệu là một bước quan trọng trong quy trình học máy, giúp làm sạch và chuẩn bị dữ liệu cho việc huấn luyện mô hình. Các bước tiền xử lý dữ liệu bao gồm:

\* Kiểm tra và xử lý giá trị thiếu

* Kiểm tra giá trị thiếu: Đầu tiên, cần xác định các giá trị thiếu trong dữ liệu. Trong Python, có thể sử dụng phương thức isnull() và sum() của Pandas để đếm số lượng giá trị thiếu trong mỗi cột.
* Xử lý giá trị thiếu: Có nhiều cách để xử lý giá trị thiếu, bao gồm
  + Loại bỏ các hàng hoặc cột có giá trị thiếu: Sử dụng phương thức dropna() của Pandas.
  + Điền giá trị thiếu: Sử dụng phương thức fillna() để điền giá trị thiếu bằng các giá trị như trung bình, trung vị, hoặc một giá trị cố định.

\* Mã hóa các biến phân loại

One-Hot Encoding: Biến các giá trị phân loại thành các biến nhị phân (0 và 1). Điều này giúp mô hình học máy xử lý dữ liệu phân loại một cách hiệu quả hơn. Thư viện category\_encoders trong Python hỗ trợ nhiều phương pháp mã hóa khác nhau, bao gồm One-Hot Encoding.

|  |
| --- |
| import category\_encoders as ce  categorical = ['symptom\_1', 'symptom\_2', 'symptom\_3', 'symptom\_4']  encoder = ce.OneHotEncoder(cols=categorical)  X\_train = encoder.fit\_transform(X\_train)  X\_test = encoder.transform(X\_test) |

\* Chuẩn hóa dữ liệu

Standardization (Chuẩn hóa theo chuẩn): Chuẩn hóa các đặc trưng để có giá trị trung bình là 0 và độ lệch chuẩn là 1. Điều này giúp các thuật toán học máy hội tụ nhanh hơn và chính xác hơn.

|  |
| --- |
| from sklearn.preprocessing import StandardScaler  scaler = StandardScaler()  X\_train = scaler.fit\_transform(X\_train)  X\_test = scaler.transform(X\_test) |

\* Chia dữ liệu thành tập huấn luyện và tập kiểm tra

\* Xử lý ngoại lệ (outliers)

* Phát hiện và loại bỏ các ngoại lệ: Ngoại lệ có thể ảnh hưởng đến hiệu quả của mô hình. Có nhiều phương pháp để phát hiện ngoại lệ, bao gồm sử dụng các biểu đồ như Box Plot hoặc sử dụng các thống kê như Z-score.

\* Chuyển đổi dữ liệu

\* Tạo ra các đặc trưng mới (feature engineering)

# **CHƯƠNG 3: HIỆN THỰC MÔ HÌNH**

## **3.1. Cơ sở dữ liệu**

### **3.1.1. Cấu trúc dữ liệu ban đầu**

Dữ liệu ban đầu gồm các cột biểu thị các triệu chứng và chẩn đoán bệnh,

1. symptom\_1: Triệu chứng thứ nhất (ví dụ: đau họng).
2. symptom\_2: Triệu chứng thứ hai (ví dụ: khó nuốt).
3. symptom\_3: Triệu chứng thứ ba (ví dụ: ho kéo dài).
4. symptom\_4: Triệu chứng thứ tư (ví dụ: khạc đàm dai dẳng).
5. diagnose: Chẩn đoán bệnh (ví dụ: viêm họng).

### **3.1.2. Các bước tiền xử lý**

#### **3.1.2.1. Kiểm tra và xử lý giá trị thiếu**

\* Kiểm tra giá trị thiếu

|  |
| --- |
| df.isnull().sum() |

\* Xử lý giá trị thiếu

|  |
| --- |
| *//Loại bỏ các hàng hoặc cột có giá trị thiếu*  df.dropna(axis=0, inplace=True) # Loại bỏ các hàng có giá trị thiếu  df.dropna(axis=1, inplace=True) # Loại bỏ các cột có giá trị thiếu  *//Điền giá trị thiếu bằng giá trị trung bình hoặc giá trị cố định*  df.fillna(df.mean(), inplace=True) # Điền giá trị thiếu bằng giá trị trung bình của cột  df.fillna('unknown', inplace=True) # Điền giá trị thiếu bằng giá trị cố định |

#### **3.1.2.2. Mã hóa các biến phân loại**

One-Hot Encoding: Biến các giá trị phân loại thành các biến nhị phân (0 và 1 )

|  |
| --- |
| import category\_encoders as ce  categorical = ['symptom\_1', 'symptom\_2', 'symptom\_3', 'symptom\_4']  encoder = ce.OneHotEncoder(cols=categorical)  X = encoder.fit\_transform(df.drop(['diagnose'], axis=1))  y = df['diagnose'] |

## **3.2. Xây dựng mô hình Naive Bayes**

### **3.2.1. Chia dữ liệu thành tập huấn luyện và kiểm tra**

Chia dữ liệu: Sử dụng thư viện train\_test\_split từ sklearn để chia dữ liệu thành tập huấn luyện và kiểm tra:

|  |
| --- |
| from sklearn.model\_selection import train\_test\_split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0) |

### **3.2.2. Mã hóa biến phân loại**

Áp dụng mã hóa cho tập huấn luyện và kiểm tra

|  |
| --- |
| X\_train = encoder.fit\_transform(X\_train)  X\_test = encoder.transform(X\_test) |

### **3.2.3. Huấn luyện mô hình**

Khởi tạo và huấn luyện mô hình Naive Bayes

|  |
| --- |
| from sklearn.naive\_bayes import GaussianNB  gnb = GaussianNB()  gnb.fit(X\_train, y\_train) |

### **3.2.4. Dự đoán và đánh giá mô hình**

Dự đoán trên tập kiểm tra

|  |
| --- |
| y\_pred = gnb.predict(X\_test) |

Đánh giá độ chính xác của mô hình

|  |
| --- |
| from sklearn.metrics import accuracy\_score  accuracy = accuracy\_score(y\_test, y\_pred)  print('Độ chính xác của mô hình: {0:0.4f} %'. format(accuracy \* 100)) |

## **3.3. Đánh giá mô hình**

### **3.3.1. Độ chính xác của mô hình**

In ra độ chính xác của mô hình

|  |
| --- |
| training\_score = gnb.score(X\_train, y\_train) \* 100  test\_score = gnb.score(X\_test, y\_test) \* 100  print('Điểm training: {:.4f} %'.format(training\_score))  print('Điểm test: {:.4f} %'.format(test\_score)) |

### **3.3.2. So sánh với các phương pháp khác**

So sánh độ chính xác với Decision Tree, Random Forest, và SVM

|  |
| --- |
| from sklearn.tree import DecisionTreeClassifier  from sklearn.ensemble import RandomForestClassifier  from sklearn.svm import SVC  # Decision Tree  dt = DecisionTreeClassifier()  dt.fit(X\_train, y\_train)  dt\_pred = dt.predict(X\_test)  dt\_accuracy = accuracy\_score(y\_test, dt\_pred)  # Random Forest  rf = RandomForestClassifier()  rf.fit(X\_train, y\_train)  rf\_pred = rf.predict(X\_test)  rf\_accuracy = accuracy\_score(y\_test, rf\_pred)  # SVM  svm = SVC()  svm.fit(X\_train, y\_train)  svm\_pred = svm.predict(X\_test)  svm\_accuracy = accuracy\_score(y\_test, svm\_pred)  print('Độ chính xác của Decision Tree: {:.4f} %'.format(dt\_accuracy \* 100))  print('Độ chính xác của Random Forest: {:.4f} %'.format(rf\_accuracy \* 100))  print('Độ chính xác của SVM: {:.4f} %'.format(svm\_accuracy \* 100)) |

## **3.4. Xây dựng ứng dụng website áp dụng thực hiện mô hình**

### **3.4.1. Nghiệp vụ của website**

Website chuẩn đoán bệnh được xây dựng bằng ngôn ngữ python flask sẽ có những tác vụ sau :

1. Nhận dữ liệu từ người dùng sau đó chuẩn đoán bệnh của họ
2. Nhận dữ liệu mới tư người dùng cung cấp
3. Gợi ý biện pháp chữa trị cho bệnh chuẩn đoán
4. Đánh giá điểm chuẩn xác, điểm xây dựng và test của mô hình dự đoán
5. Nhận phản hồi từ người dùng
6. Tạo liên hệ để người dùng có thể contact với admin

### **3.4.2. Xây dựng bộ dữ liệu của website**

Dựa trên các yêu cầu nghiệp vụ bài toán, website cần lưu trữ các data sau :

1. Dữ liệu bệnh và kết quả chuẩn đoán
2. Dữ liệu mới người dùng cung cấp
3. Phản hổi từ người dùng

Bảng bảng dữ liệu excel sẽ có các trường thông tin như sau :

Bảng data bệnh :

1. symptom\_1 : Triệu chứng bệnh thứ 1
2. symptom\_2 : Triệu chứng bệnh thứ 2
3. symptom\_3 : Triệu chứng bệnh thứ 3
4. symptom\_4 : Triệu chứng bênh thứ 4 :
5. diagnose : kết quả bệnh chuẩn đoán

****

Hình 3 : Hình ảnh minh họa dữ liệu gốc

Bảng data nhận dữ liệu từ người dùng ( bảng này để nhận dữ liệu và kiểm tra xem dữ liệu có hợp lệ không sau đó mới đưa vào database dữ liệu bên trên )

1. email : email người dùng phản hồi
2. symptom\_1 : Triệu chứng bệnh thứ 1
3. symptom\_2 : Triệu chứng bệnh thứ 2
4. symptom\_3 : Triệu chứng bệnh thứ 3
5. symptom\_4 : Triệu chứng bênh thứ 4 :
6. diagnose : kết quả bệnh chuẩn đoán

A screenshot of a computer

Description automatically generated

Hình 4 : Hinh ảnh dữ liệu người dung cung cấp

Bảng feedback từ người dùng :

1. Email
2. Phản hồi

A screenshot of a computer

Description automatically generated

Hình 5 : Hình ảnh dữ liệu phản hồi từ người dùng

### **3.4.3. Xây dựng các trang giao diện**

Trang app.py để thực hiện các thao tác chuẩn đoán cũng như tạo server khởi chạy website

|  |
| --- |
| from flask import Flask, render\_template, request  import pandas as pd  import category\_encoders as ce  from sklearn.naive\_bayes import GaussianNB  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import accuracy\_score  import openpyxl  app = Flask(\_\_name\_\_)  # Đọc dữ liệu từ file Excel  df = pd.read\_excel("./data//dataNCKH.xlsx")  # df.head()  # Đặt tên cột  col\_names = ['symptom\_1', 'symptom\_2', 'symptom\_3', 'symptom\_4', 'diagnose']  df.columns = col\_names  # Lấy biến phân loại  categorical = ['symptom\_1', 'symptom\_2', 'symptom\_3', 'symptom\_4']  # Tạo encoder  encoder = ce.OneHotEncoder(cols=categorical)  # Thực hiện one-hot encoding trên toàn bộ dữ liệu  df\_encoded = encoder.fit\_transform(df)  # Tạo và huấn luyện mô hình Naive Bayes  X = df\_encoded.drop(['diagnose'], axis=1)  y = df\_encoded['diagnose']  # Chia dữ liệu thành tập huấn luyện và tập kiểm tra  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  gnb = GaussianNB()  gnb.fit(X\_train, y\_train)    # Định nghĩa route cho trang chủ  @app.route('/')  def index():      return render\_template('index.html')  # Định nghĩa route cho việc nhận dữ liệu từ form và thực hiện dự đoán  @app.route('/predict', methods=['POST'])  def predict():      symptoms = [request.form['symptom\_1'], request.form['symptom\_2'],                  request.form['symptom\_3'], request.form['symptom\_4']]        # Lấy cột 'diagnose' từ DataFrame gốc      user\_data = df[['symptom\_1', 'symptom\_2', 'symptom\_3', 'symptom\_4', 'diagnose']].copy()        # Thêm dữ liệu người dùng vào DataFrame      user\_data.loc[len(user\_data)] = [symptoms[0], symptoms[1], symptoms[2], symptoms[3], None]        # Áp dụng encoder đã học từ toàn bộ dữ liệu      user\_data\_encoded = encoder.transform(user\_data)      # Dự đoán bằng mô hình đã huấn luyện      prediction = gnb.predict(user\_data\_encoded.iloc[-1, :-1].values.reshape(1, -1))      # Tính toán độ chính xác, điểm training và điểm test      accuracy = accuracy\_score(y\_test, gnb.predict(X\_test)) \* 100      training\_score = gnb.score(X\_train, y\_train) \* 100      test\_score = gnb.score(X\_test, y\_test) \* 100        accuracy = round(accuracy, 2)      training\_score = round(training\_score, 2)      test\_score = round(test\_score, 2)      return render\_template('result.html', prediction=prediction[0], accuracy=accuracy, training\_score=training\_score, test\_score=test\_score)  @app.route('/save-feedback', methods=['POST'])  def save\_feedback():      email = request.form.get('email')      feedback = request.form.get('feedback')      # Đọc dữ liệu từ file Excel (feedback.xlsx)      workbook = openpyxl.load\_workbook('./data/feedback.xlsx')      sheet = workbook.active      # Tìm hàng cuối cùng trong sheet      last\_row = sheet.max\_row      next\_row = last\_row + 1      # Thêm dữ liệu mới vào sheet      sheet.cell(row=next\_row, column=1, value=email)      sheet.cell(row=next\_row, column=2, value=feedback)      # Ghi lại dữ liệu vào file Excel      workbook.save('./data/feedback.xlsx')      return render\_template('index.html')  @app.route('/add-data', methods=['POST'])  def save\_data():      email = request.form.get('email')      symptom\_1 =request.form.get('symptom\_1')      symptom\_2 =request.form.get('symptom\_2')      symptom\_3 =request.form.get('symptom\_3')      symptom\_4 =request.form.get('symptom\_4')      diagnose = request.form.get('diagnose')      # Đọc dữ liệu từ file Excel (feedback.xlsx)      workbook = openpyxl.load\_workbook('./data/user\_data.xlsx')      sheet = workbook.active      # Tìm hàng cuối cùng trong sheet      last\_row = sheet.max\_row      next\_row = last\_row + 1      # Thêm dữ liệu mới vào sheet      sheet.cell(row=next\_row, column=1, value=email)      sheet.cell(row=next\_row, column=2, value=symptom\_1)      sheet.cell(row=next\_row, column=3, value=symptom\_2)      sheet.cell(row=next\_row, column=4, value=symptom\_3)      sheet.cell(row=next\_row, column=5, value=symptom\_4)      sheet.cell(row=next\_row, column=6, value=diagnose)      # Ghi lại dữ liệu vào file Excel      workbook.save('./data/user\_data.xlsx')      return render\_template('index.html')  if \_\_name\_\_ == '\_\_main\_\_':      app.run(debug=True) |

Trang index là trang chủ của người dùng bao gồm nhận dữ liệu người dùng nhập để chuẩn đoán và nhận dữ liệu mới từ người dùng cung cấp

|  |
| --- |
| <!----------------------- form dự đoán ------------------->          <form action="/predict" method="post">            <div class="form-group">              <label for="symptom\_1">Triệu chứng 1 <span style="color: red;">\*</span></label>              <input class="form-control" name="symptom\_1" required />            </div>            <div class="form-group">              <label for="symptom\_1">Nhập triệu chứng 2 <span style="color: red;">\*</span></label>              <input class="form-control" name="symptom\_2" required />            </div>            <div class="form-group">              <label for="symptom\_1">Nhập triệu chứng 3 <span style="color: red;">\*</span></label>              <input class="form-control" name="symptom\_3" required />            </div>            <div class="form-group">              <label for="symptom\_1">Nhập triệu chứng 4 <span style="color: red;">\*</span></label>              <input class="form-control" name="symptom\_4" required />            </div>            <br />            <div style="display: flex;justify-content: space-between;">              <button type="submit" class="btn btn-success">Dự đoán</button>              <button type="button" class="btn btn-light " data-bs-toggle="modal" data-bs-target="#Modal">                Đóng góp dữ liệu              </button>            </div>          </form> |

|  |
| --- |
| <!------------------------------------ Modal lấy dữ liệu   ------------------>      <div class="modal fade " id="Modal" tabindex="-1" aria-labelledby="ModalLabel" aria-hidden="true">        <div class="modal-dialog modal-dialog-centered">          <div class="modal-content">            <div class="modal-header">              <h1 class="modal-title fs-5" id="ModalLabel">Dữ liệu bệnh lí</h1>              <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>            </div>            <!------------------------------- phần lấy thông tin ------------------------>            <div class="modal-body">              <form method="post" action="/add-data">                <p class="alert alert-light" style="font-size: 14px;">                  ⚠️ Hãy gửi cho chúng tôi 4 triệu chứng bạn mắc phải kèm bệnh bạn mắc phải đã được xác nhận bởi bác sĩ !!                </p>                <div class="mb-3">                  <label for="email" class="form-label">Địa chỉ email</label>                  <input type="email" class="form-control" id="email" name="email" placeholder="name@.com" required>                </div>                <div class="form-group">                  <input class="form-control" id="symptom\_1" name="symptom\_1" required  placeholder="Nhập triệu chứng 1" />                </div>                <div class="form-group">                  <br/>                  <input class="form-control" id="symptom\_2" name="symptom\_2" required placeholder="Nhập triệu chứng 2"/>                </div>                <div class="form-group">                  <br />                  <input class="form-control" id="symptom\_3" name="symptom\_3" required placeholder="Nhập triệu chứng 3"/>                </div>                <div class="form-group">                  <br />                  <input class="form-control" id="symptom\_4" name="symptom\_4" required placeholder="Nhập triệu chứng 4"/>                </div>                <div class="form-group">                  <br/>                  <input class="form-control" id="diagnose" name="diagnose" required  placeholder="Nhập bệnh lí của bạn"/>                </div>                <div class="modal-footer">                  <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Đóng</button>                  <button type="submit" class="btn btn-primary" onclick="success()">Gửi</button>                </div>              </form>            </div>          </div>        </div>      </div> |

Trang result.html để hiển thị kết quả cho người dùng

|  |
| --- |
| <!----------------------- phần xử lí xem kết quả ------------------->              <div class="alert alert-success ketqua" role="alert">                  <p class="xemketqua\_p">                      <button class="btn btn-outline-success btshow" onclick="predict();"> Xem kết quả</button>                  </p>                  <p id="Prediction" class="result">{{ prediction }}</p>                  <p id="accuracy">Độ chính xác chuẩn đoán: {{ accuracy }} %</p>                  <p id="training\_score">                      Điểm training mô hình: {{ training\_score }} %                  </p>                  <p id="test\_score">Điểm test đánh giá mô hình: {{ test\_score }} %</p>              </div> |

|  |
| --- |
| <!----------------------- phần biện pháp và quay lại ------------------->              <div class="footerCard">                  <button id="searchButton" onclick="chuatri()" class="btn btn-outline-success ">Biện pháp chữa trị</button>                  <a href="/">                      <div class="button d-flex justify-content-between">                          <button class="btn btn-success thulai">                              Thử lại                          </button>                      </div>                  </a>              </div> |

|  |
| --- |
| <!----------------------- phần liên hệ và phản hồi ------------------->          <div class="card-footer ">              <div class="contact">                  <p>contact :</p>                  <div id="root1"></div>              </div>              <div class="feedback">                  <button type="button" class="btn btn-light " data-bs-toggle="modal" data-bs-target="#Modal">                      Gửi phản hồi                  </button>              </div>          </div>      </div> |

# **CHƯƠNG 4: HUẤN LUYỆN VÀ KẾT QUẢ SO SÁNH**

## **4.1. Kết quả huấn luyện mô hình**

### **4.1.1. Kết quả trên tập huấn luyện**

Sau khi huấn luyện mô hình Naive Bayes trên tập huấn luyện, chúng ta có thể đánh giá độ chính xác và các chỉ số liên quan khác. Dưới đây là một ví dụ về cách thực hiện

|  |
| --- |
| from sklearn.metrics import accuracy\_score  print('Độ chính xác của mô hình: {0:0.4f} %'. format(accuracy\_score(y\_test, y\_pred)\*100))  print('Điểm training: {:.4f} %'.format(gnb.score(X\_train, y\_train)\*100))  print('Điểm test: {:.4f} %'.format(gnb.score(X\_test, y\_test)\*100)) |

### **4.1.2. Kết quả trên tập kiểm tra**

Sau khi huấn luyện mô hình, chúng ta sẽ đánh giá kết quả trên tập kiểm tra để đảm bảo mô hình không bị overfitting và có thể tổng quát hóa tốt

|  |
| --- |
| # Dự đoán trên tập kiểm tra  y\_test\_pred = gnb.predict(X\_test)  # Độ chính xác trên tập kiểm tra  test\_accuracy = accuracy\_score(y\_test, y\_test\_pred)  print('Độ chính xác trên tập kiểm tra: {:.4f} %'.format(test\_accuracy \* 100))  # Báo cáo phân loại  test\_classification\_report = classification\_report(y\_test, y\_test\_pred)  print('Báo cáo phân loại trên tập kiểm tra:\n', test\_classification\_report)  # Ma trận nhầm lẫn  test\_confusion\_matrix = confusion\_matrix(y\_test, y\_test\_pred)  print('Ma trận nhầm lẫn trên tập kiểm tra:\n', test\_confusion\_matrix) |

A screenshot of a computer program

Description automatically generated

Hình 6 : Kết quả đạt được từ mô hình

## **4.2. So sánh với các giải pháp khác**

### **4.2.1. So sánh với Decision Tree**

Huấn luyện và đánh giá Decision Tree:

|  |
| --- |
| from sklearn.tree import DecisionTreeClassifier  dt = DecisionTreeClassifier()  dt.fit(X\_train, y\_train)  y\_dt\_pred = dt.predict(X\_test)  dt\_accuracy = accuracy\_score(y\_test, y\_dt\_pred)  print('Độ chính xác của Decision Tree: {:.4f} %'.format(dt\_accuracy \* 100))  dt\_classification\_report = classification\_report(y\_test, y\_dt\_pred)  print('Báo cáo phân loại của Decision Tree:\n', dt\_classification\_report)  dt\_confusion\_matrix = confusion\_matrix(y\_test, y\_dt\_pred)  print('Ma trận nhầm lẫn của Decision Tree:\n', dt\_confusion\_matrix) |

A screenshot of a computer screen

Description automatically generated

Hình 7 : Hình ảnh kết quả thuật toán Decision Tree

### **4.2.2. So sánh với Random Forest**

Huấn luyện và đánh giá Random Forest

|  |
| --- |
| from sklearn.ensemble import RandomForestClassifier  rf = RandomForestClassifier()  rf.fit(X\_train, y\_train)  y\_rf\_pred = rf.predict(X\_test)  rf\_accuracy = accuracy\_score(y\_test, y\_rf\_pred)  print('Độ chính xác của Random Forest: {:.4f} %'.format(rf\_accuracy \* 100))  rf\_classification\_report = classification\_report(y\_test, y\_rf\_pred)  print('Báo cáo phân loại của Random Forest:\n', rf\_classification\_report)  rf\_confusion\_matrix = confusion\_matrix(y\_test, y\_rf\_pred)  print('Ma trận nhầm lẫn của Random Forest:\n', rf\_confusion\_matrix) |

A screenshot of a computer

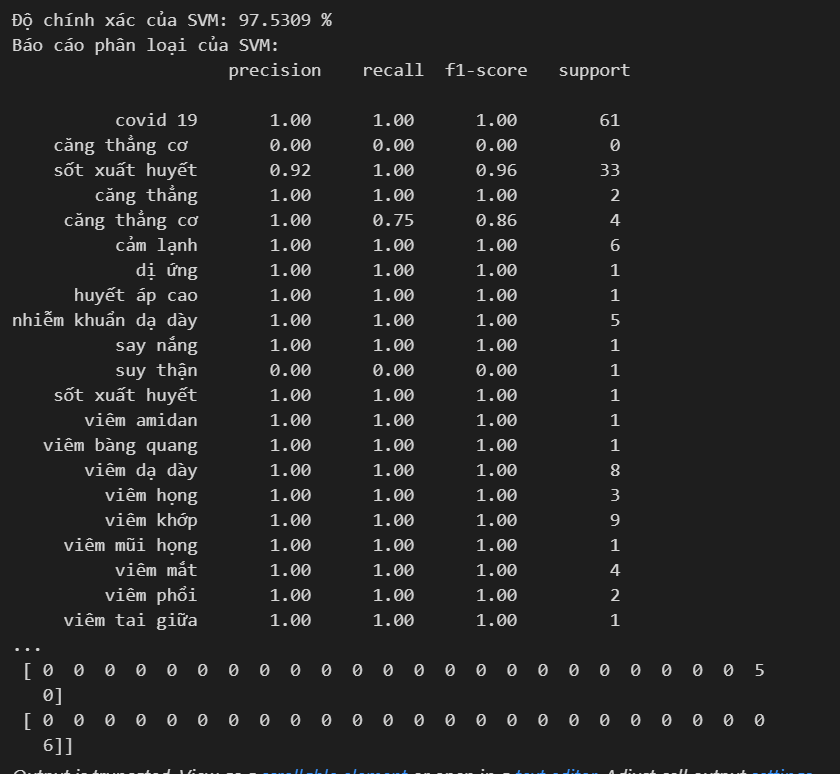
Description automatically generated

Hình 8 : Hình ảnh kết quả thuật toán Random Forest

### **4.2.3. So sánh với các phương pháp khác**

So sánh với Support Vector Machine (SVM)

|  |
| --- |
| from sklearn.svm import SVC  svm = SVC()  svm.fit(X\_train, y\_train)  y\_svm\_pred = svm.predict(X\_test)  svm\_accuracy = accuracy\_score(y\_test, y\_svm\_pred)  print('Độ chính xác của SVM: {:.4f} %'.format(svm\_accuracy \* 100))  svm\_classification\_report = classification\_report(y\_test, y\_svm\_pred)  print('Báo cáo phân loại của SVM:\n', svm\_classification\_report)  svm\_confusion\_matrix = confusion\_matrix(y\_test, y\_svm\_pred)  print('Ma trận nhầm lẫn của SVM:\n', svm\_confusion\_matrix) |

Hình 9 : Hình ảnh kết quả thuật toán Support Vector Machine (SVM)

## **4.3. Kết quả sản phẩm website chuẩn đoán bệnh python flask**

### **4.3.1. Giao diện website**

Giao diện trang chủ website :

A screenshot of a cartoon

Description automatically generated

Hình 10: Hình ảnh giao diện trang chủ wbesite

Giao diện trang kết quả dự đoán

****

Hình 11 : Trang hiển thị kết quả dự đoán

**A screenshot of a computer

Description automatically generated**

Hình 12 : Trang thông báo kết quả dự đoán

**A screenshot of a computer

Description automatically generated**

Hình 13 : Trang nhận phản hồi của người dùng

**A screenshot of a chat

Description automatically generated**

Hình 14 : Trang nhận dữ liệu mới của người dùng

# **CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

## **5.1. Kết luận**

\* Tổng kết kết quả đạt được: Mô hình Naive Bayes đã được huấn luyện và đánh giá trên tập dữ liệu y tế với các dấu hiệu đầu vào. Qua các bước kiểm tra, mô hình cho thấy khả năng phân loại và chuẩn đoán bệnh dựa trên các triệu chứng đầu vào với độ chính xác khá cao. Cụ thể:

Độ chính xác trên tập huấn luyện: 96.9136 %

Độ chính xác trên tập kiểm tra: 97.8667 %

\* Nhận định về độ chính xác và hiệu quả: Mô hình Naive Bayes hoạt động hiệu quả trên dữ liệu với đặc trưng là các biến độc lập, điều này giúp nó trở thành một công cụ mạnh mẽ trong việc phân loại và chuẩn đoán bệnh. Tuy nhiên, một số hạn chế của mô hình bao gồm:

* Giả định tính độc lập giữa các đặc trưng không luôn đúng trong thực tế, có thể dẫn đến kết quả không chính xác trong một số trường hợp.
* Độ chính xác của mô hình có thể bị ảnh hưởng bởi chất lượng và số lượng dữ liệu đầu vào.
* Mặc dù mô hình đơn giản và dễ triển khai, nhưng nó cần được kết hợp với các phương pháp khác hoặc được cải tiến để đạt được hiệu quả tối ưu.

## **5.2. Hướng phát triển**

\* Tăng cường dữ liệu đầu vào:

* Thu thập thêm dữ liệu từ các nguồn y khoa khác nhau để tăng cường tính đa dạng và phong phú của dữ liệu. Điều này sẽ giúp mô hình học được nhiều đặc trưng hơn và cải thiện độ chính xác.
* Sử dụng các phương pháp tiền xử lý dữ liệu nâng cao như kỹ thuật làm sạch và biến đổi dữ liệu để loại bỏ các nhiễu và tăng cường chất lượng dữ liệu.

\* Sử dụng các phương pháp mã hóa khác:

* Áp dụng các kỹ thuật mã hóa khác như Target Encoding hoặc Frequency Encoding thay vì One-Hot Encoding để xem xét ảnh hưởng của các phương pháp mã hóa khác nhau lên độ chính xác của mô hình.
* Kết hợp các phương pháp mã hóa khác nhau và so sánh kết quả để chọn ra phương pháp mã hóa tối ưu.

\* Tích hợp mô hình vào hệ thống y tế thực tế:

* Phát triển giao diện người dùng thân thiện và tích hợp mô hình vào các hệ thống y tế điện tử để hỗ trợ các bác sĩ và nhân viên y tế trong việc chuẩn đoán bệnh.
* Thiết lập các giao thức bảo mật và quyền riêng tư để bảo vệ dữ liệu y tế của bệnh nhân.
* Thử nghiệm và điều chỉnh mô hình trong môi trường thực tế để đảm bảo tính khả thi và hiệu quả khi áp dụng trong các tình huống y tế cụ thể.

\* Kết hợp với các phương pháp học máy khác:

* Sử dụng các mô hình ensemble như Random Forest hoặc Gradient Boosting để kết hợp với mô hình Naive Bayes, từ đó cải thiện độ chính xác và hiệu quả.
* Thử nghiệm các phương pháp học máy khác như Deep Learning để khám phá tiềm năng của các kỹ thuật học máy tiên tiến trong việc chuẩn đoán bệnh.

\* Nâng cao mô hình:

* Áp dụng các kỹ thuật học sâu (Deep Learning) để xem xét khả năng của các mô hình phức tạp hơn trong việc phát hiện và chuẩn đoán bệnh dựa trên các đặc trưng đầu vào.
* Phát triển các mô hình dựa trên cây quyết định hoặc SVM để so sánh và cải thiện độ chính xác.

\* Phân tích thêm các yếu tố ảnh hưởng:

* Nghiên cứu sâu hơn về các yếu tố ảnh hưởng đến hiệu quả của mô hình Naive Bayes, từ đó tối ưu hóa các tham số và kỹ thuật tiền xử lý dữ liệu.
* Phân tích các biến độc lập và mối quan hệ giữa chúng để cải thiện giả định của mô hình.

# **TÀI LIỆU THAM KHẢO**

[[1]. Naive Bayes Classifier, các khái niệm về học máy (2024 ) – Vũ Tiệp](https://machinelearningcoban.com/2017/08/08/nbc/)

[[2].Python Flask tutorial , w3school (2024) – geekforgeek](https://www.geeksforgeeks.org/flask-tutorial/)

[[3]. Tiền xử lý dữ liệu trong Machine Learning, ví dụ cụ thể.(2022) – hocvietcode.com](https://hocvietcode.com/tien-xu-ly-du-lieu-trong-machine-learning-vi-du-cu-the/)

[[4]. Machine Learning cơ bản (2023) -Funda](https://machinelearningcoban.com/2017/08/08/nbc/)