TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA CƠ KHÍ CHẾ TẠO MÁY
BỘ MÔN CƠ SỞ THIẾT KẾ MÁY

ଓଓଓଓଓଓଓଓ 0o0 ଌଌଌଌଌଌଌଌ

BÁO CÁO CUỐI KỲ
HỌC MÁY – 232MALE337029
ĐỀ TÀI: EMAIL-BASED SPAM DETECTION

Nhóm 02CLC

GVHD: TS Vũ Quang Huy

HỌ & TÊN SINH VIÊN	**MSSV**
Phạm Trung Hiếu	21146459
Lê Đình Tấn An	21143104

Tp. HCM, Ngày 26, tháng 5, năm 2024

Mục lục

I. Tổng quan

1.1. Lý do chọn đề tài

Sự ra đời của internet đã thay đổi đáng kể cách thức liên lạc, làm cho email trở thành một công cụ không thể thiếu trong cả đời sống cá nhân và công việc chuyên nghiệp. Tuy nhiên, sự phổ biến của email cũng mang lại những thách thức, đặc biệt là sự bùng nổ của thư rác. Thư rác, những tin nhắn không mong muốn được gửi đi hàng loạt, gây ra các vấn đề nghiêm trọng như làm đầy hộp thư với nội dung không liên quan, làm chậm tốc độ internet, và tiềm ẩn nguy cơ đánh cắp thông tin cá nhân. Giải quyết các thách thức này là điều cần thiết để duy trì tính toàn vẹn và khả năng sử dụng của email. Bài tiểu luận này khám phá các cơ chế và phương pháp phát hiện và lọc thư rác, tập trung vào các kỹ thuật học máy và hiệu quả của chúng.

Phát hiện thư rác và tin nhắn rác là một vấn đề quan trọng trong lĩnh vực an ninh mạng. Thư rác không chỉ làm giảm hiệu suất làm việc mà còn có thể chứa các phần mềm độc hại gây hại cho người dùng. Việc sử dụng học máy để phát hiện thư rác và tin nhắn rác đã trở nên phổ biến nhờ khả năng học và phân loại dựa trên dữ liệu lớn. Với các thuật toán ngày càng tiên tiến, học máy có thể phát hiện các mẫu và đặc điểm của thư rác một cách hiệu quả, giúp giảm thiểu rủi ro cho người dùng.

Với sự phát triển nhanh chóng của công nghệ thông tin, lượng thư rác và tin nhắn rác ngày càng tăng. Điều này không chỉ ảnh hưởng đến trải nghiệm người dùng mà còn tiềm ẩn nhiều nguy cơ về an ninh mạng. Thư rác có thể chứa các liên kết độc hại hoặc các tệp đính kèm nguy hiểm, đe dọa đến bảo mật thông tin cá nhân và doanh nghiệp. Việc nghiên cứu và phát triển các phương pháp phát hiện thư rác và tin nhắn rác hiệu quả là cần thiết để bảo vệ người dùng khỏi các mối đe dọa này, đảm bảo môi trường mạng an toàn và lành mạnh.

Việc sử dụng học máy không chỉ giúp phát hiện thư rác một cách hiệu quả mà còn cải thiện khả năng tự học và thích nghi với các hình thức tấn công mới. Các mô hình học máy có thể được huấn luyện liên tục với dữ liệu mới, giúp hệ thống ngày càng thông minh hơn và khả năng nhận diện thư rác ngày càng chính xác hơn. Điều này sẽ góp phần quan trọng vào việc bảo vệ người dùng và duy trì sự ổn định của hệ thống email toàn cầu.

1.2. Mục tiêu

Mục tiêu nghiên cứu là phát triển một mô hình học máy hiệu quả để phân loại thư rác và tin nhắn rác. Cụ thể, nghiên cứu tập trung vào việc áp dụng nhiều kỹ thuật học máy khác nhau trên mô hình, từ đó trực quan hóa và đánh giá các kỹ thuật học máy để lựa chọn ra kỹ thuật cho khả năng đạt được độ chính xác cao nhất trong mô hình phân loại. Từ các kết quả đánh giá, kỹ thuật học máy nào đạt được độ chính xác cao nhất và hiệu suất tốt nhất sẽ được lựa chọn để triển khai vào mô hình phân loại thực tế.

1.3. Giới hạn đề tài

-Nghiên cứu tập trung vào hai lĩnh vực chính: phát hiện thư rác email và phát hiện tin nhắn rác SMS. Các mô hình sẽ được huấn luyện và kiểm tra trên các tập dữ liệu phổ biến trong lĩnh vực này.

- Do hạn chế về thời gian, đề tài này sẽ được giới lại và tập trung vào việc tạo ra mô hình sử dụng các thuật toán tối ưu nhất dựa trên các số liệu đánh giá. Sau đó, sẽ phát triển một giao diện đơn giản để kiểm thử mô hình được cho là tốt nhất thay vì là một hệ thống Email_based_Spam_Detection hoàn thiện.

II. Cơ sở lý thuyết

2.1. Thách thức gặp phải

Phát hiện thư rác và tin nhắn rác là một vấn đề phức tạp do sự biến đổi liên tục của nội dung và hình thức của spam. Các thách thức chính bao gồm:

- Sự biến đổi liên tục của nội dung spam: Người gửi spam liên tục thay đổi nội dung và cách thức gửi để vượt qua các bộ lọc. Điều này đòi hỏi các hệ thống phát hiện phải luôn được cập nhật và thích ứng nhanh chóng.

-Đa dạng về nguồn gốc và hình thức: Spam có thể đến từ nhiều nguồn khác nhau và sử dụng nhiều hình thức khác nhau, từ văn bản thuần túy, hình ảnh đến liên kết và tệp đính kèm.

-Khối lượng lớn dữ liệu: Hàng ngày, có hàng triệu email và tin nhắn được gửi và nhận, tạo ra một khối lượng dữ liệu khổng lồ cần phải được xử lý một cách hiệu quả và nhanh chóng. Phương pháp giải quyết

2.2. Khảo Sát Tài Liệu

Trong bài báo của Shukor Bin Abd Razak và Ahmad Fahrulrazie Bin Mohamad với tiêu đề "Identification of Spam Email Based on Information from Email Header"[1], được trình bày tại Hội nghị Quốc tế lần thứ 13, các tác giả đã khám phá nhiều đặc điểm có trong tiêu đề email mà có thể được sử dụng để xác định và phân loại tin nhắn spam một cách hiệu quả. Các đặc điểm này được lựa chọn dựa trên hiệu quả của chúng trong việc phát hiện spam qua các nhà cung cấp dịch vụ email khác nhau, cụ thể là Yahoo Mail, Gmail và Hotmail. Bằng cách phân tích và chung hóa các đặc điểm này, các tác giả hướng đến đề xuất một cơ chế phát hiện spam chung áp dụng cho tất cả các nhà cung cấp email chính. Các bộ phân loại Bayes đã được sử dụng rộng rãi trong phát hiện thư rác. Các bộ phân loại này đánh giá xác suất một email là thư rác dựa trên sự xuất hiện của các từ trong email.

Những điểm chính từ bài báo bao gồm:

-Đặc điểm của Tiêu đề Email: Nghiên cứu tập trung vào các yếu tố khác nhau trong tiêu đề email, chẳng hạn như địa chỉ người gửi, dòng chủ đề và siêu dữ liệu, những yếu tố quan trọng trong việc phân biệt spam với email hợp lệ.

-Lựa chọn Dựa trên Hiệu suất: Các đặc điểm được chọn dựa trên hiệu suất của chúng trong các kịch bản phát hiện spam trước đó, đảm bảo rằng các chỉ báo hiệu quả nhất được sử dụng.

-Tính Áp dụng Qua Các Nhà Cung Cấp: Bằng cách so sánh các đặc điểm qua Yahoo Mail, Gmail và Hotmail, các tác giả tìm cách thiết lập một khung phát hiện spam thống nhất có thể áp dụng rộng rãi, không phụ thuộc vào nhà cung cấp dịch vụ email.

-Cơ Chế Phát Hiện Chung: Mục tiêu cuối cùng là tạo ra một cơ chế phát hiện spam chung, tận dụng các đặc điểm tiêu đề đã xác định, làm cho nó có thể thích ứng với các nền tảng email khác nhau và cải thiện hiệu quả chung của hệ thống lọc spam.

Nghiên cứu nhấn mạnh tầm quan trọng của thông tin tiêu đề email trong việc phát hiện spam và nhằm nâng cao độ tin cậy cũng như hiệu quả của các bộ lọc spam qua nhiều dịch vụ email bằng cách sử dụng một phương pháp chuẩn hóa.

Trong bài báo của Mohammed Reza Parsei và Mohammed Salehi[2] với tiêu đề "E-Mail Spam Detection Based on Part of Speech Tagging," được trình bày tại Hội nghị Quốc tế lần thứ 2 về Kỹ thuật Dựa trên Tri thức và Đổi mới (KBEI) năm 2015, các tác giả đã giới thiệu

một phương pháp mới dựa trên tần suất lặp lại của các từ. Phương pháp này bao gồm các bước chính sau:

-Gắn Thẻ Các Câu Chứa Từ Khóa: Các câu quan trọng trong email đến, tức là những câu chứa từ khóa, phải được gắn thẻ.

-Xác Định Vai Trò Ngữ Pháp: Vai trò ngữ pháp của tất cả các từ trong câu cần được xác định.

-Tạo Vector: Cuối cùng, các từ sẽ được đưa vào một vector để xác định mức độ tương đồng giữa các email nhận được.

-Sử Dụng Thuật Toán K-Mean: Thuật toán K-Mean được sử dụng để phân loại email nhận được. Phương pháp xác định vector được sử dụng để xác định email thuộc về loại nào.

-Phương pháp này tập trung vào việc phân tích tần suất và vai trò ngữ pháp của từ trong câu để phát hiện email spam một cách hiệu quả, sử dụng thuật toán K-Mean để phân loại email dựa trên sự tương đồng giữa chúng.

Trong bài báo của Sunil B. Rathod và Tareek M. Pattewar[3] với tiêu đề "Content Based Spam Detection in Email using Bayesian Classifier," được trình bày tại hội nghị IEEE ICCSP 2015, các tác giả đã mô tả về các cuộc tấn công mạng. Phishers và các kẻ tấn công độc hại thường sử dụng dịch vụ email để gửi những loại tin nhắn giả mạo, khiến người dùng mục tiêu có thể mất tiền và uy tín xã hội. Những cuộc tấn công này nhằm mục đích lấy cắp thông tin cá nhân như số thẻ tín dụng, mật khẩu và các dữ liệu bí mật khác.

Trong bài báo này, các tác giả đã sử dụng Bộ phân loại Bayesian (Bayesian Classifiers) để phát hiện spam. Phương pháp của họ bao gồm các điểm chính sau:

-Xem Xét Mọi Từ Trong Email: Phương pháp này xem xét từng từ trong email để phân tích nội dung.

-Thích Ứng Liên Tục Với Các Hình Thức Spam Mới: Bộ phân loại Bayesian liên tục thích ứng với các hình thức spam mới, giúp cải thiện hiệu quả phát hiện spam theo thời gian.

Phương pháp này tập trung vào việc sử dụng Bayesian Classifiers để phân tích và xác định các email spam dựa trên nội dung của chúng, đảm bảo rằng hệ thống có thể thích ứng và phản ứng với các chiến thuật spam mới và phức tạp hơn.

Trong bài báo của Aakash Atul Alurkar, Sourabh Bharat Ranade, Shreeya Vijay Joshi, Siddhesh Sanjay Ranade, Piyush A. Sonewa, Parikshit N. Mahalle, và Arvind V. Deshpande[4] với tiêu đề "A Proposed Data Science Approach for Email Spam Classification using Machine Learning Techniques," được công bố năm 2017, các tác giả đã đề xuất một hệ thống sử dụng các kỹ thuật học máy để phát hiện các mẫu từ khóa lặp đi lặp lại, được phân loại là spam. Hệ thống cũng đề xuất việc phân loại email dựa trên nhiều tham số khác trong cấu trúc của chúng như Cc/Bcc, domain và header. Mỗi tham số sẽ được coi là một đặc trưng khi áp dụng vào thuật toán học máy.

Các điểm chính của phương pháp được đề xuất bao gồm:

- Phát Hiện Từ Khóa Lặp Lại: Sử dụng các kỹ thuật học máy để phát hiện các mẫu từ khóa lặp lại trong email, giúp phân loại chúng là spam.
- Phân Loại Dựa Trên Nhiều Tham Số: Phân loại email không chỉ dựa trên từ khóa mà còn dựa trên các tham số khác như Cc/Bcc, domain và header của email. Mỗi tham số được xem là một đặc trưng trong quá trình áp dụng thuật toán học máy.

-Mô Hình Học Máy Tiền Huấn Luyện Với Cơ Chế Phản Hồi: Mô hình học máy được tiền huấn luyện và có cơ chế phản hồi để phân biệt giữa kết quả chính xác và kết quả mơ hồ. Điều này giúp cải thiện độ chính xác của hệ thống phát hiện spam theo thời gian.

-Xem Xét Nội Dung Email: Phương pháp này cũng xem xét nội dung của email, bao gồm các từ khóa và dấu câu thường được sử dụng, để tăng cường khả năng phát hiện spam.

Phương pháp này cung cấp một kiến trúc thay thế để triển khai bộ lọc spam, sử dụng các kỹ thuật học máy và một loạt các tham số cấu trúc của email để phân loại spam một cách hiệu quả.

Trong bài báo của Kriti Agarwal và Tarun Kumar[5] với tiêu đề "Email Spam Detection using integrated approach of Naïve Bayes and Particle Swarm Optimization," được trình bày tại Hội nghị Quốc tế lần thứ hai về Hệ thống Điều khiển và Tính toán Thông minh (ICICCS) năm 2018, các tác giả đã điều tra việc sử dụng các thuật toán so khớp chuỗi (string matching) để phát hiện email spam. Cụ thể, công trình này xem xét và so sánh hiệu quả của sáu thuật toán so khớp chuỗi nổi tiếng, bao gồm Longest Common Subsequence (LCS), Levenshtein Distance (LD), Jaro, Jaro-Winkler, Bi-gram, và TFIDF trên hai bộ dữ liệu khác nhau là Enron corpus và CSDMC2010 spam dataset.

Các điểm chính của nghiên cứu bao gồm:

-Các Thuật Toán So Khớp Chuỗi: Nghiên cứu kiểm tra sáu thuật toán so khớp chuỗi để đánh giá hiệu quả của chúng trong việc phát hiện email spam.

-Bộ Dữ Liệu Được Sử Dụng: Hai bộ dữ liệu được sử dụng trong nghiên cứu là Enron corpus và CSDMC2010 spam dataset, giúp kiểm tra tính nhất quán và hiệu quả của các thuật toán trên các nguồn dữ liệu khác nhau.

-Hiệu Quả của Thuật Toán Bi-gram: Các tác giả nhận thấy rằng thuật toán Bi-gram hoạt động tốt nhất trong việc phát hiện spam trên cả hai bộ dữ liệu.

Phương pháp của bài báo không chỉ kiểm tra từng thuật toán riêng lẻ mà còn kết hợp Naïve Bayes và Tối ưu hóa Bầy đàn (Particle Swarm Optimization) để cải thiện khả năng phát hiện spam. Kết quả cho thấy rằng việc tích hợp các thuật toán này có thể nâng cao hiệu quả phát hiện spam trong các hệ thống email.

2.3. Hệ thống đề xuất từ các cơ sở lý thuyết trên.

Trong hệ thống này, để giải quyết vấn đề spam, hệ thống phân loại spam được tạo ra nhằm xác định email spam và không spam. Vì những kẻ spam có thể gửi tin nhắn spam nhiều lần, nên rất khó để xác định mỗi lần một cách thủ công. Do đó, chúng tôi sẽ sử dụng một số chiến lược trong hệ thống đề xuất của chúng tôi để phát hiện spam. Giải pháp đề xuất không chỉ xác định từ spam mà còn xác định địa chỉ IP của hệ thống gửi tin nhắn spam để lần tiếp theo khi tin nhắn spam được gửi từ cùng hệ thống, hệ thống của chúng tôi sẽ trực tiếp nhận dạng nó là địa chỉ trong danh sách đen dựa trên địa chỉ IP.

Trong mô hình đề xuất, ứng dụng web được thực hiện bằng .NET và phát hiện spam được thực hiện bằng học máy. Ứng dụng web bao gồm các mô-đun sau:

1. Quản lý Người Dùng:

   -Đăng Ký: Người dùng lần đầu tiên sử dụng phải đăng ký. Việc đăng ký này giúp duy trì tài khoản riêng biệt cho mỗi người dùng. Đăng ký người dùng là bắt buộc trước khi họ đăng nhập.

   -Đăng Nhập: Người dùng sẽ đăng nhập vào trang chính với tên và mật khẩu đã đăng ký của họ. Khi người dùng đăng nhập thành công, trang được ủy quyền sẽ hiển thị, nếu không, sẽ hiển thị thông báo lỗi.
2. Soạn thư:

   -Đầu Vào: Người gửi sẽ soạn thư mới; người gửi cần thêm địa chỉ của người nhận, chủ đề và nội dung tin nhắn.

   -Đầu Ra: Thư sẽ được gửi đến địa chỉ người nhận đã đề cập.
3. Hộp thư đến

   Trang này sẽ lưu trữ tất cả các thư mà người dùng nhận được. Tất cả các thư nhận được sẽ được liệt kê theo thứ tự ngày tháng.

   -Đầu Vào: Trang hộp thư đến sẽ nhận tất cả các email gửi đến cá nhân.

   -Đầu Ra: Người nhận có thể mở và đọc email nhận được.
4. Thư đã gửi

   Thư mục này lưu trữ tất cả các thư đã gửi từ người dùng.

   -Đầu Vào: Người gửi sẽ soạn email và gửi đến người nhận.

   -Đầu Ra: Email đã gửi có thể được đọc lại.
5. Thùng rác

   Thư mục này sẽ lưu trữ tất cả các thư bị xóa bởi người dùng.

   -Đầu Vào: Chọn và xóa tất cả các email không mong muốn.

   -Đầu Ra: Tất cả các email đã xóa sẽ được thêm vào thùng rác. Thùng rác lưu trữ tất cả các email đã xóa.
6. Tin nhắn thoại

   -Đầu Vào: Email được gửi dưới dạng tin nhắn văn bản bởi người gửi.

   -Đầu Ra: Email được đọc thông qua ghi chú thoại bởi người nhận.
7. Thông báo ngoại tuyến

   -Đầu Vào: Người gửi gửi một email.

   -Đầu Ra: Người nhận nhận được một thông báo ngoại tuyến dưới dạng SMS.
8. Xóa cho mọi người

   -Đầu Vào: Người gửi xóa email mà họ đã gửi.

   -Đầu Ra: Email được xóa hoặc bị xóa cho cả người gửi và người nhận.
9. Đọc tin nhắn

   -Đầu Vào: Người nhận sẽ đọc email.

   -Đầu Ra: Người gửi sẽ nhận được thông báo rằng người gửi đã đọc tin nhắn.

Khi chúng tôi nhận được tin nhắn trong hộp thư đến, tin nhắn đó sẽ được xuất ra bộ dữ liệu. Tin nhắn này sẽ được phát hiện là spam hay không sử dụng Bộ phân loại Naïve Bayes (Naïve Bayes Classifier). Trước khi xác định xem tin nhắn nhận được có phải là spam hay không, mô hình phải được huấn luyện, điều này được giải thích trong phần dưới đây.

2.4. Phát hiện SPAM sử dụng học máy trong hệ thống đề xuất

Để giải quyết vấn đề spam, hệ thống phân loại spam được tạo ra nhằm xác định email spam và không spam. Các bước thực hiện và công cụ được sử dụng trong quá trình này được mô tả chi tiết dưới đây.

1. Sử Dụng Bộ Dữ Liệu từ Kaggle:

   Bộ dữ liệu từ Kaggle được sử dụng để huấn luyện thuật toán. Bộ dữ liệu này chứa nhiều trường (fields), một số trường không cần thiết sẽ bị loại bỏ.
2. Tiền Xử Lý Dữ Liệu:

   -Loại bỏ các cột không cần thiết từ bộ dữ liệu.

   -Đổi tên các cột để dễ dàng sử dụng.

   -Sử dụng các thư viện và công cụ như NLTK (Natural Language Toolkit) để xử lý văn bản, Matplotlib để vẽ đồ thị, biểu đồ histogram và biểu đồ cột, Word Cloud để hiển thị dữ liệu văn bản, Pandas để xử lý và phân tích dữ liệu, và NumPy để thực hiện các phép toán toán học và khoa học.
3. Chia Dữ Liệu Thành Tập Huấn Luyện và Tập Kiểm Tra:

   -Chia dữ liệu thành tập huấn luyện và tập kiểm tra với một tỷ lệ phần trăm nhất định cho mỗi tập.

   -Đặt lại chỉ số (index) cho tập huấn luyện và tập kiểm tra.
4. Tìm Từ Phổ Biến Nhất Trong Tin Nhắn Spam và Không Spam:

   -Sử dụng thư viện Word Cloud để tìm các từ được lặp lại nhiều nhất trong các tin nhắn spam và không spam
5. Tiền Xử Lý Tin Nhắn Đầu Vào:

   -Chuyển đổi tất cả các ký tự trong tin nhắn đầu vào thành chữ thường (lowercase).

   -Sử dụng quá trình phân tách từ (Tokenization) để chia nhỏ văn bản và loại bỏ dấu câu.
6. Áp Dụng Thuật Toán Porter Stemming:

   -Sử dụng thuật toán Porter Stemming để đưa các từ về dạng gốc (root word).
7. Tính Xác Suất Của Từ Trong Tin Nhắn Spam và Không Spam:

   -Tính toán xác suất xuất hiện của từ trong tin nhắn spam và không spam
8. Tính Tf-idf (Term Frequency-Inverse Document Frequency):

   -TF (Term Frequency): Đo lường số lần một từ xuất hiện trong một tài liệu.

   -IDF (Inverse Document Frequency): Đo lường tầm quan trọng của từ
9. Đánh giá mô hình

   - Sử dụng Bộ phân loại Naïve Bayes để huấn luyện mô hình.

   - Đánh giá hiệu suất của mô hình bằng cách tính toán độ chính xác (accuracy score).

III. Phương pháp đề xuất

3.1. Yêu cầu thiết kế

Hệ thống phát hiện thư rác cần phải:

-Phân loại chính xác tin nhắn rác và không phải rác.

-Xử lý nhanh và hiệu quả khối lượng lớn tin nhắn.

-Dễ dàng cập nhật và mở rộng khi có dữ liệu mới.

-Một giao diện để người dùng dễ dàng sử dụng.

3.2. Phương pháp để so sánh và lựa chọn dựa trên yêu cầu thiết kế

Dựa trên các yêu cầu thiết kế, việc sử dụng nhiều thuật toán học máy khác nhau và sau đó tìm ra phương pháp nào cho ra hiệu suất tốt nhất là một hướng tiếp cận có thể đem lại kết quả tốt cho bài toán phân loại tin nhắn spam và không spam.

3.3. Các bước được đề xuất cho hệ thống dựa trên yêu cầu thiết kế

Phương pháp đề xuất bao gồm các bước thực hiện cụ thể như sau:

1. Đọc và xử lý dữ liệu ban đầu
   - Đọc dữ liệu từ file CSV chứa các tin nhắn SMS.
   - Kiểm tra dữ liệu bằng các phương pháp như .sample(), .shape(), và .info().
   - Loại bỏ các cột không cần thiết và đổi tên các cột.
   - Mã hóa biến mục tiêu bằng LabelEncoder từ thư viện scikit-learn.
2. Xử lý các dữ liệu cần thiết, trùng lặp và trực quan hóa dữ liệu bằng biểu đồ.
   - Kiểm tra giá trị thiếu và loại bỏ các bản ghi trùng lặp.
   - Thêm các đặc trưng mới như số ký tự, số từ, và số câu trong mỗi tin nhắn.
   - Trực quan hóa các phân phối và mối quan hệ giữa các đặc trưng.
3. Tiền sử lý dữ liệu văn bản
   - Định nghĩa hàm để làm sạch và biến đổi văn bản, sau đó áp dụng cho cột văn bản.
   - Tạo đám mây từ và biểu đồ phân phối tần suất từ văn bản đã được tiền xử lý.
4. Vec hóa văn bản(TF-IDF)
   -Sử dụng TfidfVectorizer để chuyển đổi văn bản thành dạng số.
5. Huấn luyện, đánh giá so sánh hiệu xuất giữa các mô hình.
   - Chia dữ liệu thành tập huấn luyện và kiểm tra.
   - Định nghĩa và huấn luyện nhiều bộ phân loại khác nhau.
   - Đánh giá hiệu suất của từng bộ phân loại và so sánh chúng.
   - Triển khai các bộ phân loại kết hợp như Voting và Stacking.
6. Lưu mô hình, kiểm tra xem mô hình đã được huấn luyện hay chưa.
   - Lưu TfidfVectorizer và mô hình có hiệu suất tốt nhất bằng pickle.
   - Kiểm tra xem mô hình đã được huấn luyện hay chưa.
7. Tạo ra một giao diện người dùng đơn giản.
   -Tạo ra một ứng dụng Streamlit đơn để người dùng có thể tương tác với model đã được train.

IV. Kết quả và hướng phát triển

4.1. Tập dữ liệu

Mô tả Dữ liệu (1):

Tập dữ liệu: Bao gồm các tin nhắn SMS và thông tin về việc liệu chúng có phải là spam hay không.

-Số lượng mẫu: 5574 tin nhắn SMS.

-Số lượng đặc trưng: 2 đặc trưng chính:

-v1: Nhãn (spam hoặc ham).

-v2: Nội dung của tin nhắn.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

Hình 1. Thông tin Datasheet

4.2. Phương pháp đánh giá

Phương pháp đánh giá trong mã đã sử dụng hai metric chính là độ chính xác (accuracy) và độ chính xác (precision) để đánh giá hiệu suất của các mô hình phân loại trên tập dữ liệu kiểm tra.

4.3. Thực thi

1. Đọc và xử lý dữ liệu

- Đọc file CSV: Sử dụng thư viện pandas để đọc tập tin CSV từ đường dẫn đã cho. Mã hóa latin1 được sử dụng để xử lý các ký tự không phải ASCII trong tập tin.Hiển thị mẫu dữ liệu: Dùng phương thức sample() để hiển thị một số bản ghi mẫu từ DataFrame, giúp ta có cái nhìn tổng quan về dữ liệu.
- Hiển thị hình dạng của DataFrame: Sử dụng thuộc tính shape để hiển thị số lượng hàng và cột trong DataFrame, giúp ta biết được kích thước của dữ liệu.
- Hiển thị thông tin của DataFrame: Sử dụng phương thức info() để hiển thị thông tin về các cột trong DataFrame, bao gồm tên cột, số lượng giá trị không thiếu, và kiểu dữ liệu của mỗi cột.
- Loại bỏ các cột không cần thiết: Sử dụng phương thức drop() để loại bỏ các cột không cần thiết từ DataFrame. Trong trường hợp này, các cột 'Unnamed: 2', 'Unnamed: 3', và 'Unnamed: 4' được loại bỏ.
- Hiển thị mẫu dữ liệu sau khi loại bỏ các cột không cần thiết: Dùng lại phương thức sample() để kiểm tra lại DataFrame sau khi đã loại bỏ các cột không cần thiết, đảm bảo rằng các thay đổi được thực hiện đúng.
- Đổi tên các cột: Sử dụng phương thức rename() để đổi tên các cột trong DataFrame. Trong trường hợp này, cột 'v1' được đổi tên thành 'target' và cột 'v2' được đổi tên thành 'text'.

- Hiển thị mẫu dữ liệu sau khi đổi tên các cột: Dùng lại phương thức sample() để kiểm tra lại DataFrame sau khi đã đổi tên các cột, đảm bảo rằng các thay đổi được thực hiện đúng. Được thể hiện ở hình (2)

Kết quả được thể hiện ở hình(3)

```
#Reading and Initial Data Processing
# Đọc file CSV với mã hóa latin1
df = pd.read_csv('/content/drive/MyDrive/spam.csv', encoding='latin1')
# Hiển thị mẫu gồm 5 bản ghi
df.sample(5)
# Hiển thị hình dạng của DataFrame
df.shape
# Hiển thị thông tin của DataFrame
df.info()
# Loại bỏ 3 cột cuối không cần thiết
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'], inplace=True)
# Hiển thị mẫu gồm 5 bản ghi để xác nhận các thay đổi
df.sample(5)
# Đổi tên các cột
df.rename(columns={'v1': 'target', 'v2': 'text'}, inplace=True)
# Hiển thị mẫu gồm 5 bản ghi để xác nhận các thay đổi
df.sample(5)
```

Hình 2. Xử lý dữ liệu

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

	target	text
1611	ham	645
2632	ham	I WILL CAL YOU SIR. In meeting
3950	ham	Hi dude hw r u da realy mising u today
5495	ham	Good afternoon, my love ... How goes your day ...
3354	ham	Minimum walk is 3miles a day.

Hình 3. Kết quả xử lý dữ liệu

2. Xử lý các dữ liệu cần thiết, trùng lặp và trực quan hóa dữ liệu bằng biểu đồ.

2.1.1 Xử lý các dữ liệu cần thiết, trùng lặp. Hình (4)

- Mã hóa cột mục tiêu: Sử dụng LabelEncoder từ thư viện scikit-learn để mã hóa các nhãn của cột mục tiêu thành các số nguyên. Điều này thường được thực hiện để chuẩn hóa dữ liệu đầu vào cho các mô hình học máy.
- Hiển thị mẫu dữ liệu sau khi mã hóa: Dùng lại phương thức head() để kiểm tra một số bản ghi mẫu sau khi đã mã hóa cột mục tiêu, đảm bảo rằng quá trình mã hóa đã diễn ra đúng.
- Kiểm tra giá trị bị thiếu: Sử dụng phương thức isnull().sum() để đếm số lượng giá trị bị thiếu trong từng cột của DataFrame. Điều này giúp xác định xem có cần phải xử lý giá trị bị thiếu hay không.
- Kiểm tra giá trị trùng lặp: Sử dụng phương thức duplicated().sum() để đếm số lượng bản ghi trùng lặp trong DataFrame. Điều này giúp kiểm tra xem có bản ghi nào bị trùng lặp hay không.

- Loại bỏ các bản ghi trùng lặp: Sử dụng phương thức drop_duplicates() để loại bỏ các bản ghi trùng lặp từ DataFrame, chỉ giữ lại một bản ghi đầu tiên trong trường hợp có nhiều bản ghi trùng lặp.
- Hiển thị hình dạng của DataFrame sau khi loại bỏ bản ghi trùng lặp: Dùng thuộc tính shape để kiểm tra lại số lượng hàng và cột trong DataFrame sau khi đã loại bỏ các bản ghi trùng lặp, đảm bảo rằng quá trình loại bỏ diễn ra đúng.
Kết quả được thể hiện ở hình (5)

```
# Mã hóa cột mục tiêu
# Encode the target column
encoder = LabelEncoder()
df['target'] = encoder.fit_transform(df['target'])
# Hiển thị một số bản ghi đầu tiên để xác nhận các thay đổi
df.head()
# Kiểm tra giá trị thiếu
df.isnull().sum()
# Kiểm tra các bản ghi trùng lặp
df.duplicated().sum()
# Loại bỏ các bản ghi trùng lặp
df = df.drop_duplicates(keep='first')
# Xác nhận việc loại bỏ các bản ghi trùng lặp
df.duplicated().sum()
# Hiển thị hình dạng của DataFrame sau khi loại bỏ các bản ghi trùng lặp
df.shape
print(df)
```

Hình 4. Xử lý dữ liệu trùng lặp

```
      target                                                text
0          0  Go until jurong point, crazy.. Available only ...
1          0                      Ok lar... Joking wif u oni...
2          1  Free entry in 2 a wkly comp to win FA Cup fina...
3          0  U dun say so early hor... U c already then say...
4          0  Nah I don't think he goes to usf, he lives aro...
...      ...                                                ...
5567       1  This is the 2nd time we have tried 2 contact u...
5568       0              Will Ì_ b going to esplanade fr home?
5569       0  Pity, * was in mood for that. So...any other s...
5570       0  The guy did some bitching but I acted like i'd...
5571       0                        Rofl. Its true to its name

[5169 rows x 2 columns]
```

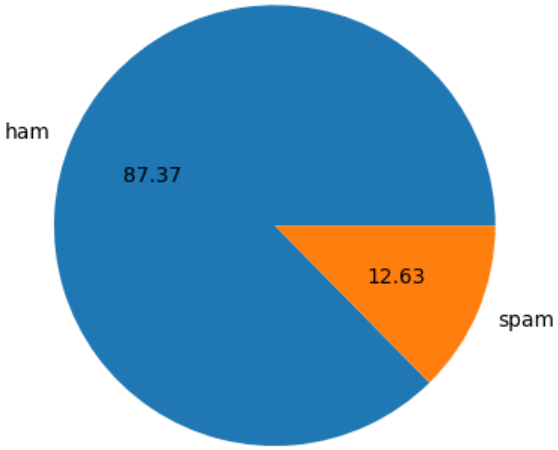
Hình 5. Kết quả xử lý dữ liệu trùng lặp

2.1.2 Phân tích dữ liệu khám phá (Exploratory Data Analysis – EDA) hình(6)
- Hiển thị một số bản ghi đầu tiên: Dùng phương thức head() để hiển thị một số bản ghi đầu tiên của DataFrame để có cái nhìn sơ bộ về dữ liệu hình (7).
- Hiển thị số lượng các nhãn trong cột mục tiêu: Sử dụng phương thức value_counts() để đếm số lượng mỗi nhãn trong cột mục tiêu. Trong trường hợp này, có hai nhãn: "ham" và "spam".
- Vẽ biểu đồ phân phối của biến mục tiêu: Sử dụng biểu đồ tròn (pie chart) để biểu diễn phân phối của các nhãn trong cột mục tiêu. Điều này giúp hiểu rõ tỷ lệ giữa các nhãn trong dữ liệu hình (7).
- Tạo các đặc trưng mới cho EDA: Sử dụng thư viện NLTK để tạo ra các đặc trưng mới như số ký tự, số từ và số câu trong mỗi tin nhắn. Điều này giúp phân tích các đặc trưng của dữ liệu.

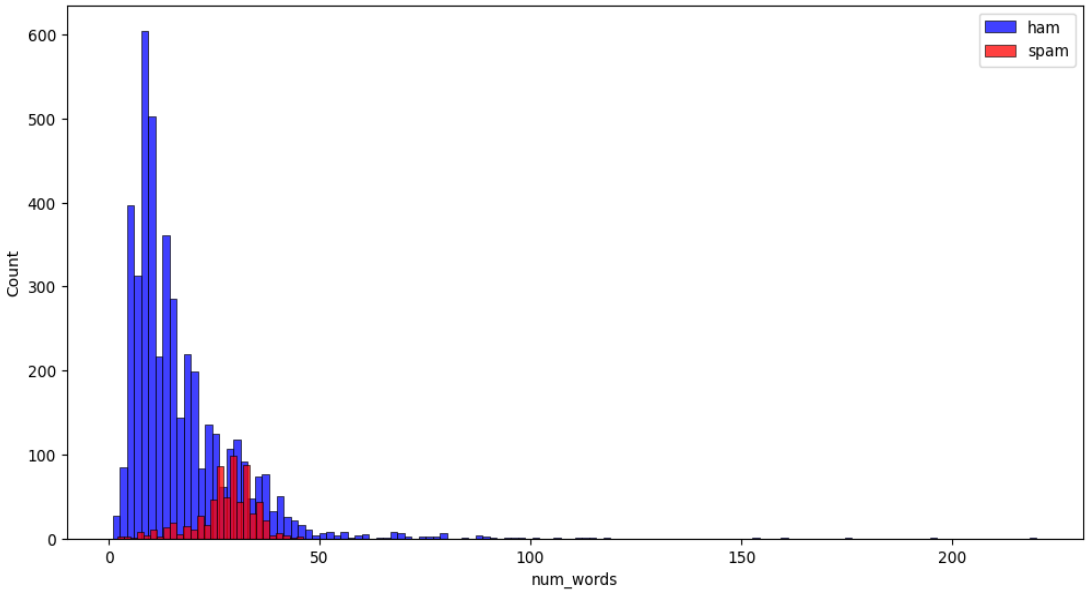
- Hiển thị các thống kê mô tả cho các đặc trưng mới: Sử dụng phương thức describe() để hiển thị các thống kê mô tả như giá trị trung bình, độ lệch chuẩn, và phân phối của các đặc trưng mới.
- Vẽ biểu đồ phân phối của các đặc trưng theo nhãn: Sử dụng biểu đồ histogram để biểu diễn phân phối của các đặc trưng mới được tạo ra, phân tách theo các nhãn trong cột mục tiêu. Điều này giúp phát hiện ra sự khác biệt trong phân phối của các đặc trưng giữa các nhãn hình (8) hình (9)
- Vẽ biểu đồ phân tán (pair plot): Sử dụng pair plot để hiển thị sự phân tán của các đặc trưng mới theo nhãn mục tiêu. Điều này giúp kiểm tra mối quan hệ giữa các đặc trưng và xem xét khả năng phân loại hình(10)
- Hiển thị heatmap tương quan: Sử dụng heatmap để hiển thị ma trận tương quan giữa các đặc trưng mới. Điều này giúp xác định mối quan hệ tương quan giữa các đặc trưng và đánh giá tính tương quan của chúng đối với mục tiêu hình (11)

```
#Exploratory Data Analysis (EDA)
# Display the first few records
df.head()
# Display value counts of the target column
df['target'].value_counts()
# Plot the distribution of the target variable
plt.pie(df['target'].value_counts(), labels=['ham', 'spam'], autopct="%0.2f")
plt.show()
# Download NLTK data
nltk.download('punkt')
# Add feature columns for EDA
df['num_characters'] = df['text'].apply(len)
df['num_words'] = df['text'].apply(lambda x: len(nltk.word_tokenize(x)))
df['num_sentences'] = df['text'].apply(lambda x: len(nltk.sent_tokenize(x)))
# Display the first few records to confirm new columns
df.head()
# Display descriptive statistics for the new columns
df[['num_characters', 'num_words', 'num_sentences']].describe()
# Display descriptive statistics for ham and spam messages separately
df[df['target'] == 0][['num_characters', 'num_words', 'num_sentences']].describe()
df[df['target'] == 1][['num_characters', 'num_words', 'num_sentences']].describe()
# Visualize distributions
plt.figure(figsize=(12, 6))
sns.histplot(df[df['target'] == 0]['num_characters'], kde=False, color='blue', label='ham')
sns.histplot(df[df['target'] == 1]['num_characters'], kde=False, color='red', label='spam')
plt.legend()
plt.show()
plt.figure(figsize=(12, 6))
sns.histplot(df[df['target'] == 0]['num_words'], kde=False, color='blue', label='ham')
sns.histplot(df[df['target'] == 1]['num_words'], kde=False, color='red', label='spam')
plt.legend()
plt.show()
# Pair plot
sns.pairplot(df, hue='target')
plt.show()
# Drop text column for correlation heatmap
df_1 = df.drop(columns=['text'])
# Display the correlation heatmap
plt.figure(figsize=(12, 6))
sns.heatmap(df_1.corr(), annot=True, cmap='coolwarm')
plt.show()
```

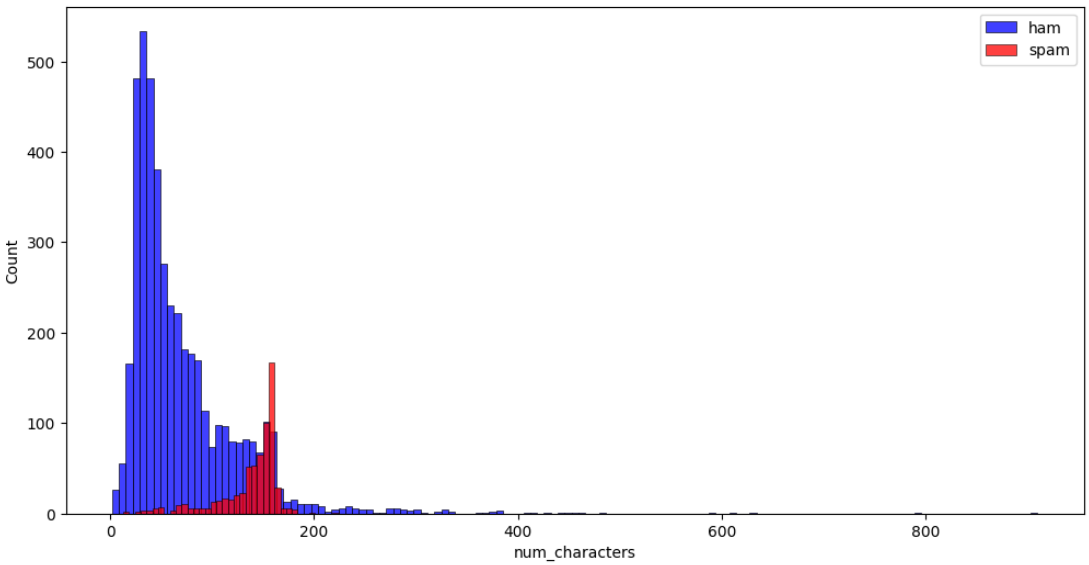
Hình 6 Exploratory Data Analysis – EDA



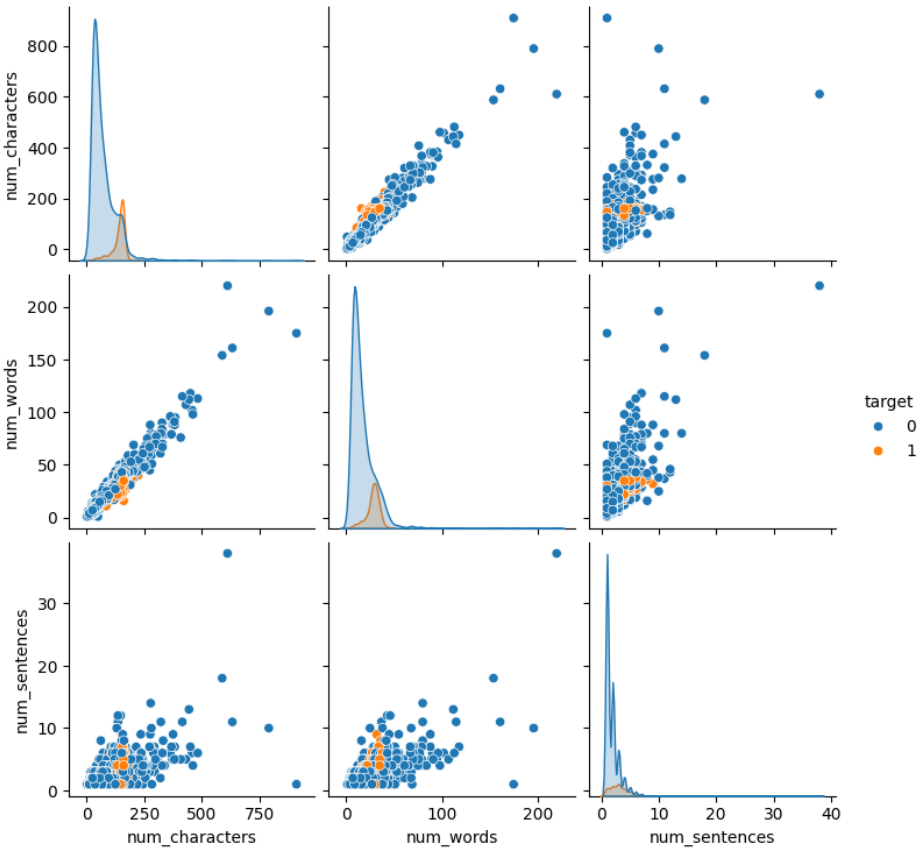
Hình 7: Biểu đồ phân phối của biến mục tiêu



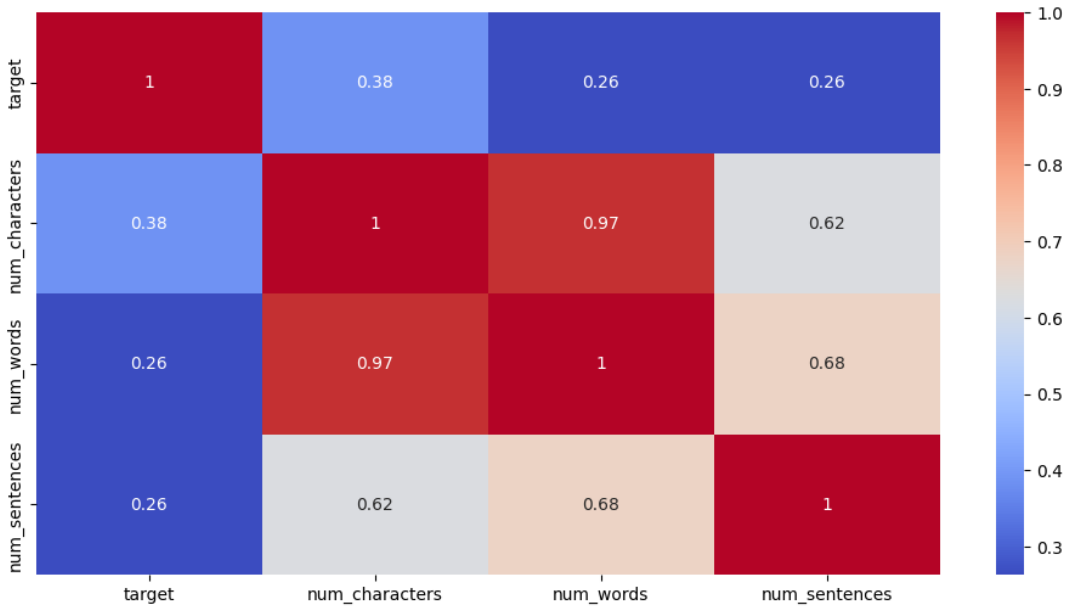
Hình 8: Biểu đồ phân phối của các đặc trưng theo nhãn (số lượng từ)



Hình 9: Biểu đồ phân phối của các đặc trưng theo nhãn(số lượng ký tự)



Hình 10: Biểu đồ phân tán (Pair Plot):



Hình 11: Heatmap tương quan

3. Tiền xử lý dữ liệu và trực quan hóa biểu đồ các dữ liệu đã xử lý

3.1 Tiền xử lý dữ liệu

- Import thư viện và định nghĩa hàm tiền xử lý text: Sử dụng thư viện nltk để tiền xử lý văn bản. Hàm transform_text được định nghĩa để thực hiện các bước tiền xử lý như

chuyển đổi văn bản thành chữ thường, tách từ, loại bỏ từ dừng (stopwords), loại bỏ dấu câu và từng từ thành dạng gốc (stemming).

- Tải danh sách từ dừng: Sử dụng nltk.download('stopwords') để tải danh sách từ dừng từ thư viện nltk, danh sách này sẽ được sử dụng để loại bỏ các từ không mang nhiều ý nghĩa như "is", "the", "and", v.v.
- Khởi tạo PorterStemmer: PorterStemmer là một trong những thuật toán stemming phổ biến được sử dụng để đưa các từ về dạng gốc.
- Áp dụng hàm transform_text vào cột văn bản: Sử dụng phương thức apply() để áp dụng hàm transform_text vào cột văn bản text của DataFrame, kết quả được lưu vào cột mới transformed_text.
- Hiển thị một số bản ghi đầu tiên để xác nhận các thay đổi: Dùng lại phương thức head() để kiểm tra một số bản ghi mẫu sau khi đã tiền xử lý văn bản, đảm bảo rằng các bước tiền xử lý đã diễn ra đúng hình (12).

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

Hình 12 Kết quả tiền xử lý

3.2 Trực quan hóa biểu đồ cho các dữ liệu đã đực xử lý

1. Tạo Word Cloud cho các tin nhắn spam và ham:

-Sử dụng thư viện WordCloud để tạo Word Cloud cho các tin nhắn spam và ham.

-Word Cloud là một biểu đồ hiển thị các từ trong văn bản, trong đó kích thước của từ thể hiện tần suất xuất hiện của từ đó trong văn bản.

Đối với mỗi loại tin nhắn (spam và ham), các từ được kết hợp lại thành một chuỗi (sử dụng str.cat()) và sau đó được truyền vào WordCloud để tạo Word Cloud tương ứng.

2. Tạo phân phối tần suất cho các từ trong tin nhắn spam và ham:

- Sử dụng Counter từ thư viện collections để đếm số lần xuất hiện của mỗi từ trong các tin nhắn spam và ham.

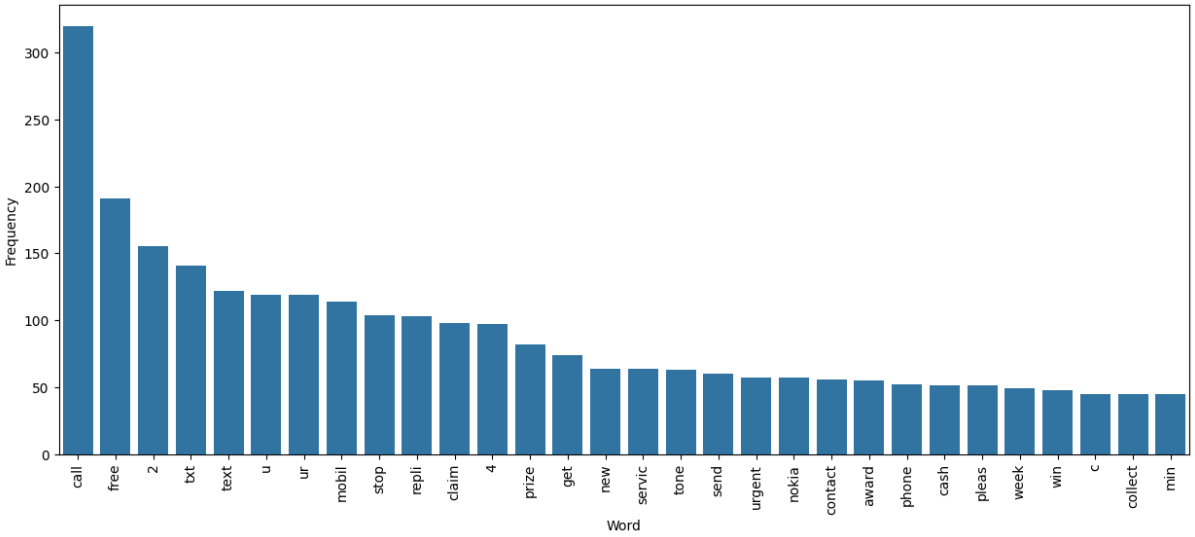
-Từ các từ đã được đếm, tạo DataFrame chứa 30 từ có tần suất xuất hiện cao nhất trong mỗi loại tin nhắn (spam và ham).

- Sử dụng Seaborn để vẽ biểu đồ cột của các từ phổ biến nhất trong mỗi loại tin nhắn.

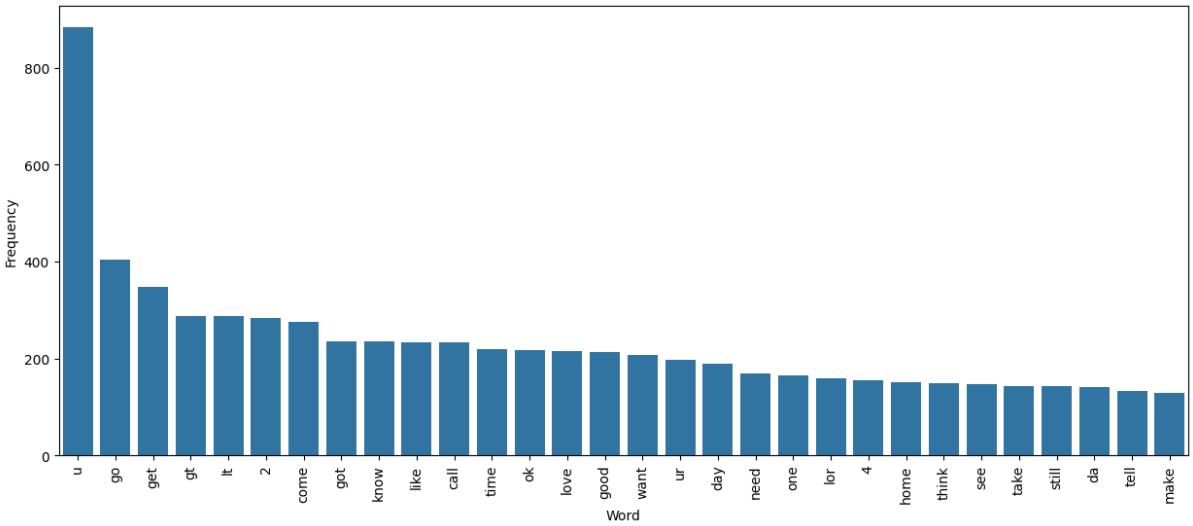
Cả hai phần này giúp hiểu rõ hơn về cấu trúc và nội dung của các tin nhắn spam và ham, cũng như tần suất xuất hiện của các từ quan trọng trong từng loại tin nhắn.

Hình 13: Biểu đồ Đám Mây Từ của Tin Nhắn Spam:

Hình 14: Biểu đồ Đám Mây Từ của Tin Nhắn Ham



Hình 15: Biểu đồ tần suất của 30 từ phổ biến nhất trong tin nhắm Spam



Hình 16: Biểu đồ tần suất của 30 từ phổ biến nhất trong tin nhắm Ham

4. Vector hóa văn bản(TF-IDF) hình 17
    1. Vector hóa dữ liệu văn bản:

        -Sử dụng TfidfVectorizer để chuyển đổi văn bản thành vectơ TF-IDF.
        max_features=3000 chỉ định số lượng tối đa các đặc trưng (từ) sẽ được sử dụng trong vectơ hóa. Chỉ các từ có tần suất xuất hiện cao nhất sẽ được chọn.
        fit_transform() được sử dụng để học từ điển từ dữ liệu huấn luyện và biến đổi dữ liệu văn bản thành vectơ TF-IDF.
    2. Xác định biến mục tiêu:

        -Định nghĩa biến mục tiêu y là cột "target" trong DataFrame, chứa nhãn của các tin nhắn (spam hoặc ham).
    3. Chia dữ liệu thành tập huấn luyện và tập kiểm tra:

        -Sử dụng train_test_split() để chia dữ liệu vectơ hóa và biến mục tiêu thành hai tập dữ liệu riêng biệt: tập huấn luyện và tập kiểm tra.
        test_size=0.2 chỉ định tỷ lệ dữ liệu được chia cho tập kiểm tra là 20%, trong khi 80% còn lại được sử dụng cho tập huấn luyện.
        random_state=2 được sử dụng để đảm bảo sự tái lập khi chạy lại mã, đảm bảo kết quả chia tập dữ liệu không thay đổi.
5. Huấn luyện, đánh giá so sánh hiệu xuất giữa các mô hình.
    1. Định nghĩa các mô hình phân loại:

        -Định nghĩa một loạt các mô hình phân loại bao gồm SVC (Support Vector Classifier), KNeighborsClassifier, MultinomialNB (Multinomial Naive Bayes), DecisionTreeClassifier, LogisticRegression, RandomForestClassifier, AdaBoostClassifier, BaggingClassifier, ExtraTreesClassifier, GradientBoostingClassifier và XGBClassifier hình (18).
        -Mỗi mô hình được khởi tạo với các tham số tương ứng để tối ưu hiệu suất.
    2. Hàm để huấn luyện và đánh giá mô hình:

        - Định nghĩa hàm train_classifier để huấn luyện mô hình trên tập huấn luyện và đánh giá hiệu suất trên tập kiểm tra hình (19).
        - Hiệu suất được đo bằng độ chính xác (accuracy) và độ chính xác dự báo (precision).
    3. Huấn luyện và đánh giá các mô hình:

- Sử dụng một vòng lặp để lặp qua mỗi mô hình trong danh sách các mô hình đã định nghĩa và đánh giá hiệu suất của từng mô hình bằng cách gọi hàm train_classifier.
- Kết quả của việc đánh giá hiệu suất được in ra màn hình cho mỗi mô hình.

4. So sánh hiệu suất của các mô hình:
-Dữ liệu hiệu suất được sắp xếp vào DataFrame performance_df để so sánh hiệu suất của các mô hình.
-Biểu đồ được vẽ để trực quan hóa hiệu suất của các mô hình.

5. Voting Classifier và Stacking Classifier:
-Định nghĩa một Voting Classifier và một Stacking Classifier để kết hợp nhiều mô hình lại với nhau.
-Huấn luyện hai mô hình kết hợp này và đánh giá hiệu suất của chúng trên tập kiểm tra.

Thông qua quá trình này, ta có thể so sánh và lựa chọn mô hình phù hợp nhất cho bài toán phân loại văn bản.

```
#Text Vectorization
# Vectorize text data using TfidfVectorizer
tfidf = TfidfVectorizer(max_features=3000)
X = tfidf.fit_transform(df['transformed_text']).toarray()

# Define target variable
y = df['target'].values

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

Hình 17: TF-IDF

```
# Define classifiers with probability=True for SVC
svc = SVC(kernel='sigmoid', gamma=1.0, probability=True)  # Enable probability estimates
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50, random_state=2)
xgb = XGBClassifier(n_estimators=50, random_state=2)

clfs = {
    'SVC': svc,
    'KN': knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT': gbdt,
    'xgb': xgb
}
```

Hình 18: Định nghĩa các mô hình phân loại

```python
# Function to train and evaluate classifiers
def train_classifier(clf, X_train, y_train, X_test, y_test):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    return accuracy, precision
```

Hình 19: Định nghĩa hàm huấn luyện

```python
# Evaluate all classifiers
accuracy_scores = []
precision_scores = []

for name, clf in clfs.items():
    current_accuracy, current_precision = train_classifier(clf, X_train, y_train, X_test, y_test)
    print(f"For {name}")
    print(f"Accuracy - {current_accuracy}")
    print(f"Precision - {current_precision}")
    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
```

Hình 20: Huấn luyện và đánh giá các mô hình

```python
# Create DataFrame to compare model performance
performance_df = pd.DataFrame({'Algorithm': clfs.keys(), 'Accuracy': accuracy_scores, 'Precision':
                               precision_scores}).sort_values('Precision', ascending=False)
# Visualize model performance
performance_df_melted = pd.melt(performance_df, id_vars='Algorithm')
sns.catplot(x='Algorithm', y='value', hue='variable', data=performance_df_melted, kind='bar', height=5)
plt.ylim(0.5, 1.0)
plt.xticks(rotation='vertical')
plt.show()
```

Hình 21: So sánh hiệu suất của các mô hình

```python
# Voting Classifier with probability=True for SVC
voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc)], voting='soft')
voting.fit(X_train, y_train)
y_pred = voting.predict(X_test)
print("Voting Classifier - Accuracy:", accuracy_score(y_test, y_pred))
print("Voting Classifier - Precision:", precision_score(y_test, y_pred))

# Stacking Classifier
estimators = [('svm', svc), ('nb', mnb), ('et', etc)]
final_estimator = RandomForestClassifier()
stacking = StackingClassifier(estimators=estimators, final_estimator=final_estimator)
stacking.fit(X_train, y_train)
y_pred = stacking.predict(X_test)
print("Stacking Classifier - Accuracy:", accuracy_score(y_test, y_pred))
print("Stacking Classifier - Precision:", precision_score(y_test, y_pred))
```

Hình 22: Voting Classifier và Stacking Classifier

6. Lưu mô hình, kiểm tra xem mô hình đã được huấn luyện hay chưa hình (22)
    1. Lưu TfidfVectorizer với mô hình tốt nhất:
        -Sử dụng pickle.dump() để lưu TfidfVectorizer vào file 'vectorizer.pkl' và mô hình tốt nhất (trong trường hợp này là Multinomial Naive Bayes) vào file 'model.pkl'.
        -Đối tượng pickle được sử dụng để lưu trữ các đối tượng Python vào file.
    2. Kiểm tra xem mô hình đã được huấn luyện chưa:

-Sử dụng hasattr() để kiểm tra xem mô hình có thuộc tính 'classes_' không. Thuộc tính này thường tồn tại sau khi mô hình đã được huấn luyện.
-Nếu thuộc tính này tồn tại, in ra thông báo "Model has been trained." ngược lại in ra "Model has not been trained.".

Việc lưu trữ TfidfVectorizer và mô hình cho phép ta tái sử dụng chúng mà không cần huấn luyện lại từ đầu, tiết kiệm thời gian và công sức khi triển khai mô hình trong các ứng dụng thực tế.

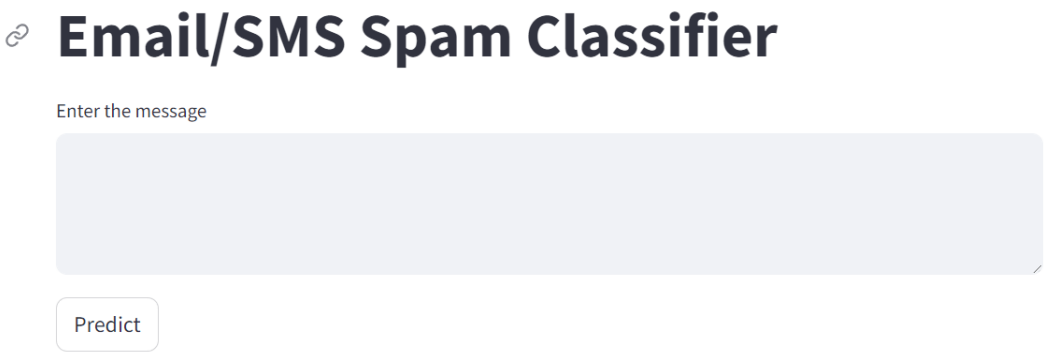
```
# Save the TfidfVectorizer and the best model
pickle.dump(tfidf, open('vectorizer.pkl', 'wb'))
pickle.dump(mnb, open('model.pkl', 'wb'))

# Check if the model has been trained
if hasattr(mnb, 'classes_'):
    print("Model has been trained.")
else:
    print("Model has not been trained.")
```

Hình 22: Lưu mô hình, kiểm tra xem mô hình đã được huấn luyện hay chưa

7. Tạo ra một giao diện người dùng đơn giản.

- Thiết lập: Nhập các thư viện cần thiết, định nghĩa hàm chuyển đổi văn bản, tải các mô hình đã được huấn luyện trước và thiết lập giao diện Streamlit.
- Tiền xử lý văn bản: Sử dụng NLTK để tách từ, loại bỏ stopwords và stemming từ.
- Tải mô hình: Tải vectorizer TF-IDF và mô hình phân loại từ đĩa.
- Giao diện: Tạo giao diện đơn giản với một vùng nhập liệu cho văn bản và một nút để thực hiện dự đoán hình (23).
- Dự đoán: Xử lý văn bản đầu vào, vector hóa nó, thực hiện dự đoán bằng mô hình đã huấn luyện và hiển thị kết quả.



Hình 23: Giao diện người dùng đơn giản

4.4. Kết quả

Kết quả đánh giá hiệu suất của các mô hình phân loại được thực hiện bằng cách tính toán độ chính xác (accuracy) và độ chính xác dự báo (precision) trên tập dữ liệu kiểm tra cho từng mô hình dự đoán và trực quan hóa biểu đồ đánh giá cho từng mô hình (2). Dưới đây là một số đánh giá dựa trên 2 hình:

1. Đánh giá từng mô hình độc lập bảng (1):

-Mô hình Support Vector Classifier (SVC) đạt được độ chính xác và độ chính xác dự báo cao nhất với accuracy khoảng 97.6% và precision gần 97.5%.

-Mô hình Naive Bayes (NB) và Random Forest (RF) cũng có hiệu suất tốt, với precision đạt 100% trên tập dữ liệu kiểm tra.

-Các mô hình khác như KNeighbors (KN), Logistic Regression (LR), và Extra Trees (ETC) cũng đạt được kết quả tốt, với accuracy trên 95% và precision trên 90%.

2. So sánh hiệu suất giữa các mô hình bảng (2):

-Khi so sánh hiệu suất giữa các mô hình, SVC và NB có vẻ là hai lựa chọn hàng đầu với cả độ chính xác và độ chính xác dự báo cáo.

-Mô hình Stacking (kết hợp) và Voting (bỏ phiếu) cả hai đều có hiệu suất cao hơn so với các mô hình đơn lẻ, với Voting Classifier có precision cao nhất là khoảng 99.2% và Stacking Classifier đạt precision khoảng 94.7%.

3. Trực quan hóa hiệu suất mô hình dựa trên biểu đồ:

-Kết quả cho thấy rằng mô hình SVC, NB và RF có hiệu suất tốt trên tập dữ liệu kiểm tra, với khả năng phân loại chính xác cao.

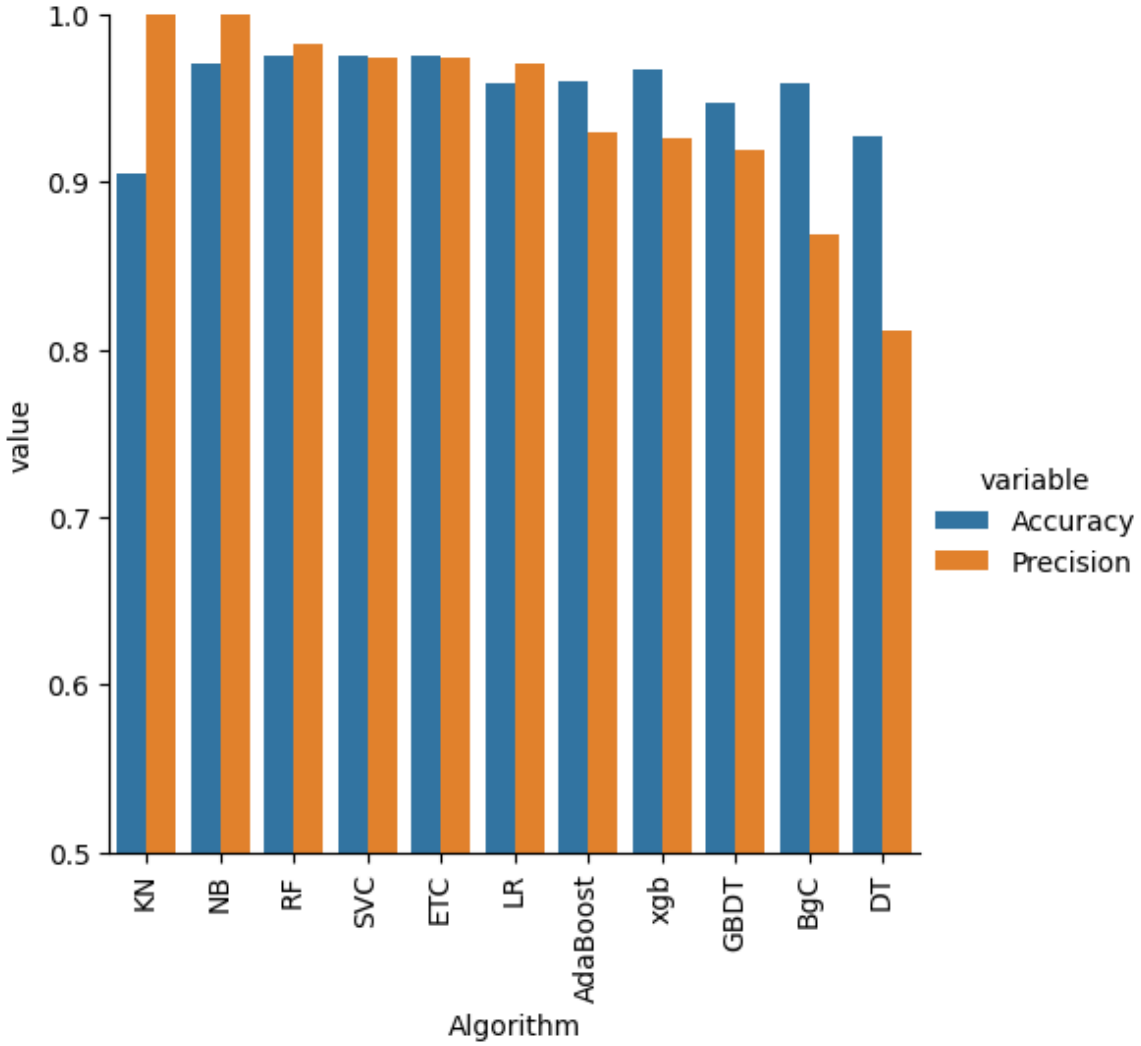
-Sự kết hợp giữa các mô hình thông qua Voting và Stacking cũng mang lại cải thiện đáng kể về độ chính xác dự báo.

Mô hình	Accuracy	Precision
Support Vector Classifier (SVC)	97.58%	97.48%
KNeighbors Classifier (KN)	90.52%	100.0%
Naive Bayes (NB)	97.10%	100.0%
Decision Tree (DT)	92.75%	81.19%
Logistic Regression (LR)	95.84%	97.03%
Random Forest (RF)	97.58%	98.29%
AdaBoost	96.03%	92.92%
Bagging Classifier (BgC)	95.84%	86.82%
Extra Trees Classifier (ETC)	97.49%	97.46%
Gradient Boosting (GBDT)	94.68%	91.92%

Bảng 1: Đánh giá độc lập các hiệu suất của mô hình

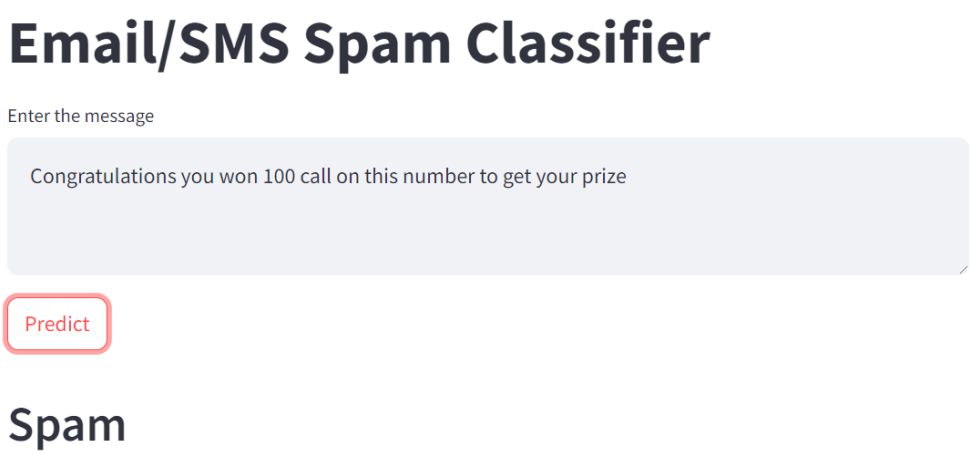
Mô hình	Precision
Support Vector Classifier (SVC)	~97.5%
Naive Bayes (NB)	100.0%
Stacking Classifier	~94.7%
Voting Classifier	~99.2%

Bảng 2: Hiệu suất giữa các mô hình



Hình 23: Hiệu suất từng mô hình trên biểu đồ

- Từ những số liệu trên, ta thấy rằng Naive Bayes (NB) sẽ là mô hình cho ra chỉ số Accuracy và Precision gần như là cao nhất, nên mô hình Naive Bayes (NB) sẽ được sử dụng để xây dựng mô hình Email_based_Spam_Detection.
- Dự đoán: Thực hiện dự đoán trên tập dữ liệu bằng mô hình Naive Bayes (NB) đã được huấn luyện và hiển thị kết quả hình (24), hình (25)



Hình 24: Predict trên model Naive Bayes (NB) với tin nhắn quảng cáo

Hình 25: Predict trên model Naive Bayes (NB) với tin nhắn thoại

4.5. Hạn chế

- Sensitivity đối với dữ liệu không cân bằng: Trong bài toán phân loại tin nhắn spam, dữ liệu thường không cân bằng, với số lượng tin nhắn spam ít hơn nhiều so với số lượng tin nhắn không phải spam. Điều này có thể dẫn đến một số mô hình có thể không nhạy cảm đối với các trường hợp spam.
- Overfitting: Một số mô hình có nguy cơ bị overfitting, đặc biệt là các mô hình có độ phức tạp cao như Random Forest, Gradient Boosting, và XGBoost. Overfitting có thể xảy ra khi mô hình quá tập trung vào các chi tiết nhỏ trong dữ liệu huấn luyện mà không thể tổng quát hóa tốt cho dữ liệu mới.
- Tính tổng quát hóa: Một số mô hình có thể không tổng quát hóa tốt đối với dữ liệu mới ngoài dữ liệu huấn luyện, đặc biệt là các mô hình có độ phức tạp cao như Gradient Boosting và XGBoost.
- Thời gian huấn luyện và dự đoán: Các mô hình có độ phức tạp cao như Random Forest và XGBoost có thể tốn nhiều thời gian để huấn luyện và dự đoán, đặc biệt là khi tập dữ liệu lớn.
- Đối với các mô hình kết hợp (Voting và Stacking): Việc kết hợp nhiều mô hình có thể làm tăng chi phí tính toán và làm cho mô hình trở nên phức tạp hơn. Đôi khi, sự kết hợp có thể không mang lại cải thiện đáng kể so với việc sử dụng một mô hình đơn lẻ.

V. Kết luận

5.1. Kết luận

Email đã trở thành phương tiện giao tiếp quan trọng nhất trong thời đại hiện nay, thông qua kết nối internet, bất kỳ tin nhắn nào cũng có thể được gửi đến khắp nơi trên thế giới. Hàng ngày có hơn 270 tỷ email được trao đổi, khoảng 57% trong số này chỉ là email rác. Email rác, còn được biết đến là email không mong muốn, là những email thương mại không mong muốn hoặc độc hại, ảnh hưởng hoặc hack thông tin cá nhân như thông tin ngân hàng, liên quan đến tiền bạc hoặc bất cứ điều gì gây ra phá hủy cho một cá nhân, một tập đoàn hoặc một nhóm người. Ngoài việc quảng cáo, chúng có thể chứa liên kết đến các trang web lừa đảo hoặc lưu trữ phần mềm độc hại được thiết lập để đánh cắp thông tin nhạy cảm. Email rác là một vấn đề nghiêm trọng không chỉ làm phiền người dùng cuối mà còn gây tổn thất về mặt tài chính và mối nguy hiểm về mặt an ninh. Do đó, hệ thống này được thiết kế để phát hiện và ngăn chặn email không mong muốn và không mong muốn, từ đó giúp giảm thiểu tin nhắn rác, điều này sẽ mang lại lợi ích lớn cho cá nhân cũng như cho công ty. Trong tương lai, hệ thống này có thể được triển khai bằng cách sử dụng các thuật toán khác nhau và cũng có thể thêm các tính năng mới vào hệ thống hiện tại.

5.2. Hướng phát triển của đề tài

1. Phát triển giao diện người dùng:
   - -Tạo giao diện người dùng dễ sử dụng và thân thiện với người dùng.
   - -Tối ưu hóa trải nghiệm người dùng trên các thiết bị di động và máy tính để bàn.
   - -Cải thiện tính linh hoạt và khả năng tương tác của giao diện.
2. Tích hợp chức năng quản lý người dùng:
   - -Phát triển chức năng đăng ký và đăng nhập người dùng.
   - -Xây dựng cơ sở dữ liệu để lưu trữ thông tin người dùng.
   - -Cải thiện tính bảo mật của hệ thống đăng nhập.
3. Phát triển chức năng gửi và nhận email:
   - -Tạo chức năng soạn thư và gửi email.
   - -Xây dựng chức năng lưu trữ và quản lý hộp thư đến (inbox), hộp thư đã gửi (sent), và hộp thư rác (trash).
   - -Cải thiện tính năng đính kèm tập tin và hình ảnh trong email.
4. Tích hợp tính năng phát hiện và lọc spam:
   - -Phát triển hệ thống phân loại email thành spam và không phải spam.
   - -Sử dụng kỹ thuật học máy, như Naïve Bayes Classifier, để phát hiện spam.
   - -Tối ưu hóa hệ thống phát hiện spam bằng cách sử dụng thông tin về IP address của hệ thống gửi spam.
5. Tích hợp các tính năng bổ sung:
   - -Phát triển tính năng ghi âm và gửi tin nhắn bằng giọng nói.
   - -Tạo tính năng thông báo ngoại tuyến qua tin nhắn văn bản.
   - -Cải thiện tính năng xóa email cho cả người gửi và người nhận.
6. Kiểm thử và triển khai:
   - -Tiến hành kiểm thử hệ thống để đảm bảo tính ổn định và bảo mật.
   - -Triển khai hệ thống trên môi trường thực tế và điều chỉnh nếu cần thiết dựa trên phản hồi từ người dùng.
7. Huấn luyện và cập nhật mô hình học máy:

-Huấn luyện mô hình học máy để phát hiện spam và cập nhật định kỳ dựa trên dữ liệu mới.

-Kiểm tra và đánh giá hiệu suất của mô hình để đảm bảo tính đáng tin cậy của hệ thống.

8. Cải thiện và bảo trì:

-Liên tục cải thiện và bảo trì hệ thống dựa trên phản hồi từ người dùng và thay đổi trong nhu cầu.

-Theo dõi và giải quyết các vấn đề bảo mật và hiệu suất.

Tài liệu tham khảo

[1] Shukor Bin Abd Razak, Ahmad Fahrulrazie Bin Mohamad "Identification of Spam Email Based on Information from Email Header" 13th International Conference on Intelligent SystemsDesign and Applications (ISDA), 2013.

[2] Mohammed Reza Parsei, Mohammed Salehi "E-Mail Spam Detection Based on Part of Speech Tagging" 2nd International Conference on Knowledge Based Engineering and Innovation (KBEI), 2015.

[3] Sunil B. Rathod, Tareek M. Pattewar "Content Based Spam Detection in Email using Bayesian Classifier", presented at the IEEE ICCSP 2015 conference.

[4] Aakash Atul Alurkar, Sourabh Bharat Ranade, Shreeya Vijay Joshi, Siddhesh Sanjay Ranade, Piyush A. Sonewa, Parikshit N. Mahalle, Arvind V. Deshpande "A Proposed Data Science Approach for Email Spam Classification using Machine Learning Techniques", 2017.

[5] Kriti Agarwal, Tarun Kumar "Email Spam Detection using integrated approach of Naïve Bayes and Particle Swarm Optimization", Proceedings of the Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2018.