

NGUYỄN VĂN MẠNH

BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC PHENIKAA



ĐỒ ÁN TỐT NGHIỆP

**XÂY DỰNG CHƯƠNG TRÌNH PHÂN LOẠI THƯ RÁC BẰNG
THUẬT TOÁN HỌC MÁY**

Sinh viên: Nguyễn Văn Mạnh

Mã số sinh viên: 21013115

Khóa: 15

Ngành: Công nghệ thông tin

Hệ: Chính quy

Giảng viên hướng dẫn: ThS. Nguyễn Thị Khánh Trâm

CÔNG NGHỆ THÔNG TIN

Hà Nội - Năm 2025

NGUYỄN VĂN MẠNH

BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC PHENIKAA



PHENIKAA
UNIVERSITY

ĐỒ ÁN TỐT NGHIỆP

**XÂY DỰNG CHƯƠNG TRÌNH PHÂN LOẠI THƯ RÁC BẰNG
THUẬT TOÁN HỌC MÁY**

Sinh viên: Nguyễn Văn Tài Anh

Mã số sinh viên: 21013115

Khóa: 15

Ngành: Công nghệ thông tin

Hệ: Chính quy

Giảng viên hướng dẫn: ThS. Nguyễn Thị Khánh Trâm

CÔNG NGHỆ THÔNG TIN

Hà Nội - Năm 2025

Lời cam kết

Họ và tên sinh viên: Nguyễn Văn Mạnh

Điện thoại liên lạc: 0342881863 Email: 21013115@st.phenikaa-uni.edu.vn

Lớp: Công nghệ thông tin 2

Hệ đào tạo: Chính quy

Tôi cam kết đồ án là công trình nghiên cứu của bản thân/nhóm tôi. Các kết quả nêu trong đồ án là trung thực, là thành quả của riêng tôi, không sao chép theo bất kỳ công trình nào khác. Tất cả những tham khảo trong đồ án – bao gồm hình ảnh, bảng biểu, số liệu, và các câu trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục tài liệu tham khảo. Tôi/chúng tôi xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế của nhà trường.

Hà Nội, ngày 5 tháng 12 năm 2025

Tác giả ĐATN

LỜI MỞ ĐẦU

Trong kỷ nguyên số hóa phát triển mạnh mẽ như hiện nay, email đã trở thành một phương tiện trao đổi thông tin không thể thiếu trong hầu hết các lĩnh vực của đời sống, từ giao tiếp cá nhân, học tập cho đến hoạt động kinh doanh và quản lý doanh nghiệp. Tính tiện lợi, nhanh chóng và khả năng truyền tải lượng thông tin lớn chính là những ưu điểm khiến email trở thành một trong những kênh liên lạc quan trọng nhất. Tuy nhiên, song song với sự phổ biến ấy, số lượng thư rác (spam) và email giả mạo (phishing) cũng gia tăng với tốc độ nhanh chóng. Điều này không chỉ gây ảnh hưởng đến hiệu suất làm việc của tôi khi phải xử lý một lượng lớn thư không mong muốn, mà còn tiềm ẩn nhiều nguy cơ nghiêm trọng liên quan đến an toàn thông tin, bảo mật hệ thống và quyền riêng tư của người sử dụng.

Thư rác ngày nay không chỉ dừng lại ở các nội dung quảng cáo phiền nhiễu mà còn có thể chứa mã độc, liên kết dẫn đến website lừa đảo hoặc các đoạn script nguy hiểm. Đặc biệt, email giả mạo đang trở thành một trong những hình thức tấn công mạng phổ biến và nguy hiểm nhất. Thông qua việc ngụy trang dưới danh nghĩa tổ chức tài chính, dịch vụ uy tín hoặc thậm chí là người thân, kẻ tấn công có thể đánh lừa người dùng cung cấp thông tin cá nhân, mật khẩu, mã OTP hoặc truy cập vào các trang độc hại. Chính vì vậy, việc nghiên cứu các giải pháp nhằm tự động phân loại thư rác và phát hiện email giả mạo đóng vai trò then chốt trong việc bảo vệ hệ thống thông tin và người dùng cuối trước những mối đe dọa ngày càng tinh vi.

Xuất phát từ tính cấp thiết đó, tôi quyết định chọn đề tài “Phân loại thư rác và nhận diện email giả mạo bằng kỹ thuật Machine Learning và Natural Language Processing” với mục tiêu xây dựng một mô hình có khả năng phân tích nội dung email và tự động xác định xem email đó có phải là spam, ham hay có dấu hiệu giả mạo. Để đạt được mục tiêu này, tôi ứng dụng các kỹ thuật xử lý ngôn ngữ tự nhiên (NLP) như tách từ, làm sạch dữ liệu, loại bỏ stopwords, lemmatization và chuẩn hóa văn bản nhằm rút trích các đặc trưng quan trọng phục vụ cho quá trình học của mô hình. Bên cạnh đó, các phương pháp biểu diễn văn bản như Bag-of-Words và

TF-IDF cũng được tôi áp dụng để chuyển đổi email thành dạng vector số mà mô hình có thể phân tích.

Để đảm bảo tính toàn diện và khách quan, tôi triển khai và đánh giá nhiều thuật toán học máy khác nhau, bao gồm Naive Bayes, K-Nearest Neighbors, Decision Tree, Support Vector Machine, Random Forest và kết hợp các mô hình thông qua Voting Classifier. Các thuật toán được tôi so sánh dựa trên các chỉ số Accuracy, Precision, Recall và F1-score nhằm lựa chọn ra mô hình có hiệu năng tối ưu nhất. Ngoài ra, tôi cũng mở rộng hướng phân tích email giả mạo bằng cách xem xét các đặc trưng đặc thù như ngôn ngữ khẩn cấp, từ khóa có tính lừa đảo, đường link đáng ngờ và các dấu hiệu bất thường trong tiêu đề hoặc cấu trúc câu — những yếu tố thường xuất hiện trong các cuộc tấn công phishing.

Phạm vi nghiên cứu của tôi tập trung vào việc xử lý dữ liệu văn bản email đã được gán nhãn, thử nghiệm các mô hình học máy trên tập dữ liệu mẫu và đánh giá khả năng nhận diện chính xác giữa thư rác, thư hợp lệ và email có nguy cơ giả mạo. Phương pháp tiếp cận dựa trên dữ liệu (data-driven) kết hợp với các kỹ thuật NLP hiện đại mang lại nền tảng vững chắc cho việc xây dựng hệ thống phân loại email tự động, có thể áp dụng trong thực tế đối với các hệ thống email doanh nghiệp hoặc cá nhân.

Thông qua đề tài này, tôi mong muốn đóng góp một phần nhỏ vào việc nâng cao hiệu quả bảo mật thông tin và hỗ trợ người dùng trong việc sử dụng email an toàn hơn. Đồng thời, quá trình nghiên cứu cũng giúp tôi hiểu sâu hơn về cách các mô hình học máy hoạt động, cách xử lý ngôn ngữ tự nhiên và cách ứng dụng chúng vào các bài toán thực tế trong lĩnh vực an ninh mạng và phân tích dữ liệu.

Báo cáo này sẽ trình bày môi trường nghiên cứu, phương pháp được sử dụng, quá trình triển khai mô hình, xây dựng ứng dụng, và kết quả đạt được. Nội dung báo cáo sẽ được trình bày trong bốn chương:

Chương 1: Tổng quan về nhận diện email- Trình bày tổng quan về đề tài và các lĩnh vực liên quan.

Chương 2: Phân tích xây dựng mô hình - Trình bày về quá trình xây dựng bộ dữ liệu, lựa chọn, huấn luyện và đánh giá kết quả mô hình.

Chương 3: Triển khai ứng dụng - Dựa vào các kết quả của mô hình và triển khai hệ thống.

Chương 4: Kết luận và hướng phát triển - Đưa ra kết luận về đề tài và định hướng phát triển hệ thống.

MỤC LỤC

LỜI CAM KẾT.....	ii
LỜI MỞ ĐẦU	iii
MỤC LỤC DANH MỤC HÌNH ẢNH.....	vi
DANH MỤC HÌNH ẢNH	ix
DANH MỤC BẢNG BIỂU	xii
DANH MỤC TỪ VIẾT TẮT.....	xii
CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI.....	1
1.1. Giới thiệu chung về đề tài.....	1
1.1.1. Giới thiệu đề tài.....	1
1.1.2. Tính cấp thiết của đề tài.....	1
1.1.3. Mục tiêu đề tài.....	2
1.1.4. Tính ứng dụng.....	3
1.2. Các vấn đề liên quan đến Spam Email.....	6
1.2.1. <i>Gây quá tải, mất thời gian và giảm hiệu suất làm việc</i>	6
1.2.2. <i>Spam là công cụ để tấn công mạng thông qua liên kết độc hại và tệp đính kèm</i>	6
1.2.3. <i>Các chiến dịch lừa đảo (Phishing) đánh cắp thông tin cá nhân</i>	7
1.2.4. <i>Email giả mạo (Email Spoofing) – kỹ thuật nguy hiểm ngày càng tinh vi</i>	7
1.2.5. <i>Sự hạn chế của các bộ lọc spam truyền thống</i>	7
1.2.6. <i>Tiêu tốn tài nguyên hệ thống và chi phí vận hành</i>	8
1.3. Giới thiệu về Google Colab.....	9
1.3.1. <i>Ưu điểm của Google Colab</i>	9
1.3.2. <i>Nhược điểm của Google Colab</i>	10
1.4. Giới thiệu về Visual Studio Code	11
1.5. Giới thiệu về Machine Learning	12

1.6. Giới thiệu về Natural Language Processing (NLP).....	14
1.7. Các thuật toán học máy	16
1.7.1. Giới thiệu về Naive Bayes	18
1.7.2. Giới thiệu về K-Nearest Neighbors.....	19
1.7.3. Giới thiệu về Decision Tree.....	20
1.7.4. Giới thiệu về Support Vector Machine.....	21
1.7.5. Giới thiệu về Random Forest.....	23
1.7.6. Giới thiệu về Voting Classifier.....	24
CHƯƠNG 2: PHÂN TÍCH XÂY DỰNG MÔ HÌNH	26
2.1. Tổng quan mô hình	26
2.2. Xây dựng tập dữ liệu.....	26
2.2.1. Thu thập dữ liệu	26
2.2.2. Tiên xử lý dữ liệu.....	27
2.2.3. Xác định đặc trưng và lựa chọn đặc trưng	28
2.2.4. Trích xuất đặc trưng	29
2.3. Tiêu chí đánh giá	32
2.4. Đào tạo và đánh giá kết quả	36
2.4.1. Thư viện	36
2.4.2. Mô hình triển khai	36
2.4.3. Đào tạo mô hình	36
2.4.4. Đánh giá và so sánh kết quả	36
CHƯƠNG 3: TRIỂN KHAI ỨNG DỤNG	46
3.1. Môi trường và công cụ triển khai.	46
3.1.1. Ngôn ngữ lập trình và Framework.....	46
3.1.1.1. Ngôn ngữ lập trình Python 3.8+.....	46
3.1.1.2. Tổng quan về Framework giao diện web Flask.....	47
3.1.1.3. Thư viện Machine Learning.....	50

3.1.1.3. Các Thư Viện Kỹ Thuật Cốt Lõi (Core Technical Libraries)	51
3.1.2. Yêu Cầu Hệ Thống Ứng Dụng	53
3.2. Cấu trúc tổng quan ứng dụng	53
3.2.1. Sơ đồ kiến trúc hệ thống	53
3.2.2. Cấu trúc thư mục dự án	55
3.2.3. Triển khai chi tiết ứng dụng và giao diện	60
3.2.2.1. Giao diện ứng dụng và chức năng đăng nhập.	63
3.2.2.2. Giao diện chức năng Dashboard	68
3.2.2.3. Giao diện Auto Checker	74
CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	76
4.1. Kết luận về đề tài	76
4.2. Hướng phát triển	76
TÀI LIỆU THAM KHẢO	78

DANH MỤC HÌNH ẢNH

Hình 1.1. Khung dự kiến của mô hình phân loại thư rác được đề xuất	17
Hình 1.2. Khung dự kiến của mô hình phân loại thư rác được đề xuất	18
Hình 2.1. Dataset.....	26
Hình 2.2. Tiền xử lý dữ liệu	28
Hình 2.3.Xác định và trích xuất đặc trưng	32
Hình 2.4. Kết quả của Naive Bayes	37
Hình 2.5. Kết quả của KNN	37
Hình 2.6. Kết quả của Decision Tree	37
Hình 2.7. Kết quả của SVM	37
Hình 2.8. Kết quả của Random Forest.....	38
Hình 2.9. Kết quả so sánh kết quả của 5 thuật toán	38
Hình 2.10. Kết quả so sánh confusion matrix của 5 thuật toán	38
Hình 2.11. Tối ưu hóa mô hình Naive Bayes.....	41
Hình 2.12. Tối ưu hóa mô hình Decision Tree	42
Hình 2.13. Tối ưu hóa mô hình Random Forest	42
Hình 2.14. Tối ưu hóa mô hình SVM	43
Hình 2.15. Kết quả so sánh của các thuật toán sau khi được tối ưu	45
Hình 3.1. Streamlit	48
Hình 3.2. Cấu trúc thư mục dự án.....	55
Hình 3.3. Popup ứng dụng được khởi động	63
Hình 3.4. Menu chức năng ứng dụng	63
Hình 3.5. Giao diện đăng nhập Tray app	64
Hình 3.6. File Config.json.....	65
Hình 3.7. Giao diện web checker	68
Hình 3.8. Giao diện Email Monitor – kiểm tra bán thủ công	70
Hình 3.9. Giao diện sau khi email monitor kết nối thành công với email ...	72

Hình 3.10. Giao diện auto checker được khởi động 74

DANH MỤC BẢNG BIỂU

Bảng 2.1 Công thức tính tiêu chí đánh giá 35

Bảng 2.2. Bảng kết quả của các thuật toán sau khi được tối ưu hóa 44

DANH MỤC TỪ VIẾT TẮT

Chữ viết tắt	Ý nghĩa
Accuracy	Độ chính xác (Tỷ lệ dự đoán đúng)
AdaBoost	Adaptive Boosting
AUC	Area Under the Curve (Diện tích dưới đường cong)
Bag of Words (BoW)	Mô hình đại diện văn bản bằng tập hợp các từ xuất hiện trong văn bản
Batch Size	Kích thước batch (Số lượng mẫu được sử dụng trong một lần cập nhật tham số)
Confusion Matrix	Ma trận nhầm lẫn (Bảng đo lường hiệu quả mô hình phân loại)
Count Vectorizer	Biến đổi văn bản thành ma trận tần suất từ khóa
Cross-validation	Kiểm tra chéo (Phương pháp kiểm tra mô hình bằng cách phân chia dữ liệu thành nhiều phần)
Dropout	Kỹ thuật ngừng kết nối một số nơ-ron trong quá trình huấn luyện để giảm overfitting
Epoch	Một vòng lặp qua toàn bộ dữ liệu trong huấn luyện
F1-Score	Đo lường sự cân bằng giữa độ chính xác và độ phủ (precision và recall)
FP	False Positive (Dự đoán sai là spam)
Grid Search	Tìm kiếm qua lưới tham số (Kỹ thuật tối ưu hóa siêu tham số)
Hyperparameter	Tham số siêu tham số, các tham số không học trong mô hình
K-NN	K-Nearest Neighbors

K-fold Cross-validation	Phương pháp kiểm tra chéo với K phần chia
Lemmatization	Chuyển đổi từ về dạng chuẩn (giảm từ về dạng từ điển)
Learning Rate	Tốc độ học (Chỉ số quyết định mức độ điều chỉnh trong quá trình huấn luyện)
Logistic Regression	Hồi quy Logistic
ML	Machine Learning (Học máy)
NB	Naive Bayes
NLP	Natural Language Processing (Xử lý ngôn ngữ tự nhiên)
Overfitting	Hiện tượng học quá chi tiết trên dữ liệu huấn luyện, dẫn đến giảm khả năng tổng quát
PCA	Principal Component Analysis (Phân tích thành phần chính)
Precision	Độ chính xác (Tỷ lệ dự đoán đúng là spam trong tổng số thư dự đoán là spam)
Recall	Độ phủ (Tỷ lệ dự đoán đúng là spam trong tổng số thư thực tế là spam)
RF	Random Forest
ROC	Receiver Operating Characteristic (Đặc tính nhận diện tín hiệu)
SVM	Support Vector Machine
SPAM	Thư rác, thư không hợp lệ
Stop Words	Các từ dừng (Từ không mang nhiều thông tin, thường bị loại bỏ trong xử lý văn bản)
TF	Term Frequency (Tần suất từ trong văn bản)

True Negative (TN)	Dự đoán đúng là ham
True Positive (TP)	Dự đoán đúng là spam
Underfitting	Hiện tượng mô hình học không đủ chi tiết, dẫn đến khả năng dự đoán kém
Word2Vec	Mô hình học từ ngữ dạng vecto
XGBoost	Extreme Gradient Boosting (Thuật toán tăng cường cực đại)

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1. Giới thiệu chung về đề tài

1.1.1. Giới thiệu đề tài

Email, hay thư điện tử, đã trở thành một công cụ giao tiếp không thể thiếu trong xã hội hiện đại, đặc biệt trong môi trường làm việc và các hoạt động kinh doanh. Với sự phát triển mạnh mẽ của công nghệ thông tin và sự phổ biến rộng rãi của Internet, email đã trở thành phương tiện truyền thông chính thức, cho phép người dùng dễ dàng gửi nhận thông tin và kết nối với nhau trong thời gian ngắn. Tuy nhiên, bên cạnh những tiện ích mà email mang lại, một vấn đề nghiêm trọng cũng đang ngày càng gia tăng – đó chính là thư rác (spam).

Thư rác là các email không mong muốn, không được yêu cầu, và thường mang tính chất quảng cáo, lừa đảo, hoặc chứa các phần mềm độc hại (malware). Chúng có thể làm gián đoạn công việc của người dùng, gây lãng phí tài nguyên và thời gian xử lý email. Hơn nữa, thư rác còn tiềm ẩn nguy cơ lớn về bảo mật thông tin cá nhân, vì nhiều email giả mạo có thể chứa liên kết dẫn đến các trang web lừa đảo hoặc thậm chí cài đặt phần mềm gián điệp, đánh cắp thông tin quan trọng.

Do đó, việc phát triển các hệ thống phát hiện thư rác trở thành một nhiệm vụ quan trọng, giúp giảm thiểu các nguy cơ này và nâng cao hiệu quả sử dụng email. Sự phát triển của các thuật toán học máy đã mở ra một hướng đi mới trong việc phát hiện và xử lý thư rác. Thay vì chỉ dựa vào các quy tắc đơn giản, các hệ thống này có thể tự động học hỏi và thích nghi với các hình thức thư rác mới, nâng cao khả năng bảo vệ người dùng và cải thiện chất lượng của hộp thư điện tử.

1.1.2. Tính cấp thiết của đề tài

Thư rác không chỉ làm mất thời gian và tài nguyên của người dùng mà còn tạo ra các mối đe dọa về bảo mật nghiêm trọng. Theo các nghiên cứu gần đây, tỷ lệ thư rác trong tổng số email gửi đi đã vượt quá 50%, và con số này vẫn đang có xu hướng tăng lên. Các cuộc tấn công lừa đảo (phishing), phát tán phần mềm độc hại (malware), và tấn công từ chối dịch vụ (DDoS) là những ví dụ điển hình của các nguy cơ mà thư rác gây ra.

Với sự gia tăng về quy mô và độ tinh vi của các cuộc tấn công qua email, các hệ thống lọc thư rác truyền thống dựa trên quy tắc đã trở nên kém hiệu quả. Các phương pháp này chỉ có thể phát hiện được thư rác khi chúng tuân theo một mẫu cố định, trong khi thư rác mới liên tục thay đổi chiến thuật để tránh bị phát hiện. Do đó, việc ứng dụng các thuật toán học máy (machine learning) vào bài toán phát hiện thư rác là một giải pháp tiên tiến, giúp hệ thống tự động phát hiện các mẫu thư rác mới và thích nghi với các thay đổi của chúng theo thời gian.

Máy học không chỉ giúp mô hình phát hiện thư rác trở nên linh hoạt và chính xác hơn, mà còn mang lại khả năng học từ dữ liệu và cải tiến hiệu suất qua mỗi lần huấn luyện. Điều này sẽ giúp người dùng tiết kiệm thời gian, bảo vệ an ninh thông tin và giảm thiểu thiệt hại do thư rác gây ra.

1.1.3. Mục tiêu đề tài

Bài toán Email-Spam_Detection nhằm mục tiêu phát triển một hệ thống thông minh có khả năng phân loại email thành hai loại chính: “spam” và “ham” (thư hợp lệ). Để đạt được mục tiêu này, các bước nghiên cứu và triển khai sẽ bao gồm:

Xây dựng mô hình phân loại thư rác: Sử dụng các thuật toán học máy như Naive Bayes, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree, và Random Forest để huấn luyện mô hình. Mỗi thuật toán sẽ được thử nghiệm và đánh giá để xác định thuật toán nào mang lại hiệu quả tốt nhất cho bài toán phân loại email.

Để kết hợp các điểm mạnh của các thuật toán riêng lẻ này, chúng tôi triển khai Voting Classifier, một kỹ thuật học tập tổng hợp các dự đoán từ nhiều mô hình. Voting Classifier kết hợp các đầu ra của năm mô hình (Naive Bayes, SVM, KNN, Random Forest và Decision Tree) để đưa ra dự đoán cuối cùng. Phương pháp này đặc biệt có lợi trong các tình huống như phát hiện thư rác, trong đó các thuật toán khác nhau có thể vượt trội trong việc nắm bắt các mẫu khác nhau trong dữ liệu.

Đánh giá hiệu suất mô hình: Các chỉ số quan trọng như độ chính xác (accuracy), độ nhạy (recall), độ đặc hiệu (specificity), và F1-Score sẽ được sử dụng để đánh giá hiệu suất của mô hình. Việc so sánh các thuật toán khác nhau sẽ giúp lựa chọn phương pháp tối ưu cho bài toán.

Ứng dụng thực tế: Mô hình phát hiện thư rác sau khi hoàn thiện sẽ được triển khai vào một hệ thống thực tế (ví dụ như ứng dụng web, ứng dụng di động hoặc dịch vụ

email). Mô hình này sẽ giúp người dùng nhận diện và loại bỏ thư rác tự động, từ đó giảm thiểu nguy cơ bị tấn công qua email.

Mục tiêu cuối cùng là tạo ra một hệ thống học máy có khả năng phát hiện thư rác nhanh chóng và chính xác, giúp người dùng tránh khỏi các phiền toái và nguy cơ bảo mật do thư rác gây ra, đồng thời tối ưu hóa hiệu quả trong việc quản lý thông tin qua email.

1.1.4. Tính ứng dụng

Bài toán phát hiện thư rác không chỉ có ứng dụng trong việc lọc thư rác mà còn có nhiều ứng dụng quan trọng trong các lĩnh vực khác nhau. Các hệ thống phát hiện thư rác sử dụng thuật toán học máy có thể giúp nâng cao hiệu quả làm việc và bảo vệ an ninh mạng trong nhiều môi trường. Dưới đây là các phạm vi ứng dụng cụ thể:

Lọc thư rác (Spam Filtering)

Mô tả: Các hệ thống phát hiện thư rác tự động phân loại email vào hộp thư rác hoặc hộp thư chính, giúp người dùng tránh được sự phiền toái của các email không mong muốn.

Ứng dụng:

Dịch vụ email: Các nền tảng dịch vụ email như Gmail, Yahoo Mail, Outlook, và nhiều dịch vụ khác sử dụng hệ thống phát hiện spam để giảm thiểu thư rác, bảo vệ người dùng khỏi các email không mong muốn.

Doanh nghiệp: Các doanh nghiệp triển khai hệ thống lọc email spam trong hệ thống nội bộ để tăng hiệu quả làm việc, giảm thiểu thời gian xử lý các email không quan trọng và đảm bảo rằng nhân viên chỉ nhận được thông tin cần thiết.

Tăng hiệu quả công việc

Mô tả: Hệ thống phát hiện thư rác giúp người dùng tập trung vào các email quan trọng, không bị phân tâm bởi thư rác.

Ứng dụng:

Môi trường doanh nghiệp: Hệ thống này giúp giảm thiểu các email không liên quan trong hộp thư đến, hỗ trợ quản lý email hiệu quả hơn, tiết kiệm thời gian xử lý thông tin không cần thiết và nâng cao năng suất làm việc.

Phòng chống lừa đảo (Phishing Detection)

Mô tả: Phát hiện các email lừa đảo (phishing emails) được thiết kế để đánh cắp thông tin cá nhân, tài khoản ngân hàng hoặc các dữ liệu quan trọng.

Ứng dụng:

Ngân hàng và tài chính: Hệ thống phát hiện phishing giúp ngăn chặn các cuộc tấn công lừa đảo qua email trong các ngành ngân hàng, tài chính, bảo hiểm và các tổ chức chính phủ.

An ninh cá nhân: Hệ thống này bảo vệ người dùng cá nhân khỏi các mối đe dọa an ninh mạng, giảm thiểu nguy cơ mất mát thông tin quan trọng.

Bảo vệ thông tin nhạy cảm

Mô tả: Phát hiện các email độc hại chứa phần mềm gián điệp (spyware) hoặc virus.

Ứng dụng:

Hệ thống công nghệ thông tin (IT): Các doanh nghiệp có thể bảo vệ hệ thống IT của mình khỏi các cuộc tấn công bằng phần mềm độc hại, bảo vệ các dữ liệu nhạy cảm và đảm bảo hoạt động liên tục của hệ thống.

An toàn dữ liệu cá nhân: Người dùng cá nhân cũng có thể bảo vệ thông tin cá nhân khỏi các email chứa virus hoặc phần mềm độc hại.

Hỗ trợ marketing và bán hàng

Mô tả: Phân biệt giữa email quảng cáo hợp pháp và spam, giúp các nhà tiếp thị đạt hiệu quả trong chiến dịch email marketing.

Ứng dụng:

Công ty marketing: Các công ty sử dụng hệ thống phát hiện thư rác để tối ưu hóa chiến lược gửi email, tránh việc bị đánh dấu là spam, từ đó cải thiện tỷ lệ mở email (open rate) và tỷ lệ nhấp chuột (click-through rate) trong chiến dịch quảng cáo.

Tăng hiệu quả marketing: Phân loại đúng các email quảng cáo giúp doanh nghiệp tiếp cận khách hàng mục tiêu hiệu quả hơn, tránh gây phiền toái cho người dùng.

Cải thiện trải nghiệm người dùng

Mô tả: Giúp hộp thư của người dùng gọn gàng, chỉ hiển thị những email có giá trị.

Ứng dụng:

Trải nghiệm email cá nhân hóa: Người dùng sẽ không còn bị phân tâm bởi các email không mong muốn, đồng thời có thể trải nghiệm một hộp thư gọn gàng và hiệu quả hơn, từ đó nâng cao sự hài lòng với các dịch vụ email.

Tăng tính hiệu quả: Việc chỉ hiển thị các email quan trọng giúp người dùng tiết kiệm thời gian và cải thiện năng suất làm việc.

Phân tích dữ liệu email

Mô tả: Thu thập và phân tích dữ liệu từ các email spam để hiểu rõ hơn về các xu hướng, chiến lược của kẻ tấn công.

Ứng dụng:

An ninh mạng: Các tổ chức an ninh mạng sử dụng dữ liệu thu thập được từ các email spam để nghiên cứu các loại thư rác và phát triển các biện pháp phòng ngừa phù hợp.

Cải tiến hệ thống học máy: Dữ liệu thu thập từ các email spam có thể được sử dụng để cải thiện và tối ưu hóa các mô hình học máy, giúp các hệ thống phát hiện thư rác ngày càng chính xác hơn.

Tối ưu hóa các mô hình nlp

Mô tả: Phát hiện spam giúp cải thiện các mô hình xử lý ngôn ngữ tự nhiên thông qua việc phân loại văn bản.

Ứng dụng:

Đào tạo mô hình học máy: Việc phát hiện thư rác đóng vai trò quan trọng trong việc đào tạo và tối ưu hóa các mô hình học máy cho các ứng dụng NLP khác như phân tích cảm xúc, nhận dạng thực thể và phân loại văn bản.

Kết luận

Ứng dụng của hệ thống phát hiện thư rác không chỉ giúp lọc thư rác mà còn mang lại nhiều lợi ích quan trọng trong các lĩnh vực khác nhau, từ bảo mật thông tin đến tối ưu hóa hiệu quả công việc và nâng cao trải nghiệm người dùng. Việc ứng dụng máy học

trong việc phát hiện thư rác và các mối đe dọa qua email ngày càng trở nên cần thiết và hữu ích trong mọi lĩnh vực từ công nghệ, tài chính, marketing đến an ninh mạng.

1.2. Các vấn đề liên quan đến Spam Email

Spam email từ lâu đã trở thành một trong những thách thức lớn nhất trong an toàn thông tin. Khi khối lượng trao đổi dữ liệu qua email ngày càng tăng, thư rác không chỉ gây ảnh hưởng đến trải nghiệm người dùng mà còn trở thành môi trường thuận lợi cho các hoạt động tấn công mạng. Dưới đây là phân tích chi tiết về các vấn đề cụ thể liên quan đến spam email.

1.2.1. Gây quá tải, mất thời gian và giảm hiệu suất làm việc

Spam email là nguyên nhân trực tiếp làm hộp thư đến bị quá tải, đặc biệt trong bối cảnh khối lượng trao đổi dữ liệu qua email ngày càng lớn. Việc liên tục phải rà soát, sàng lọc và xóa bỏ các email không mong muốn khiến tốn rất nhiều thời gian, đồng thời làm giảm mức độ tập trung trong công việc. Khi hộp thư chứa quá nhiều thư rác, nguy cơ bỏ lỡ các email quan trọng cũng tăng lên vì các email có giá trị dễ bị chìm lẫn giữa vô số thư quảng cáo hoặc nội dung không liên quan. Đối với doanh nghiệp, vấn đề này còn nghiêm trọng hơn khi nó ảnh hưởng đến năng suất làm việc của nhân viên và gây gián đoạn quy trình giao tiếp với khách hàng và đối tác. Như vậy, dù nhìn bề ngoài tưởng chừng chỉ gây phiền toái, spam email thực chất làm giảm hiệu quả hoạt động của cả cá nhân lẫn tổ chức.

1.2.2. Spam là công cụ để tấn công mạng thông qua liên kết độc hại và tệp đính kèm

Ngày nay, spam email không chỉ dừng lại ở việc quấy rối mà đã trở thành phương tiện được tin tặc tận dụng để triển khai các cuộc tấn công mạng có mức độ tinh vi cao. Nhiều email chứa các liên kết độc hại dẫn tới trang web cài mã độc tự động, thu thập thông tin hoặc khai thác lỗ hổng trình duyệt. Bên cạnh đó, không ít email đính kèm các tệp chứa virus, mã độc, ransomware hoặc các macro nguy hiểm được ngụy trang như tài liệu văn phòng thông thường. Chỉ cần người dùng bát cẩn mở tệp hoặc nhấp vào liên kết, thiết bị có thể bị nhiễm mã độc, từ đó gây ra mất dữ liệu, bị chiếm quyền truy cập hoặc ảnh hưởng đến toàn bộ hệ thống mạng nội bộ. Tính chất phức tạp và khó lường của các tấn công này cho thấy spam email đã trở thành một trong những kênh phát tán mã độc hiệu quả nhất của tin tặc.

1.2.3. Các chiến dịch lừa đảo (Phishing) đánh cắp thông tin cá nhân

Phishing là một dạng tấn công đặc biệt nguy hiểm được triển khai thông qua spam email. Tin tặc thường thiết kế nội dung email tinh vi đến mức người dùng khó phân biệt thật – giả, với giao diện, tên miền và thông điệp mô phỏng y hệt email của ngân hàng, tổ chức tài chính, nền tảng thương mại điện tử hoặc cơ quan nhà nước. Nội dung thường mang tính chất khẩn cấp nhằm kích hoạt tâm lý lo lắng của người dùng, chẳng hạn như cảnh báo khóa tài khoản hoặc yêu cầu cập nhật thông tin bảo mật. Khi người nhận làm theo hướng dẫn và cung cấp thông tin như mật khẩu, mã OTP hay dữ liệu cá nhân, tin tặc có thể nhanh chóng chiếm đoạt tài khoản, tiền bạc hoặc truy cập vào các hệ thống quan trọng. Một cuộc tấn công phishing thành công có thể dẫn tới những hậu quả nghiêm trọng, vượt xa thiệt hại về tài chính, bao gồm rò rỉ dữ liệu, mất quyền kiểm soát tài khoản và ảnh hưởng đến an toàn thông tin của toàn tổ chức.

1.2.4. Email giả mạo (Email Spoofing) – kỹ thuật nguy hiểm ngày càng tinh vi

Một trong những phương thức nguy hiểm gắn liền với spam email là kỹ thuật giả mạo email (spoofing), cho phép kẻ tấn công tạo ra các email có vẻ như được gửi từ những tổ chức hoặc cá nhân uy tín. Bằng cách thao túng thông tin trong tiêu đề email hoặc giả mạo tên miền, tin tặc có thể khiến người dùng tin rằng email đến từ nguồn đáng tin cậy như ngân hàng hoặc các dịch vụ trực tuyến quen thuộc. Sự tương đồng gần như tuyệt đối về hình thức, nội dung và cách trình bày khiến việc phát hiện email giả mạo bằng mắt thường trở nên vô cùng khó khăn. Khi người dùng bị đánh lừa và thực hiện các hành động như nhập mật khẩu, mở liên kết hoặc tải tệp đính kèm, tin tặc có thể dễ dàng thu thập được các dữ liệu nhạy cảm. Kỹ thuật giả mạo ngày càng tinh vi khiến email spoofing trở thành một trong những mối đe dọa nguy hiểm nhất đối với người dùng cá nhân và doanh nghiệp.

1.2.5. Sự hạn chế của các bộ lọc spam truyền thống

Các phương pháp lọc Spam truyền thống chủ yếu dự trên bộ quy tắc cố định, từ khóa hoặc danh sách đen địa chỉ gửi. Tuy nhiên, các kỹ thuật này ngày càng trở nên kém hiệu quả do tin tặc liên tục thay đổi các phương pháp khác nhau để có thể né tránh. Việc chèn kí tự đặc biệt vào từ khóa, sử dụng hình ảnh thay vì văn bản, thay đổi cấu trúc câu hoặc tạo ra hàng loạt địa chỉ email mới là những phương pháp phổ biến giúp email rác

vượt qua bộ lọc. Sự linh hoạt và liên tục thích nghi của các tội phạm mạng khiến các hệ thống rule-based khong thể nhận diện được các mẫu spam mới.

Điều này dẫn đến tình trạng email rác vẫn tràn vào hộp thư, trong khi một số email hợp lệ lại bị đánh dấu nhầm làm giảm hiệu quả tổng thể của việc lọc spam.

1.2.6. Tiêu tốn tài nguyên hệ thống và chi phí vận hành

Lượng lớn spam email không chỉ ảnh hưởng đến người dùng mà còn gây áp lực lên toàn bộ hạ tầng email của tổ chức. Các hệ thống email phải xử lý, lưu trữ và phân loại khói lượng thư rác không lồ, dẫn đến tiêu tốn băng thông mạng, chiếm dụng dung lượng lưu trữ và tăng tải cho máy chủ. Đối với các doanh nghiệp có quy mô lớn, chi phí duy trì hệ thống để xử lý lượng email rác có thể rất đáng kể, bao gồm cả chi phí phần mềm, phần cứng và nhân lực kỹ thuật. Việc hệ thống phải xử lý quá nhiều email rác cũng có thể làm giảm tốc độ truyền tải, gây chậm trễ trong việc nhận và gửi email hợp lệ, từ đó ảnh hưởng trực tiếp đến hoạt động kinh doanh. Như vậy, spam email không chỉ là vấn đề an toàn thông tin mà còn là gánh nặng về tài nguyên và chi phí vận hành.

Kết luận

Từ tất cả các vấn đề đã phân tích, có thể thấy rằng spam email không chỉ dừng lại ở mức độ gây phiền toái mà đã trở thành một rủi ro phức tạp, vượt xa khả năng xử lý của các kỹ thuật lọc truyền thống vốn dựa trên quy tắc cố định. Để đối phó hiệu quả với sự tinh vi và liên tục thay đổi của các hình thức thư rác, phishing hay email giả mạo, việc áp dụng các phương pháp hiện đại như Machine Learning, Natural Language Processing (NLP), trích xuất đặc trưng nội dung (Feature Extraction), và thậm chí là các mô hình học sâu (Deep Learning) đối với tập dữ liệu lớn là điều cần thiết. Các mô hình tiên tiến này không chỉ có khả năng tự học từ dữ liệu, mà còn có thể nhận diện các mẫu spam mới, thích nghi với chiến thuật tấn công thay đổi liên tục, phát hiện các hành vi bất thường, đồng thời giảm thiểu tỷ lệ phân loại sai (false positives và false negatives). Do đó, nghiên cứu và triển khai các hệ thống phát hiện spam email thông minh trở thành nhiệm vụ cấp bách nhằm bảo vệ người dùng, ngăn chặn tấn công mạng, tăng cường khả năng phòng vệ của hệ thống và đảm bảo an toàn thông tin trong kỷ nguyên số hóa hiện nay.

1.3. Giới thiệu về Google Colab

Google Colab, viết tắt của Collaboratory, là một dịch vụ của Google cho phép người dùng viết và chạy mã Python trong trình duyệt mà không cần phải cài đặt phần mềm trên máy tính cá nhân. Google Colab hỗ trợ các công cụ mạnh mẽ cho việc xử lý dữ liệu và phát triển mô hình học máy, đồng thời cho phép người dùng truy cập miễn phí vào các tài nguyên tính toán mạnh mẽ như GPU (Graphics Processing Unit) và TPU (Tensor Processing Unit).

1.3.1. Ưu điểm của Google Colab

Miễn phí và dễ sử dụng: Colab là một công cụ miễn phí, người dùng chỉ cần có tài khoản Google để sử dụng. Người dùng có thể bắt đầu lập trình mà không cần phải cài đặt bất kỳ phần mềm nào trên máy tính, chỉ cần trình duyệt web là đủ. Đây là lợi thế lớn đối với những người mới bắt đầu hoặc không muốn lo lắng về việc cài đặt và cấu hình môi trường phát triển.

Tài nguyên tính toán miễn phí (GPU/TPU): Một trong những ưu điểm nổi bật của Google Colab là việc cung cấp miễn phí tài nguyên tính toán như GPU và TPU. Điều này giúp người dùng có thể huấn luyện các mô hình học sâu (deep learning) và thực hiện các tác vụ tính toán phức tạp mà không cần phải đầu tư vào phần cứng đắt tiền. Việc sử dụng GPU/TPU giúp giảm thiểu thời gian huấn luyện mô hình, đặc biệt đối với các mô hình học sâu hoặc các bộ dữ liệu lớn.

Khả năng làm việc nhóm và chia sẻ tài liệu: Colab cho phép người dùng dễ dàng chia sẻ các notebook (sổ tay) với người khác và làm việc cùng nhau trên một dự án. Điều này rất hữu ích trong các dự án nhóm hoặc khi làm việc với các cộng tác viên từ xa. Các tài liệu trong Colab có thể được lưu trữ trên Google Drive, giúp người dùng dễ dàng truy cập và làm việc từ bất kỳ thiết bị nào có kết nối Internet.

Tích hợp với Google Drive: Colab có khả năng tích hợp với Google Drive, cho phép người dùng lưu trữ và quản lý dữ liệu trực tiếp trên đám mây. Điều này giúp giảm thiểu các rủi ro về việc mất dữ liệu và giúp người dùng dễ dàng chia sẻ dữ liệu với các thành viên trong nhóm.

Khả năng mở rộng và tích hợp với các công cụ khác: Google Colab cho phép người dùng tích hợp với nhiều công cụ khác như GitHub, BigQuery, và TensorBoard, giúp quản lý mã nguồn và theo dõi quá trình huấn luyện mô hình dễ dàng. Các tính năng

này giúp mở rộng khả năng sử dụng Colab trong nhiều tình huống khác nhau, từ việc kiểm thử mô hình đến phân tích dữ liệu.

1.3.2. Nhược điểm của Google Colab

Giới hạn tài nguyên miễn phí: Mặc dù Colab cung cấp tài nguyên tính toán miễn phí như GPU/TPU, nhưng tài nguyên này có giới hạn. Google Colab có chính sách hạn chế thời gian sử dụng tài nguyên tính toán (mỗi phiên làm việc kéo dài khoảng 12 giờ), và nếu sử dụng quá mức, tài khoản người dùng có thể bị tạm thời khóa hoặc hạn chế quyền truy cập vào tài nguyên tính toán. Điều này có thể gây khó khăn khi làm việc với các mô hình yêu cầu huấn luyện lâu dài hoặc xử lý bộ dữ liệu lớn.

Khả năng lưu trữ hạn chế: Dù người dùng có thể lưu trữ tài liệu trên Google Drive, nhưng bộ nhớ tạm thời trên Colab có hạn, và tất cả dữ liệu tải lên trong phiên làm việc sẽ bị mất khi phiên làm việc kết thúc. Điều này có nghĩa là người dùng cần phải lưu trữ dữ liệu quan trọng của mình ở nơi khác nếu không muốn mất mát dữ liệu khi phiên Colab kết thúc.

Khả năng kết nối Internet và phụ thuộc vào trình duyệt: Colab là một công cụ dựa trên nền tảng đám mây, vì vậy nó yêu cầu kết nối Internet ổn định để hoạt động. Nếu kết nối Internet bị gián đoạn, người dùng sẽ không thể tiếp tục làm việc trên các notebook của mình. Điều này có thể là một bất lợi đối với những người làm việc trong môi trường không ổn định hoặc ở những nơi có kết nối Internet yếu.

Thời gian xử lý lâu đói với các tác vụ phức tạp: Mặc dù Google Colab cung cấp GPU và TPU miễn phí, nhưng so với các dịch vụ tính toán chuyên nghiệp như Google Cloud AI hoặc AWS, tài nguyên này có thể bị giới hạn về tốc độ xử lý. Khi làm việc với các mô hình rất lớn hoặc huấn luyện trên các bộ dữ liệu khổng lồ, việc sử dụng Colab có thể không hiệu quả bằng các dịch vụ tính toán trả phí.

Không hỗ trợ các công cụ phần mềm phức tạp: Colab không hỗ trợ cài đặt các công cụ phần mềm phức tạp hoặc các phần mềm bên ngoài mà không thuộc về môi trường Python hoặc những công cụ được tích hợp sẵn. Điều này có thể hạn chế một số dự án yêu cầu cài đặt các phần mềm đặc biệt hoặc các công cụ không có sẵn trên nền tảng Colab.

Khó khăn trong việc quản lý môi trường ảo: Một số người dùng có thể gặp khó khăn trong việc quản lý các môi trường ảo (virtual environments) khi làm việc trên

Google Colab, vì nền tảng này đã thiết lập sẵn môi trường Python. Điều này có thể gây khó khăn khi cần phải cài đặt và sử dụng các phiên bản thư viện cụ thể hoặc khi người dùng cần phải kiểm soát môi trường phát triển.

1.4. Giới thiệu về Visual Studio Code

Visual Studio Code (VS Code) là một trình soạn thảo mã nguồn mạnh mẽ và miễn phí được phát triển bởi Microsoft. Đây là một công cụ rất phổ biến trong cộng đồng lập trình, đặc biệt là khi phát triển ứng dụng Python, học máy, và khoa học dữ liệu. Với khả năng mở rộng mạnh mẽ và các tính năng tích hợp sẵn, VS Code là lựa chọn lý tưởng cho việc phát triển các dự án phần mềm và học máy.

VS Code có các tính năng nổi bật như sau:

Tính năng linh hoạt và dễ sử dụng: Một trong những đặc điểm nổi bật của Visual Studio Code là tính linh hoạt và giao diện người dùng dễ sử dụng. VS Code có thể được tùy chỉnh với các tiện ích mở rộng (extensions) để hỗ trợ nhiều ngôn ngữ lập trình và công nghệ khác nhau. Người dùng có thể cài đặt các plugin phù hợp với nhu cầu của mình, giúp làm việc hiệu quả hơn với các dự án học máy và phân tích dữ liệu. Đặc biệt, VS Code có hỗ trợ ngôn ngữ Python, ngôn ngữ chính được sử dụng trong học máy. Các tính năng như IntelliSense (tự động hoàn thiện mã nguồn) và gợi ý cú pháp giúp lập trình viên nhanh chóng viết mã mà không gặp phải lỗi cú pháp. Điều này giúp tăng hiệu suất làm việc và giảm thiểu sai sót khi phát triển các mô hình học máy.

Hỗ trợ tích hợp Git và quản lý phiên bản: VS Code tích hợp sẵn hệ thống quản lý phiên bản Git, giúp người dùng dễ dàng theo dõi và quản lý các thay đổi trong mã nguồn. Với Git, người dùng có thể lưu trữ các phiên bản khác nhau của mã nguồn, điều này rất quan trọng khi phát triển các mô hình học máy. Quá trình huấn luyện mô hình học máy thường yêu cầu các thử nghiệm liên tục và điều chỉnh tham số, vì vậy việc theo dõi và quản lý phiên bản mã nguồn trở nên rất quan trọng. Bằng cách sử dụng Git trong VS Code, người phát triển có thể dễ dàng quay lại các phiên bản trước của mã nguồn nếu cần thiết, đảm bảo quá trình phát triển không bị gián đoạn. Điều này cũng hỗ trợ các nhóm làm việc, khi nhiều người có thể cùng nhau phát triển và kiểm tra các thay đổi mà không gặp phải xung đột mã nguồn.

Hỗ trợ gỡ lỗi và kiểm tra mã: VS Code hỗ trợ một hệ thống gỡ lỗi mạnh mẽ, giúp lập trình viên kiểm tra và sửa lỗi mã nguồn một cách nhanh chóng và hiệu quả. Khi làm

việc với các mô hình học máy phức tạp, quá trình gỡ lỗi rất quan trọng để xác định các vấn đề trong quá trình huấn luyện mô hình, như lỗi tính toán hoặc vấn đề với dữ liệu đầu vào. Các công cụ gỡ lỗi trong VS Code cho phép người dùng thiết lập các điểm dừng (breakpoints), theo dõi biến và trạng thái của mã trong quá trình thực thi, giúp dễ dàng phát hiện và khắc phục các lỗi trong mô hình học máy. Điều này giúp giảm thiểu thời gian phát triển và nâng cao hiệu quả của dự án.

Khả năng mở rộng thông qua extension: VS Code rất mạnh mẽ trong việc mở rộng tính năng thông qua các extension. Những extension này cho phép người dùng bổ sung các công cụ hỗ trợ cho các công việc như phân tích dữ liệu, học máy, xử lý văn bản, và nhiều lĩnh vực khác. Các extension phổ biến bao gồm Python extension, Jupyter extension, và Docker extension, giúp cải thiện quy trình phát triển và thử nghiệm mã. Các extension này giúp VS Code trở thành một công cụ linh hoạt có thể phục vụ cho nhiều loại dự án, từ việc phân tích dữ liệu cơ bản đến phát triển các mô hình học sâu phức tạp. Người dùng có thể cài đặt các công cụ cần thiết cho dự án của mình một cách dễ dàng và nhanh chóng.

Tóm lại, Visual Studio Code là công cụ phát triển linh hoạt, nhẹ nhàng nhưng đầy đủ tính năng, giúp lập trình viên dễ dàng xây dựng, kiểm tra và tối ưu hóa các mô hình học máy.

1.5. Giới thiệu về Machine Learning

Máy học (Machine Learning - ML) là một lĩnh vực con trong trí tuệ nhân tạo (AI), nghiên cứu và phát triển các thuật toán, phương pháp giúp máy tính có thể học hỏi từ dữ liệu và cải thiện hiệu suất làm việc mà không cần sự can thiệp hay lập trình chi tiết từ con người. Trái ngược với các phương pháp lập trình truyền thống, nơi con người cần phải chỉ rõ từng bước và quy tắc, trong học máy, các thuật toán tự động xây dựng các mô hình dựa trên dữ liệu đầu vào và học từ các mẫu, giúp cải thiện khả năng đưa ra quyết định hoặc dự đoán.

Học máy có thể được chia thành ba loại chính, mỗi loại có các ứng dụng và phương pháp khác nhau:

Học có giám sát (Supervised Learning): Trong học có giám sát, thuật toán học từ dữ liệu đã được gắn nhãn, tức là mỗi dữ liệu đầu vào đều đi kèm với một kết quả (nhãn). Mục tiêu của thuật toán là học cách chuyển từ đầu vào (input) sang đầu ra

(output) một cách chính xác. Đây là phương pháp phổ biến nhất trong bài toán phát hiện thư rác. Ví dụ, trong việc phân loại email thành "spam" và "ham", mỗi email đã được gắn nhãn trước đó, và thuật toán sẽ học cách nhận diện các đặc điểm của spam so với thư hợp lệ.

Các thuật toán học có giám sát điển hình bao gồm:

Naive Bayes: Một phương pháp đơn giản nhưng hiệu quả trong phân loại văn bản, dựa trên xác suất có điều kiện.

Support Vector Machine (SVM): Thuật toán phân loại mạnh mẽ, tìm kiếm siêu phẳng tối ưu để phân tách các lớp dữ liệu.

Decision Trees: Cây quyết định, sử dụng một cấu trúc cây để phân loại dữ liệu dựa trên các đặc trưng của chúng.

Random Forest: Một phương pháp sử dụng tập hợp nhiều cây quyết định để cải thiện độ chính xác và giảm overfitting.

Học không giám sát (Unsupervised Learning): Khác với học có giám sát, trong học không giám sát, dữ liệu không có nhãn và mục tiêu của thuật toán là tìm ra cấu trúc hoặc mẫu trong dữ liệu. Đây là phương pháp hữu ích khi không có sẵn dữ liệu đã được phân loại. Các thuật toán học không giám sát thường được sử dụng trong việc phân nhóm (clustering), giảm chiều dữ liệu (dimensionality reduction), và phát hiện bất thường (anomaly detection).

Một ví dụ điển hình trong học không giám sát là việc phân nhóm các email có cùng chủ đề hoặc nội dung vào các nhóm tương tự, giúp phát hiện các mẫu thư rác mà không cần phải biết trước liệu chúng là spam hay không.

Học tăng cường (Reinforcement Learning): Trong học tăng cường, mô hình học từ các phản hồi nhận được từ môi trường. Mô hình này không học từ dữ liệu đã biết mà thay vào đó sẽ thực hiện các hành động trong môi trường và nhận về phản hồi dưới dạng thưởng hoặc phạt. Mục tiêu của thuật toán là tối đa hóa tổng phần thưởng nhận được. Học tăng cường thường được áp dụng trong các bài toán như chơi game, robot tự động hoặc tối ưu hóa chiến lược trong các tình huống phức tạp.

Mặc dù học tăng cường ít được áp dụng trong phát hiện thư rác, nhưng nó có thể được sử dụng để tối ưu hóa các chiến lược lọc thư rác tự động trong môi trường thay đổi liên tục.

Học máy đã chứng minh được giá trị vượt trội trong việc phân tích dữ liệu và hỗ trợ ra quyết định trong nhiều lĩnh vực, từ nhận dạng hình ảnh, phân tích văn bản, đến dự báo tài chính. Trong bài toán phát hiện thư rác, học máy cho phép hệ thống không chỉ dựa vào các quy tắc cố định mà còn học hỏi từ các mẫu dữ liệu thực tế, cải thiện khả năng nhận diện thư rác theo thời gian.

1.6. Giới thiệu về Natural Language Processing (NLP)

Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) là một lĩnh vực của khoa học máy tính và trí tuệ nhân tạo, nghiên cứu cách máy tính có thể hiểu, phân tích và sinh ra ngôn ngữ tự nhiên của con người. NLP kết hợp các phương pháp từ ngôn ngữ học và học máy để xử lý và phân tích văn bản hoặc ngôn ngữ nói. Mục tiêu của NLP là giúp máy tính "hiểu" được nội dung, ngữ nghĩa, và ngữ pháp của ngôn ngữ con người một cách tự động, giúp ứng dụng được trong nhiều lĩnh vực khác nhau như dịch thuật tự động, tóm tắt văn bản, tìm kiếm thông tin và phân tích cảm xúc.

Trong bài toán phát hiện thư rác, NLP đóng một vai trò quan trọng trong việc trích xuất và xử lý các đặc trưng từ văn bản của email. Các kỹ thuật NLP giúp hệ thống học máy hiểu được cấu trúc và nội dung của email, từ đó phân loại chính xác xem email đó có phải là spam hay không. Các kỹ thuật NLP điển hình trong bài toán này bao gồm:

Tiền xử lý dữ liệu văn bản (Text Preprocessing):

Tiền xử lý là bước đầu tiên và rất quan trọng trong việc xử lý dữ liệu văn bản. Mục tiêu là chuẩn hóa và làm sạch dữ liệu để chuẩn bị cho việc phân tích và xây dựng mô hình. Các kỹ thuật tiền xử lý bao gồm:

Loại bỏ dấu câu và các ký tự đặc biệt: Loại bỏ các yếu tố không cần thiết trong văn bản.

Chuyển đổi chữ hoa thành chữ thường: Giúp giảm thiểu số lượng từ khác nhau trong dữ liệu.

Loại bỏ các từ không mang ý nghĩa (stop words): Như "the", "and", "is",... Những từ này thường không mang nhiều thông tin có ích cho việc phân loại.

Tách từ (Tokenization): Phân tách văn bản thành các đơn vị nhỏ hơn, thường là từ hoặc cụm từ, giúp máy tính dễ dàng xử lý.

Trích xuất đặc trưng (Feature Extraction):

Sau khi tiền xử lý, bước tiếp theo là chuyển đổi văn bản thành các đặc trưng mà mô hình học máy có thể hiểu được. Các phương pháp phổ biến trong trích xuất đặc trưng bao gồm:

Bag of Words (BoW): Là phương pháp đơn giản trong việc chuyển đổi văn bản thành một ma trận các tần suất từ, nơi mỗi cột đại diện cho một từ trong từ điển và mỗi dòng là một văn bản.

TF-IDF (Term Frequency - Inverse Document Frequency): Đây là phương pháp cải tiến BoW, giúp làm giảm trọng số của các từ xuất hiện quá thường xuyên trong toàn bộ bộ dữ liệu, vì chúng không mang lại nhiều thông tin.

Word Vectors (Word Embeddings): Sử dụng các mô hình học sâu để chuyển từ văn bản thành các vector trong không gian n-dimensional, giúp mô hình hiểu được mối quan hệ giữa các từ và cụm từ.

Ứng dụng mô hình học máy:

Sau khi trích xuất đặc trưng, các thuật toán học máy sẽ được áp dụng để phân loại email. Dữ liệu văn bản sẽ được đưa vào các mô hình học máy đã được huấn luyện từ trước (như Naive Bayes, SVM, Random Forest, v.v.) để dự đoán xem một email có phải là spam hay không. Mỗi thuật toán học máy có thể học các mẫu khác nhau trong dữ liệu và có thể mang lại hiệu quả khác nhau tùy thuộc vào đặc điểm của dữ liệu.

Trong bài toán phát hiện thư rác, NLP giúp các mô hình học máy khai thác được các đặc trưng ngữ nghĩa và ngữ pháp của email, từ đó giúp phân loại thư rác chính xác hơn. Việc áp dụng NLP không chỉ giúp cải thiện chất lượng của mô hình học máy mà còn giúp giảm thiểu sai sót trong việc phân loại thư rác, qua đó bảo vệ người dùng khỏi các mối đe dọa và tăng hiệu quả sử dụng email.

Kết luận

Việc kết hợp giữa học máy và NLP trong bài toán phát hiện thư rác không chỉ giúp hệ thống tự động hóa việc phân loại email mà còn giúp nâng cao độ chính xác của các mô hình phân loại. Các kỹ thuật NLP sẽ hỗ trợ trong việc hiểu và trích xuất các đặc trưng quan trọng từ dữ liệu văn bản, trong khi học máy sẽ giúp mô hình học từ các dữ

liệu thực tế và liên tục cải thiện khả năng phát hiện thư rác theo thời gian. Sự kết hợp này đã và đang mang lại hiệu quả cao trong việc bảo vệ người dùng khỏi các mối đe dọa an ninh mạng qua email.

1.7. Các thuật toán học máy

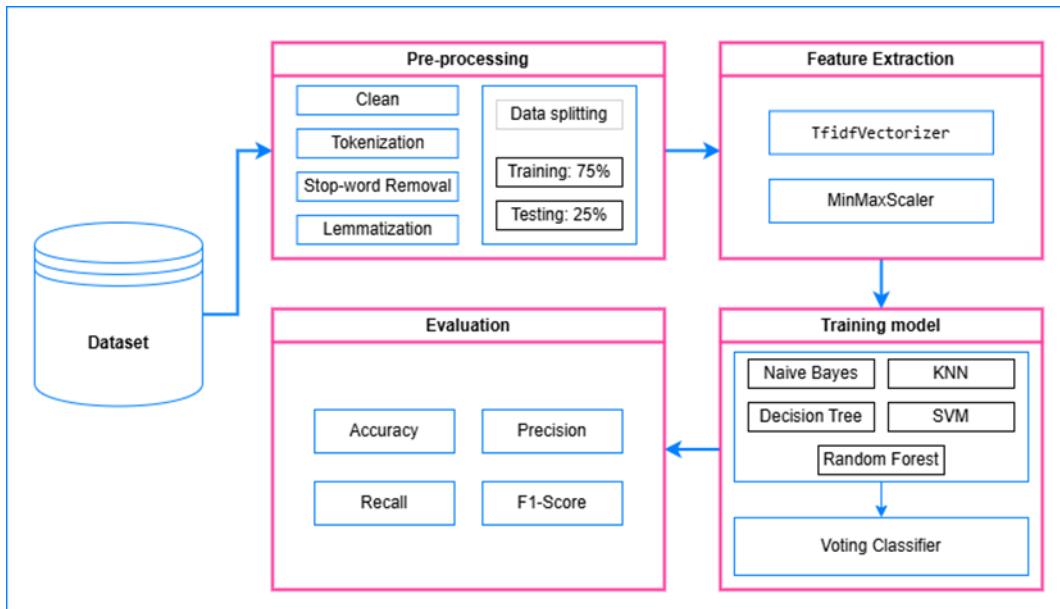
Nghiên cứu này điều tra hiệu quả của các thuật toán học máy trong việc giải quyết vấn đề phát hiện thư rác email, tập trung vào các thuật toán được biết đến với thành công của chúng trong các tác vụ phân loại văn bản. Các thuật toán được chọn là Naive Bayes, SVM, KNN, Random Forest và Decision Tree được chọn vì các điểm mạnh bổ sung và các phương pháp tiếp cận đa dạng để phân loại.

Naive Bayes, một mô hình xác suất giả định tính độc lập của tính năng, được công nhận rộng rãi vì hiệu quả của nó trong các tác vụ dựa trên văn bản, đặc biệt là khi làm việc với dữ liệu có chiều cao như nội dung email. Tính đơn giản và khả năng diễn giải của nó khiến nó trở thành một mô hình cơ sở mạnh mẽ để phát hiện thư rác.

Mặt khác, Support Vector Machine (SVM) rất phù hợp cho các tác vụ phân loại nhị phân. Tận dụng hạt nhân tuyến tính, SVM nhằm mục đích tìm siêu phẳng tối ưu để phân tách thư rác và email ham trong khi tối đa hóa biên độ giữa chúng. Tính mạnh mẽ của nó chống lại quá mức khiến nó trở thành một bổ sung thiết yếu cho phương pháp luận của chúng tôi.

K-Nearest Neighbors (KNN) được đưa vào vì tính đơn giản và dựa vào các số liệu tương đồng. Là một phương pháp phi tham số, KNN hoạt động tốt khi phân biệt các mẫu cục bộ, cung cấp thông tin chi tiết về hiệu suất của nó so với các mô hình tính toán chuyên sâu hơn.

Việc đưa vào các phương pháp tổng hợp—Random Forest và Decision Tree đảm bảo việc đánh giá các bộ phân loại tiên tiến, mạnh mẽ có khả năng xử lý các mối quan hệ phức tạp, phi tuyến tính trong dữ liệu. Random Forest xây dựng nhiều cây quyết định và tổng hợp các đầu ra của chúng, giảm phương sai và quá khớp.



Hình 1.1. Khung dự kiến của mô hình phân loại thư rác được đề xuất

Để kết hợp các điểm mạnh của các thuật toán riêng lẻ này, chúng tôi triển khai Voting Classifier, một kỹ thuật học tập tổng hợp các dự đoán từ nhiều mô hình. Voting Classifier kết hợp các đầu ra của năm mô hình (Naive Bayes, SVM, KNN, Random Forest và Decision Tree) để đưa ra dự đoán cuối cùng. Phương pháp này đặc biệt có lợi trong các tình huống như phát hiện thư rác, trong đó các thuật toán khác nhau có thể vượt trội trong việc nắm bắt các mẫu khác nhau trong dữ liệu.

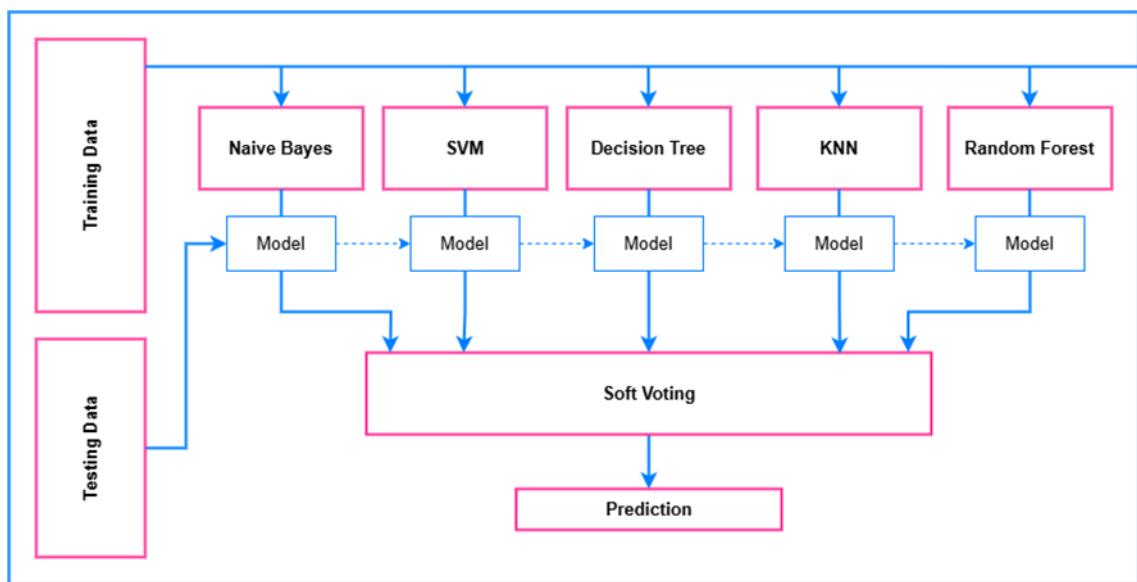
Voting Classifier dễ triển khai và có thể kết hợp các bộ phân loại có nhiều loại khác nhau (ví dụ: xác suất, dựa trên cây và dựa trên trường hợp). Tính linh hoạt này đảm bảo rằng các điểm mạnh riêng biệt của từng mô hình được tận dụng hiệu quả.

Bằng cách tổng hợp các dự đoán, Voting Classifier làm giảm khả năng xảy ra lỗi của từng mô hình chi phối quyết định cuối cùng. Nó đặc biệt có lợi khi các bộ phân loại cơ sở có hiệu suất tương đương và các điểm mạnh đa dạng.

So với các phương pháp như xếp chồng, dựa trên siêu phân loại, Voting Classifier ít có khả năng bị quá khớp hơn vì nó không đưa thêm các tham số hoặc độ phức tạp trong quá trình đào tạo.

Voting Classifier cung cấp những lợi thế riêng biệt so với các phương pháp tổng hợp khác bằng cách tận dụng tính đa dạng của các mô hình cơ sở. So với các phương pháp bagging như Random Forest, giúp giảm phương sai bằng cách đào tạo nhiều trường hợp của cùng một mô hình trên các tập hợp dữ liệu khác nhau, Voting Classifier kết hợp các dự đoán từ nhiều mô hình khác nhau, do đó nắm bắt hiệu quả các mẫu bổ sung trong

tập dữ liệu. Không giống như các phương pháp tăng cường như XGBoost, tinh chỉnh theo từng lần lặp lại các trình học yếu và có thể trở nên nhạy cảm với nhiễu, Voting Classifier tổng hợp các dự đoán mà không nhấn mạnh quá mức vào bất kỳ mô hình đơn lẻ nào, tăng cường tính mạnh mẽ chống lại các tập dữ liệu nhiễu. Hơn nữa, không giống như xếp chồng, kết hợp các trình học cơ sở thông qua một siêu phân loại và đưa vào thêm chi phí tính toán và rủi ro quá khó khăn, Voting Classifier sử dụng biểu quyết đa số (biểu quyết cứng) hoặc xác suất trung bình (biểu quyết mềm) cho các dự đoán cuối cùng, đảm bảo tính đơn giản và hiệu quả tính toán.



Hình 1. 2. Khung dự kiến của mô hình phân loại thư rác được đề xuất

1.7.1. Giới thiệu về Naive Bayes

Naive Bayes là một thuật toán phân loại đơn giản nhưng rất hiệu quả, đặc biệt trong các bài toán phân loại văn bản như Email Spam Detection. Naive Bayes dựa trên định lý Bayes và giả định rằng các đặc trưng của dữ liệu là độc lập với nhau (giả định "naive").

Quá trình hoạt động:

Giả định độc lập:

Naive Bayes giả định rằng tất cả các từ trong email là độc lập với nhau. Điều này có nghĩa là sự xuất hiện của một từ trong email không ảnh hưởng đến sự xuất hiện của các từ khác. Giả định này giúp đơn giản hóa quá trình tính toán xác suất.

Xác suất theo định lý Bayes:

Naive Bayes sử dụng định lý Bayes để tính xác suất của một email thuộc về một lớp nhất định (spam hoặc không spam). Định lý Bayes cho phép tính xác suất có điều kiện, dựa trên các đặc trưng có sẵn (ví dụ: các từ trong email).

Công thức cơ bản của định lý Bayes là:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

Trong đó:

$P(C|X)$: Đây là xác suất mà email X thuộc về lớp C (spam hoặc không spam), khi ta đã biết các từ trong email X.

$P(X|C)$: Đây là xác suất mà các từ trong email X xuất hiện nếu email thuộc lớp C (spam hoặc không spam).

$P(C)$: Đây là xác suất mà email là lớp C (spam hoặc không spam) trong dữ liệu huấn luyện. Ví dụ, tỉ lệ của các thư rác trong tổng số email đã biết.

$P(X)$: Đây là xác suất tổng quát của các từ trong email X, nó giúp chuẩn hóa xác suất

Huấn luyện mô hình:

Trong quá trình huấn luyện, Naive Bayes tính toán xác suất $P(X|C)$ cho mỗi từ trong email và xác suất $P(C)$ cho từng lớp (spam và ham) dựa trên dữ liệu huấn luyện.

Dự đoán: Khi dự đoán một email mới, Naive Bayes tính toán xác suất $P(C|X)$ cho mỗi lớp (spam và ham) và phân loại email vào lớp có xác suất cao nhất.

Xử lý với nhiều từ: Để tính toán xác suất với nhiều từ trong email, Naive Bayes sử dụng giả định độc lập và tính toán xác suất của tất cả các từ trong email. Xác suất của một email là sự nhân của xác suất các từ trong email.

1.7.2. Giới thiệu về K-Nearest Neighbors

K-Nearest Neighbors (KNN) là một thuật toán học máy đơn giản nhưng mạnh mẽ, thuộc nhóm các thuật toán học không giám sát (instance-based learning). KNN được sử dụng để phân loại các đối tượng trong một không gian nhiều chiều dựa trên sự tương đồng của chúng với các đối tượng trong tập huấn luyện.

KNN hoạt động theo nguyên lý rất đơn giản: khi cần phân loại một đối tượng (trong trường hợp email), thuật toán sẽ tìm k đối tượng (email) gần nhất trong không gian đặc trưng của tập huấn luyện, sau đó phân loại email đó dựa trên các nhãn (labels) của các đối tượng gần nhất.

Quá trình hoạt động của KNN như sau:

Tiền xử lý và chuyển đổi văn bản thành đặc trưng (features): Các email đầu vào được biến đổi thành các đặc trưng có thể so sánh được, thường là các vector số học. Các đặc trưng này có thể được tạo ra thông qua các phương pháp như CountVectorizer hoặc TF-IDF.

Tính khoảng cách giữa các đối tượng: Sau khi đã có đặc trưng của email cần phân loại, thuật toán tính toán khoảng cách giữa email này và tất cả các email trong tập huấn luyện, thường sử dụng các chỉ số khoảng cách như Euclidean Distance hoặc Cosine Similarity.

Chọn k đối tượng gần nhất: Thuật toán chọn k email gần nhất (tùy thuộc vào giá trị của k, ví dụ k=3, k=5, k=7), và các email này sẽ có ảnh hưởng lớn đến việc phân loại email hiện tại.

Phân loại email: Sau khi tìm được k email gần nhất, thuật toán sẽ dựa trên nhãn của chúng để phân loại email cần dự đoán. Cách phân loại có thể là majority voting (bỏ phiếu phổ biến), tức là nhãn mà xuất hiện nhiều nhất trong k đối tượng gần nhất sẽ là nhãn của email hiện tại.

KNN không có quá trình huấn luyện phức tạp. Nó là một thuật toán kiểu "lazy learning", tức là không tạo ra một mô hình học trước mà chỉ lưu trữ các điểm dữ liệu để sử dụng khi cần phân loại.

Trong trường hợp email spam, KNN sẽ phân loại email thành "spam" hoặc "ham" (không spam) dựa trên sự tương tự với các email trong tập huấn luyện đã được gán nhãn.

1.7.3. Giới thiệu về Decision Tree

Cây quyết định (Decision tree) là một thuật toán học máy trong nhóm các thuật toán học có giám sát, được sử dụng rộng rãi trong phân loại và hồi quy. Trong bài toán phân loại email spam, cây quyết định có thể được sử dụng để xác định xem một email có phải là spam hay không dựa trên các đặc trưng của nó.

Cây quyết định xây dựng một mô hình phân loại bằng cách sử dụng một cấu trúc cây, trong đó mỗi nút đại diện cho một câu hỏi hoặc quyết định về một đặc trưng, và các nhánh của nó dẫn đến các kết quả hoặc các quyết định tiếp theo. Cây được xây dựng bằng cách phân chia các đối tượng trong tập huấn luyện thành các nhóm con, dựa trên các đặc trưng sao cho sự phân chia này tối ưu cho việc phân loại.

Quá trình hoạt động của cây quyết định:

Chọn đặc trưng phân chia: Đầu tiên, cây quyết định sẽ chọn đặc trưng tốt nhất (thường là bằng cách tối ưu hóa một tiêu chí như Entropy hoặc Gini Impurity) để phân chia tập dữ liệu tại nút gốc. Tiêu chí này quyết định mức độ "thuần khiết" của các nhóm con sau khi phân chia.

Tạo các nhánh: Sau khi phân chia, dữ liệu được chia thành các nhóm con, mỗi nhóm con sẽ có các đặc trưng giống nhau. Cây tiếp tục tạo các nhánh con cho đến khi đạt được một số điều kiện dừng, như tất cả các dữ liệu trong một nhóm đều có cùng nhãn (hoặc rất ít sự khác biệt), hoặc độ sâu của cây đạt đến giới hạn.

Dự đoán: Để phân loại một email, chúng ta đi qua các nhánh của cây, trả lời các câu hỏi cho đến khi đạt đến một lá (nút cuối cùng của cây). Nút lá chứa nhãn cuối cùng, xác định liệu email là "spam" hay "ham" (không spam).

Tiêu chí chia:

Gini Impurity: Đo lường mức độ "thuần khiết" của các lớp trong một nhóm con. Mục tiêu là giảm Gini impurity, nghĩa là giảm sự không đồng nhất trong các nhóm con.

Entropy: Đo lường sự không chắc chắn của dữ liệu. Mục tiêu là giảm entropy, tức là giảm sự không chắc chắn trong các nhánh.

Cắt tia cây (Pruning): Để tránh overfitting, các cây quyết định có thể được "cắt tia" bằng cách loại bỏ các nhánh không cần thiết, giúp cây trở nên đơn giản hơn và có khả năng tổng quát tốt hơn.

1.7.4. Giới thiệu về Support Vector Machine

Support Vector Machine (SVM) là một thuật toán học máy mạnh mẽ được sử dụng rộng rãi trong phân loại và hồi quy, đặc biệt là trong các bài toán phân loại văn bản như Email Spam Detection.

Quá trình hoạt động:

Tìm siêu phẳng phân tách (Hyperplane):

Mục tiêu của SVM là tìm một siêu phẳng (hyperplane) phân tách các điểm dữ liệu trong không gian đặc trưng sao cho khoảng cách giữa các điểm dữ liệu thuộc các lớp khác nhau (spam và không spam) là lớn nhất. Siêu phẳng này sẽ được sử dụng để phân loại các điểm dữ liệu mới.

SVM tìm kiếm siêu phẳng tối ưu bằng cách tối đa hóa khoảng cách (margin) giữa các điểm dữ liệu của hai lớp. Điểm dữ liệu gần siêu phẳng nhất được gọi là các vector hỗ trợ (support vectors), và chúng là những điểm quan trọng nhất trong việc xác định siêu phẳng.

Kernel Trick (Kỹ thuật Kernel):

Trong trường hợp các điểm dữ liệu không thể phân tách bằng một siêu phẳng trong không gian đặc trưng ban đầu (ví dụ, khi các lớp không tuyến tính), SVM sử dụng một kỹ thuật gọi là kernel trick.

Kernel trick chuyển dữ liệu vào một không gian đặc trưng cao hơn, nơi các lớp có thể được phân tách bằng siêu phẳng. Một số loại kernel phổ biến là linear kernel, polynomial kernel, và radial basis function (RBF) kernel.

Huấn luyện mô hình:

Trong quá trình huấn luyện, SVM tối ưu hóa siêu phẳng bằng cách sử dụng các thuật toán tối ưu như Gradient Descent hoặc Quadratic Programming để tối đa hóa khoảng cách giữa các lớp.

Mô hình sẽ học được một siêu phẳng phân tách sao cho khi gặp các dữ liệu mới, mô hình có thể phân loại chúng chính xác.

Dự đoán:

Sau khi mô hình đã được huấn luyện, khi có một email mới, SVM sẽ sử dụng siêu phẳng đã học để phân loại email đó là spam hoặc non-spam.

Nếu email nằm phía một bên của siêu phẳng, nó sẽ được phân loại là spam; nếu nó nằm phía bên kia, nó sẽ được phân loại là non-spam.

Cách xử lý với dữ liệu phi tuyến tính:

Khi các dữ liệu không thể phân tách bằng một siêu phẳng trong không gian ban đầu (dữ liệu phi tuyến tính), kernel được sử dụng để ánh xạ các điểm vào không gian có chiều cao hơn, nơi dữ liệu có thể được phân tách một cách tuyến tính.

1.7.5. Giới thiệu về Random Forest

Random Forest là một thuật toán học máy dựa trên ensemble learning (học tập kết hợp), được sử dụng rộng rãi trong phân loại và hồi quy. Trong bài toán Email Spam Detection, Random Forest hoạt động như sau:

Quá trình hoạt động:

Tạo các cây quyết định (Decision Trees):

Random Forest xây dựng nhiều cây quyết định (có thể là hàng nghìn cây) từ các tập con ngẫu nhiên của dữ liệu huấn luyện. Mỗi cây quyết định sẽ học cách phân loại email là spam hay không spam dựa trên các đặc trưng của email.

Mỗi cây sử dụng một phần mẫu ngẫu nhiên (bootstrapping) và chỉ xét một tập hợp con ngẫu nhiên của các đặc trưng (features) tại mỗi điểm chia (split) trong cây.

Huấn luyện mô hình:

Quá trình huấn luyện diễn ra qua việc xây dựng nhiều cây quyết định độc lập từ các dữ liệu con khác nhau. Các cây này được huấn luyện độc lập và không có mối liên hệ với nhau.

Các cây quyết định này có thể cho ra những dự đoán khác nhau, nhưng Random Forest sẽ kết hợp tất cả kết quả của các cây quyết định.

Dự đoán (Prediction):

Khi dự đoán một email mới, Random Forest sử dụng phương pháp bỏ phiếu (voting): mỗi cây quyết định sẽ đưa ra một kết quả (spam hay không spam), và kết quả cuối cùng được quyết định dựa trên số lượng cây đồng thuận.

Phân loại đa số (Majority Voting): Email được phân loại là spam nếu phần lớn cây quyết định đưa ra kết quả spam, ngược lại nếu đa số cây cho kết quả không spam.

Đánh giá mô hình:

Mô hình Random Forest có thể được đánh giá dựa trên các chỉ số như accuracy, precision, recall, và F1-score. Vì Random Forest sử dụng nhiều cây quyết định, nó thường ít bị overfitting và có thể đạt độ chính xác cao.

1.7.6. Giới thiệu về Voting Classifier

Voting Classifier là một kỹ thuật học tập tổng hợp kết hợp nhiều mô hình để cải thiện độ chính xác của phân loại. Thay vì dựa vào một mô hình duy nhất, nó tổng hợp các dự đoán từ nhiều bộ phân loại (ví dụ: Naïve Bayes, SVM, KNN, Decision Tree, Random Forest) để đưa ra quyết định cuối cùng.

Bằng cách sử dụng Voting Classifier, ta có thể tận dụng thế mạnh của từng mô hình, dẫn đến độ chính xác cao hơn và khả năng khai quát hóa tốt hơn trong việc phát hiện thư rác qua email.

Voting Classifier kết hợp các dự đoán bằng hai phương pháp chính:

(a) Hard Voting (Biểu quyết đa số)

Mỗi mô hình dự đoán Spam hoặc Non-Spam và phân loại cuối cùng dựa trên biểu quyết đa số:

$$Prediction(x) = \arg \max_y \sum_{i=1}^N I(Model_i(x) = y)$$

Trong đó:

N là số lượng mô hình.

$Model_i(x)$ là dự đoán của mô hình thứ i .

$I(Model_i(x) = y)$ là 1 nếu mô hình dự đoán lớp y , nếu không thì 0.

Lớp có nhiều phiếu bầu nhất sẽ là lớp dự đoán cuối cùng.

(b) Soft Voting (Xác suất trung bình có trọng số)

Nếu các mô hình đưa ra xác suất thay vì chỉ nhãn, ta tính tổng có trọng số của các xác suất:

$$Prediction(x) = \arg \max_y \sum_{i=1}^N \omega_i \cdot P(Model_i(x) = y)$$

Trong đó:

ω_i là trọng số được gán cho mỗi mô hình (mặc định là trọng số bằng nhau).

$P(Model_i(x) = y)$ là xác suất của lớp y được dự đoán bởi mô hình i .

Soft Voting thường hoạt động tốt hơn Hard Voting vì nó xem xét mức độ tin cậy của mô hình.

Bước 1: Đào tạo các mô hình cá nhân

Đào tạo nhiều bộ phân loại trên cùng một tập dữ liệu, chẳng hạn như:

Naïve Bayes (tốt cho dữ liệu văn bản)

Support Vector Machines (SVM) (hiệu quả cho dữ liệu có chiều cao)

K-Nearest Neighbors (KNN) (dựa trên sự tương đồng)

Decision Tree (quy tắc có thể diễn giải)

Random Forest (tập hợp các cây quyết định)

Bước 2: Kết hợp Dự đoán với Voting

Đối với Hard Voting: Mỗi mô hình đưa ra Spam hoặc Non-Spam và dự đoán phổ biến nhất sẽ được chọn.

Đối với Soft Voting: Các mô hình đưa ra xác suất và tính toán một giá trị trung bình có trọng số.

Bước 3: Phân loại Email mới

Email được chuyển qua tất cả các bộ phân loại.

Đầu ra của chúng được tổng hợp bằng cách sử dụng biểu quyết đa số (Hard) hoặc trung bình xác suất (Soft).

Nhận cuối cùng là Spam hoặc Non-Spam.

CHƯƠNG 2: PHÂN TÍCH XÂY DỰNG MÔ HÌNH

2.1. Tổng quan mô hình

Trong việc phát hiện thư rác (email spam detection), mô hình học máy (Machine Learning - ML) đã trở thành công cụ mạnh mẽ giúp tự động phân loại email thành hai nhóm chính: thư rác (spam) và thư hợp lệ (ham). Mô hình này dựa vào các thuật toán học máy để nhận diện các mẫu và đặc trưng từ dữ liệu, từ đó dự đoán xem một email mới có phải là spam hay không.

2.2. Xây dựng tập dữ liệu

2.2.1. Thu thập dữ liệu

Bộ dữ liệu: [spam.csv](#).

Gồm 5572 dòng dữ liệu gồm các tin nhắn và nhãn của từng tin nhắn đi kèm.

Có 2 nhãn:

“spam”: tin nhắn này là spam.

“ham”: tin nhắn này không phải là spam

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	v1	v2													
2	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...													
3	ham	OK lar... Joking wif u oni...													
4	spam	Free entry in 2 wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's													
5	ham	U dun say so early hor... U c already then say...													
6	ham	Nah I don't think he goes to usf, he lives around here though													
7	spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! Xxx std chgs to send, 老1.50 to rcv													
8	ham	Even my brother is not like to speak with me. They treat me like aids patient.													
9	ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune													
10	spam	WINNER!! As a valued network customer you have been selected to receive a 老900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.													
11	spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030													
12	ham	I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.													
13	spam	SIX chances to win CASH! From 100 to 20,000 pounds txt CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info													
14	spam	URGENT! You have won a 1 week FREE membership in our 老100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD POBOX 4403LDNW1A7RW18													
15	ham	I've been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulfil my promise. You have been wonderful and a blessing at all time!													
16	ham	I HAVE A DATE ON SUNDAY WITH WILL!!													
17	spam	XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap. xxmobilemovieclub.com?n=QJKGIGHJJGCBL													
18	ham	Oh k...i'm watching here:)													
19	ham	Eh u remember how 2 spell his name... Yes i did. He v naughty make until i v wet.													
20	ham	Fine if that 老s the way u feel. That 老s the way its gotta b													
21	spam	England v Macedonia - dont miss the goals/team news. Txt ur national team to 87077 eg ENGLAND to 87077 Try:WALES, SCOTLAND 4txt/7老1.20 POBOXox36504W45WQ 16+													
22	ham	Is that seriously how you spell his name?													
23	ham	I 老 going to try for 2 months ha only joking													
24	ham	So 老, pay first lar... Then when is da stock comin...													
25	ham	Aft i finish my lunch then i go str down lor. Ard 3 smth lor. U finish ur lunch already?													

Hình 2.1. Dataset

2.2.2. Tiền xử lý dữ liệu

Loại bỏ giá trị trùng lặp: Trong bộ dữ liệu Spam, có một số dòng bị trùng lặp. Điều này có thể gây ảnh hưởng đến độ chính xác của mô hình, vì vậy các dòng trùng lặp cần được loại bỏ để đảm bảo chất lượng dữ liệu.

Xử lý các giá trị Null: Mặc dù cột v1 và v2 không có giá trị Null, nhưng một số cột không tên (Unnamed: 2, Unnamed: 3, Unnamed: 4) có chứa nhiều giá trị Null. Các cột này có thể không cần thiết cho quá trình phân tích, do đó chúng sẽ bị loại bỏ để giảm thiểu độ phức tạp của dữ liệu.

Đổi tên cột: Các cột v1 và v2 không rõ ràng, vì vậy cần đổi tên chúng thành Label (cho cột v1) và Message (cho cột v2) để dễ hiểu và thuận tiện cho việc xử lý sau này.

Tạo cột nhị phân 'Spam': Cột Label chứa hai giá trị là 'ham' và 'spam'. Để thuận tiện cho việc huấn luyện mô hình, chúng ta thêm một cột mới với tên Spam, trong đó đánh dấu giá trị 1 cho 'spam' và 0 cho 'ham'. Điều này giúp mô hình phân biệt dễ dàng hơn giữa các loại tin nhắn.

Trước khi đưa dữ liệu vào mô hình học máy, quá trình tiền xử lý văn bản giúp loại bỏ các yếu tố gây nhiễu, chuẩn hóa dữ liệu và tối ưu hóa hiệu suất mô hình. Điều này đặc biệt quan trọng trong nhận diện email spam, nơi nội dung email có thể chứa nhiều ký tự dư thừa hoặc không cần thiết.

Loại bỏ mã HTML: Các email thường chứa mã HTML như <html>, <body> hoặc các thẻ không cần thiết khác. Việc loại bỏ những thẻ này giúp mô hình tập trung vào nội dung văn bản thay vì các thành phần trình bày.

Loại bỏ liên kết, số và địa chỉ email: Email spam thường chứa nhiều đường dẫn đến các trang web lừa đảo hoặc các địa chỉ email nhắm mục đích thu thập thông tin người dùng. Những thành phần này không đóng vai trò quan trọng trong việc phân loại email nên cần được loại bỏ.

Chuyển đổi chữ thường: Để giảm thiểu sự phân biệt không cần thiết giữa chữ hoa và chữ thường, toàn bộ văn bản sẽ được chuyển về dạng chữ thường. Điều này giúp đảm bảo rằng từ viết hoa và từ viết thường được xử lý như nhau.

Xóa ký tự đặc biệt và dấu câu: Các ký tự như !, @, #, %, dấu chấm, dấu phẩy,... thường không mang nhiều ý nghĩa trong phân loại email, do đó chúng sẽ được loại bỏ để giữ lại nội dung chính của văn bản.

Loại bỏ từ dừng (stopwords): Các từ phổ biến như "the", "and", "is" trong tiếng Anh hoặc "và", "là" trong tiếng Việt không giúp phân biệt email spam và ham, nên được loại bỏ để giảm độ phức tạp của mô hình.

Lemmatization: Đây là quá trình đưa các từ về dạng nguyên thể (ví dụ: "running" → "run", "better" → "good"). Bằng cách này, mô hình có thể nhận diện các từ có cùng ý nghĩa mà không bị ảnh hưởng bởi các biến thể của chúng.

Sau khi áp dụng các bước trên, dữ liệu văn bản trở nên gọn gàng và có ý nghĩa hơn, giúp mô hình học máy đưa ra dự đoán chính xác hơn về email spam.

```
# Hàm làm sạch dữ liệu văn bản
def clean_text(text):
    # Loại bỏ thẻ HTML
    text = re.sub(r'<.*?>', '', text)

    # Loại bỏ các liên kết website
    text = re.sub(r"http\S+", '', text)

    # Loại bỏ các chữ số
    text = re.sub(r'\d+', '', text)

    # Loại bỏ email
    text = re.sub(r"\S*@\S*\s?", '', text)

    # Chuyển tất cả ký tự thành chữ thường
    text = text.lower()

    # Tokenize: tách văn bản thành các từ
    words = nltk.word_tokenize(text)

    # Giữ lại các từ chỉ gồm chữ cái (loại bỏ ký tự đặc biệt và số)
    words = [w for w in words if w.isalpha()]

    # Loại bỏ stopwords (từ dừng) tiếng Anh
    stop_words = set(stopwords.words('english'))
    words = [w for w in words if w not in stop_words]

    # Lemmatization: đưa các từ về dạng gốc
    lemmatizer = WordNetLemmatizer()
    words = [lemmatizer.lemmatize(w) for w in words]

    # Nối các từ lại thành chuỗi văn bản
    text = ' '.join(words)

return text
```

Hình 2.2. Tiền xử lý dữ liệu

2.2.3. Xác định đặc trưng và lựa chọn đặc trưng

Trong bước này, mục tiêu là tìm ra những đặc trưng quan trọng giúp phân biệt thư rác và thư không phải thư rác. Đặc trưng trong bài toán này chủ yếu là các từ hoặc nhóm từ (n-gram) trong nội dung thư. Bằng cách phân tích các từ khóa hoặc cụm từ thường xuất hiện trong thư rác hoặc không phải thư rác, chúng ta có thể xác định được những đặc trưng nổi bật.

Tiền xử lý dữ liệu và trích xuất đặc trưng: Sau khi chia bộ dữ liệu, chúng ta tiến hành tiền xử lý và trích xuất đặc trưng từ các tin nhắn trong bộ dữ liệu. Để chuyển đổi

các tin nhắn thành dạng mà mô hình có thể xử lý, chúng ta sử dụng phương pháp TfidfVectorizer. Phương pháp này không chỉ đếm số lần xuất hiện của từ trong văn bản mà còn tính toán tầm quan trọng của từ đó trong toàn bộ tập dữ liệu. Điều này giúp mô hình tập trung vào các từ mang ý nghĩa quan trọng thay vì chỉ dựa vào tần suất xuất hiện, từ đó cải thiện khả năng phân loại email spam chính xác hơn.

Lựa chọn đặc trưng: Sau khi xác định được các đặc trưng tiềm năng, việc lựa chọn đặc trưng sẽ giúp giảm thiểu độ phức tạp và cải thiện hiệu suất mô hình. Chúng ta có thể sử dụng các phương pháp như CountVectorizer hoặc TF-IDF để chọn ra những đặc trưng (từ hoặc nhóm từ) quan trọng nhất.

2.2.4. Trích xuất đặc trưng

Trong lĩnh vực xử lý ngôn ngữ và văn bản, trích xuất tính năng đóng vai trò quan trọng trong việc chuyển đổi dữ liệu văn bản thô thành định dạng có cấu trúc phù hợp với các thuật toán học máy. Một trong những phương pháp được sử dụng phổ biến nhất để trích xuất tính năng trong dữ liệu văn bản là Count Vectorizer (CV). Đây là một phương pháp đơn giản nhưng hiệu quả, chuyển đổi văn bản thành dữ liệu số bằng cách tạo ma trận đếm mã thông báo từ một tập hợp các tài liệu văn bản.

Ý tưởng cơ bản đằng sau Count Vectorizer là các tài liệu có nội dung tương tự có xu hướng có số lượng từ tương tự. Phương pháp này coi mỗi tài liệu là một Túi từ, trong đó tần suất của từng từ được ghi lại trên tất cả các tài liệu. Kết quả là một ma trận thưa thớt, trong đó mỗi hàng biểu thị một tài liệu, mỗi cột biểu thị một từ trong vốn từ vựng và các giá trị biểu thị tần suất của một từ cụ thể trong một tài liệu cụ thể. Phương pháp này đặc biệt hữu ích cho các tác vụ như phân loại văn bản, phân tích tình cảm và phát hiện thư rác, trong đó sự hiện diện hoặc vắng mặt của một số từ nhất định rất quan trọng để phân biệt giữa các danh mục khác nhau.

Mặc dù đơn giản và dễ triển khai, Count Vectorizer có một hạn chế: nó không xem xét tầm quan trọng của các từ trong toàn bộ ngữ liệu. Nói cách khác, nó xử lý tất cả các từ như nhau, bất kể chúng xuất hiện thường xuyên như thế nào trong tập dữ liệu. Điều này có thể gây ra vấn đề, vì các từ phổ biến xuất hiện trong nhiều tài liệu có thể chiếm ưu thế trong không gian tính năng và làm lu mờ các từ có ý nghĩa hơn nhưng ít phổ biến hơn.

Để giải quyết vấn đề này, một phương pháp thay thế được gọi là Tần suất thuật ngữ-Tần suất tài liệu nghịch đảo (TF-IDF) thường được sử dụng. Không giống như Count Vectorizer, TF-IDF không chỉ xem xét tần suất của một từ trong một tài liệu duy nhất (Tần suất thuật ngữ - TF) mà còn đo lường mức độ hiếm của từ đó trong tất cả các tài liệu (Tần suất tài liệu nghịch đảo - IDF). Điều này giúp giảm tác động của các từ thường gặp như "a", "the" và "and", những từ không đóng góp nhiều ý nghĩa cho chủ đề của tài liệu.

Về mặt toán học, TF-IDF được tính là tích của TF và IDF. Thành phần TF đo lường tần suất xuất hiện của một từ trong một tài liệu cụ thể, trong khi thành phần IDF đánh giá ý nghĩa của từ đó trong toàn bộ tập dữ liệu. Do đó, các từ xuất hiện thường xuyên chỉ trong một vài tài liệu (chẳng hạn như "học máy" trong các bài viết liên quan đến AI) có tầm quan trọng cao hơn, trong khi các từ xuất hiện trong tất cả các tài liệu có tầm quan trọng thấp hơn.

Nghiên cứu này chủ yếu tập trung vào việc sử dụng TF-IDF để trích xuất tính năng. Trong khi Count Vectorizer đơn giản hơn để triển khai và hiệu quả về mặt tính toán, TF-IDF đặc biệt phù hợp để xử lý các tập dữ liệu lớn với các tài liệu có độ dài khác nhau. Nó đặc biệt hiệu quả trong phân loại tài liệu, phân cụm và mô hình chủ đề, vì nó giúp làm nổi bật các thuật ngữ có liên quan nhất để phân biệt các loại tài liệu khác nhau.

Tóm lại, trong khi Count Vectorizer là một kỹ thuật thiết yếu để chuyển đổi văn bản phi cấu trúc thành các biểu diễn số, thì nó không tính đến tầm quan trọng của từ trong ngữ liệu. Mặt khác, TF-IDF cân bằng tần suất thuật ngữ với tần suất tài liệu nghịch đảo, khiến nó trở thành lựa chọn phù hợp hơn cho các tác vụ yêu cầu phân biệt giữa các từ quan trọng và không quan trọng. Bằng cách áp dụng TF-IDF, các mô hình học máy có thể nắm bắt tốt hơn bản chất thực sự của dữ liệu văn bản và cải thiện hiệu suất phân loại của chúng.

Bag-of-Words with N-grams

Mô hình Term Frequency-Inverse Document Frequency (TF-IDF) biểu diễn dữ liệu văn bản bằng cách đo lường tầm quan trọng của các từ trong một tài liệu so với toàn bộ tập dữ liệu. Không giống như mô hình Bag-of-Words (BoW), chỉ tính số lần xuất hiện của từ, TF-IDF điều chỉnh tần suất từ bằng cách giảm ảnh hưởng của các từ thường xuất hiện, giúp nó hiệu quả hơn đối với các tác vụ phân loại văn bản.

Biểu diễn Unigram

Sử dụng TfidfVectorizer() với các thiết lập mặc định của nó, trích xuất các unigram (từ đơn) trong khi xem xét tầm quan trọng của chúng trên toàn bộ ngũ liệu.

Ví dụ về chuyển đổi

Văn bản gốc: "Get FREE iphone now!".

Trích xuất các unigram có giá trị TF-IDF: 'get', 'free', 'iphone', 'now'.

Mỗi unigram được gán một trọng số dựa trên tầm quan trọng của nó trong tập dữ liệu, giúp cải thiện độ chính xác của phân loại.

Bằng cách sử dụng các thiết lập mặc định của TfidfVectorizer(), mô hình đạt được hiệu suất tốt hơn so với việc sử dụng bigram (ngram_range=(2,2)). Điều này cho thấy rằng các từ riêng lẻ là các tính năng hiệu quả hơn để phân biệt giữa các tin nhắn spam và không phải spam trong tập dữ liệu đã cho.

Đặc trưng số

Các tính năng được trích xuất:

Word Count: Tổng số từ trong một email.

Char Count: Tổng số ký tự trong một email.

Sentence Count: Tổng số câu trong một email.

Chuẩn hóa: Để xử lý các phạm vi khác nhau của các thuộc tính này, hãy sử dụng MinMaxScaler để chuẩn hóa chúng theo thang điểm [0, 1]. Các thuộc tính này có thể tiết lộ các đặc điểm phân biệt, chẳng hạn như sử dụng quá nhiều từ hoặc câu trong email rác.

```

# Tao đối tượng TfidfVectorizer
tfidf_vect = TfidfVectorizer()
tfidf_vect.fit(train_set['cleaned_text']) # Huấn luyện vectorizer trên tập huấn luyện

# Biểu diễn dữ liệu văn bản dưới dạng vector TF-IDF
x_train_text = tfidf_vect.transform(train_set['cleaned_text']) # Chuyển đổi tập huấn luyện
x_test_text = tfidf_vect.transform(test_set['cleaned_text']) # Chuyển đổi tập kiểm tra

# Lưu mô hình TfidfVectorizer đã huấn luyện vào file để sử dụng sau
with open('tfidf_vect_model.pkl', 'wb') as f:
    pickle.dump(tfidf_vect, f)

# Chuẩn hóa các đặc trưng số (Numeric Features)
scaler = MinMaxScaler()

# Danh sách các cột đặc trưng số
numeric_features = ['Num_Char', 'Num_Word', 'Num_Sen', 'num_words_transform']

# Chuẩn hóa dữ liệu tập huấn luyện về khoảng [0,1]
x_train_numeric = scaler.fit_transform(train_set[numeric_features])

# Chuẩn hóa dữ liệu tập kiểm tra dựa trên thông số đã học từ tập huấn luyện
x_test_numeric = scaler.transform(test_set[numeric_features])

# Lưu scaler
with open('scaler_model.pkl', 'wb') as f:
    pickle.dump(scaler, f)

# Kết hợp các đặc trưng văn bản và đặc trưng số thành một ma trận đặc trưng chung
x_train_combined = hstack([x_train_text, x_train_numeric])
x_test_combined = hstack([x_test_text, x_test_numeric])

```

Hình 2.3.Xác định và trích xuất đặc trưng

2.3. Tiêu chí đánh giá

Để đánh giá hiệu quả của mô hình phân loại thư rác (Email Spam Detection), chúng ta sẽ sử dụng một số tiêu chí đánh giá phổ biến trong học máy, giúp đo lường chính xác khả năng phân loại đúng các tin nhắn là thư rác hay không phải thư rác.

Để nâng cao hiệu quả của hệ thống hơn nữa, quá trình tối ưu hóa siêu tham số đã được thực hiện bằng GridSearchCV. Phương pháp này tìm kiếm có hệ thống qua các giá trị siêu tham số được xác định trước để tìm cấu hình có hiệu suất tốt nhất cho từng mô hình. Các phiên bản được tối ưu hóa của Random Forest, SVM và Voting Classifier đã được đánh giá và so sánh với các phiên bản không được tối ưu hóa của chúng. Kết quả chứng minh rằng các mô hình được tối ưu hóa luôn vượt trội hơn các phiên bản mặc định của chúng, đạt được độ chính xác, độ chính xác, khả năng thu hồi và F1-score cao hơn.

Bộ phân loại Naïve Bayes, được biết đến với tính đơn giản và hiệu quả với các tập dữ liệu nhỏ, giả định tính độc lập của các đặc điểm, đây có thể là một hạn chế trong các phân phối dữ liệu phức tạp. Decision Tree, một mô hình dựa trên cây, xử lý hiệu quả các tương tác đặc điểm và giá trị bị thiếu nhưng có xu hướng quá khớp. Random Forest, một phương pháp tiếp cận tổng hợp, giảm thiểu tình trạng quá khớp này bằng cách lấy trung bình nhiều cây quyết định. SVM hoạt động tốt trong không gian có nhiều chiều và

có thể xử lý cả dữ liệu tuyến tính và phi tuyến tính, trong khi KNN, một thuật toán dựa trên khoảng cách, dựa vào việc bỏ phiếu đa số giữa các láng giềng.

Voting Classifier tổng hợp các dự đoán từ nhiều mô hình để cải thiện độ mạnh của phân loại. Tuy nhiên, KNN đã bị loại khỏi cấu hình Voting Classifier cuối cùng do khả năng thu hồi kém và F1-score thấp, ảnh hưởng tiêu cực đến hiệu suất chung của hệ thống. Việc loại bỏ KNN đã mang lại độ chính xác và khả năng thu hồi cao hơn cho bộ phân loại cuối cùng.

Bằng cách kết hợp GridSearchCV, hệ thống có thể tinh chỉnh các siêu tham số mô hình, dẫn đến cải thiện hiệu suất đáng kể. Các mô hình Random Forest và SVM được tối ưu hóa đã chứng minh khả năng khai quát hóa tốt hơn, trong khi Voting Classifier được tối ưu hóa đạt được độ chính xác và khả năng thu hồi cao nhất. Những cải tiến này làm nổi bật tầm quan trọng của việc điều chỉnh siêu tham số trong việc đạt được hệ thống phát hiện thư rác email hiệu quả và đáng tin cậy.

Hiệu quả của hệ thống phát hiện thư rác được đề xuất được đánh giá bằng các tham số hiệu suất như độ precision, recall, accuracy, và F1-score.

Độ chính xác (Accuracy):

Độ chính xác là tỷ lệ các dự đoán chính xác so với tổng số các dự đoán.

Công thức tính độ chính xác:

$$\text{Accuracy} = \frac{\text{Số lượng dự đoán đúng}}{\text{Tổng số dự đoán}}$$

Độ chính xác cho phép đánh giá tổng quát về hiệu suất của mô hình, tuy nhiên, nếu tập dữ liệu mất cân bằng (chẳng hạn, có nhiều "ham" hơn "spam"), độ chính xác có thể không phản ánh chính xác hiệu quả mô hình trong việc phân biệt thư rác.

Ma trận nhầm lẫn (Confusion Matrix):

Ma trận nhầm lẫn là công cụ đánh giá giúp ta thấy rõ số lượng các dự đoán đúng và sai, phân chia thành 4 loại:

True Positive (TP): Mô hình dự đoán thư rác đúng là thư rác.

False Positive (FP): Mô hình dự đoán thư rác nhưng thực tế không phải là thư rác.

True Negative (TN): Mô hình dự đoán không phải thư rác đúng là không phải thư rác.

False Negative (FN): Mô hình dự đoán không phải thư rác nhưng thực tế là thư rác.

Ma trận nhầm lẫn cung cấp cái nhìn chi tiết hơn về sự phân bố các dự đoán của mô hình, giúp hiểu rõ hơn về những lỗi mà mô hình mắc phải.

Precision (Độ chính xác):

Precision đo lường tỷ lệ thư rác mà mô hình dự đoán là đúng so với tổng số thư rác mà mô hình dự đoán.

Công thức tính precision:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Precision cao cho thấy mô hình không đưa ra nhiều cảnh báo sai (false positives), nghĩa là ít thư không phải rác bị dự đoán là thư rác.

Recall (Độ nhạy):

Recall đo lường tỷ lệ thư rác mà mô hình dự đoán đúng so với tổng số thư rác trong dữ liệu thực tế.

Công thức tính recall:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Recall cao cho thấy mô hình không bỏ sót nhiều thư rác, nghĩa là ít thư rác bị bỏ qua.

F1-Score:

F1-Score là trung bình điều hòa của Precision và Recall. F1-Score có giá trị từ 0 đến 1, trong đó giá trị 1 là tốt nhất.

Công thức tính F1-Score:

$$\text{F1 - Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1-Score cao cho thấy mô hình cân bằng tốt giữa việc dự đoán chính xác thư rác và việc không bỏ sót thư rác.

ROC Curve và AUC (Area Under Curve):

Đường cong ROC (Receiver Operating Characteristic) thể hiện mối quan hệ giữa tỷ lệ dương tính giả (False Positive Rate - FPR) và tỷ lệ dương tính thật (True Positive Rate - TPR) tại các điểm ngưỡng khác nhau của mô hình.

AUC (Area Under Curve) đo lường diện tích dưới đường cong ROC, với giá trị càng gần 1 thì mô hình càng tốt.

ROC và AUC giúp đánh giá khả năng phân biệt giữa các lớp của mô hình mà không phụ thuộc vào ngưỡng phân loại

Bảng 2.1 Công thức tính tiêu chí đánh giá

Đo lường đánh giá	Mô tả	Công thức
True Positive (TP)	Số lượng email spam được dự đoán đúng là spam.	N/A
False Positive (FP)	Số lượng email ham (không phải spam) được dự đoán sai là spam.	N/A
True Negative (TN)	Số lượng email ham được dự đoán đúng là ham.	N/A
False Negative (FN)	Số lượng email spam được dự đoán sai là ham.	N/A
Precision	Xác định hiệu quả của bộ phân loại.	$TP/(TP+FP)$
Recall(True Positive Rate)	Trong tổng số dữ liệu lớp, dữ liệu được gán nhãn dương do bộ phân loại trả về.	$TP/(TP+FN)$
Accuracy	Tỷ lệ giá trị dự đoán dương so với tổng dữ liệu.	$(TP+TN)/(TP+FN+FP+TN)$
F1-Score	Hiệu suất tổng thể bằng cách hiển thị kết quả dương hiệu quả.	$(2 \times Precision \times Recall) / (Precision + Recall)$

2.4. Đào tạo và đánh giá kết quả

2.4.1. Thư viện

Scikit-learn: một thư viện mã nguồn mở và hỗ trợ hầu hết các thuật toán dành cho học tự động. Chúng ta sử dụng scikit-learn khi cần giải quyết bài toán về học tự động.

Nltk: phục vụ xử lý ngôn ngữ tự nhiên, ngoài ra nó còn có tác dụng làm sạch dữ liệu, xử lý dữ liệu đầu vào cho các thuật toán học tự động.

2.4.2. Mô hình triển khai

Naive Bayes

K-Nearest Neighbors

Decision Tree

Support Vector Machine (SVM)

Random Forest

2.4.3. Đào tạo mô hình

Trong phần này, chúng ta chia bộ dữ liệu thành hai tập: tập huấn luyện (train set) và tập kiểm tra (test set). Tập huấn luyện sẽ được sử dụng để huấn luyện mô hình, trong khi tập kiểm tra sẽ giúp đánh giá hiệu suất của mô hình đã được huấn luyện.

Chia bộ dữ liệu: Chúng ta chia bộ dữ liệu thành hai phần: 75% dữ liệu được sử dụng để huấn luyện mô hình và 25% còn lại được sử dụng để kiểm tra hiệu suất của mô hình. Việc chia này giúp chúng ta kiểm tra khả năng tổng quát của mô hình, tức là khả năng áp dụng mô hình vào các dữ liệu chưa từng thấy.

Tiền xử lý dữ liệu và trích xuất đặc trưng: Sau khi chia bộ dữ liệu, chúng ta tiến hành tiền xử lý và trích xuất đặc trưng từ các tin nhắn trong bộ dữ liệu. Để chuyển đổi các tin nhắn thành dạng mà mô hình có thể xử lý, chúng ta sử dụng phương pháp đếm từ (CountVectorizer). Phương pháp này giúp chúng ta chuyển các tin nhắn thành các đặc trưng số, từ đó mô hình có thể học được các đặc điểm quan trọng trong tin nhắn.

Huấn luyện và lưu mô hình: Sau khi huấn luyện và đánh giá, mô hình được lưu lại để có thể sử dụng trong các dự đoán sau này mà không cần phải huấn luyện lại từ đầu. Việc này giúp tiết kiệm thời gian và tài nguyên khi triển khai mô hình vào thực tế.

2.4.4. Đánh giá và so sánh kết quả

Naive Bayes

	precision	recall	f1-score	support
0	0.95	1.00	0.97	1201
1	1.00	0.68	0.81	191
accuracy			0.96	1392
macro avg	0.98	0.84	0.89	1392
weighted avg	0.96	0.96	0.95	1392

Hình 2.4. Kết quả của Naive Bayes

K-Nearest Neighbors

	precision	recall	f1-score	support
0	0.92	1.00	0.96	1201
1	1.00	0.45	0.62	191
accuracy			0.92	1392
macro avg	0.96	0.73	0.79	1392
weighted avg	0.93	0.92	0.91	1392

Hình 2.5. Kết quả của KNN

Decision Tree

	precision	recall	f1-score	support
0	0.98	0.97	0.98	1201
1	0.84	0.85	0.85	191
accuracy			0.96	1392
macro avg	0.91	0.91	0.91	1392
weighted avg	0.96	0.96	0.96	1392

Hình 2.6. Kết quả của Decision Tree

Support Vector Machine (SVM)

	precision	recall	f1-score	support
0	0.98	1.00	0.99	1201
1	0.99	0.87	0.93	191
accuracy			0.98	1392
macro avg	0.99	0.94	0.96	1392
weighted avg	0.98	0.98	0.98	1392

Hình 2.7. Kết quả của SVM

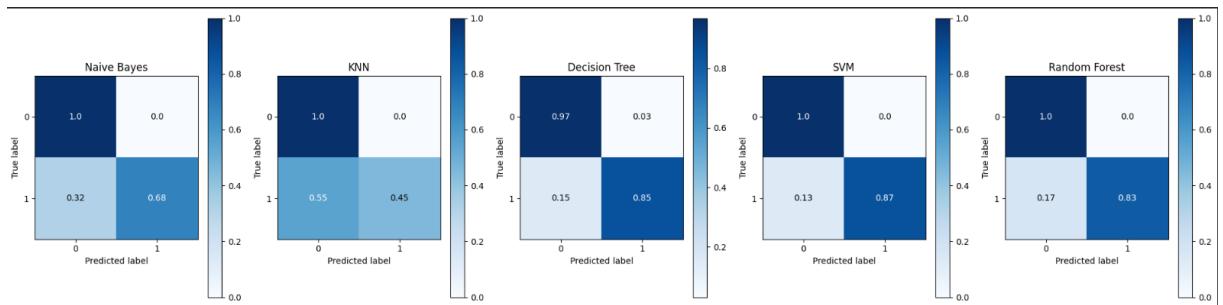
- Random Forest

	precision	recall	f1-score	support
0	0.97	1.00	0.99	1201
1	0.99	0.83	0.90	191
accuracy			0.98	1392
macro avg	0.98	0.92	0.94	1392
weighted avg	0.98	0.98	0.97	1392

Hình 2.8. Kết quả của Random Forest

Classifier	Accuracy	Precision	Recall	F1-score
Naive Bayes	0.95546	1	0.675393	0.80625
K-Nearest Neighbors	0.924569	1	0.450262	0.620939
Decision Tree	0.957615	0.840206	0.853403	0.846753
Support Vector Machine	0.98204	0.994048	0.874346	0.930362
Random Forest	0.975575	0.987578	0.832461	0.903409

Hình 2.9. Kết quả so sánh kết quả của 5 thuật toán



Hình 2.10. Kết quả so sánh confusion matrix của 5 thuật toán

Nhận xét và so sánh Accuracy của 5 mô hình

Naive Bayes (95.55%)

Mô hình Naive Bayes đạt độ chính xác 95.55%, một kết quả khá ấn tượng trong bài toán phân loại thư rác. Mặc dù không phải là mô hình có độ chính xác tuyệt đối cao nhất trong các mô hình so sánh, Naive Bayes vẫn thể hiện được sự hiệu quả của nó, đặc biệt trong các bài toán xử lý văn bản. Mô hình này dựa trên giả định độc lập giữa các đặc trưng, tức là mỗi từ trong email được xem như một đặc trưng độc lập, không phụ thuộc vào các từ khác. Điều này giúp mô hình học nhanh và phân loại hiệu quả, đặc biệt khi dữ liệu chứa các từ khóa đặc trưng, như từ ngữ quảng cáo hoặc các nội dung lừa đảo thường gặp trong thư rác.

Tuy nhiên, một trong những nhược điểm chính của Naive Bayes là khả năng không bắt được các mối quan hệ phức tạp giữa các từ trong email. Trong các bài toán phân loại có tính chất phức tạp hơn, hoặc khi sự tương tác giữa các từ có ảnh hưởng lớn đến kết quả phân loại, Naive Bayes có thể gặp khó khăn. Dù vậy, trong bài toán phân loại thư rác, nơi mà các từ khóa đóng vai trò quan trọng, Naive Bayes vẫn là một lựa chọn rất mạnh mẽ và có thể xử lý tốt

K-Nearest Neighbors (KNN) (92.46%)

Mô hình K-Nearest Neighbors (KNN) đạt độ chính xác 92.46%, thấp hơn so với các mô hình khác. KNN hoạt động theo nguyên lý đơn giản là phân loại một điểm dữ liệu dựa trên lớp của các điểm gần nhất. Tuy nhiên, trong các bài toán với dữ liệu văn bản có nhiều chiều, KNN gặp khó khăn vì không thể xử lý hiệu quả các đặc trưng đa chiều. KNN rất nhạy cảm với nhiễu trong dữ liệu, và không có khả năng học từ dữ liệu như các mô hình khác, điều này làm giảm hiệu quả phân loại trong trường hợp dữ liệu có nhiều đặc trưng không quan trọng.

Một trong những lý do khiến KNN hoạt động kém hơn là mô hình này rất nhạy cảm với nhiễu trong dữ liệu. Khi có quá nhiều đặc trưng không quan trọng hoặc dữ liệu có nhiều chiều (ví dụ như trong bài toán xử lý văn bản), mô hình có thể gặp khó khăn trong việc phân biệt thư rác và thư hợp lệ, dẫn đến kết quả phân loại sai. Hơn nữa, KNN không có khả năng "học" từ dữ liệu như các mô hình học máy khác mà chỉ dựa vào thông tin của các điểm dữ liệu xung quanh, làm cho mô hình trở nên kém hiệu quả khi gặp phải dữ liệu có nhiều sự biến động hoặc tính chất không tuyến tính.

Decision Tree (95.76%)

Decision Tree đạt độ chính xác 95.76%, là một trong những mô hình mạnh mẽ và dễ hiểu trong các bài toán phân loại. So với Naive Bayes, Decision Tree có khả năng nắm bắt các mối quan hệ phức tạp giữa các đặc trưng. Mô hình này xây dựng một cây quyết định, nơi mỗi nút trong cây đại diện cho một đặc trưng hoặc câu hỏi về đặc trưng, và mỗi nhánh là một phân nhánh quyết định tiếp theo dựa trên giá trị của đặc trưng đó. Điều này cho phép Decision Tree dễ dàng phân loại các dữ liệu có cấu trúc rõ ràng và giải thích được các quyết định phân loại.

Tuy nhiên, Decision Tree có thể gặp phải vấn đề overfitting, đặc biệt khi dữ liệu huấn luyện có nhiều đặc trưng không quan trọng hoặc có sự phân bố không đồng đều giữa các lớp. Overfitting xảy ra khi mô hình học quá chi tiết các đặc trưng từ dữ liệu huấn luyện và không thể tổng quát hóa khi gặp dữ liệu mới. Để giảm thiểu vấn đề này, có thể áp dụng các kỹ thuật như cắt tỉa cây (pruning) để loại bỏ các nhánh không quan trọng. Với những cải tiến này, Decision Tree có thể đạt được độ chính xác tốt và trở thành một lựa chọn hợp lý cho bài toán phân loại thư rác.

Support Vector Machine (SVM) (98.2%)

SVM đạt độ chính xác cao nhất trong các mô hình với 98.20%. Đây là một trong những mô hình mạnh mẽ và hiệu quả nhất trong các bài toán phân loại có cấu trúc phân tách rõ ràng. SVM tìm kiếm một siêu phẳng tối ưu trong không gian đặc trưng, phân tách các lớp dữ liệu sao cho khoảng cách giữa các điểm dữ liệu của các lớp là lớn nhất. Chính vì vậy, SVM rất mạnh mẽ trong các bài toán có sự phân tách rõ ràng giữa các lớp.

Mặc dù SVM có độ chính xác rất cao, mô hình này lại có thể gặp phải vấn đề khi dữ liệu có nhiều chiều hoặc khi dữ liệu chứa nhiều nhiễu. Việc huấn luyện mô hình SVM, đặc biệt khi sử dụng các kernel phức tạp hoặc khi dữ liệu có quá nhiều đặc trưng, có thể tiêu tốn rất nhiều thời gian và tài nguyên tính toán.Thêm vào đó, SVM không phải là một lựa chọn tối ưu cho các bài toán có nhiều nhiễu hoặc không có sự phân tách rõ ràng giữa các lớp. Tuy nhiên, khi được tối ưu hóa tốt và áp dụng đúng cho bài toán có cấu trúc phân tách rõ ràng, SVM mang lại độ chính xác cao và khả năng phân loại rất hiệu quả.

Random Forest (97.56%)

Random Forest Random Forest đạt độ chính xác 97.56%, chỉ thấp hơn SVM một chút. Đây là một mô hình ensemble, trong đó nhiều cây quyết định được kết hợp với nhau để tạo ra dự đoán cuối cùng. Mỗi cây trong Random Forest được huấn luyện trên một tập con ngẫu nhiên của dữ liệu và các đặc trưng, giúp mô hình tổng quát hóa tốt hơn so với một cây quyết định đơn lẻ. Một trong những lợi thế lớn nhất của Random Forest là khả năng giảm thiểu overfitting, nhờ vào việc kết hợp nhiều cây quyết định và giảm ảnh hưởng của các cây yếu kém.

Tuy nhiên, Random Forest yêu cầu nhiều tài nguyên tính toán hơn so với Decision Tree và không dễ dàng giải thích như Decision Tree vì tính chất của mô hình ensemble (tập hợp nhiều mô hình). Dù vậy, với độ chính xác cao và khả năng xử lý tốt các vấn đề như overfitting, Random Forest vẫn là một lựa chọn tuyệt vời cho các bài toán phân loại phức tạp.

Để kết quả của Voting Classifier tốt nhất tôi đã tối ưu hóa từng mô hình

```
# # Định nghĩa lưới tham số cho Naïve Bayes
# param_grid_nb = {
#   'alpha': [0.1, 0.5, 1.0, 5.0, 10.0] # Tham số Laplace smoothing
# }

# # GridSearchCV để tìm tham số tốt nhất
# nb_grid = GridSearchCV(MultinomialNB(), param_grid_nb, cv=5, n_jobs=-1)
# nb_grid.fit(x_train_combined, train_set['Spam'])

# # Lấy mô hình tốt nhất
# best_nb = nb_grid.best_estimator_
# print("Best Parameters:", nb_grid.best_params_)

# # Huấn luyện mô hình tốt nhất
# best_nb.fit(x_train_combined, train_set['Spam'])

# Khởi tạo và huấn luyện Naïve Bayes với alpha=0.1
best_nb = MultinomialNB(alpha=0.1)
best_nb.fit(x_train_combined, train_set['spam'])

# Dự đoán nhãn cho tập kiểm tra
predicted_values_NB = best_nb.predict(x_test_combined)

# Lưu các chỉ số đánh giá vào bảng metrics tối ưu
add_metrics_to_prediction_optimized("Naïve Bayes", test_set.Spam, predicted_values_NB)
```

Hình 2.11. Tối ưu hóa mô hình Naive Bayes

```

# # Định nghĩa lưới tham số cho Decision Tree
# param_grid_dt = {
#     'criterion': ['gini', 'entropy'], # Chọn hàm đo độ bất bình đẳng (impurity)
#     'max_depth': [None, 10, 20, 30], # Độ sâu tối đa của cây
#     'min_samples_split': [2, 5, 10], # Số mẫu tối thiểu để chia nhánh
#     'min_samples_leaf': [1, 2, 5]    # Số mẫu tối thiểu ở mỗi lá
# }

# # GridSearchCV để tìm tham số tối ưu
# dt_grid = GridSearchCV(DecisionTreeClassifier(random_state=42), param_grid_dt, cv=5, n_jobs=-1)
# dt_grid.fit(x_train_combined, train_set['spam'])

# # Lấy mô hình tốt nhất
# best_dt = dt_grid.best_estimator_
# print("Best Parameters:", dt_grid.best_params_)

# # Huấn luyện mô hình tốt nhất
# best_dt.fit(x_train_combined, train_set['spam'])

# Khởi tạo và huấn luyện Decision Tree với các tham số tối ưu
best_dt = DecisionTreeClassifier(
    random_state=42,
    criterion='gini',
    max_depth=30,
    min_samples_leaf=1,
    min_samples_split=2
)
best_dt.fit(x_train_combined, train_set['spam'])

# Dự đoán nhãn cho tập kiểm tra
predicted_values_DT = best_dt.predict(x_test_combined)

# Lưu các chỉ số đánh giá vào bảng metrics tối ưu
add_metrics_to_prediction_optimized("Decision Tree", test_set.Spam, predicted_values_DT)

```

Hình 2.12. Tối ưu hóa mô hình Decision Tree

```

# # Định nghĩa lưới tham số cho Random Forest
# param_grid = {
#     'n_estimators': [100, 300],          # Số cây trong rừng
#     'max_depth': [None, 10],            # Độ sâu tối đa của cây
#     'min_samples_split': [2, 5],        # Số mẫu tối thiểu để chia nhánh
#     'min_samples_leaf': [1, 2],         # Số mẫu tối thiểu ở mỗi lá
#     'class_weight': ['balanced']       # Cân bằng số lượng mẫu của các lớp
# }

# # GridSearchCV để tìm tham số tối ưu
# rf_grid = GridsearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5, n_jobs=-1)
# rf_grid.fit(x_train_combined, train_set['Spam'])

# # Lấy mô hình với các tham số tối ưu
# best_rf = rf_grid.best_estimator_
# print("Best Parameters:", rf_grid.best_params_)

# Khởi tạo và huấn luyện Random Forest với các tham số tối ưu
best_rf = RandomForestClassifier(
    random_state=42,
    class_weight='balanced',
    max_depth=None,
    min_samples_leaf=1,
    min_samples_split=5,
    n_estimators=100
)
best_rf.fit(x_train_combined, train_set['Spam'])

# Dự đoán nhãn cho tập kiểm tra
predicted_values_RF_1 = best_rf.predict(x_test_combined)

# Lưu các chỉ số đánh giá vào bảng metrics tối ưu
add_metrics_to_prediction_optimized("Random Forest", test_set.Spam, predicted_values_RF_1)

```

Hình 2. 13. Tối ưu hóa mô hình Random Forest

```

# # Định nghĩa lưới tham số cho Random Forest
# param_grid = {
#     'n_estimators': [100, 300],          # Số cây trong rừng
#     'max_depth': [None, 10],            # Độ sâu tối đa của cây
#     'min_samples_split': [2, 5],        # Số mẫu tối thiểu để chia nhánh
#     'min_samples_leaf': [1, 2],         # Số mẫu tối thiểu ở mỗi lá
#     'class_weight': ['balanced']       # Cân bằng số lượng mẫu của các lớp
# }

# # GridSearchCV để tìm tham số tối ưu
# rf_grid = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5, n_jobs=-1)
# rf_grid.fit(x_train_combined, train_set['spam'])

# # Lấy mô hình với các tham số tối ưu
# best_rf = rf_grid.best_estimator_
# print("Best Parameters:", rf_grid.best_params_)

# Khởi tạo và huấn luyện Random Forest với các tham số tối ưu
best_rf = RandomForestClassifier(
    random_state=42,
    class_weight='balanced',
    max_depth=None,
    min_samples_leaf=1,
    min_samples_split=5,
    n_estimators=100
)
best_rf.fit(x_train_combined, train_set['spam'])

# Dự đoán nhãn cho tập kiểm tra
predicted_values_RF_1 = best_rf.predict(x_test_combined)

# Lưu các chỉ số đánh giá vào bảng metrics tối ưu
add_metrics_to_prediction_optimized("Random Forest", test_set.spam, predicted_values_RF_1)

```

Hình 2. 14. Tối ưu hóa mô hình SVM

Cuối cùng kết hợp 5 mô hình bằng Voting Classifier.

Chương trình phát hiện thư rác email đạt độ chính xác tổng thể là 98.56%, với Voting Classifier được tối ưu hóa vượt trội hơn tất cả các mô hình khác. Trong số các bộ phân loại riêng lẻ, Naive Bayes đã chứng minh hiệu suất mạnh mẽ với accuracy 97.99%, precision 96.05%, và recall 89.01%, khiến nó có hiệu quả cao nhưng hơi dễ bỏ sót một số trường hợp thư rác. Bộ phân loại Decision Tree, mặc dù đạt accuracy 95.91%, nhưng có precision thấp hơn (85.64%) và recall (84.29%), cho thấy tỷ lệ phân loại sai cao hơn. Random Forest hoạt động tốt, đạt accuracy 97.84%, với precision 97.63% và recall 86.38%, cho thấy khả năng phát hiện thư rác cân bằng. Support Vector Machine (SVM) cũng mang lại kết quả tuyệt vời, đạt accuracy 98.35%, precision 98.28%, và recall 89.53%, khiến nó trở thành một mô hình phân loại đáng tin cậy.

Tuy nhiên, Voting Classifier, kết hợp nhiều mô hình, đã chứng minh hiệu suất tổng thể tốt nhất với accuracy 98.56%, precision 99.42% và recall 90.05%, đảm bảo sự cân bằng mạnh mẽ giữa việc xác định đúng thư rác và giảm thiểu các kết quả dương tính giả. Những kết quả này làm nổi bật hiệu quả của việc học tập tổng hợp trong việc cải

thiện độ chính xác của phân loại, cho thấy rằng việc tối ưu hóa thêm thông qua kỹ thuật tính năng và điều chỉnh siêu tham số có thể tăng cường khả năng thu hồi trong khi vẫn duy trì độ chính xác cao.

Bảng 2.2. Bảng kết quả của các thuật toán sau khi được tối ưu hóa

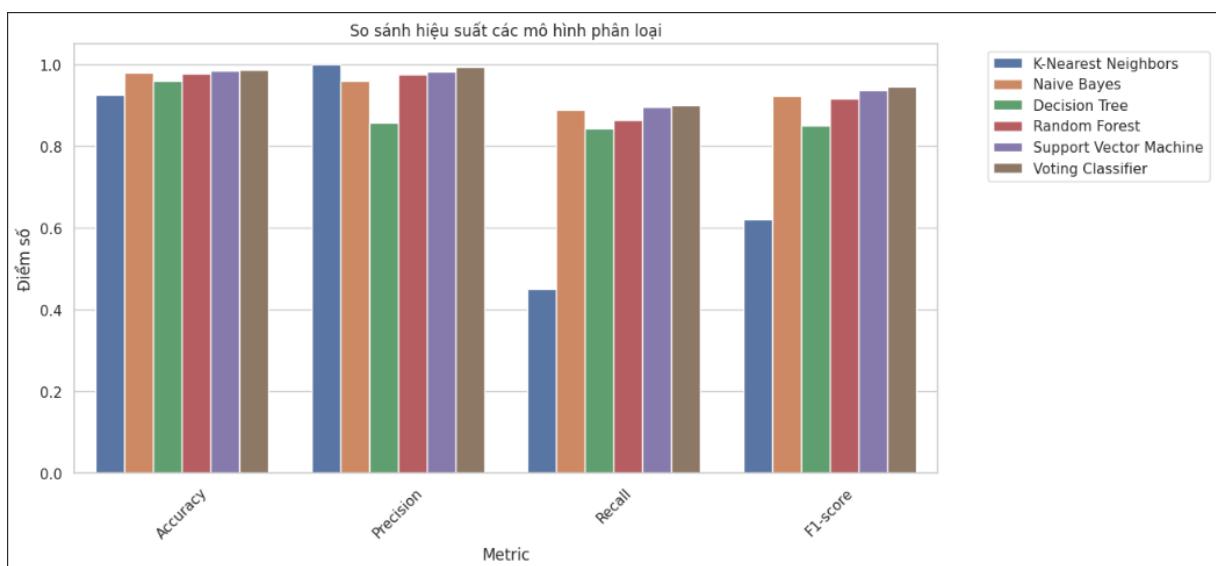
Classifier	Accuracy	Precision	Recall	F1-score
Naive Bayes	0.979885	0.960452	0.890052	0.923913
K-Nearest Neighbors	0.924569	1	0.450262	0.620939
Decision Tree	0.959052	0.856383	0.842932	0.849604
Support Vector Machine (SVM)	0.983477	0.982759	0.895288	0.936986
Random Forest	0.978448	0.976331	0.863874	0.916667
Voting Classifier	0.985632	0.99422	0.900524	0.945055

Trong quá trình phát triển Voting Classifier, cả năm thuật toán Naive Bayes, K-Nearest Neighbors (KNN), Decision Tree, Support Vector Machine (SVM) và Random Forest ban đầu đều được đưa vào. Tuy nhiên, sau khi đánh giá hiệu suất của từng mô hình, KNN đã bị loại vì nó ảnh hưởng tiêu cực đến hiệu quả chung của mô hình tổng hợp.

Một trong những lý do chính để loại bỏ KNN là khả năng thu hồi kém (0,4503), cho thấy mô hình đã không xác định đúng một phần đáng kể các email rác thực tế. Vì các hệ thống phát hiện thư rác yêu cầu khả năng thu hồi cao để giảm thiểu các kết quả âm tính giả (tức là email rác được phân loại nhầm là ham), khả năng thu hồi thấp của KNN khiến nó không phù hợp để đưa vào. Ngoài ra, điểm F1 thấp (0.621) cân bằng giữa độ chính xác và khả năng thu hồi càng chứng minh rằng KNN không đáng tin cậy trong việc phân biệt thư rác với ham.

Việc đưa các mô hình yếu vào một phương pháp tổng hợp như Voting Classifier có thể ảnh hưởng tiêu cực đến độ chính xác dự đoán chung, vì bộ phân loại tổng hợp các dự đoán từ tất cả các mô hình. Vì KNN hoạt động kém hơn so với các bộ phân loại khác nên nó đã tạo ra nhiều và làm giảm hiệu quả của nhóm. Bằng cách loại bỏ KNN, độ chính xác của Bộ phân loại bỏ phiếu đã cải thiện lên 97,7% và sau khi tối ưu hóa bằng GridSearchCV, độ chính xác này tiếp tục tăng lên 98,56%.

Ngoài ra, khả năng recall của mô hình được tối ưu hóa đạt 0,8115, nâng cao khả năng phát hiện email rác chính xác của mô hình này. Quyết định loại trừ KNN này đã cho phép các mô hình mạnh hơn SVLiterature ReviewM, Random Forest, Naive Bayes và Decision Tree chiếm ưu thế, dẫn đến hệ thống phát hiện thư rác hiệu quả và đáng tin cậy hơn. Các kết quả nêu bật tầm quan trọng của việc lựa chọn cẩn thận các mô hình trong quá trình học nhóm, vì việc kết hợp các bộ phân loại yếu hơn có thể làm giảm hiệu suất tổng thể thay vì cải thiện nó.



Hình 2.15. Kết quả so sánh của các thuật toán sau khi được tối ưu

Kết quả chứng minh rằng phương pháp tiếp cận tổng hợp sử dụng Voting Classifier cải thiện đáng kể độ chính xác phân loại và giảm lỗi phân loại sai. Mô hình cuối cùng cung cấp hệ thống phát hiện thư rác email đáng tin cậy và ổn định hơn. Các bộ phân loại được tối ưu hóa nâng cao hơn nữa hiệu suất của mô hình, đảm bảo độ chính xác phát hiện thư rác ở mức cao

CHƯƠNG 3: TRIỂN KHAI ỨNG DỤNG

3.1. Môi trường và công cụ triển khai.

3.1.1. Ngôn ngữ lập trình và Framework

Ứng dụng nhận diện email rác được xây dựng dựa trên các công nghệ hiện đại, đảm bảo khả năng mở rộng, dễ triển khai và dễ bảo trì. Các thành phần công nghệ chính bao gồm:

3.1.1.1. Ngôn ngữ lập trình Python 3.8+

Python là lựa chọn ưu tiên hàng đầu để phát triển các ứng dụng phân loại văn bản (ví dụ: phát hiện spam email) nhờ vào sự kết hợp giữa hệ sinh thái mạnh mẽ, khả năng tích hợp linh hoạt, và cộng đồng hỗ trợ rộng lớn.

Hệ Sinh Thái Phong Phú cho Xử Lý Ngôn ngữ Tự nhiên (NLP) và Học Máy (ML).

Python sở hữu bộ thư viện khoa học dữ liệu và học máy (Data Science & ML) vô song, là nền tảng cốt lõi cho các dự án NLP. Các thư viện chính bao gồm:

- **Xử lý ngôn ngữ tự nhiên (NLP):** Các thư viện chuyên biệt như **NLTK** (Natural Language Toolkit) và **spaCy** cung cấp các công cụ mạnh mẽ và tối ưu hóa để tiến xử lý văn bản, tokenization, phân tích cú pháp và nhận dạng thực thể (NER).
 - **Học Máy (ML):** **Scikit-learn** là tiêu chuẩn công nghiệp cho các mô hình ML truyền thống (như Naive Bayes, SVM), thường được sử dụng trong các tác vụ phân loại văn bản cơ bản. Đối với các mô hình mạng nơ-ron phức tạp hơn (Deep Learning), các framework hàng đầu như **TensorFlow** và **PyTorch** đều được xây dựng với API Python, cho phép lập trình viên dễ dàng xây dựng và huấn luyện các mô hình phân loại văn bản nâng cao.
- *Khả năng tích hợp linh hoạt (Full-Stack Integration)*
Khả năng đóng vai trò là "mã keo" (glue code) là một lợi thế lớn của Python. Nó cho phép tích hợp linh hoạt logic nghiệp vụ (business logic) và các mô hình ML vào nhiều loại giao diện người dùng khác nhau:

- **Backend:** Sử dụng các framework web như **Django** và **Flask** để triển khai mô hình phân loại văn bản dưới dạng API dịch vụ (RESTful API), cho phép các ứng dụng di động hoặc web gọi đến.
 - **Desktop/GUI:** Mặc dù ít phổ biến hơn, Python vẫn hỗ trợ các thư viện GUI (như Tkinter, PyQt) để xây dựng giao diện máy tính để bàn.
 - **Tích hợp:** Một mô hình spam email được huấn luyện bằng Scikit-learn có thể được đóng gói bằng thư viện **Pickle** và được phục vụ bởi một server Flask, sau đó được tiêu thụ bởi một ứng dụng giao diện web (frontend) viết bằng JavaScript.
- *Khả năng hỗ trợ và tài liệu phong phú*

Sự phổ biến của Python đã tạo ra một cộng đồng toàn cầu khổng lồ.

 - **Hỗ trợ và Học tập:** Cộng đồng lớn đảm bảo rằng mọi vấn đề kỹ thuật đều có giải pháp sẵn có thông qua các diễn đàn và trang web như Stack Overflow.
 - **Tài liệu (Documentation):** Các thư viện lớn đều có tài liệu chi tiết, giúp quá trình học tập và xây dựng ứng dụng nhanh chóng, đặc biệt phù hợp cho việc **tạo mẫu nhanh (rapid prototyping)** các ứng dụng như phân loại văn bản. Sự ổn định và khả năng đọc mã cao của Python cũng giúp các nhóm phát triển phối hợp hiệu quả.

Tóm lại, Python 3.8 và các phiên bản tiếp theo đã mang lại sự cân bằng hoàn hảo giữa tính dễ đọc, khả năng viết mã ngắn gọn và hiệu suất. Các tính năng như toán tử Walrus không chỉ là sự thay đổi cú pháp mà còn là công cụ tư duy mới, cho phép lập trình viên tối ưu hóa cấu trúc mã. **Hơn thế nữa, đối với lĩnh vực Phân loại Văn bản và Trí tuệ Nhân tạo, Python là ngôn ngữ không thể thay thế** do sở hữu hệ sinh thái thư viện mạnh mẽ và khả năng tích hợp vượt trội, giúp nó trở thành công cụ lý tưởng để xây dựng các ứng dụng chuyên sâu từ backend đến frontend. Việc nắm vững và áp dụng các tính năng từ Python 3.8+ trở lên là điều cần thiết để viết mã Python hiện đại, hiệu quả, và dễ bảo trì trong môi trường phát triển ngày nay.

3.1.1.2. Tổng quan về Framework giao diện web Flask

a, Mục đích và tính năng sử dụng



Hình 3.1. Flask

Flask là một micro-framework mã nguồn mở của Python, được sử dụng rộng rãi để xây dựng các ứng dụng web với cấu trúc linh hoạt và hiệu năng cao. Trong bài toán phân loại thư rác, Flask đóng vai trò là nền tảng để xây dựng giao diện web giúp người dùng dễ dàng tương tác với hệ thống, gửi dữ liệu email cần phân loại và xem kết quả trả về từ các mô hình học máy (Machine Learning). Thay vì tạo ứng dụng dạng kéo-thả như Streamlit, Flask cung cấp khả năng tùy biến sâu hơn thông qua HTML, CSS, JavaScript, giúp hệ thống dễ dàng mở rộng, triển khai và tích hợp vào các nền tảng doanh nghiệp.

Ứng dụng phân loại thư rác dựa trên Flask được thiết kế nhằm cung cấp cho người dùng một công cụ phân tích email rõ ràng, trực quan và có khả năng xử lý cả dữ liệu đơn lẻ lẫn dữ liệu theo tệp. Hệ thống sử dụng nhiều mô hình học máy đã được huấn luyện như Naive Bayes, K-Nearest Neighbors, SVM, Decision Tree và Random Forest để đưa ra dự đoán “Spam” hoặc “Ham”.

Các tính năng chính của ứng dụng gồm:

1. Nhập nội dung email trực tiếp:

Người dùng có thể nhập nội dung email vào form HTML. Sau khi gửi yêu cầu, Flask xử lý dữ liệu và trả kết quả phân loại ngay trên trang web.

2. Tải lên tệp CSV:

Ứng dụng hỗ trợ tải lên file CSV chứa nhiều nội dung email. Flask đọc tệp, thực hiện phân loại hàng loạt, sau đó trả về bảng kết quả cùng tùy chọn tải xuống file CSV đã xử lý.

3. Auto Check:

Ứng dụng hỗ trợ người dùng sau khi đăng nhập có thể mở tính năng Auto Check thực hiện việc tự động phân loại 10 email mới sau mỗi 5 phút nhằm phân loại các email mới được gửi về(nếu có).

4. Hiển thị kết quả phân loại:

Kết quả dự đoán của từng mô hình học máy sẽ được hiển thị dưới dạng bảng hoặc thẻ thông tin (card) trên giao diện web. Người dùng có thể xem mô hình nào cho kết quả phù hợp nhất với dữ liệu của họ.

5. Tải xuống kết quả:

Sau khi hệ thống xử lý tệp CSV, Flask cho phép người dùng tải về file chứa nhãn phân loại mới, phục vụ cho việc lưu trữ hoặc phân tích tiếp theo.

6. Hỗ trợ nhiều mô hình:

Flask dễ dàng tích hợp các mô hình ML nhờ kiến trúc mở, cho phép người dùng chọn mô hình muốn sử dụng và so sánh hiệu quả giữa Naive Bayes, KNN, SVM, Decision Tree và Random Forest.

b, Lợi ích khi sử dụng Flask

Flask mang đến nhiều ưu điểm quan trọng khi áp dụng vào phát triển ứng dụng phân loại thư rác:

- **Kiến trúc linh hoạt và dễ tùy biến:**

Không giống như các framework giao diện tự động, Flask cho phép lập trình viên toàn quyền xây dựng giao diện bằng HTML/CSS/JS, kết hợp với các template engine như Jinja2. Điều này giúp ứng dụng dễ dàng mở rộng, tích hợp API hoặc triển khai vào hệ thống lớn của doanh nghiệp.

- **Phân tách rõ ràng giữa backend và frontend:**

Flask cho phép tách biệt phần xử lý mô hình học máy và phần giao diện web, giúp việc bảo trì, nâng cấp hoặc thay đổi mô hình ML trở nên đơn giản hơn.

- **Tích hợp tốt với các thư viện khoa học dữ liệu:**

Flask hoạt động hoàn hảo với Pandas, NumPy, Scikit-learn, TensorFlow, Keras,... giúp dễ dàng áp dụng mô hình đã huấn luyện vào ứng dụng.

- **Hỗ trợ xử lý file mạnh mẽ:**

Flask cho phép upload và xử lý file CSV trực tiếp từ giao diện web, giúp người dùng dễ dàng phân loại hàng loạt email.

- **Hiệu năng tốt và triển khai đơn giản:**

Flask nhẹ, dễ triển khai trên Heroku, AWS, Google Cloud, hoặc thông qua Docker. Điều này giúp ứng dụng phân loại thư rác có thể chạy ổn định trong môi trường sản phẩm hoặc nội bộ doanh nghiệp.

- **Bảo trì và mở rộng dễ dàng:**

Nhờ cấu trúc micro-framework, ứng dụng có thể dễ dàng mở rộng thêm các tính năng như xác thực người dùng, lưu lịch sử phân loại, kết nối cơ sở dữ liệu hoặc tích hợp API xử lý email tự động.

- **Tính thực tế cao:**

Flask là framework phổ biến trong công nghiệp, giúp sản phẩm dễ tích hợp vào hệ thống thực tế hơn so với Streamlit. Điều này đặc biệt quan trọng nếu ứng dụng phân loại thư rác được triển khai trong môi trường doanh nghiệp hoặc tích hợp vào hệ thống email thật.

Flask là một framework mạnh mẽ, linh hoạt và rất phù hợp để xây dựng ứng dụng phân loại thư rác dựa trên mô hình học máy. Với khả năng tùy biến cao, hỗ trợ tốt các thư viện ML và xử lý file hiệu quả, Flask mang lại nền tảng vững chắc cho việc triển khai một hệ thống phân loại email thực tế, dễ mở rộng và có tính ứng dụng cao trong môi trường số hóa hiện nay, trở thành một công cụ lý tưởng để triển khai ứng dụng phân loại thư rác trong thực tế.

3.1.1.3. Thư viện Machine Learning

Scikit-learn là thư viện cốt lõi được sử dụng để xây dựng và triển khai các giải pháp Học máy truyền thống cho bài toán phân loại văn bản. Thư viện này được lựa chọn vì tính ổn định, API đồng nhất và hiệu quả trong môi trường sản xuất.

Hệ thống tận dụng Scikit-learn cho các tác vụ then chốt sau:

Lựa chọn và Huấn luyện Mô hình: Scikit-learn cung cấp một tập hợp đa dạng các thuật toán phân loại hiệu suất cao như Naive Bayes (thường dùng cho văn bản), Support Vector Machines (SVM), Random Forest, K-Nearest Neighbors (KNN), và Decision Tree.

Mô hình Tổ hợp (Ensemble Modeling): Tạo ra các mô hình tổ hợp mạnh mẽ hơn thông qua Voting Classifier hoặc Stacking, bằng cách kết hợp kết quả dự đoán của nhiều thuật toán khác nhau để tăng độ chính xác và tính ổn định.

Tiền xử lý và Chuẩn hóa Dữ liệu: Sử dụng các công cụ như StandardScaler để chuẩn hóa các giá trị đầu vào số học (nếu có).

Trích xuất Đặc trưng Văn bản: Sử dụng module feature_extraction.text với TF-IDF Vectorizer (Term Frequency-Inverse Document Frequency) hoặc CountVectorizer để chuyển đổi dữ liệu văn bản thô thành định dạng số học mà các mô hình ML có thể xử lý.

Triển khai và Bảo trì Mô hình: Sử dụng thư viện Joblib để lưu trữ (serialization) và tải lại (deserialization) các mô hình đã huấn luyện một cách hiệu quả, đảm bảo thời gian phản hồi nhanh khi triển khai mô hình vào môi trường sản xuất.

Đối với các mô hình mạng nơ-ron phức tạp hơn (Deep Learning), các framework hàng đầu như TensorFlow và PyTorch đều được xây dựng với API Python, cho phép lập trình viên dễ dàng xây dựng và huấn luyện các mô hình phân loại văn bản nâng cao.

3.1.1.3. Các Thư Viện Kỹ Thuật Cốt Lõi (Core Technical Libraries)

a. Thư viện xử lý dữ liệu

Các thư viện này cung cấp nền tảng tính toán và cấu trúc dữ liệu hiệu quả cao, giúp đảm bảo khả năng xử lý dữ liệu lớn, tốc độ nhanh và tính ổn định khi chạy mô hình.

- Pandas 2.0.0: Được sử dụng để quản lý dữ liệu bảng (tabular data) một cách mạnh mẽ. Đây là công cụ thiết yếu cho việc tải, làm sạch và biến đổi tập dữ liệu email. Các tác vụ cụ thể gồm: tiền xử lý tập huấn luyện, lưu kết quả log và phân tích thống kê hiệu suất mô hình.
- NumPy 1.24.0: Là thư viện nền tảng cho mọi tính toán khoa học trong Python. Nó hỗ trợ tính toán ma trận, tối ưu hóa hiệu năng và cung cấp cấu trúc dữ liệu mảng đa chiều (nd-array) nền tảng, được sử dụng trực tiếp bởi Scikit-learn và Pandas.

b. Thư viện xử lý ngôn ngữ tự nhiên

NLTK 3.8.0 (Natural Language Toolkit): Một thư viện mạnh mẽ phục vụ NLP, được sử dụng để nâng cao chất lượng dữ liệu đầu vào trước khi đưa vào mô hình học máy.

Tokenization (tách từ): Chia văn bản thành các đơn vị nhỏ (từ, câu) để xử lý.

- Loại bỏ Stopwords: Lọc bỏ các từ chức năng không mang nhiều ý nghĩa ngữ pháp (như "là", "và", "của"), giúp giảm số chiều của đặc trưng (feature dimension).

- Tiền xử lý văn bản: Bao gồm các kỹ thuật như *stemming* (cắt bỏ hậu tố để đưa từ về dạng gốc) và *chuẩn hóa* (chuyển chữ hoa thành chữ thường), giúp giảm số lượng đặc trưng và cải thiện hiệu suất mô hình.

c. Thư viện giao diện desktop & hệ thống

Ứng dụng được thiết kế để hoạt động như một tiện ích desktop thực tế, cung cấp khả năng chạy ngầm (background service) thông qua biểu tượng khay hệ thống (system tray).

- Tkinter: Thư viện giao diện người dùng tiêu chuẩn của Python, dùng để hiển thị các thành phần giao diện đơn giản như hộp thoại thông báo, popup cấu hình, hoặc các form cấu hình đơn giản.
- PyStray 0.19.4: Thư viện cốt lõi giúp ứng dụng hoạt động linh hoạt và chạy ngầm.
 - Hiển thị icon: Đặt icon ứng dụng ở khay hệ thống (system tray).
 - Tạo menu tương tác: Cho phép người dùng tương tác nhanh chóng với các chức năng chính (ví dụ: Start/Stop dịch vụ kiểm tra email định kỳ, Cấu hình, Thoát).
 - Chạy ngầm: Cho phép ứng dụng kiểm tra email định kỳ mà không cần mở giao diện web, mang lại trải nghiệm tiện ích desktop hiệu quả.

d. Các thư viện hỗ trợ khác

openpyxl 3.1.0: Dùng để xuất báo cáo, log hoặc danh sách email đã phân loại dưới dạng file Excel (.xlsx), hỗ trợ kiểm tra dữ liệu đầu ra và nhu cầu báo cáo.

email-validator 2.0.0: Hỗ trợ kiểm tra định dạng email hợp lệ trong quá trình đọc và xử lý dữ liệu, giúp giảm thiểu lỗi trong quá trình phân tích và đảm bảo chất lượng dữ liệu.

requests 2.31.0: Thư viện tiêu chuẩn cho HTTP, được sử dụng khi cần giao tiếp với API hoặc dịch vụ ngoài (ví dụ: kiểm tra URL an toàn, gửi thông báo hoặc truy vấn dữ liệu bên thứ ba).

Pillow 10.0.0: Dùng để xử lý hình ảnh, đặc biệt là tạo và hiển thị icon cho system tray (kết hợp với PyStray) hoặc các hình minh họa trong giao diện.

joblib 1.3.0: Đảm bảo tính nhất quán và hiệu suất khi triển khai mô hình. Dùng để lưu trữ (serialize) và tải lại (deserialize) mô hình máy học đã huấn luyện, TF-IDF vectorizer và scaler, giúp rút ngắn thời gian khởi động ứng dụng.

3.1.2. Yêu Cầu Hệ Thống Ứng Dụng

Đây là các yêu cầu tối thiểu và khuyến nghị để cài đặt và chạy ứng dụng phân loại văn bản:

1. Yêu Cầu Phần Cứng (Tối thiểu)

CPU: Intel Core i3 hoặc bộ xử lý tương đương.

RAM: 4GB (Khuyến nghị 8GB cho hiệu suất ổn định).

Dung lượng Ổ cứng: 500MB (cho ứng dụng và các thư viện liên quan).

Mạng: Kết nối Internet ổn định (tối thiểu 1Mbps).

2. Hệ Điều Hành Hỗ Trợ

Ứng dụng tương thích với các hệ điều hành phổ biến sau:

Windows 10/11

macOS 10.14 trở lên

Linux (ví dụ: Ubuntu 20.04+, Debian, CentOS)

3. Các Yêu Tố Phụ Thuộc (Dependencies)

Để ứng dụng hoạt động, cần có:

Python: Phiên bản 3.8 trở lên.

Tài khoản Gmail: Cần có tài khoản Gmail đã bật tính năng IMAP.

Mật khẩu Ứng dụng: Yêu cầu Mật khẩu Ứng dụng (App Password) nếu tài khoản Gmail có bật Xác minh 2 bước (2FA).

3.2. Cấu trúc tổng quan ứng dụng

3.2.1. Sơ đồ kiến trúc hệ thống

Hệ thống được thiết kế theo kiến trúc modular với 3 tầng chính:

Tầng 1 - Presentation Layer (Giao diện):

Flask Web Interface (app.py): Giao diện web chính của ứng dụng

System Tray Application (tray_launcher.py): Ứng dụng chạy nền trên system tray

Auto Checker GUI (auto_checker.py): Giao diện kiểm tra email tự động

Templates HTML (templates/): Các template HTML cho web interface

Static Resources (static/css, static/js): Tài nguyên tĩnh CSS và JavaScript

Tầng 2 - Business Logic Layer (Xử lý nghiệp vụ):

Email Monitor Module (src/services/): Theo dõi và quản lý email

Gmail Login Service (gmail_login_standalone.py): Xác thực và đăng nhập Gmail

Core Processing Logic (src/core/): Logic xử lý nghiệp vụ cốt lõi

Text Processing Utilities (src/utils/): Các tiện ích xử lý văn bản

Launcher Controller (launcher.py): Điều khiển khởi động ứng dụng

Tầng 3 - Data Layer (Dữ liệu và Model):

Pre-trained ML Models (models/classifiers/*.pkl): Các mô hình ML đã được huấn luyện

Preprocessor Models (models preprocessors/*.pkl): Các mô hình tiền xử lý dữ liệu

Configuration Storage (config/config.json): Lưu trữ cấu hình hệ thống

Log Files (logs/app.log): File ghi log hoạt động của ứng dụng

Luồng xử lý dữ liệu:

Input: Người dùng nhập văn bản/email hoặc hệ thống tự động đọc email từ Gmail

Preprocessing: Làm sạch văn bản, tokenization, loại bỏ stopwords, lemmatization

Feature Extraction: TF-IDF vectorization + 4 features bổ sung (num_char, num_word, num_sen, num_words_transform)

Scaling: MinMaxScaler chuẩn hóa features

Prediction: Model ML dự đoán spam/ham

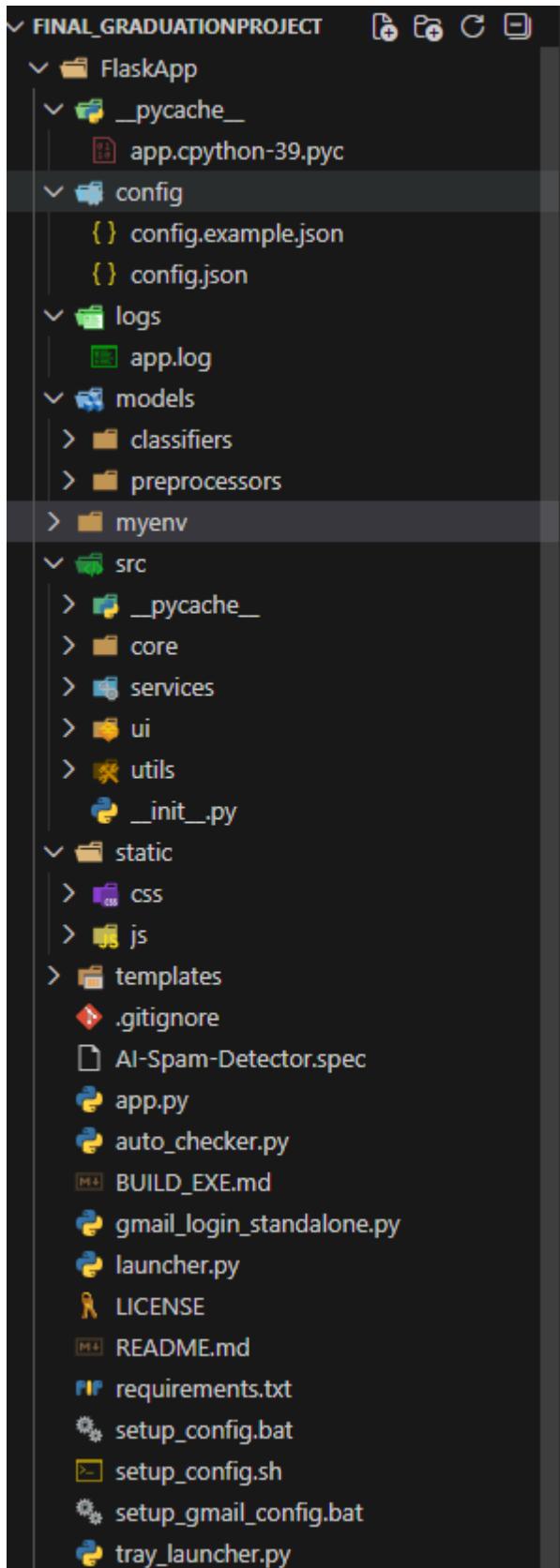
Output: Hiển thị kết quả, gắn nhãn email, gửi thông báo

External Services (Dịch vụ bên ngoài):

Gmail API: Tích hợp OAuth 2.0 cho xác thực Gmail

Email Server: Kết nối IMAP/SMTP để đọc/gửi email

3.2.2. Cấu trúc thư mục dự án



Hình 3.2. Cấu trúc thư mục dự án

Giải thích vai trò của từng file quan trọng:

Thư mục gốc - FINAL_GRADUATIONPROJECT

Files Python chính:

app.py

File khởi động chính của ứng dụng Flask web server

Định nghĩa các routes, endpoints API

Khởi tạo Flask application và cấu hình middleware

Điểm vào chính để chạy web interface

launcher.py

Script khởi động tổng hợp cho ứng dụng

Có thể chứa logic để khởi động nhiều service cùng lúc

Quản lý việc khởi tạo các module phụ thuộc

Thường được sử dụng để start ứng dụng từ command line

tray_launcher.py

Khởi động ứng dụng dưới dạng system tray icon

Cho phép ứng dụng chạy nền và truy cập nhanh từ taskbar

Tạo menu context với các chức năng như start/stop, settings, exit

Phù hợp cho ứng dụng cần chạy liên tục trong background

auto_checker.py

Module kiểm tra email spam tự động

Có thể chạy như một service độc lập hoặc được gọi từ scheduler

Thực hiện việc quét email định kỳ và phân loại spam

Tích hợp với email monitor để xử lý email mới

gmail_login_standalone.py

Module xử lý xác thực Gmail OAuth 2.0 độc lập

Quản lý token authentication và refresh token

Có thể chạy riêng để setup/test kết nối Gmail

Lưu trữ credentials an toàn cho việc truy cập Gmail API

Thư mục config/

config.json

File cấu hình chính của ứng dụng (không được commit vào Git)

Chứa thông tin nhạy cảm: API keys, credentials, database connection

Cấu hình các tham số: email settings, model paths, thresholds

Format JSON để dễ đọc và parse

config.example.json

Template mẫu cho file config.json

Hiển thị cấu trúc và các field cần thiết

Người dùng copy file này thành config.json và điền thông tin thực tế

An toàn để commit vào Git vì không chứa dữ liệu nhạy cảm

Thư mục logs/

app.log

File log tập trung của toàn bộ ứng dụng

Ghi lại mọi sự kiện: errors, warnings, info, debug messages

Hỗ trợ troubleshooting và monitoring hệ thống

Có thể cấu hình log rotation để quản lý kích thước

Thư mục models/

Thư mục con classifiers/

Chứa các file .pkl của mô hình machine learning đã train

Các model phân loại spam như: Naive Bayes, SVM, Random Forest, etc.

Load để thực hiện prediction trên dữ liệu mới

Mỗi file .pkl là một model đã serialize sẵn sàng sử dụng

Thư mục con preprocessors/

Chứa các preprocessor đã train: TfidfVectorizer, MinMaxScaler

Feature extractors và transformers

Đảm bảo dữ liệu mới được xử lý giống như lúc training

Cần load cùng với classifier để đảm bảo tính nhất quán

Thư mục src/

__init__.py

Biến thư mục src thành Python package

Có thể chứa code khởi tạo package-level

Cho phép import các module bên trong src

Có thể định nghĩa __all__ để control public API

Thư mục con core/

Chứa business logic cốt lõi của ứng dụng

Xử lý quy trình chính: nhận input → process → trả về kết quả

Không phụ thuộc vào UI hay database cụ thể

Clean architecture, dễ test và maintain

Thư mục con services/

Các service modules: email service, notification service

Tương tác với external APIs và services

Wrapper cho third-party libraries

Separation of concerns với business logic

Thư mục con ui/

Code liên quan đến user interface

Form handlers, request validators

View logic cho Flask routes

Tách biệt presentation layer với business logic

Thư mục con utils/

Helper functions và utility modules

Code tái sử dụng nhiều nơi: text processing, file I/O, validators

Không chứa business logic cụ thể

Pure functions, dễ test

Thư mục static/

Thư mục css/

Stylesheet files cho web interface

Định nghĩa giao diện, colors, layouts, responsive design

Có thể dùng CSS frameworks như Bootstrap, Tailwind

Thư mục js/

JavaScript files cho interactive features

AJAX calls, form validation, dynamic UI updates

Có thể dùng libraries như jQuery, Alpine.js

Thư mục templates/

Chứa Jinja2 templates cho Flask

HTML files với template syntax để render động

Kế thừa base templates, include partials

Tách biệt presentation markup với Python code

Files cấu hình và deployment:

.gitignore

Liệt kê files/folders không commit vào Git

Loại trừ: __pycache__, .pyc, config.json, logs/, virtual environments

Bảo vệ thông tin nhạy cảm và files tạm thời

requirements.txt

Danh sách tất cả Python packages và versions

Dùng cho pip install -r requirements.txt

Đảm bảo môi trường dev/prod giống nhau

Generated bằng pip freeze > requirements.txt

setup_config.bat và setup_config.sh

Scripts tự động setup cấu hình ban đầu

Copy config.example.json → config.json

Có thể install dependencies, create folders

.bat cho Windows, .sh cho Linux/Mac

setup_gmail_config.bat

Script đặc biệt để cấu hình Gmail integration

Hướng dẫn người dùng setup OAuth credentials

Có thể test connection với Gmail API

Simplify quá trình onboarding

LICENSE

Giấy phép sử dụng mã nguồn (MIT, GPL, Apache, etc.)

Xác định quyền và nghĩa vụ của người dùng

Bảo vệ quyền tác giả

README.md

Tài liệu chính của dự án

Hướng dẫn cài đặt, sử dụng, cấu hình

Mô tả features, screenshots, troubleshooting

First stop cho developers mới

3.2.3. Triển khai chi tiết ứng dụng và giao diện

Ứng dụng được xây dựng dựa trên hai thành phần chính hoạt động song song và liên kết chặt chẽ với nhau nhằm đảm bảo quá trình thu thập – phân loại – hiển thị – phản hồi diễn ra tự động và liên tục.

Thành phần thứ nhất: Dịch vụ nền (Background Service/Worker) Đây là một tiến trình chạy độc lập dưới dạng daemon thông qua các module auto_checker.py và gmail_login_standalone.py, có nhiệm vụ kết nối tới hộp thư Gmail người dùng thông qua Gmail API với xác thực OAuth 2.0. Worker được khởi động từ tray_launcher.py, chạy nền trên system tray, cho phép người dùng kiểm soát trạng thái hoạt động (start/stop/restart) một cách thuận tiện.

Worker liên tục lắng nghe email mới thông qua email_monitor trong thư mục src/services/, lấy dữ liệu thô từ Gmail, giải mã nội dung email (MIME parsing), sau đó đưa qua pipeline tiền xử lý văn bản được định nghĩa trong src/utils/. Pipeline này thực

hiện các bước: làm sạch HTML tags, loại bỏ ký tự đặc biệt, tokenization, loại bỏ stopwords, lemmatization, và trích xuất các đặc trưng (TF-IDF vectorization kết hợp với 4 features bổ sung: num_char, num_word, num_sen, num_words_transform).

Sau khi tiền xử lý, dữ liệu được chuẩn hóa bằng MinMaxScaler (load từ models/preprocessors/) và đưa vào các mô hình Machine Learning đã được train sẵn (load từ models/classifiers/) để phân loại spam/ham cùng tính toán điểm tin cậy. Kết quả xử lý được ghi log vào logs/app.log và có thể được lưu trữ để phục vụ các bước tiếp theo.

Ngoài vai trò phân loại, worker còn thực hiện các tác vụ mở rộng như:

Gửi thông báo desktop khi phát hiện email spam nguy hiểm

Tự động gắn nhãn hoặc di chuyển email spam vào thư mục tương ứng thông qua Gmail API

Ghi nhận feedback của người dùng để cải thiện độ chính xác của mô hình trong tương lai

Thực hiện các tác vụ định kỳ theo lịch được cấu hình trong config/config.json

Thành phần thứ hai: Giao diện Web (Flask Application)

Đây là Flask web server được khởi động từ app.py, đóng vai trò là dashboard trực quan cho người dùng cuối. Giao diện được render thông qua Jinja2 templates (thư mục templates/) kết hợp với CSS và JavaScript (thư mục static/) để tạo trải nghiệm người dùng mượt mà và responsive.

Tại đây, người dùng có thể:

Xem và quản lý email:

Xem danh sách email đã được xử lý cùng nhãn phân loại (spam/ham), điểm dự đoán (confidence score), thời gian nhận và các metadata liên quan

Xem chi tiết nội dung email đầy đủ

Chỉnh sửa nhãn nếu phân loại sai (feedback mechanism) để cải thiện mô hình

Theo dõi thống kê và phân tích:

Dashboard hiển thị các metrics: tỷ lệ spam/ham, độ chính xác của mô hình

Biểu đồ số lượng email xử lý theo ngày/tuần/tháng

Phân tích xu hướng và patterns của spam emails

Cấu hình hệ thống:

Quản lý tài khoản Gmail: thêm/xóa/sửa thông tin xác thực

Thiết lập phương thức xác thực OAuth 2.0

Điều chỉnh ngưỡng phân loại (threshold) cho mô hình

Cấu hình lịch xử lý email tự động (polling interval)

Bật/tắt các tính năng: auto-labeling, notifications, auto-move

Thao tác thủ công:

Kích hoạt quét hộp thư ngay lập tức (manual trigger)

Test phân loại với văn bản mẫu

Xem và quản lý logs hệ thống

Tải xuống báo cáo theo khoảng thời gian tùy chỉnh

Export dữ liệu để phân tích ngoại vi

Flask application sử dụng các module trong src/core/ để xử lý business logic, src/services/ để tương tác với Gmail API, và src/ui/ để xử lý presentation logic. Các routes được định nghĩa trong app.py phục vụ cả trang web và RESTful API endpoints, cho phép tích hợp với các hệ thống khác nếu cần.

Cơ chế liên kết giữa hai thành phần

Hai thành phần này hoạt động theo mô hình tách biệt nhưng chia sẻ:

Cấu hình chung: Thông qua config/config.json, đảm bảo cả worker và web app đều sử dụng cùng settings

Log tập trung: Cả hai ghi vào logs/app.log, giúp dễ dàng troubleshooting

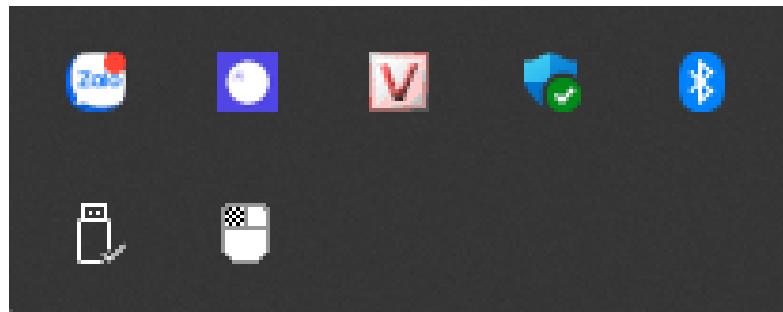
Models chia sẻ: Cùng sử dụng các file .pkl trong models/, đảm bảo tính nhất quán trong phân loại

Launcher thống nhất: launcher.py có thể khởi động đồng thời cả background worker và web server

Kiến trúc này giúp hệ thống vừa linh hoạt (có thể chạy riêng từng component), dễ mở rộng (scale worker và web app độc lập), vừa đảm bảo tính ổn định và đồng bộ trong toàn bộ quy trình phân loại email spam. Người dùng có thể chọn chạy đầy đủ với system tray app (dành cho desktop), hoặc chỉ chạy web interface (dành cho server deployment), tùy theo nhu cầu sử dụng.

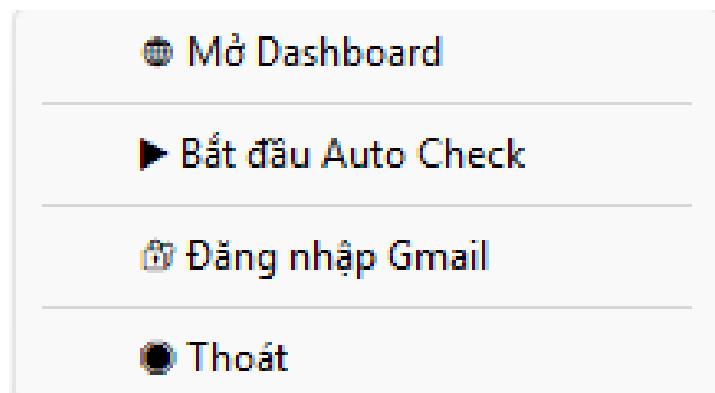
3.2.2.1. Giao diện ứng dụng và chức năng đăng nhập.

Sau khi khởi chạy tray app và sẽ có 1 icon ứng dụng hiện ra ở góc dưới bên phải của màn hình.



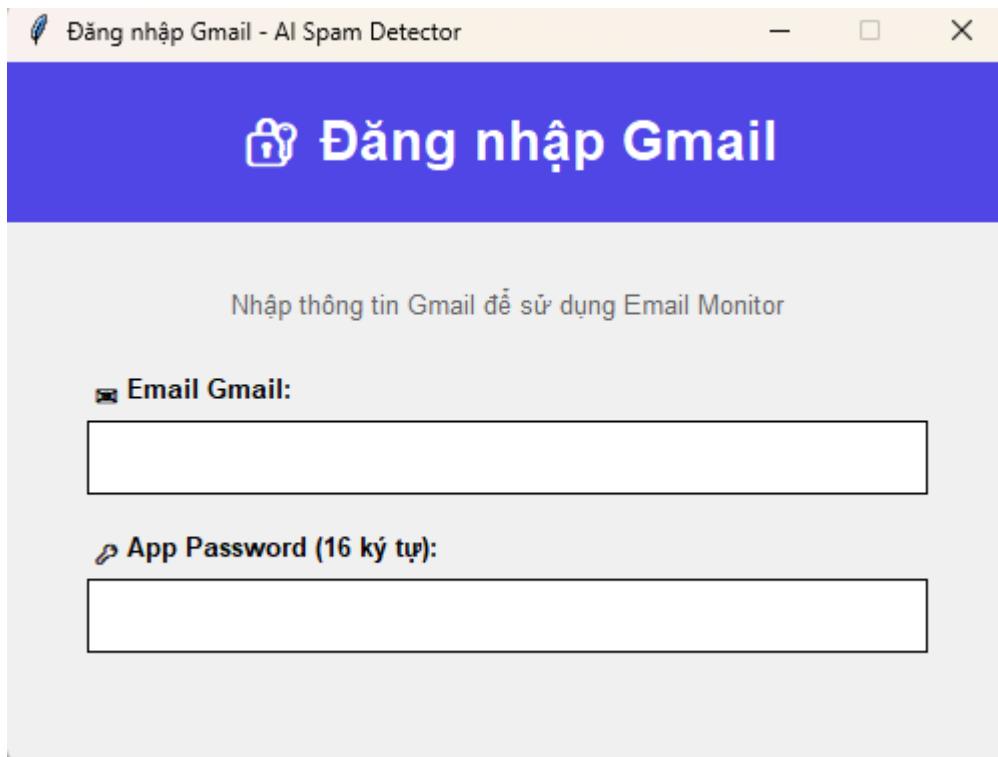
Hình 3. 3. Ứng dụng được khởi động

Tại khu vực thanh tác vụ, người dùng chỉ cần nhấp chuột phải lên biểu tượng của ứng dụng. Ngay lập tức, một bảng menu chức năng sẽ hiện ra, cho phép người dùng thao tác nhanh các tùy chọn như đăng nhập, Mở Dashboard, Bắt đầu Auto Check hoặc Thoát để tắt ứng dụng.



Hình 3. 4. Menu chức năng ứng dụng

Đầu tiên người dùng sẽ bắt đầu với tính năng đăng nhập để ứng dụng kết nối đến với tài khoản email mong muốn.



Hình 3.5. Giao diện đăng nhập Tray app

Giao diện Đăng nhập Gmail là cửa sổ xác thực ban đầu của ứng dụng AI Spam Detector, được thiết kế với header màu xanh tím nổi bật mang icon Gmail và tiêu đề "Đăng nhập Gmail" căn giữa. Phía dưới header là dòng hướng dẫn "Nhập thông tin Gmail để sử dụng Email Monitor" giải thích ngắn gọn mục đích của form đăng nhập. Giao diện bao gồm hai trường nhập liệu chính: trường "Email Gmail" với icon envelope cho phép người dùng nhập địa chỉ email đầy đủ, và trường "App Password (16 ký tự)" với icon key yêu cầu nhập mật khẩu ứng dụng đặc biệt. Điểm quan trọng cần lưu ý là App Password không phải mật khẩu Gmail thông thường mà là mã 16 ký tự do Google tạo ra dành riêng cho ứng dụng bên thứ ba khi tài khoản đã bật xác thực 2 bước. Người dùng cần truy cập Google Account → Security → App passwords để tạo mật khẩu này, việc sử dụng App Password đảm bảo an toàn hơn vì có thể thu hồi bất cứ lúc nào mà không ảnh hưởng đến tài khoản chính. Sau khi người dùng nhập thông tin và submit, hệ thống sẽ thực hiện validation kiểm tra format, tiến hành authentication kết nối đến Gmail API/IMAP, verify credentials, lưu thông tin đã mã hóa vào file config.json, kích hoạt Email Monitor service tự động và chuyển đến dashboard chính hoặc chạy nền. Giao diện có xử lý các trường hợp lỗi như email không hợp lệ, App Password sai format, xác thực thất bại, hoặc lỗi kết nối mạng với các thông báo rõ ràng giúp người dùng khắc phục. Đây là bước quan trọng đầu tiên để tích hợp ứng dụng với Gmail, cho phép hệ thống tự

động giám sát và phân loại email spam một cách liên tục với các credentials được bảo mật bằng mã hóa và quyền truy cập chỉ ở mức read-only để đảm bảo an toàn cho tài khoản người dùng.



```
1  {
2      "email": "your-email@gmail.com",
3      "password": "your-app-password-16-chars",
4      "check_interval": 300,
5      "auto_label": true,
6      "initial_load": 20,
7      "model_to_use": "Voting Classifier",
8      "notification_settings": {
9          "notify_on_spam": true,
10         "notify_on_ham": false,
11         "telegram_token": "",
12         "telegram_chat_id": ""
13     },
14     "advanced_settings": {
15         "max_emails_per_check": 50,
16         "mark_as_read": false,
17         "move_spam_to_folder": false,
18         "spam_folder_name": "[Gmail]/Spam"
19     }
20 }
```

Hình 3. 6. File Config.json

File config.example.json là template mẫu cho cấu hình hệ thống, được thiết kế với cấu trúc JSON rõ ràng và chia thành các nhóm cài đặt chính. Đầu tiên là thông tin xác thực Gmail bao gồm trường "email" với giá trị mẫu "your-email@gmail.com" để người dùng thay thế bằng địa chỉ Gmail thực tế, và trường "password" với giá trị "your-app-password-16-chars" yêu cầu nhập App Password 16 ký tự đã tạo từ Google Account. Tiếp theo là các tham số giám sát email tự động với "check_interval" được đặt là 300 (giây), tức là hệ thống sẽ kiểm tra email mới mỗi 5 phút một lần; "auto_label" được set là true để tự động gắn nhãn spam/ham cho email sau khi phân loại; "initial_load" có giá trị 20 nghĩa là lần đầu chạy sẽ load 20 email gần nhất để xử lý; và "model_to_use" được cấu hình là "Voting Classifier" chỉ định mô hình machine learning nào sẽ được sử dụng cho việc phân loại (có thể thay đổi thành các model khác như "Naive Bayes", "SVM", "Random Forest" tùy vào file .pkl có sẵn trong thư mục models/classifiers/). Phần notification_settings quản lý các thông báo với "notify_on_spam" set là true để gửi thông báo khi phát hiện spam, "notify_on_ham" set là false vì email bình thường không cần thông báo, "telegram_token" và "telegram_chat_id" để trống (dấu "") cho phép tích hợp Telegram bot gửi thông báo

nếu người dùng muốn bật tính năng này trong tương lai. Cuối cùng là advanced_settings chứa các cài đặt nâng cao như "max_emails_per_check" có giá trị 50 giới hạn số lượng email xử lý tối đa mỗi lần quét để tránh overload; "mark_as_read" được set false nghĩa là không tự động đánh dấu email đã đọc sau khi xử lý, giữ nguyên trạng thái unread để người dùng có thể kiểm tra lại; "move_spam_to_folder" cũng là false tức là không tự động di chuyển spam vào thư mục riêng mà chỉ gắn nhãn; và "spam_folder_name" được đặt là "[Gmail]/Spam" chỉ định tên thư mục Gmail spam mặc định nếu bật tính năng auto-move trong tương lai. File template này được commit an toàn vào Git vì không chứa thông tin nhạy cảm thực tế, người dùng chỉ cần copy file này thành config.json và điền các giá trị thực tế của họ, đặc biệt là email và App Password, đồng thời có thể tùy chỉnh các tham số check_interval, model_to_use, notification và advanced settings theo nhu cầu sử dụng cá nhân để tối ưu hiệu suất và trải nghiệm của ứng dụng AI Spam Detector.

Người dùng cần thực hiện các bước dưới đây để có thể kết nối gmail cá nhân với ứng dụng.

B1 — Bật IMAP trong Gmail

Mục đích: Cho phép ứng dụng kết nối tới hòm thư Gmail.

Cách thực hiện:

1. Mở Gmail trên trình duyệt → Cài đặt (Settings) → Xem tất cả chế độ cài đặt (See all settings).
 2. Chọn tab **Forwarding and POP/IMAP**.
 3. Tìm phần **IMAP Access** → chọn **Enable IMAP**.
 4. Bấm **Save Changes**.
-

B2 — Bật Xác minh 2 bước (2-Step Verification / 2FA)

Mục đích: Đây là yêu cầu bắt buộc để Google cho phép bạn tạo App Password.

Cách bật 2FA:

1. Vào trang: <https://accounts.google.com> → My Account → Security.
2. Tìm mục **Signing in to Google**.
3. Bật **2-Step Verification**.

- Xác minh bằng số điện thoại hoặc thiết bị di động.
-

B3 — Tạo Mật khẩu Ứng dụng (App Password)

Mục đích:

Ứng dụng **không được phép** dùng mật khẩu Gmail chính.

Google yêu cầu bắt buộc sử dụng App Password — một mã cô lập trong trường hợp rõ rỉ, không ảnh hưởng đến tài khoản chính.

Cách tạo App Password:

- Vào **Google Account** → **Security**.
- Trong mục **Signing in to Google**, chọn **App Passwords**.
- Đăng nhập lại nếu được yêu cầu.
- Tại mục *Select App*: chọn **Mail**.
- Tại mục *Select device*: chọn **Other (Custom name)** → nhập Spam Checker.
- Nhấn **Generate**.
- Google sẽ hiển thị một chuỗi 16 ký tự — đó là App Password.

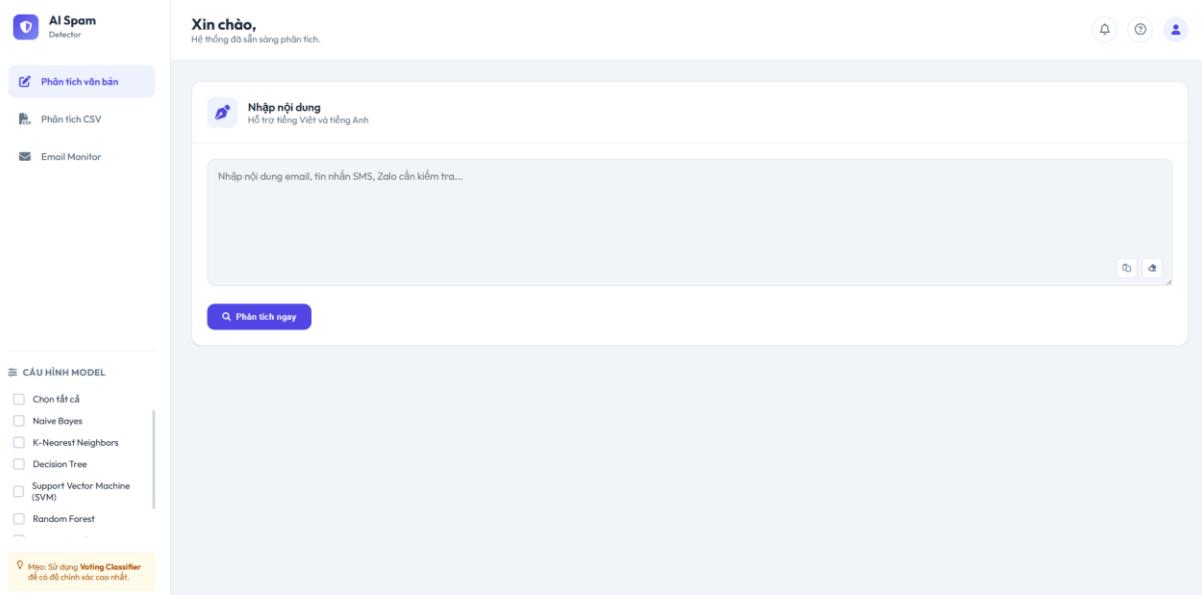
B4 — Nhập App Password vào config.json (Bước này không bắt buộc)

Mục đích: Hoàn tất cấu hình đăng nhập.

Tóm tắt cơ chế bảo mật

Thành phần	Vai trò	Lợi ích
IMAP + SSL/TLS	Kết nối mã hóa đến Gmail	Chống nghe lén/chặn dữ liệu
2FA	Xác thực 2 bước	Ngăn đăng nhập trái phép
App Password	Token truy cập riêng	Không lộ mật khẩu chính
config.json	Lưu cấu hình	Người dùng tự kiểm soát biến mật

3.2.2.2. Giao diện chức năng Dashboard



Hình 3.7. Giao diện web khi mở chức năng Dashboard

Tổng quan

Giao diện Phân tích văn bản là trang chủ chính của ứng dụng AI Spam Detector, được thiết kế theo layout hai cột với sidebar bên trái và khu vực làm việc chính bên phải, tạo trải nghiệm người dùng trực quan và dễ sử dụng.

Phần Header

Header hiển thị logo ứng dụng "AI Spam Detector" ở góc trái trên cùng, kèm theo ba icon chức năng ở góc phải: icon chuông cho thông báo, icon dấu hỏi cho trợ giúp, và icon người dùng cho quản lý tài khoản.

Sidebar bên trái

Sidebar bao gồm ba menu chính được bố trí theo chiều dọc: "Phân tích văn bản" với icon bút chì đang được active (highlight màu tím), "Phân tích CSV" với icon bảng dữ liệu cho phép upload và phân tích hàng loạt email từ file CSV, và "Email Monitor" với icon envelope để truy cập tính năng giám sát email tự động từ Gmail.

Cấu hình Model

Phía dưới sidebar là phần "CẤU HÌNH MODEL" chứa danh sách checkbox các mô hình machine learning có sẵn bao gồm "Chọn tất cả", "Naive Bayes", "K-Nearest Neighbors", "Decision Tree", "Support Vector Machine (SVM)", và "Random Forest", cho phép người dùng chọn một hoặc nhiều model để so sánh hiệu suất. Đặc biệt, ở cuối sidebar có hộp thông báo màu vàng nhạt với icon lightbulb ghi chú "Mẹo: Sử dụng **Voting Classifier** để có độ chính xác cao nhất", hướng dẫn người dùng nên dùng ensemble method để đạt kết quả tốt nhất.

Khu vực làm việc chính

Khu vực làm việc chính bên phải hiển thị lời chào "Xin chào," với dòng phụ đề "Hệ thống đã sẵn sàng phân tích" thể hiện trạng thái ready của ứng dụng. Phía dưới là phần "Nhập nội dung" với tiêu đề kèm lưu ý "Hỗ trợ tiếng Việt và tiếng Anh", sau đó là một textarea lớn màu xám nhạt với placeholder text "Nhập nội dung email, tin nhắn SMS, Zalo cần kiểm tra..." cho phép người dùng paste hoặc gõ trực tiếp văn bản cần phân loại.

Textarea và Controls

Textarea có kích thước responsive, chiếm phần lớn diện tích màn hình để dễ dàng nhập nội dung dài, và ở góc phải dưới có hai icon nhỏ: icon copy để sao chép nội dung và icon upload để tải file text lên. Cuối cùng, phía dưới textarea là nút "Phân tích ngay" với màu tím nổi bật và icon kính lúp.

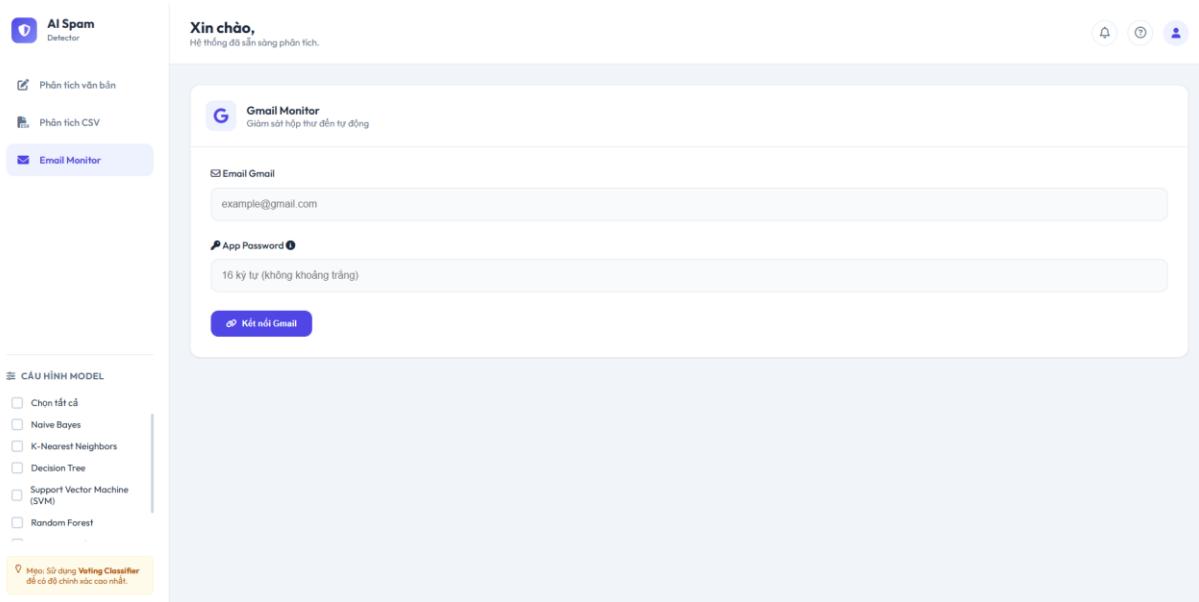
Quy trình xử lý

Khi click nút "Phân tích ngay", hệ thống sẽ trigger quá trình xử lý: lấy văn bản từ textarea, đưa qua pipeline tiền xử lý (cleaning, tokenization, stopwords removal, lemmatization), trích xuất features (TF-IDF + 4 features bổ sung), chuẩn hóa bằng MinMaxScaler, và chạy qua các model đã chọn trong sidebar để trả về kết quả phân loại spam/ham kèm confidence score và các metrics đánh giá.

Ưu điểm của giao diện

Giao diện này tối ưu cho việc test nhanh và demo, cho phép người dùng kiểm tra ngay lập tức bất kỳ đoạn văn bản nào mà không cần cấu hình phức tạp, đồng thời cung cấp

khả năng tùy chỉnh model và so sánh kết quả giữa nhiều thuật toán khác nhau để đánh giá hiệu suất và độ chính xác của từng phương pháp phân loại.



Hình 3. 8. Giao diện Email Monitor – kiểm tra bán thủ công

Tổng quan

Giao diện Email Monitor là trang quản lý kết nối Gmail trong ứng dụng AI Spam Detector, được truy cập thông qua menu "Email Monitor" trên sidebar với icon envelope màu tím đang active. Giao diện này cho phép người dùng cấu hình tài khoản Gmail để kích hoạt tính năng giám sát và phân loại email tự động.

Phần Header

Header giữ nguyên thiết kế nhất quán với các trang khác, hiển thị logo "AI Spam Detector" ở góc trái, lời chào "Xin chào," với dòng phụ đề "Hệ thống đã sẵn sàng phân tích", và ba icon chức năng ở góc phải (thông báo, trợ giúp, tài khoản).

Card Gmail Monitor

Khu vực làm việc chính hiển thị một card trắng với tiêu đề "Gmail Monitor" kèm icon Gmail màu đỏ-vàng-xanh-xanh lá đặc trưng của Google, bên dưới là dòng mô tả "Giám sát hộp thư đến tự động" giải thích ngắn gọn chức năng của trang này.

Form nhập thông tin

Card chứa form đăng nhập Gmail với hai trường nhập liệu chính. Trường đầu tiên là "Email Gmail" với icon envelope, hiển thị placeholder "example@gmail.com" gợi ý format địa chỉ email đúng. Trường thứ hai là "App Password" với icon key và icon thông tin (i) bên cạnh để người dùng click xem hướng dẫn chi tiết, placeholder hiển thị "16 ký tự (không khoảng trắng)" nhấn mạnh yêu cầu format của App Password phải là chuỗi 16 ký tự liên tục không có khoảng trắng.

Nút kết nối

Phía dưới form là nút "Kết nối Gmail" màu tím nổi bật với icon link, khi click sẽ thực hiện xác thực credentials, test connection với Gmail API, và lưu thông tin vào config.json nếu thành công.

Sidebar Model Configuration

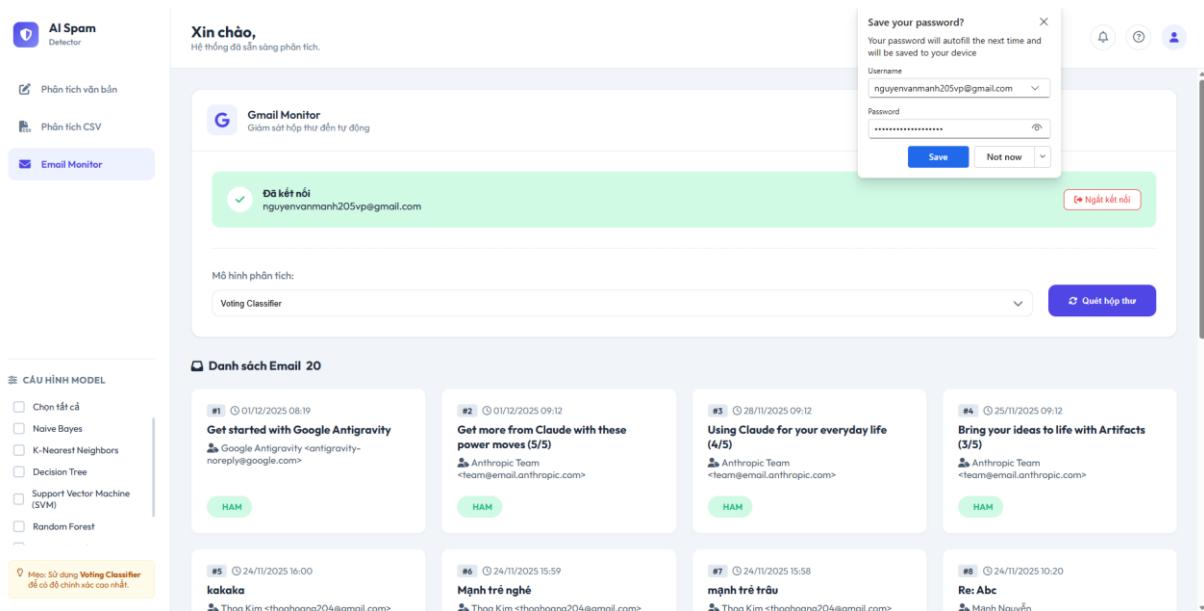
Sidebar bên trái vẫn hiển thị phần "CẤU HÌNH MODEL" với danh sách checkbox các thuật toán: "Chọn tất cả", "Naive Bayes", "K-Nearest Neighbors", "Decision Tree", "Support Vector Machine (SVM)", và "Random Forest", cho phép người dùng chọn model sẽ được sử dụng khi Email Monitor tự động phân loại email. Phía dưới là hộp thông báo màu vàng với mèo "Sử dụng Voting Classifier để có độ chính xác cao nhất", khuyến nghị người dùng dùng ensemble method cho kết quả tối ưu.

Luồng hoạt động

Sau khi người dùng nhập email và App Password, click "Kết nối Gmail", hệ thống sẽ validate thông tin, thực hiện OAuth 2.0 authentication với Gmail API, verify quyền truy cập (read-only scope), lưu credentials đã mã hóa vào file config.json, và hiển thị thông báo thành công. Từ đó, background service (auto_checker.py) sẽ tự động kích hoạt, bắt đầu giám sát hộp thư theo check_interval đã cấu hình, và mỗi khi có email mới, sẽ tự động phân loại bằng model đã chọn trong sidebar, gắn nhãn spam/ham, và thực hiện các action tương ứng theo cấu hình trong advanced_settings.

Ưu điểm thiết kế

Giao diện này tách biệt rõ ràng giữa việc cấu hình kết nối Gmail và cấu hình model phân loại, giúp người dùng dễ dàng setup từng bước mà không bị overwhelm bởi quá nhiều options cùng lúc. Việc tích hợp trực tiếp form đăng nhập vào web interface thay vì yêu cầu chạy script riêng biệt tạo trải nghiệm user-friendly và phù hợp với người dùng không có kiến thức kỹ thuật sâu.



Hình 3. 9. Giao diện sau khi email monitor kết nối thành công với email

Tổng quan

Giao diện Email Monitor sau khi kết nối Gmail thành công hiển thị trạng thái đã xác thực và danh sách email được giám sát, cho phép người dùng theo dõi kết quả phân loại và quản lý hộp thư một cách trực quan.

Popup lưu mật khẩu

Ở góc phải trên màn hình xuất hiện popup "Save your password?" của trình duyệt với thông báo "Your password will autofill the next time and will be saved to your device". Popup hiển thị Username "nguyenvanmanh205vp@gmail.com" và Password được ẩn dưới dạng dots, kèm hai nút "Save" (màu xanh) và "Not now" để người dùng lựa chọn lưu credentials vào trình duyệt cho lần đăng nhập sau.

Thông báo kết nối thành công

Phía dưới card Gmail Monitor hiển thị banner màu xanh lá nhạt với icon checkmark tròn, text "Đã kết nối" và địa chỉ email "nguyenvanmanh205vp@gmail.com" xác nhận việc authentication đã thành công. Bên phải banner có nút "Ngắt kết nối" màu đỏ nhạt với icon unlink, cho phép người dùng disconnect tài khoản Gmail khỏi ứng dụng khi cần.

Dropdown chọn Model

Ngay dưới banner kết nối là section "Mô hình phân tích:" với dropdown menu hiển thị "Voting Classifier" - mô hình đang được sử dụng để phân loại email. Người dùng có thể click vào dropdown để chọn model khác từ danh sách các thuật toán có sẵn (Naive Bayes, SVM, Random Forest, etc.) hoặc giữ nguyên Voting Classifier như khuyến nghị để đạt độ chính xác cao nhất.

Nút quét hộp thư

Bên phải dropdown là nút "Quét hộp thư" màu tím nổi bật với icon refresh, cho phép người dùng trigger manual scan để kiểm tra email mới ngay lập tức thay vì chờ theo chu kỳ tự động (check_interval). Khi click, hệ thống sẽ kết nối Gmail API, fetch email mới, và chạy qua pipeline phân loại.

Danh sách Email

Phần chính của giao diện là section "Danh sách Email 20" với icon inbox, hiển thị 20 email gần nhất đã được xử lý. Danh sách được trình bày dưới dạng grid 4 cột, mỗi email card chứa các thông tin: số thứ tự (#1, #2, #3...), timestamp nhận email (format: DD/MM/YYYY HH:MM), tiêu đề email (subject line), thông tin người gửi với icon user và địa chỉ email trong ngoặc nhọn, và nhãn phân loại ở dưới cùng.

3.2.2.3. Giao diện Auto Checker

```
warnings.warn(
C:\Users\Admin\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X
does not have valid feature names, but MinMaxScaler was fitted with feature names
    warnings.warn(
C:\Users\Admin\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X
does not have valid feature names, but MinMaxScaler was fitted with feature names
    warnings.warn(
C:\Users\Admin\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X
does not have valid feature names, but MinMaxScaler was fitted with feature names
    warnings.warn(
C:\Users\Admin\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X
does not have valid feature names, but MinMaxScaler was fitted with feature names
    warnings.warn(
C:\Users\Admin\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X
does not have valid feature names, but MinMaxScaler was fitted with feature names
    warnings.warn(
C:\Users\Admin\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X
does not have valid feature names, but MinMaxScaler was fitted with feature names
    warnings.warn(
=====
✉ NEW EMAILS DETECTED!
=====
📊 Total: 10 new emails (0 spam, 10 ham)
=====
🕒 Real-time stats: 10 emails (0 spam)
🕒 Waiting 300s before next check...
```

Hình 3.10. Giao diện auto checker được khởi động

Console hiển thị output của background worker service (auto_checker.py) đang chạy nền để giám sát và phân loại email tự động, thể hiện hai bước quan trọng trong quy trình xử lý. **Step 1 (Khởi tạo ban đầu)** được kích hoạt ngay sau khi người dùng kết nối Gmail thành công, hệ thống tự động fetch và phân loại **20 email mới nhất** trong hộp thư inbox như được định nghĩa bởi tham số initial_load: 20 trong config.json, các email này được đưa qua pipeline xử lý đầy đủ bao gồm preprocessing (làm sạch HTML, tokenization, stopwords removal, lemmatization), feature extraction (TF-IDF vectorization kết hợp 4 features bổ sung: num_char, num_word, num_sen, num_words_transform), scaling bằng MinMaxScaler, và cuối cùng prediction bằng Voting Classifier để phân loại spam/ham kèm confidence score, kết quả 20 email này sẽ được hiển thị ngay lập tức trong web interface ở section "Danh sách Email 20" để người dùng có thể xem và tương tác. Console thông báo kết quả Step 1 với message "NEW EMAILS DETECTED!" theo sau là thông kê "Total: 10 new emails (0 spam, 10 ham)" cho thấy trong batch đầu tiên đã xử lý 10 email và không phát hiện spam nào, kèm real-time stats "Real-time stats: 10 emails (0 spam)" xác nhận lại kết quả phân

loại. Sau khi hoàn thành Step 1, hệ thống chuyển sang **Step 2 (Giám sát liên tục)** với message "⏳ Waiting 300s before next check..." cho biết worker đang chờ 300 giây (5 phút) theo check_interval: 300 được cấu hình trước khi thực hiện lần quét tiếp theo, và từ Step 2 trở đi, worker sẽ hoạt động theo vòng lặp vô tận: kết nối Gmail API → fetch chỉ email mới phát sinh kể từ lần quét trước (incremental scan thay vì lấy 20 email như Step 1) → xử lý qua cùng pipeline → phân loại → cập nhật web interface → sleep 300s → lặp lại, đảm bảo hộp thư được giám sát liên tục mà không bị xử lý trùng lặp hay overload hệ thống. Cơ chế hai bước này giúp người dùng có trải nghiệm tốt ngay từ đầu với 20 email được hiển thị instant sau khi kết nối, đồng thời đảm bảo monitoring real-time hiệu quả với interval hợp lý cho các email mới trong tương lai.

CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1. Kết luận về đề tài

Trong báo cáo này, hệ thống phân loại thư rác dựa trên các mô hình học máy đã được xây dựng và triển khai thành công thông qua giao diện Flask. Quá trình thực hiện bao gồm từ việc thu thập và xử lý dữ liệu, xây dựng các mô hình học máy đến việc triển khai ứng dụng thực tế. Những điểm nổi bật có thể tóm tắt như sau:

Các mô hình học máy này đều có những đặc điểm riêng biệt, từ cách xử lý các đặc trưng dữ liệu cho đến khả năng phân loại chính xác các email rác và hợp lệ. Các mô hình Naive Bayes, SVM, Random Forest, Decision Tree và KNN đã được áp dụng hiệu quả trong việc phân loại email.

Giao diện người dùng qua Flask: Giao diện của ứng dụng mang lại giao diện trực quan, thân thiện, cho phép người dùng kiểm tra nội dung email hoặc tải lên file CSV để phân loại hàng loạt.

Xây dựng ứng dụng Tray app: Việc xây dựng Tray app mang lại tính tiện lợi và tăng khả năng truy cập, quản lý tập trung nhiều tính năng, lưu trữ thông tin đăng nhập, thông báo người dùng, phù hợp hơn với việc sử dụng thực tế.

Tính năng phân loại linh hoạt và dễ sử dụng: Một trong những điểm nổi bật của dự án là tính năng phân loại email với khả năng kiểm tra nội dung email ngay lập tức và cung cấp kết quả nhanh chóng. Các tùy chọn tải xuống kết quả dự đoán cũng giúp người dùng dễ dàng lưu trữ và sử dụng kết quả cho mục đích lâu dài.

Dự án không chỉ giải quyết vấn đề phát hiện thư rác mà còn cung cấp giải pháp có khả năng mở rộng và tích hợp với các hệ thống email hiện có.

4.2. Hướng phát triển

Dựa trên các kết quả đạt được và những hạn chế còn tồn tại, dự án có thể được phát triển theo các hướng sau:

Nâng cao độ chính xác của mô hình: Áp dụng các kỹ thuật ensemble (như Bagging, XGBoost) để cải thiện khả năng phân loại. Tăng cường tập dữ liệu huấn luyện với các nguồn dữ liệu thực tế hơn để đảm bảo tính đa dạng.

Mở rộng tính năng của ứng dụng: Phát triển thêm tính năng lọc và phân loại email tự động trong thời gian thực. Cung cấp giao diện API để tích hợp với các dịch vụ email như Gmail hoặc Outlook.

Tăng cường trải nghiệm người dùng: Tối ưu hóa hiệu suất xử lý khi làm việc với file CSV lớn. Thêm tính năng phân tích chi tiết nội dung email, ví dụ: hiển thị các từ khóa hoặc cụm từ nghi ngờ là SPAM.

Ứng dụng công nghệ mới: Thủ nghiệm và tích hợp các mô hình ngôn ngữ hiện đại như BERT hoặc GPT để xử lý nội dung email chính xác hơn. Áp dụng học sâu (deep learning) trong trường hợp yêu cầu độ chính xác cao hơn.

Phát triển bảo mật: Bổ sung các cơ chế mã hóa để bảo vệ dữ liệu email của người dùng trong quá trình xử lý. Đảm bảo tuân thủ các quy định về bảo mật thông tin, như GDPR hoặc các quy định tương tự.

TÀI LIỆU THAM KHẢO

1. Venkatesh Garnepudi, *Spam Mails Dataset*,
https://www.kaggle.com/datasets/venky73/spam-mails-dataset?select=spam_ham_dataset.csv, 2019
2. KANAGALINGAM S M, Email-Spam-Detection, <https://github.com/kanagalingamsm/Email-Spam-Detection>, 2023
3. *Detecting Spam Emails Using Tensorflow in Python*,
<https://www.geeksforgeeks.org/detecting-spam-emails-using-tensorflow-in-python/>, 2023
4. Azim Khan, *Email Spam Detection with Machine Learning: A Comprehensive Guide*, <https://medium.com/@azimkhan8018/email-spam-detection-with-machine-learning-a-comprehensive-guide-b65c6936678b>, 2024
5. Ứng dụng học máy trong lọc thư rác, <https://m.antoanthongtin.vn/giai-phap-khac/ung-dung-hoc-may-trong-loc-thu-rac-107401>, 2021
6. Truong-LV, Spam-filter, <https://github.com/truong-lv/Spam-filter>, 2021
7. Arindam Paul, Email_Spam_Detection_with_Machine_Learning,
https://github.com/Apaulgithub/oibspip_taskno4/blob/main/Email_Spam_Detection_with_Machine_Learning.ipynb, 2023