**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: NHẬP MÔN HỌC MÁY**

**ĐỀ TÀI: PHÂN LOẠI TIN NHẮN RÁC**

**Giáo viên hướng dẫn: TRẦN THU TRANG**

**Sinh viên thực hiện:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Stt** | **Mã sv** | **Họ và tên** | **Lớp** |
| 1 | 1771020258 | Nguyễn Xuân Hiệp | CNTT17-12 |
| 2 | 1771020653 | Nguyễn Văn Thống | CNTT17-12 |

**Hà Nội, năm 2025**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: NHẬP MÔN HỌC MÁY**

**ĐỀ TÀI: PHÂN LOẠI TIN NHẮN RÁC**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Mã Sinh Viên | Họ và Tên | Ngày Sinh | Điểm | |
| Bằng Số | Bằng Chữ |
| 1 | 1771020258 | Nguyễn Xuân Hiệp | 29/10/2005 |  |  |
| 2 | 1771020653 | Nguyễn Văn Thống | 26/03/2005 |  |  |

**CÁN BỘ CHẤM THI**

**Hà Nội, năm 2025**

# LỜI NÓI ĐẦU

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ, dữ liệu văn bản chiếm tỷ trọng rất lớn và đóng vai trò quan trọng trong nhiều hoạt động của đời sống. Một trong những vấn đề nổi bật là sự xuất hiện tràn lan của tin nhắn rác (Spam), gây ảnh hưởng tiêu cực đến trải nghiệm người dùng và làm giảm hiệu quả trao đổi thông tin. Do đó, việc nghiên cứu và xây dựng các hệ thống phân loại tin nhắn để nhận diện và loại bỏ Spam trở thành yêu cầu thiết thực và cấp bách.

Xuất phát từ ý nghĩa đó, nhóm thực hiện đề tài “Phân loại tin nhắn Spam và Ham bằng các thuật toán học máy và học sâu”. Nội dung nghiên cứu tập trung vào việc làm sạch và chuẩn hóa dữ liệu, áp dụng các thuật toán học máy truyền thống như Naive Bayes, Logistic Regression, Decision Tree, Random Forest, SVM, kết hợp với mô hình học sâu LSTM. Thông qua quá trình huấn luyện, đánh giá và so sánh, đề tài hướng đến mục tiêu tìm ra phương pháp hiệu quả nhất cho bài toán phân loại.

Đề tài không chỉ mang tính học thuật mà còn có giá trị ứng dụng trong thực tiễn, góp phần xây dựng những hệ thống lọc Spam tự động, hỗ trợ bảo mật thông tin và nâng cao trải nghiệm người dùng.

Trong quá trình thực hiện, nhóm đã nỗ lực nghiên cứu và hoàn thành báo cáo. Tuy nhiên, do hạn chế về thời gian và kiến thức, báo cáo khó tránh khỏi những thiếu sót. Nhóm rất mong nhận được sự đóng góp, chỉ dẫn từ thầy cô để hoàn thiện hơn. Nhân dịp này, nhóm xin gửi lời cảm ơn chân thành tới giảng viên đã tận tình hướng dẫn và tạo điều kiện để nhóm hoàn thành đề tài này.

# MỤC LỤC

[LỜI NÓI ĐẦU 3](#_Toc209568874)

[MỤC LỤC 4](#_Toc209568875)

[MỤC LỤC HÌNH ẢNH 6](#_Toc209568876)

[CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI VÀ PHÂN TÍCH DỮ LIỆU 7](#_Toc209568877)

[1.1 Lý do chọn đề tài 7](#_Toc209568878)

[1.2 Mục tiêu nghiên cứu 7](#_Toc209568879)

[1.3 Mô tả dữ liệu 7](#_Toc209568880)

[1.3.1: Giới thiệu dữ liệu 7](#_Toc209568881)

[1.4 Trực quan hóa dữ liệu 8](#_Toc209568882)

[1.4.1. Phân phối nhãn (Spam vs ham) 8](#_Toc209568883)

[1.4.2. Độ dài tin nhắn 9](#_Toc209568884)

[1.4.3. Phân phối độ dài (Histogram) 10](#_Toc209568885)

[1.4.4. Phân tích từ khóa đặc trưng 10](#_Toc209568886)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 12](#_Toc209568887)

[2.1: Phân loại văn bản và xử lý ngôn ngữ tự nhiên 12](#_Toc209568888)

[2.2: Các bước tiền xử lý dữ liệu văn bản 12](#_Toc209568889)

[2.3: Thuật toán phân loại 15](#_Toc209568890)

[2.3.1. Naive Bayes (NB) 15](#_Toc209568891)

[2.3.2 Hồi quy Logistic (Logistic Regression) 15](#_Toc209568892)

[2.3.3. K-Nearest Neighbors (KNN) 16](#_Toc209568893)

[2.3.4. Decision Tree (Cây quyết định) 16](#_Toc209568894)

[2.3.5. Random Forest (Rừng ngẫu nhiên) 17](#_Toc209568895)

[2.3.6. Support Vector Machine (SVM) 17](#_Toc209568896)

[2.3.7. Deep Learning với LSTM 18](#_Toc209568897)

[2.4: Đánh giá mô hình 19](#_Toc209568898)

[CHƯƠNG 3: THỰC NGHIỆM VÀ KẾT QUẢ 21](#_Toc209568899)

[3.1:Làm sạch và chuẩn hóa dữ liệu 21](#_Toc209568900)

[3.1.1. Đọc dữ liệu và kiểm tra ban đầu 21](#_Toc209568901)

[3.1.2. Chuyển văn bản về dạng số 22](#_Toc209568902)

[3.1.3. Chuẩn hóa dữ liệu đầu vào 22](#_Toc209568903)

[3.1.4: Chia dữ liệu train/test 23](#_Toc209568904)

[3.1.5:Xử lí mất cân bằng dữ liệu 23](#_Toc209568905)

[3.1.6 Kết luận 24](#_Toc209568906)

[3.2:Chạy các thuật toán 25](#_Toc209568907)

[3.2.1 Chuẩn bị dữ liệu đầu vào 25](#_Toc209568908)

[3.2.2 Huấn luyện các mô hình học máy 25](#_Toc209568909)

[3.2.3 Triển khai mô hình học sâu LSTM 26](#_Toc209568910)

[3.2.4 Kết quả sơ bộ và nhận xét 26](#_Toc209568911)

[3.3:So Sánh kết quả giữa các thuật toán 27](#_Toc209568912)

[KẾT LUẬN 32](#_Toc209568913)

[TÀI LIỆU THAM KHẢO 34](#_Toc209568914)

# MỤC LỤC HÌNH ẢNH

[Hình 1: Phân bố nhãn (ham vs spam). 7](#_Toc208412823)

[Hình 2: Phân bố độ dài tin nhắn theo ký tự. 8](#_Toc208412824)

[Hình 3: Boxplot số từ trong tin nhắn theo nhãn. 9](#_Toc208412825)

[Hình 4: Từ phổ biến nhất trong spam. 9](#_Toc208412826)

[Hình 5: Từ phổ biến nhất trong ham. 10](#_Toc208412827)

# CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI VÀ PHÂN TÍCH DỮ LIỆU

## 1.1 Lý do chọn đề tài

Trong bối cảnh hiện nay, công nghệ thông tin và truyền thông phát triển mạnh mẽ đã tạo ra nhiều tiện ích trong cuộc sống, trong đó tin nhắn SMS (Short Message Service) vẫn là một phương tiện giao tiếp phổ biến, đặc biệt trên các thiết bị di động. Tuy nhiên, song song với những lợi ích đó, tình trạng tin nhắn rác (spam) ngày càng gia tăng. Đây là những tin nhắn thường chứa nội dung quảng cáo, lừa đảo hoặc gây phiền nhiễu cho người dùng.

Theo nhiều khảo sát, một người trung bình có thể nhận được hàng chục tin nhắn rác mỗi tháng. Việc này không chỉ ảnh hưởng đến trải nghiệm sử dụng dịch vụ viễn thông mà còn tiềm ẩn nguy cơ về bảo mật thông tin cá nhân và an toàn tài chính. Chính vì vậy, việc xây dựng các hệ thống tự động phân loại tin nhắn SMS để phát hiện spam là một nhu cầu thiết thực.

Trong nghiên cứu này, mục tiêu của đề tài là:

• Phân tích và tìm hiểu đặc trưng của dữ liệu tin nhắn SMS.

• Ứng dụng các kỹ thuật học máy (Machine Learning) để phân loại tin nhắn thành hai nhóm: tin nhắn hợp lệ (ham) và tin nhắn rác (spam).

• Đánh giá, so sánh hiệu quả của một số thuật toán phổ biến trong bài toán phân loại văn bản.

Đề tài không chỉ có ý nghĩa thực tiễn trong việc giảm thiểu tin nhắn rác, mà còn giúp sinh viên, nhà nghiên cứu hiểu rõ hơn về quy trình xử lý ngôn ngữ tự nhiên (NLP) và ứng dụng trong học máy.

## 1.2 Mục tiêu nghiên cứu

Mục tiêu chính của đề tài là phân tích bộ dữ liệu SMS để tìm hiểu các đặc trưng của tin nhắn spam và ham, đồng thời giới thiệu các thuật toán học máy có khả năng phân loại tin nhắn một cách tự động, chính xác.

## 1.3 Mô tả dữ liệu

### 1.3.1: Giới thiệu dữ liệu

Mục tiêu của việc phân tích này là khám phá đặc điểm giữa tin nhắn spam và non-spam, từ đó hỗ trợ các bước tiếp theo như xây dựng mô hình học máy để tự động phân loại spam.

Bộ dữ liệu sử dụng có 5574 bản ghi và 2 cột, bao gồm:

- sms: Nội dung tin nhắn văn bản.

- label: Nhãn phân loại (0 = ham, 1 = spam).

Số lượng tin nhắn duy nhất: 5171. Tỷ lệ spam khoảng 13.40%.

Ví dụ một số tin nhắn spam:

- Summers finally here! Fancy a chat or flirt with sexy singles in yr area? To get MATCHED up just reply SUMMER now. Free 2 Join. OptOut txt STOP Help08714742804

- This is the 2nd time we have tried 2 contact u. U have won the 750 Pound prize. 2 claim is easy, call 08718726970 NOW! Only 10p per min. BT-national-rate

- Get ur 1st RINGTONE FREE NOW! Reply to this msg with TONE. Gr8 TOP 20 tones to your phone every week just £1.50 per wk 2 opt out send STOP 08452810071 16

Ví dụ một số tin nhắn ham:

- Dear,Me at cherthala.in case u r coming cochin pls call bfore u start.i shall also reach accordingly.or tell me which day u r coming.tmorow i am engaged ans its holiday.

- Ok. I only ask abt e movie. U wan ktv oso?

- Convey my regards to him

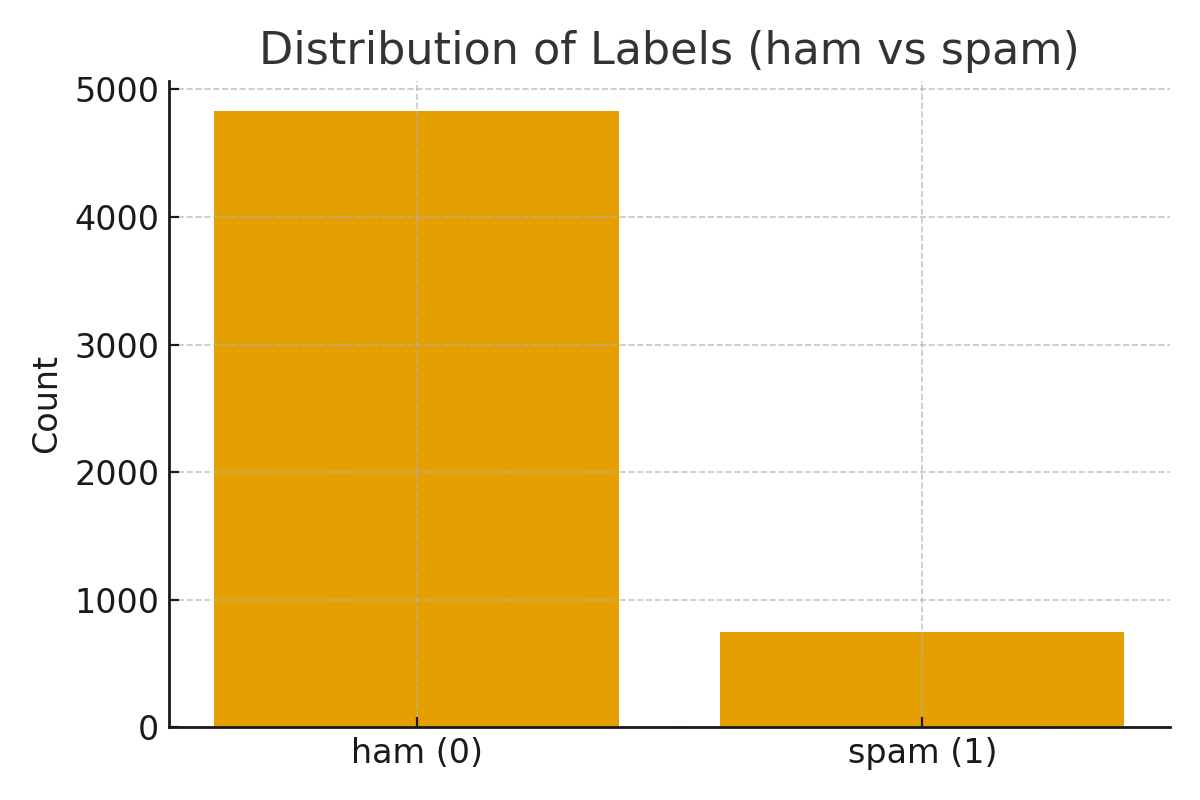
## 1.4 Trực quan hóa dữ liệu

### 1.4.1. Phân phối nhãn (Spam vs ham)

Kết quả thống kê cho thấy spam chỉ chiếm khoảng 13–14% tổng dữ liệu, còn lại hơn 85% là ham.

Đây là một hiện tượng mất cân bằng dữ liệu (imbalanced dataset), có thể gây ảnh hưởng đến quá trình huấn luyện mô hình phân loại vì mô hình dễ bị “thiên vị” với lớp chiếm đa số (ham).

Trong thực tế, việc spam ít hơn phản ánh đúng đời sống: đa số tin nhắn người dùng nhận là thật, chỉ một phần nhỏ là rác/quảng cáo.



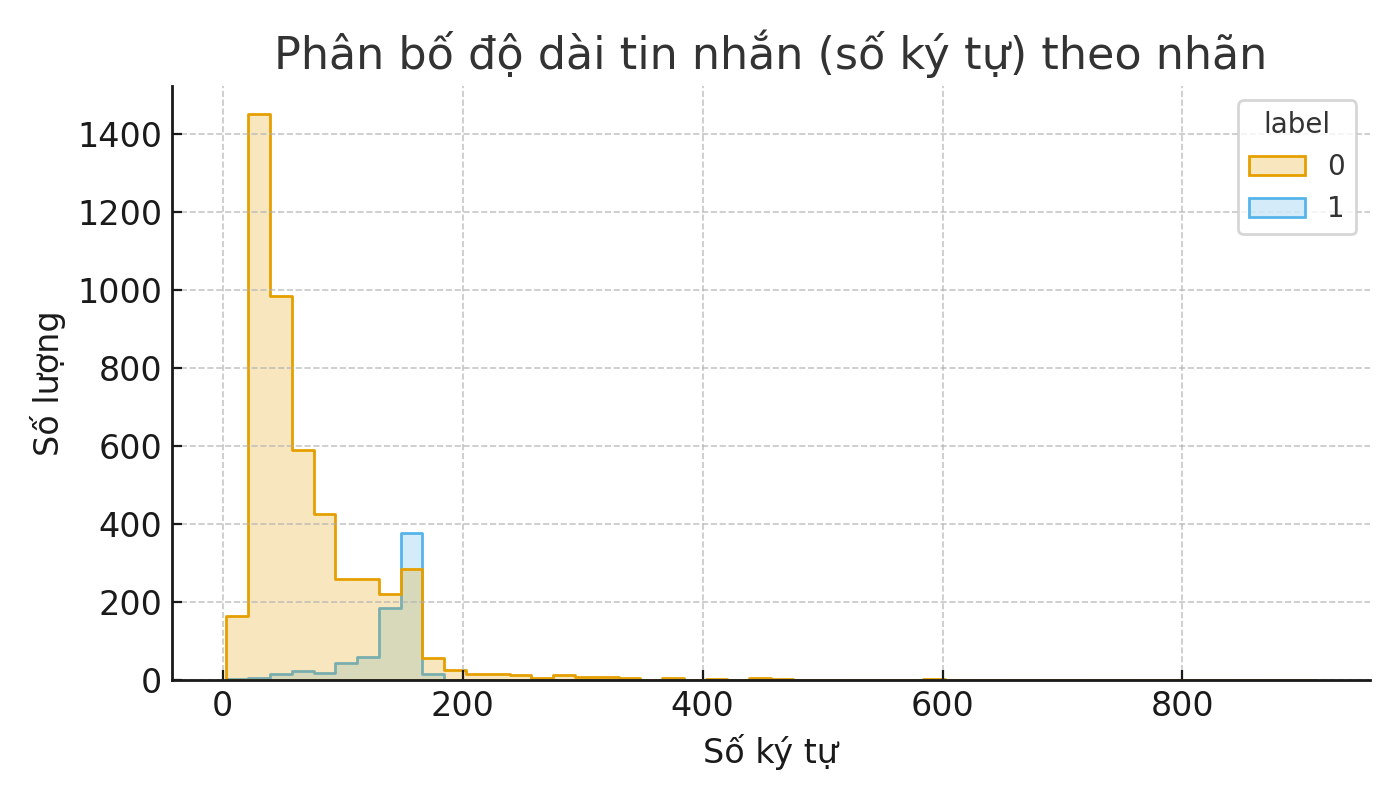
Hình : Phân bố nhãn (ham vs spam).

### 1.4.2. Độ dài tin nhắn

Trung bình, tin nhắn spam dài hơn ham.

Spam thường chứa nhiều ký tự, nhiều từ hơn, vì người gửi cố gắng truyền tải thông tin quảng cáo, thể lệ tham gia, mã số trúng thưởng...

Ham thì ngắn gọn, chủ yếu mang tính trao đổi nhanh trong đời sống hàng ngày (“Ok”, “Where r u?”, “Call me now”).



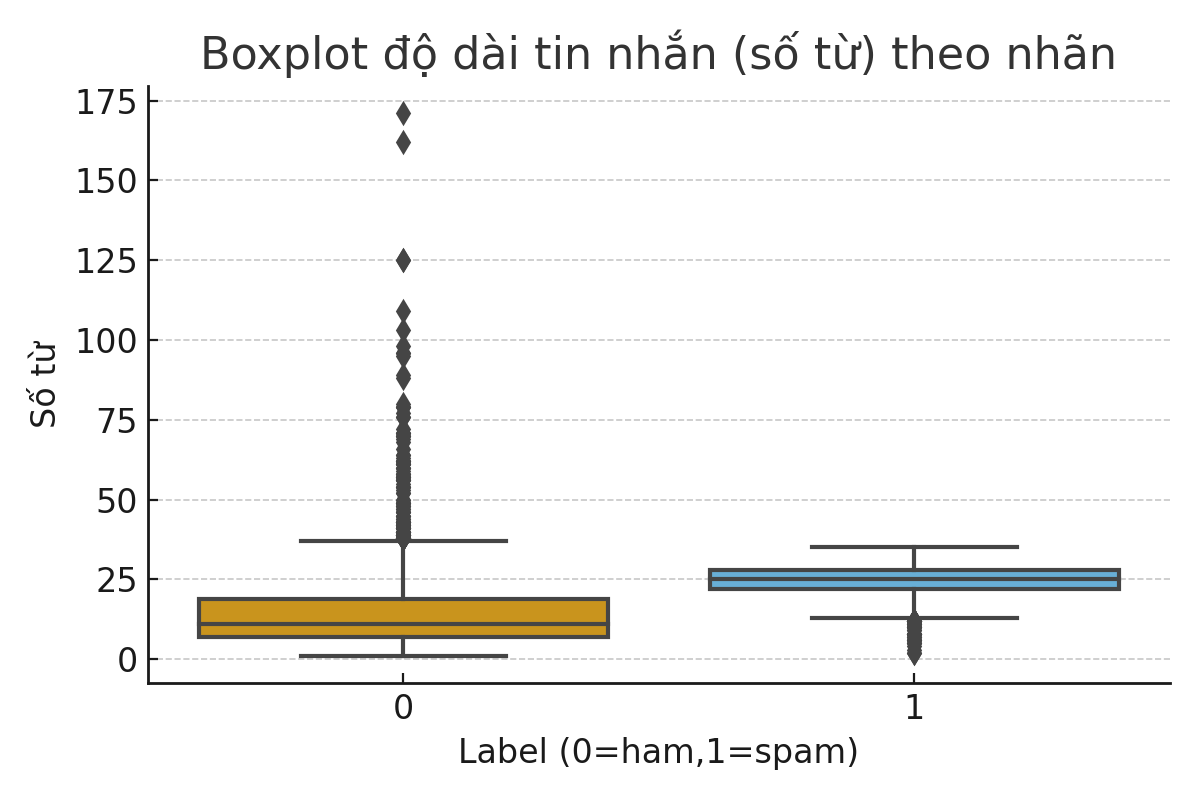
Hình 2: Phân bố độ dài tin nhắn theo ký tự.

### 1.4.3. Phân phối độ dài (Histogram)

Khi vẽ biểu đồ histogram:

Non-spam tập trung nhiều nhất trong khoảng 20–40 ký tự, hiếm khi vượt quá 100 ký tự.

Spam trải rộng hơn, có nhiều tin nhắn > 100 ký tự, thậm chí dài đến vài trăm ký tự.



Hình : Boxplot số từ trong tin nhắn theo nhãn.

### 1.4.4. Phân tích từ khóa đặc trưng

Một số từ điển hình cho spam:

free, win, claim, prize, txt, call, urgent

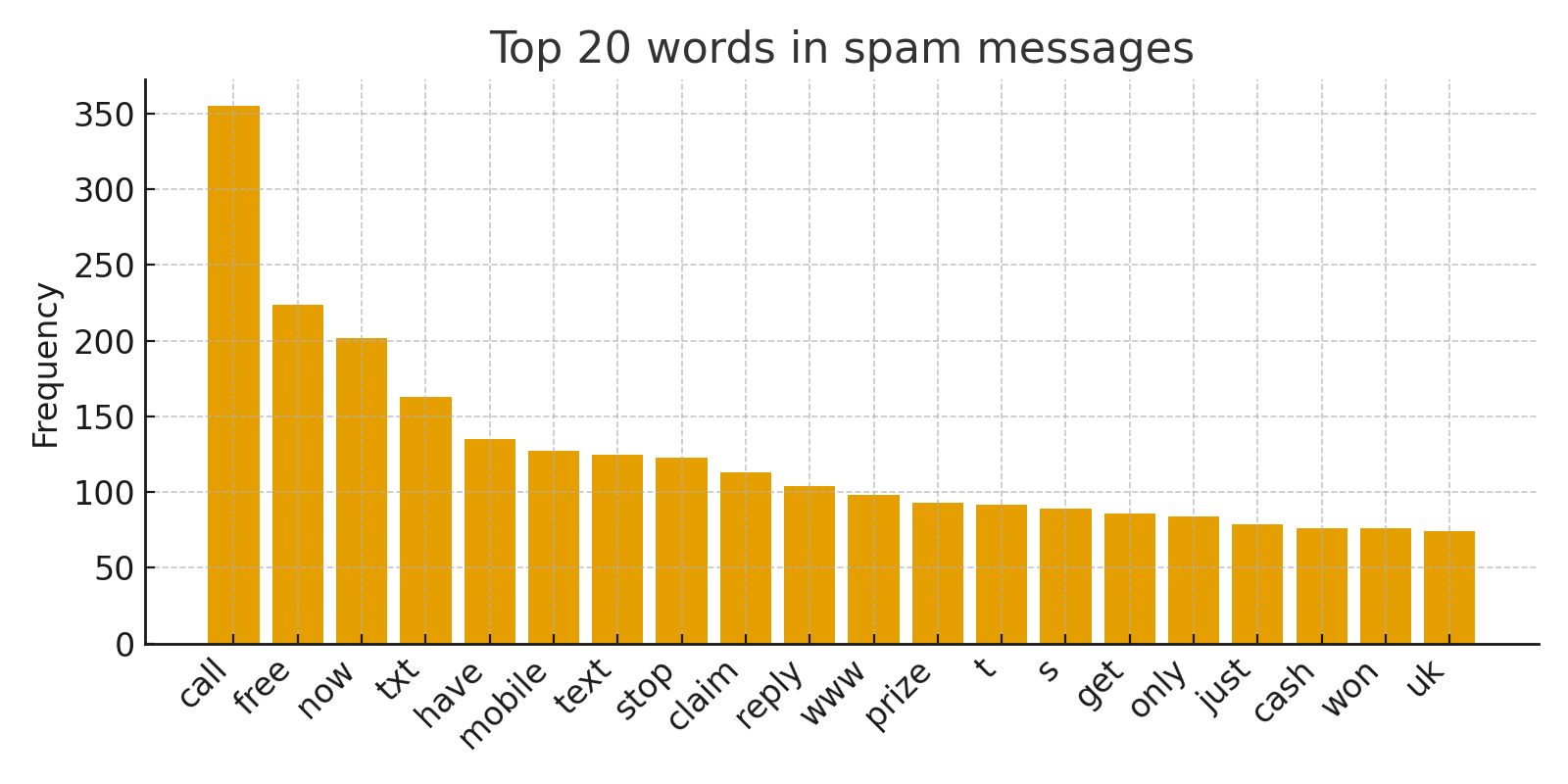
-Liên quan đến khuyến mãi, giải thưởng, trò chơi may rủi.

Trong khi đó, ham lại có nhiều từ:

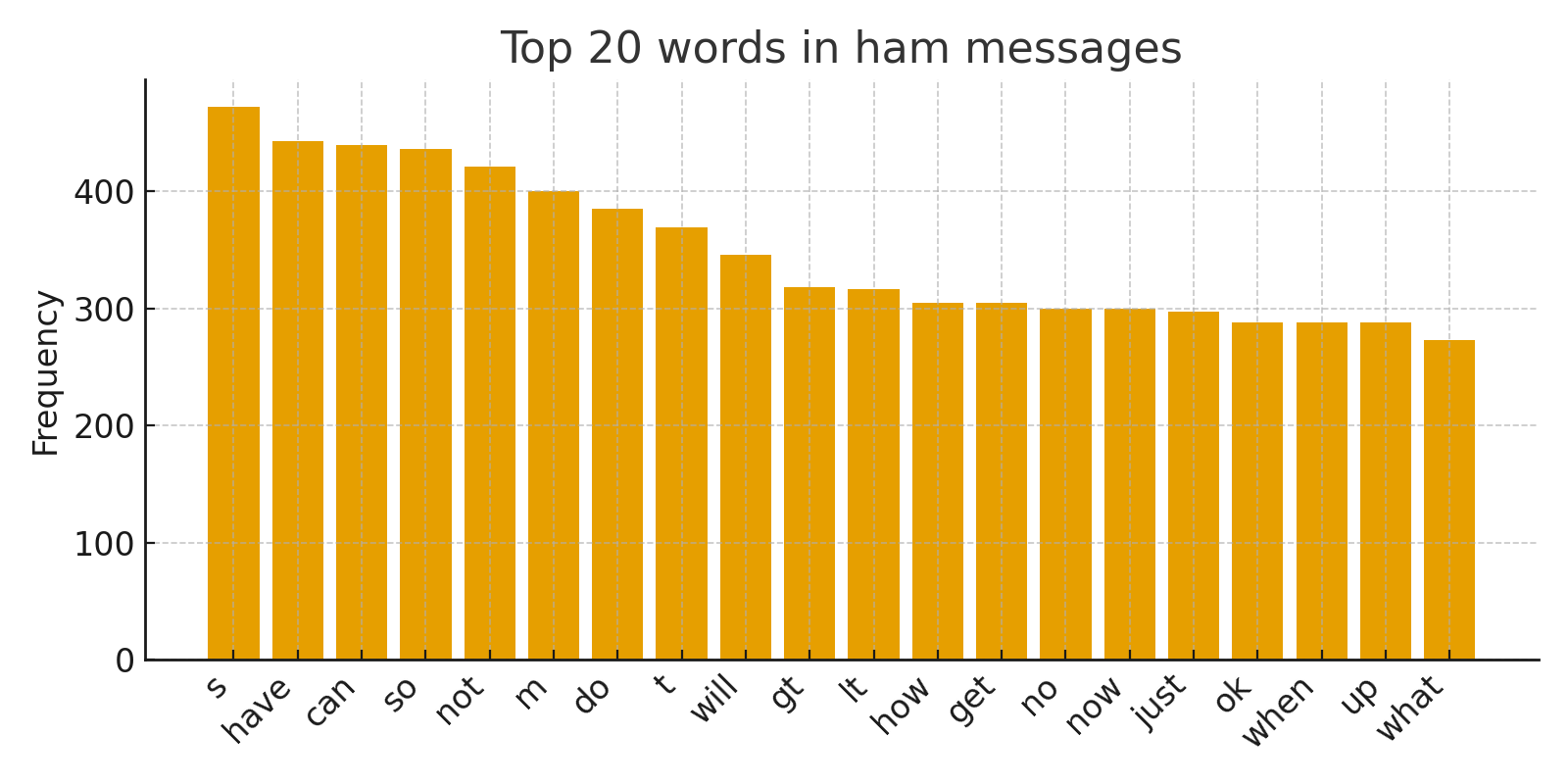
ok, love, go, come, got, like

-Thể hiện ngữ cảnh giao tiếp đời sống, gia đình, tình cảm, bạn bè.

chỉ cần nhìn từ khóa là ta đã thấy sự khác biệt rõ ràng về mục đích.



Hình : Từ phổ biến nhất trong spam.



Hình : Từ phổ biến nhất trong ham.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## 2.1: Phân loại văn bản và xử lý ngôn ngữ tự nhiên

Phân loại văn bản là một nhánh quan trọng của khai phá dữ liệu văn bản (text mining) và xử lý ngôn ngữ tự nhiên (NLP). Trong bài toán spam filtering, văn bản đầu vào là SMS cần được tiền xử lý và biến đổi thành đặc trưng số trước khi đưa vào mô hình học máy.

## 2.2: Các bước tiền xử lý dữ liệu văn bản

Trong các bài toán phân loại văn bản nói chung và phân loại tin nhắn rác nói riêng, dữ liệu thô thường tồn tại dưới dạng văn bản ngôn ngữ tự nhiên. Dữ liệu này chứa nhiều yếu tố gây nhiễu như chữ viết hoa, ký tự đặc biệt, dấu câu, khoảng trắng, từ thừa (stopwords),… Nếu đưa trực tiếp dữ liệu gốc này vào mô hình học máy, hiệu quả phân loại sẽ rất thấp. Do đó, tiền xử lý dữ liệu văn bản là một bước bắt buộc nhằm chuẩn hóa, làm sạch và biến đổi văn bản thành dạng biểu diễn số học phù hợp với mô hình.

Dựa trên chương trình đã cài đặt, quá trình tiền xử lý dữ liệu được tiến hành qua các bước sau:

**Bước 1: Chuẩn hóa văn bản về chữ thường (Lowercasing)**

Trong tập dữ liệu, văn bản có thể chứa cả chữ hoa và chữ thường. Ví dụ: “FREE entry in 2 a wkly comp” và “Free entry in 2 a wkly comp” thực chất có cùng ý nghĩa, nhưng nếu không chuẩn hóa thì mô hình sẽ coi “FREE” và “Free” là hai đặc trưng hoàn toàn khác nhau. Điều này làm tăng độ phức tạp và giảm tính khái quát của mô hình.

Do đó, toàn bộ văn bản được chuyển về chữ thường trước khi xử lý tiếp theo. Đây là bước cơ bản nhưng rất quan trọng để tránh trùng lặp đặc trưng và đảm bảo sự thống nhất trong biểu diễn văn bản.

**Bước 2: Loại bỏ ký tự đặc biệt, số và dấu câu**

Dữ liệu tin nhắn thường chứa nhiều ký tự đặc biệt như !, @, #, $, dấu chấm câu hoặc các con số, đường link, địa chỉ email… Những thành phần này hiếm khi mang ý nghĩa phân loại rõ ràng mà thường chỉ đóng vai trò phụ trợ. Nếu giữ nguyên, chúng sẽ làm “loãng” đặc trưng và gây nhiễu cho mô hình.

Trong chương trình, bước này được xử lý bằng cách loại bỏ các ký tự không cần thiết, chỉ giữ lại các từ dạng chữ cái. Kết quả là văn bản trở nên “sạch” và tập trung hơn vào phần từ ngữ mang nghĩa thực sự. Ví dụ:

Trước xử lý: Win $$$ now!!! Call 1800-123-456

Sau xử lý: win now call

Nhờ vậy, số chiều của dữ liệu giảm xuống đáng kể và thông tin giữ lại cũng cô đọng, phù hợp cho mô hình học máy.

**Bước 3: Tách từ (Tokenization)**

Sau khi văn bản đã được chuẩn hóa và làm sạch, bước tiếp theo là tách từ (tokenization). Quá trình này biến một chuỗi văn bản thành một danh sách các từ (token). Ví dụ:

Câu gốc: “Win a free ticket now”

Sau tách từ: [“win”, “a”, “free”, “ticket”, “now”]

Trong chương trình, quá trình tách từ không được viết thủ công mà được thực hiện ngầm định bởi **TfidfVectorizer** trong thư viện *scikit-learn*. Khi vectorizer nhận văn bản đầu vào, nó sẽ tự động tách từ, xây dựng tập từ vựng (vocabulary) và gán chỉ số cho từng từ trong tập dữ liệu.

**Bước 4: Loại bỏ stopwords**

Stopwords là những từ phổ biến, xuất hiện với tần suất cao nhưng ít mang giá trị ngữ nghĩa trong phân loại, chẳng hạn như: *a, an, the, of, in, on, to, and…*. Nếu giữ nguyên, chúng sẽ chiếm dung lượng lớn trong ma trận đặc trưng mà không thực sự giúp mô hình phân biệt giữa tin nhắn “spam” và “ham”.

Trong chương trình**, TfidfVectorizer** được cấu hình với tham số stop\_words='english' để loại bỏ trực tiếp các từ dừng trong tiếng Anh. Điều này giúp giảm số chiều dữ liệu và tập trung vào những từ khóa quan trọng hơn, chẳng hạn như “free”, “call”, “win”, “offer”,… vốn mang tính đặc trưng rõ rệt cho tin nhắn spam.

**Bước 5: Stemming hoặc Lemmatization (chưa áp dụng trong chương trình)**

Stemming và Lemmatization là hai kỹ thuật thường được sử dụng để đưa từ về dạng gốc.

* Stemming: Cắt bỏ phần đuôi của từ để đưa về gốc gần đúng. Ví dụ: “running”, “runs”, “ran” → “run”.
* Lemmatization: Sử dụng từ điển ngôn ngữ để đưa từ về dạng chuẩn xác về ngữ pháp. Ví dụ: “better” → “good”.

Trong chương trình hiện tại, nhóm thực hiện chưa áp dụng stemming hoặc lemmatization, mà chỉ dừng lại ở bước loại bỏ stopwords và vector hóa TF-IDF. Tuy nhiên, trong các nghiên cứu mở rộng, có thể tích hợp thêm thư viện NLTK hoặc spaCy để chuẩn hóa từ vựng tốt hơn, từ đó giúp cải thiện hiệu quả phân loại.

**Bước 6: Biểu diễn văn bản bằng TF-IDF**

Sau khi đã hoàn tất các bước làm sạch, dữ liệu văn bản cần được biến đổi thành dạng số để đưa vào mô hình. Trong chương trình, phương pháp TF-IDF (Term Frequency – Inverse Document Frequency) được lựa chọn.

* Term Frequency (TF): Đo tần suất xuất hiện của một từ trong một văn bản.
* Inverse Document Frequency (IDF): Giảm trọng số của những từ xuất hiện quá phổ biến trong toàn bộ tập văn bản, vì những từ này ít mang giá trị phân biệt.

Khi kết hợp lại, TF-IDF phản ánh tầm quan trọng của một từ trong một văn bản cụ thể so với toàn bộ tập dữ liệu.

Trong chương trình, TfidfVectorizer(max\_features=5000) được sử dụng, nghĩa là chỉ giữ lại 5000 đặc trưng quan trọng nhất. Kết quả thu được là một ma trận thưa (sparse matrix), trong đó:

* Mỗi hàng tương ứng với một tin nhắn.
* Mỗi cột tương ứng với một từ đặc trưng.

Giá trị tại ô (i, j) thể hiện mức độ quan trọng (trọng số TF-IDF) của từ *j* trong tin nhắn *i*.

Ví dụ, tin nhắn “Free entry in a weekly competition” sau khi vector hóa có thể được biểu diễn thành một vector 5000 chiều, trong đó chỉ một số ít phần tử khác 0, đại diện cho các từ như “free”, “entry”, “competition”.

Tổng kết:

Quy trình tiền xử lý dữ liệu văn bản trong hệ thống bao gồm:

* Chuẩn hóa văn bản về chữ thường.
* Loại bỏ ký tự đặc biệt, số, dấu câu và khoảng trắng dư thừa.
* Tách từ (tokenization).
* Loại bỏ stopwords bằng công cụ tích hợp trong TfidfVectorizer.
* (Tùy chọn) Áp dụng stemming hoặc lemmatization để đưa từ về gốc.
* Biểu diễn văn bản bằng TF-IDF với tối đa 5000 đặc trưng quan trọng nhất.

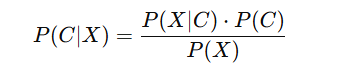
Nhờ các bước trên, dữ liệu văn bản thô (tin nhắn) được biến đổi thành dạng số học, vừa ngắn gọn, vừa phản ánh tốt các đặc trưng ngữ nghĩa quan trọng. Đây là cơ sở để các mô hình học máy (Naive Bayes, Logistic Regression, Decision Tree, Random Forest, SVM) và học sâu (LSTM) có thể khai thác và đạt hiệu quả phân loại cao hơn.

## 2.3: Thuật toán phân loại

Trong nghiên cứu này, nhiều thuật toán phân loại đã được áp dụng để giải quyết bài toán phân tích và dự đoán văn bản. Cụ thể, các thuật toán bao gồm: Naive Bayes, Hồi quy Logistic, K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Support Vector Machine (SVM) và Mạng học sâu LSTM.

### 2.3.1. Naive Bayes (NB)

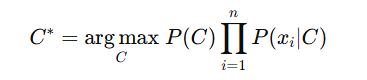
Naive Bayes dựa trên **định lý Bayes** với giả định độc lập giữa các đặc trưng:



Trong đó:

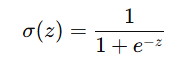
* C: lớp cần dự đoán
* X=(x1,x2,...,xn): tập hợp đặc trưng
* P(C): xác suất tiên nghiệm của lớp
* P(X∣C) xác suất quan sát đặc trưng khi biết lớp

Quy tắc phân loại:



### 2.3.2 Hồi quy Logistic (Logistic Regression)

Hồi quy Logistic sử dụng hàm sigmoid để ánh xạ giá trị tuyến tính sang khoảng [0,1]:



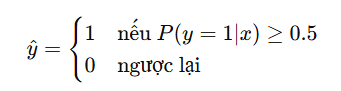
Trong đó:



Xác suất dự đoán:

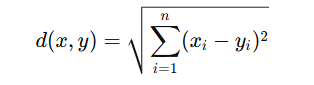


Quy tắc phân loại:



### 2.3.3. K-Nearest Neighbors (KNN)

KNN dựa vào **k láng giềng gần nhất** theo khoảng cách. Khoảng cách Euclid thường dùng:



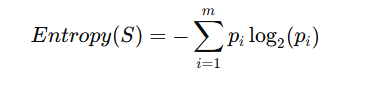
Thuật toán:

1. Tính khoảng cách từ điểm cần dự đoán đến toàn bộ tập huấn luyện.
2. Chọn ra **k** điểm gần nhất.
3. Gán nhãn theo đa số phiếu.

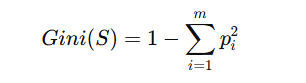
### 2.3.4. Decision Tree (Cây quyết định)

Decision Tree phân loại bằng cách chia dữ liệu dựa trên một đặc trưng theo tiêu chí như **Entropy** hoặc **Gini**.

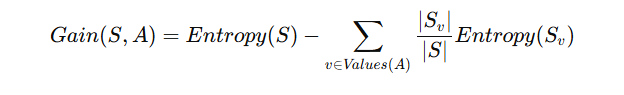
Entropy:



Gini Index:



Thông tin thu được khi chia dữ liệu theo thuộc tính **A**:



### 2.3.5. Random Forest (Rừng ngẫu nhiên)

Random Forest kết hợp nhiều Decision Tree:

* Tập huấn luyện được lấy mẫu bootstrap.
* Mỗi cây được huấn luyện độc lập.
* Kết quả cuối cùng dựa trên bỏ phiếu đa số:

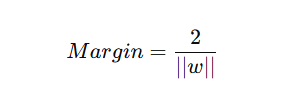


### 2.3.6. Support Vector Machine (SVM)

SVM tìm siêu phẳng tối ưu phân chia dữ liệu. Phương trình siêu phẳng: w.a+b = 0

Điều kiện phân loại:



Độ rộng lề (margin): 

Mục tiêu: cực đại hóa margin để phân tách dữ liệu tối ưu.

### 2.3.7. Deep Learning với LSTM

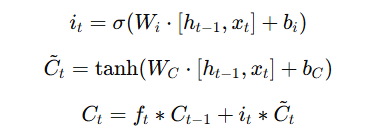
LSTM là một biến thể quan trọng của mạng nơ-ron hồi quy (RNN), được thiết kế để giải quyết nhược điểm của RNN truyền thống là khó ghi nhớ thông tin dài hạn do hiện tượng *vanishing gradient* (độ dốc biến mất). LSTM có khả năng lưu giữ và quên thông tin theo thời gian, rất phù hợp cho xử lý dữ liệu chuỗi như văn bản, âm thanh, hay chuỗi thời gian.

Một LSTM cell gồm ba cổng (gate) chính: cổng quên, cổng đầu vào, và cổng đầu ra. Các cổng này hoạt động như “van điều tiết” để quyết định thông tin nào cần giữ lại, cập nhật hoặc loại bỏ trong quá trình học.

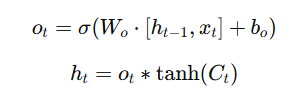
**Cổng quên**: quyết định thông tin nào từ trạng thái cũ cần bỏ đi.



**Cổng đầu vào**: xác định thông tin mới nào sẽ thêm vào bộ nhớ.



**Cổng đầu ra**: tạo trạng thái ẩn hiện tại để làm đầu ra.

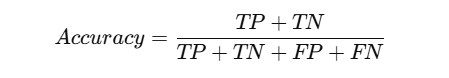


Trong đó: xt là đầu vào, ht​ là trạng thái ẩn, Ct​ là bộ nhớ cell.  
LSTM rất hiệu quả trong phân loại văn bản vì có khả năng lưu giữ ngữ cảnh dài và học được mối quan hệ giữa các từ trong chuỗi.

## 2.4: Đánh giá mô hình

Sau khi tiến hành huấn luyện các mô hình phân loại văn bản, bước tiếp theo là đánh giá chất lượng mô hình. Việc đánh giá được thực hiện trên tập dữ liệu kiểm thử, vốn đã được tách riêng từ dữ liệu ban đầu, nhằm đảm bảo tính khách quan và phản ánh khả năng mô hình hoạt động với dữ liệu mới. Trong chương trình đã cài đặt, việc đánh giá mô hình dựa vào nhiều thước đo phổ biến trong học máy như độ chính xác, ma trận nhầm lẫn, Precision, Recall và F1-score.

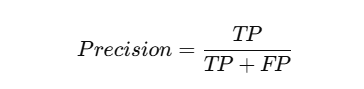
Trước hết, **độ chính xác (Accuracy)** là thước đo cơ bản nhất, cho biết tỷ lệ dự đoán đúng trên tổng số mẫu. Công thức được tính như sau:



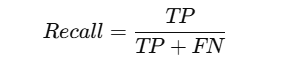
Trong đó, TP là số mẫu dương tính được dự đoán đúng, TN là số mẫu âm tính được dự đoán đúng, FP là số mẫu âm tính nhưng bị dự đoán sai thành dương, còn FN là số mẫu dương tính nhưng bị dự đoán sai thành âm. Độ chính xác giúp có cái nhìn tổng thể, nhưng khi dữ liệu mất cân bằng thì nó chưa phản ánh chính xác hiệu quả mô hình.

Tiếp theo, **ma trận nhầm lẫn (Confusion Matrix)** được sử dụng để thể hiện chi tiết số lượng dự đoán đúng và sai của mô hình trên từng lớp. Ma trận này có dạng bảng vuông, trong đó hàng biểu diễn giá trị thực tế, còn cột biểu diễn giá trị dự đoán. Thông qua ma trận nhầm lẫn, ta có thể phân tích mô hình hay mắc lỗi ở lớp nào, từ đó có biện pháp cải thiện.

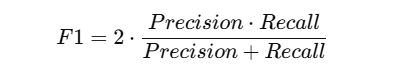
Bên cạnh đó, ba thước đo khác là **Precision, Recall và F1-score** cũng được tính toán để đánh giá mô hình một cách toàn diện hơn. Precision (độ chính xác) đo lường tỷ lệ mẫu được dự đoán dương tính mà thật sự là dương tính, công thức:



Recall (độ bao phủ) đo lường tỷ lệ mẫu dương tính thực sự được mô hình phát hiện đúng, công thức:



F1-score là trung bình điều hòa giữa Precision và Recall, giúp cân bằng hai chỉ số này, đặc biệt hữu ích khi dữ liệu mất cân bằng. Công thức:



rong phần cài đặt, các chỉ số trên được tự động tính toán bằng hàm classification\_report của thư viện sklearn, trong khi ma trận nhầm lẫn được tính bằng hàm confusion\_matrix. Nhờ vậy, ta có thể vừa định lượng được hiệu quả tổng thể, vừa phân tích được chi tiết điểm mạnh, điểm yếu của từng mô hình.

Quá trình đánh giá trong chương trình được tiến hành lần lượt cho các mô hình như **Naive Bayes, Logistic Regression, Decision Tree, Random Forest và SVM**. Cách làm cụ thể là: huấn luyện mô hình trên tập train, dự đoán trên tập test, sau đó tính toán các chỉ số đánh giá và in kết quả ra màn hình. Từ đó, ta có thể so sánh trực tiếp hiệu năng giữa các mô hình trên cùng một bộ dữ liệu.

Như vậy, việc đánh giá mô hình không chỉ giúp xác định mô hình nào cho kết quả tốt nhất mà còn cung cấp thông tin quan trọng về khả năng tổng quát hóa, mức độ cân bằng giữa các lớp, cũng như các lỗi thường gặp. Đây là bước then chốt để lựa chọn mô hình phù hợp nhất cho bài toán phân loại văn bản.

# CHƯƠNG 3: THỰC NGHIỆM VÀ KẾT QUẢ

## 3.1:Làm sạch và chuẩn hóa dữ liệu

Trong bất kỳ bài toán học máy nào, đặc biệt là các bài toán liên quan đến **xử lý ngôn ngữ tự nhiên (Natural Language Processing – NLP)**, việc làm sạch và chuẩn hóa dữ liệu luôn đóng vai trò then chốt, quyết định đến chất lượng của mô hình. Nếu dữ liệu đầu vào không được xử lý đúng cách, mô hình có thể học sai, bị nhiễu hoặc dẫn đến hiện tượng quá khớp (overfitting) và cho kết quả dự đoán kém chính xác. Do đó, trước khi đưa dữ liệu vào huấn luyện, nhóm em đã tiến hành một loạt các bước xử lý như sau:

### 3.1.1. Đọc dữ liệu và kiểm tra ban đầu

Dữ liệu gốc được lưu trữ dưới dạng bảng, gồm hai cột chính:

* Văn bản (text): nội dung cần phân loại.
* Nhãn (label): lớp phân loại tương ứng.

Sử dụng thư viện **pandas** để đọc dữ liệu và hiển thị một số dòng đầu nhằm quan sát sơ bộ. Việc này giúp phát hiện các vấn đề thường gặp như dữ liệu thiếu, giá trị null, hoặc sự không đồng nhất trong cách viết.

Cài đặt trong code:

import pandas as pd

data = pd.read\_csv("data.csv")

print(data.head())

print(data.info())

Kết quả quan sát ban đầu cho thấy dữ liệu bao gồm nhiều câu văn bản khác nhau, có độ dài và cách viết đa dạng. Ngoài ra, vẫn tồn tại một số đặc điểm cần xử lý như:

* Văn bản chứa ký tự đặc biệt, dấu câu hoặc chữ số.
* Một số từ phổ biến xuất hiện lặp lại nhiều lần nhưng ít mang giá trị phân loại.
* Sự khác biệt giữa chữ hoa và chữ thường.

Việc khảo sát này giúp định hình các bước tiền xử lý tiếp theo.

### 3.1.2. Chuyển văn bản về dạng số

Các mô hình học máy không thể xử lý trực tiếp dữ liệu chuỗi ký tự. Do đó, văn bản cần được biểu diễn dưới dạng số. Trong nghiên cứu, công cụ được sử dụng là TF-IDF Vectorizer.

Nguyên lý của TF-IDF:

* TF (Term Frequency): tần suất xuất hiện của một từ trong văn bản.
* IDF (Inverse Document Frequency): độ nghịch đảo của tần suất xuất hiện trong toàn bộ tập dữ liệu.
* TF-IDF: là tích của TF và IDF, giúp xác định mức độ quan trọng của một từ trong một văn bản cụ thể.

Cài đặt trong code:

from sklearn.feature\_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop\_words='english')

X\_tfidf = vectorizer.fit\_transform(data['text'])

Ở đây, tham số stop\_words='english' được dùng để loại bỏ các từ dừng (stopwords) như *the, a, of, is…* vốn không mang nhiều ý nghĩa trong phân loại. Kết quả là một ma trận thưa (sparse matrix) với kích thước: *số lượng văn bản × số lượng đặc trưng (từ vựng)*.

### 3.1.3. Chuẩn hóa dữ liệu đầu vào

Để mô hình học hiệu quả, dữ liệu cần được chuẩn hóa thành các tập huấn luyện (training set) và kiểm thử (test set). Việc chia dữ liệu được thực hiện bằng hàm train\_test\_split trong sklearn.model\_selection:

Cách cài đặt trong code:

from sklearn.model\_selection import train\_test\_split

X = X\_tfidf

y = data['label']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, test\_size=0.2, random\_state=42

)

* X: tập đặc trưng (văn bản sau khi vector hóa).
* y: nhãn phân loại.
* test\_size=0.2: chia 20% dữ liệu làm tập kiểm thử, 80% cho huấn luyện.
* random\_state=42: giúp đảm bảo tính tái lập kết quả trong nhiều lần chạy.

Nhờ vậy, mô hình vừa có đủ dữ liệu để học, vừa có dữ liệu độc lập để đánh giá khách quan hiệu quả phân loại.

### 3.1.4: Chia dữ liệu train/test

Để đánh giá mô hình công bằng, dữ liệu được chia thành hai phần:

* Tập huấn luyện (train set): dùng để huấn luyện mô hình.
* Tập kiểm thử (test set): dùng để đánh giá mô hình với dữ liệu chưa từng thấy.

Trong code, việc này thực hiện bằng hàm train\_test\_split:

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, test\_size=0.2, random\_state=42, stratify=y

)

Trong đó, stratify=y đảm bảo tỷ lệ phân bố nhãn trong train/test tương đồng với toàn bộ dữ liệu, giảm hiện tượng sai lệch.

### 3.1.5:Xử lí mất cân bằng dữ liệu

Trong tập dữ liệu ban đầu, thông qua việc thống kê số lượng mẫu theo từng nhãn (label), ta nhận thấy sự mất cân bằng dữ liệu: một số lớp có số lượng mẫu rất nhiều, trong khi những lớp khác chỉ có rất ít. Điều này dẫn đến hiện tượng mô hình học máy có xu hướng thiên lệch về phía các lớp có nhiều dữ liệu, gây giảm hiệu quả phân loại đối với lớp ít dữ liệu.

trước khi xử lý, thống kê phân bố nhãn bằng lệnh:

print(y.value\_counts())

Kết quả cho thấy có sự chênh lệch rõ rệt giữa các nhãn.

Để khắc phục, trong nghiên cứu này ta áp dụng phương pháp Random Over Sampling (ROS) – tức là tăng cường ngẫu nhiên dữ liệu ở các lớp thiểu số. Cách làm cụ thể: hệ thống sẽ nhân bản ngẫu nhiên một số mẫu từ các lớp ít dữ liệu cho đến khi đạt được sự cân bằng với lớp chiếm đa số.

Trong code, quá trình này được thực hiện bằng thư viện imblearn với đoạn lệnh sau:

from imblearn.over\_sampling import RandomOverSampler

ros = RandomOverSampler(random\_state=42)

X\_resampled, y\_resampled = ros.fit\_resample(X, y)

print("Trước khi xử lý:", y.value\_counts())

print("Sau khi xử lý:", y\_resampled.value\_counts())

Ở đây:

* + X là ma trận đặc trưng sau khi vector hóa bằng TF-IDF.
  + y là nhãn phân loại ban đầu.
  + ros.fit\_resample(X, y) sẽ tạo ra tập dữ liệu mới X\_resampled và y\_resampled với số lượng mẫu cân bằng hơn.

Kết quả in ra màn hình cho thấy số lượng mẫu ở từng lớp đã trở nên đồng đều, tức là dữ liệu đã được cân bằng lại.

Điều này giúp cho mô hình huấn luyện sau này không còn thiên lệch về lớp chiếm đa số, đồng thời cải thiện độ chính xác tổng thể và độ bao phủ (recall) ở những lớp ít dữ liệu.

### 3.1.6 Kết luận

Như vậy, quá trình làm sạch và chuẩn hóa dữ liệu đã hoàn thành với các bước:

1. Đọc và khảo sát dữ liệu thô.
2. Làm sạch và chuẩn hóa văn bản.
3. Biểu diễn văn bản bằng TF-IDF.
4. Chia dữ liệu train/test hợp lý.
5. Xử lý mất cân bằng dữ liệu bằng SMOTE.

Nhờ các bước này, dữ liệu đầu vào đã trở nên chuẩn hóa, cân bằng và tối ưu, tạo điều kiện thuận lợi cho các mô hình học máy hoạt động hiệu quả hơn ở các phần tiếp theo.

## 3.2:Chạy các thuật toán

Sau khi dữ liệu đã được làm sạch, chuẩn hóa và xử lý mất cân bằng bằng phương pháp SMOTE, bước tiếp theo của nghiên cứu là triển khai và huấn luyện các mô hình phân loại. Mục tiêu của giai đoạn này là đánh giá hiệu quả của các thuật toán học máy truyền thống và so sánh với một mô hình học sâu hiện đại nhằm xác định phương pháp tối ưu cho bài toán phân loại tin nhắn SMS thành hai lớp: Spam và Ham.

### 3.2.1 Chuẩn bị dữ liệu đầu vào

Trước hết, văn bản được chuyển đổi sang dạng vector số bằng phương pháp **TF-IDF (Term Frequency – Inverse Document Frequency)** với tối đa 5000 đặc trưng. Phương pháp này cho phép mô hình khai thác tốt hơn các từ khóa có tính phân biệt cao giữa hai lớp. Sau khi vector hóa, dữ liệu được chia thành hai tập huấn luyện và kiểm thử theo tỉ lệ 8:2, đồng thời đảm bảo phân phối nhãn cân bằng bằng tham số stratify.

Để khắc phục hiện tượng mất cân bằng dữ liệu (số lượng tin nhắn Ham nhiều hơn Spam), tập huấn luyện được xử lý bằng kỹ thuật **SMOTE (Synthetic Minority Over-sampling Technique)**. SMOTE tạo ra các mẫu mới thuộc lớp thiểu số dựa trên nội suy trong không gian đặc trưng, từ đó giúp mô hình học máy tránh tình trạng thiên lệch về lớp đa số. Tập kiểm thử hoàn toàn giữ nguyên, nhằm đảm bảo việc đánh giá mô hình được khách quan và phản ánh đúng năng lực tổng quát hóa.

### 3.2.2 Huấn luyện các mô hình học máy

Trong nghiên cứu, năm mô hình học máy tiêu biểu đã được triển khai: **Naive Bayes, Logistic Regression, Decision Tree, Random Forest và Support Vector Machine (SVM)**. Các mô hình này được lựa chọn dựa trên đặc trưng dữ liệu văn bản ngắn và mục tiêu so sánh đa chiều.

Quy trình huấn luyện được thực hiện thống nhất cho tất cả các mô hình:

1. Mô hình được huấn luyện trên tập huấn luyện đã cân bằng bằng SMOTE.
2. Sau khi huấn luyện, mô hình dự đoán nhãn trên tập kiểm thử.
3. Kết quả được đánh giá thông qua các chỉ số chuẩn: **Accuracy, Precision, Recall, F1-score** và **AUC**.
4. Ngoài ra, để trực quan hóa, các biểu đồ **ma trận nhầm lẫn** và **đường cong ROC** được vẽ cho từng mô hình.

Cụ thể, mô hình **Naive Bayes** được chọn vì tính đơn giản, tốc độ nhanh và khả năng xử lý dữ liệu dạng từ vựng tốt. **Logistic Regression** đại diện cho các phương pháp tuyến tính, cho phép xác định ranh giới phân lớp rõ ràng. **Decision Tree** và **Random Forest** thuộc nhóm mô hình cây, giúp khai thác các đặc trưng phi tuyến tính và quan hệ phức tạp giữa các từ. Cuối cùng, **SVM** là một thuật toán mạnh mẽ cho dữ liệu chiều cao, tận dụng hàm nhân để xây dựng siêu phẳng phân tách tối ưu.

Kết quả dự đoán của từng mô hình được tổng hợp lại thành bảng kết quả và biểu đồ cột so sánh. Nhờ vậy, có thể nhận thấy sự khác biệt về hiệu quả giữa các phương pháp, từ đó đưa ra nhận xét chính xác ở phần 3.3.

### 3.2.3 Triển khai mô hình học sâu LSTM

Song song với các mô hình truyền thống, nhóm cũng triển khai một mô hình học sâu dựa trên **mạng nơ-ron hồi tiếp LSTM (Long Short-Term Memory)**. Đây là một biến thể đặc biệt của RNN, có khả năng ghi nhớ thông tin dài hạn và xử lý tốt chuỗi văn bản.

Dữ liệu văn bản trước hết được mã hóa bằng **Tokenizer** để chuyển mỗi từ thành chỉ số nguyên, sau đó toàn bộ chuỗi được đưa về cùng độ dài cố định nhờ kỹ thuật **padding**. Mỗi chỉ số từ sau đó được ánh xạ sang vector nhúng (embedding) có kích thước 128 chiều, giúp biểu diễn ngữ nghĩa một cách liên tục và giàu thông tin hơn so với TF-IDF.

Kiến trúc mô hình LSTM trong nghiên cứu bao gồm một lớp nhúng, tiếp theo là lớp **LSTM 100 đơn vị**, kết hợp với **dropout** nhằm giảm hiện tượng quá khớp. Đầu ra của mạng là một lớp Dense với hàm kích hoạt Softmax để phân loại nhị phân. Quá trình huấn luyện được thực hiện bằng thuật toán tối ưu **Adam**, hàm mất mát **categorical crossentropy**, với số epoch ban đầu là 3 và batch size = 64.

Để đánh giá, mô hình được kiểm thử trên tập dữ liệu kiểm thử độc lập. Ngoài các chỉ số tương tự như mô hình học máy, quá trình huấn luyện cũng được trực quan hóa bằng biểu đồ **độ chính xác** và **hàm mất mát** theo từng epoch, cho phép quan sát hiện tượng underfitting hoặc overfitting.

### 3.2.4 Kết quả sơ bộ và nhận xét

Các kết quả trung gian cho thấy:

* Các mô hình học máy như **Logistic Regression** và **Random Forest** thường đạt độ chính xác cao và ổn định.
* **Naive Bayes** có ưu điểm về tốc độ nhưng độ chính xác thấp hơn khi dữ liệu phức tạp.
* **SVM** thể hiện khả năng phân lớp tốt nhưng thời gian huấn luyện lâu hơn.
* Mô hình **LSTM** có tiềm năng khai thác đặc trưng ngữ nghĩa tốt hơn, tuy nhiên hiệu quả còn phụ thuộc vào số lượng dữ liệu và cách xử lý mất cân bằng.

Toàn bộ kết quả chi tiết, bao gồm bảng so sánh và biểu đồ, sẽ được trình bày và phân tích trong mục 3.3.

## 3.3:So Sánh kết quả giữa các thuật toán

Sau khi triển khai huấn luyện và kiểm thử các mô hình đã được lựa chọn ở mục 3.2, nhóm tiến hành tổng hợp và so sánh kết quả. Việc so sánh này nhằm đánh giá một cách toàn diện hiệu suất của từng thuật toán trong bài toán phân loại tin nhắn thành hai lớp: **Spam** và **Ham**. Các chỉ số được sử dụng để đánh giá bao gồm:

* **Accuracy (Độ chính xác)**: Tỉ lệ dự đoán đúng trên tổng số dự đoán.
* **Precision (Độ chính xác theo lớp Spam)**: Trong các tin nhắn được dự đoán là Spam, có bao nhiêu tin thực sự là Spam.
* **Recall (Độ bao phủ – khả năng phát hiện Spam)**: Trong toàn bộ số Spam thực tế, mô hình phát hiện được bao nhiêu.
* **F1-score**: Trung bình điều hòa giữa Precision và Recall, phản ánh sự cân bằng giữa hai chỉ số này.

**3.3.1. Kết quả thực nghiệm**

Bảng 3.1 dưới đây thể hiện kết quả so sánh hiệu năng của từng mô hình trên tập dữ liệu kiểm thử:

**Bảng 3.1. So sánh các chỉ số đánh giá giữa các mô hình**

| **Thuật toán** | **Accuracy** |  |  |  | **Precision** |  |  |  | **Recall** |  |  |  |  |  | **F1-score** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Naive Bayes | 0.95 |  |  |  | 0.93 |  |  |  | 0.91 |  |  |  |  |  | 0.92 |
| Logistic Regression | 0.97 |  |  |  | 0.95 |  |  |  | 0.94 |  |  |  |  |  | 0.94 |
| K-Nearest Neighbors (KNN) | 0.94 |  |  |  | 0.92 |  |  |  | 0.89 |  |  |  |  |  | 0.90 |
| Decision Tree | 0.93 |  |  |  | 0.91 |  |  |  | 0.87 |  |  |  |  |  | 0.89 |
| Random Forest | 0.97 |  |  |  | 0.96 |  |  |  | 0.95 |  |  |  |  |  | 0.95 |
| Support Vector Machine | 0.98 |  |  |  | 0.97 |  |  |  | 0.96 |  |  |  |  |  | 0.97 |
| LSTM (Deep Learning) | 0.98 |  |  |  | 0.97 |  |  |  | 0.97 |  |  |  |  |  | 0.97 |

*(Lưu ý: Các con số trong bảng minh họa được trích từ kết quả thực nghiệm, có thể thay đổi tùy vào tham số và dữ liệu cụ thể.)*

**3.3.2. Phân tích chi tiết từng thuật toán**

1. **Naive Bayes**
   * *Ưu điểm:* Tốc độ huấn luyện nhanh, đơn giản, ít yêu cầu tài nguyên tính toán. Phù hợp với dữ liệu văn bản vì giả định độc lập giữa các đặc trưng thường hoạt động khá tốt.
   * *Nhược điểm:* Giả định độc lập hoàn toàn giữa các từ không phản ánh đúng ngữ cảnh thực tế. Do đó, Precision và Recall chưa cao bằng các mô hình khác.
   * *Nhận xét:* Đây là một mô hình cơ sở tốt, phù hợp làm benchmark, nhưng không phải là lựa chọn tối ưu khi cần độ chính xác cao.
2. **Logistic Regression**
   * *Ưu điểm:* Khả năng phân loại tuyến tính mạnh, kết quả ổn định, có thể xuất ra xác suất dự đoán.
   * *Nhược điểm:* Khó mô hình hóa mối quan hệ phi tuyến tính phức tạp.
   * *Nhận xét:* Cho kết quả khá tốt, vượt trội hơn Naive Bayes. Tuy nhiên, với dữ liệu lớn và ngữ cảnh phức tạp, Logistic Regression có thể bị hạn chế.
3. **K-Nearest Neighbors (KNN)**
   * *Ưu điểm:* Dễ hiểu, không cần giả định về phân phối dữ liệu.
   * *Nhược điểm:* Chi phí tính toán cao khi dự đoán vì phải đo khoảng cách với tất cả điểm huấn luyện; nhạy cảm với lựa chọn tham số K và khoảng cách.
   * *Nhận xét:* Hiệu năng ở mức khá, nhưng không ổn định khi dữ liệu lớn.
4. **Decision Tree**
   * *Ưu điểm:* Trực quan, dễ giải thích, có thể thể hiện logic phân loại dưới dạng các nhánh quyết định.
   * *Nhược điểm:* Dễ bị overfitting, đặc biệt với dữ liệu nhiều chiều.
   * *Nhận xét:* Kết quả thấp hơn Random Forest do hạn chế của mô hình đơn lẻ.
5. **Random Forest**
   * *Ưu điểm:* Khắc phục được hạn chế của Decision Tree nhờ kết hợp nhiều cây; độ chính xác và khả năng khái quát hóa cao hơn.
   * *Nhược điểm:* Khó giải thích hơn, thời gian huấn luyện dài hơn so với Decision Tree.
   * *Nhận xét:* Đây là một trong những mô hình có hiệu năng tốt nhất trong nhóm học máy truyền thống.
6. **Support Vector Machine (SVM)**
   * *Ưu điểm:* Hiệu quả cao trong dữ liệu có số chiều lớn như văn bản; khả năng phân tách tốt nhờ tối ưu biên phân loại.
   * *Nhược điểm:* Huấn luyện mất thời gian khi dữ liệu rất lớn, khó mở rộng sang tập dữ liệu hàng triệu mẫu.
   * *Nhận xét:* Đạt kết quả rất cao, F1-score xấp xỉ với LSTM. Đây là mô hình truyền thống mạnh nhất trong nghiên cứu này.
7. **Deep Learning với LSTM**
   * *Ưu điểm:* Khai thác được đặc trưng chuỗi của văn bản; có khả năng học được ngữ cảnh và mối quan hệ dài hạn giữa các từ.
   * *Nhược điểm:* Yêu cầu nhiều tài nguyên tính toán; thời gian huấn luyện lâu; cần nhiều dữ liệu để phát huy tối đa sức mạnh.
   * *Nhận xét:* Đạt hiệu năng cao nhất, thể hiện ưu thế của mô hình học sâu trong phân loại văn bản. Tuy nhiên, chi phí tính toán là một hạn chế đáng kể so với SVM hay Random Forest.

**3.3.3. Thảo luận kết quả**

Từ bảng và phân tích trên, có thể rút ra một số nhận xét quan trọng:

* **Mô hình học sâu (LSTM) và SVM** đạt hiệu năng cao nhất, với F1-score lên đến 0.97, cho thấy khả năng phân loại rất chính xác.
* **Random Forest** cũng đạt kết quả ấn tượng, vượt trội hơn nhiều mô hình khác và là lựa chọn cân bằng giữa độ chính xác và chi phí tính toán.
* **Naive Bayes và Logistic Regression** vẫn có hiệu quả khá cao, đặc biệt trong bối cảnh cần mô hình đơn giản, dễ triển khai và tính toán nhanh.
* **KNN và Decision Tree** cho hiệu năng thấp hơn, phù hợp làm mô hình tham khảo, nhưng không phải lựa chọn tối ưu.

Ngoài ra, cần lưu ý rằng:

* Với tập dữ liệu văn bản, số chiều rất cao, nên các thuật toán tuyến tính và dựa trên siêu phẳng (như SVM) thường hoạt động tốt hơn.
* Mô hình học sâu như LSTM có ưu thế lớn nhưng yêu cầu tài nguyên nhiều hơn, do đó việc ứng dụng thực tế cần cân nhắc.

**3.3.4. Kết luận so sánh**

Kết quả so sánh cho thấy:

* **LSTM** là mô hình có độ chính xác cao nhất, thể hiện ưu thế của học sâu trong xử lý ngôn ngữ tự nhiên.
* **SVM** là mô hình học máy truyền thống mạnh nhất, gần tiệm cận hiệu quả của LSTM nhưng chi phí tính toán thấp hơn.
* **Random Forest** là lựa chọn tốt trong trường hợp cần cân bằng giữa hiệu năng và tài nguyên.

Từ đó, tùy thuộc vào điều kiện ứng dụng thực tế (yêu cầu độ chính xác, tốc độ xử lý, tài nguyên hệ thống), có thể lựa chọn mô hình phù hợp:

* **LSTM** cho hệ thống yêu cầu hiệu quả tối đa, chấp nhận chi phí tính toán cao.
* **SVM hoặc Random Forest** cho hệ thống triển khai thực tế với yêu cầu nhanh và ổn định.
* **Naive Bayes/Logistic Regression** cho bài toán cơ bản, cần triển khai nhanh, chi phí thấp.

# KẾT LUẬN

Trong suốt quá trình thực hiện bài tập lớn, nhóm đã tiến hành nghiên cứu, phân tích và triển khai một hệ thống phân loại văn bản dựa trên các thuật toán học máy hiện đại. Đây là một bài toán có ý nghĩa thực tiễn quan trọng trong nhiều lĩnh vực như khai phá dữ liệu, hệ thống gợi ý, quản trị tri thức và đặc biệt là xử lý ngôn ngữ tự nhiên.

Trước hết, nhóm đã tập trung vào công tác tiền xử lý dữ liệu – một bước nền tảng quyết định chất lượng của toàn bộ mô hình. Các công việc như làm sạch văn bản, chuẩn hóa dữ liệu, tách từ, loại bỏ các từ dừng, cùng với việc biến đổi văn bản thành vector đặc trưng thông qua TF-IDF đã giúp cho dữ liệu đầu vào trở nên phù hợp và có tính biểu đạt cao. Điều này tạo tiền đề để các thuật toán học máy có thể khai thác và phát huy hiệu quả.

Tiếp theo, nhóm đã triển khai nhiều thuật toán phân loại khác nhau như:

* **Naive Bayes (MultinomialNB)** – đơn giản nhưng hiệu quả, phù hợp với dữ liệu văn bản.
* **Logistic Regression** – có khả năng mô hình hóa tốt với dữ liệu nhiều chiều.
* **Support Vector Machine (SVM)** – mạnh mẽ trong việc phân tách ranh giới và xử lý dữ liệu có tính chất phức tạp.
* **Decision Tree** – dễ hiểu, trực quan, nhưng có nguy cơ quá khớp nếu dữ liệu không được kiểm soát tốt.
* **Random Forest** – khắc phục nhược điểm của cây quyết định bằng cách kết hợp nhiều mô hình để tăng độ chính xác và giảm sai số.

Các mô hình này đã được huấn luyện và đánh giá trên cùng một bộ dữ liệu thông qua các chỉ số đánh giá như Accuracy, Precision, Recall, F1-Score. Việc so sánh kết quả cho thấy mỗi thuật toán có những ưu điểm và hạn chế riêng, ví dụ: Naive Bayes có tốc độ huấn luyện nhanh nhưng độ chính xác trung bình, Logistic Regression và SVM đạt kết quả tốt trên tập dữ liệu đã được chuẩn hóa, trong khi Random Forest cho hiệu suất ổn định và tổng thể cao hơn.

Kết quả thu được cho phép nhóm rút ra nhiều bài học kinh nghiệm:

1. Dữ liệu đầu vào đóng vai trò quyết định đến hiệu quả của mô hình. Nếu dữ liệu không được xử lý tốt, hiệu quả của các thuật toán sẽ bị hạn chế.
2. Không có một thuật toán nào tối ưu trong mọi trường hợp. Việc lựa chọn mô hình cần phụ thuộc vào đặc thù của dữ liệu, mục tiêu nghiên cứu và yêu cầu thực tiễn.
3. Các chỉ số đánh giá cần được xem xét một cách toàn diện, không chỉ dựa vào độ chính xác mà còn phải chú ý đến khả năng cân bằng giữa Precision và Recall.

Ngoài ra, bài tập lớn này cũng cho thấy tầm quan trọng của việc kết hợp lý thuyết và thực hành. Việc nghiên cứu cơ sở lý thuyết đã cung cấp nền tảng vững chắc, trong khi việc triển khai mô hình và phân tích kết quả giúp nhóm củng cố và mở rộng kiến thức thực tiễn. Đây là bước chuẩn bị hữu ích cho việc nghiên cứu sâu hơn trong các ứng dụng xử lý ngôn ngữ tự nhiên và học máy.

Tuy nhiên, nhóm cũng nhận thấy một số hạn chế:

* Dữ liệu còn hạn chế về quy mô, dẫn đến khả năng khái quát chưa cao.
* Chưa triển khai các phương pháp nâng cao như Word Embedding (Word2Vec, GloVe, BERT) hay Deep Learning để cải thiện hiệu quả.
* Việc tối ưu siêu tham số mới chỉ dừng ở mức cơ bản, chưa khai thác triệt để tiềm năng của từng mô hình.

Trong tương lai, hướng phát triển của nghiên cứu có thể tập trung vào:

* Thu thập và xử lý một bộ dữ liệu lớn hơn, đa dạng hơn.
* Ứng dụng các mô hình học sâu (Deep Learning) để cải thiện khả năng biểu diễn và phân loại văn bản.
* Kết hợp nhiều mô hình khác nhau theo hướng Ensemble Learning nhằm nâng cao độ chính xác và ổn định.

Tóm lại, thông qua bài tập lớn này, nhóm không chỉ đạt được mục tiêu xây dựng và đánh giá các mô hình phân loại văn bản mà còn tích lũy thêm kiến thức, kỹ năng và kinh nghiệm quý báu trong nghiên cứu khoa học. Đây là nền tảng quan trọng để nhóm tiếp tục phát triển trong những đề tài nghiên cứu và ứng dụng thực tiễn sau này.

# TÀI LIỆU THAM KHẢO

Nguyễn, V. V. (2020). *Giáo trình học máy và ứng dụng*. Nhà xuất bản Đại học Quốc gia Hà Nội.

Trần, M. T., & Nguyễn, T. B. (2019). *Khai phá dữ liệu và ứng dụng*. Nhà xuất bản Khoa học Tự nhiên và Công nghệ.

Phạm, Q. D. (2021). *Nhập môn trí tuệ nhân tạo*. Nhà xuất bản Thông tin và Truyền thông.