ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

MÔN HỌC

TRÍ TUỆ NHÂN TẠO CHO AN NINH THÔNG TIN

Lab 1: Phân loại thư điện tử

Thành viên:
21120107 - Nguyễn Minh Nhật
21120056 - Nguyễn Đặng Tường Duy

Giảng viên:
PGS.TS Nguyễn Đình Thúc
ThS. Vũ Quốc Hoàng
ThS. Nguyễn Ngọc Toàn
ThS. Nguyễn Thị Hường

Ngày 18 tháng 10 năm 2024

Mục lục

1 Thông tin chung **3**
1.1 Đánh giá kết quả . . . 3

2 Giới thiệu bài toán **4**
2.1 Bộ dữ liệu Enron-Spam . . . 4

3 Tiền xử lý dữ liệu **5**
3.1 Làm sạch dữ liệu . . . 5
3.1.1 Xử lý trên cấp độ dòng dữ liệu . . . 5
3.1.2 Xử lý trong từng dòng dữ liệu . . . 5
3.1.3 Xử lý giá trị outlier . . . 5
3.2 Tăng cường dữ liệu . . . 6
3.3 Trích xuất đặc trưng dữ liệu . . . 6
3.3.1 Trích xuất đặc trưng TF-IDF . . . 6
3.3.2 Trích xuất đặc trưng word2vec . . . 8

4 Phân tích dữ liệu Enron-Spam **10**
4.1 Phân bố nhãn dữ liệu . . . 10
4.2 Một số phân bố dữ liệu khác . . . 11
4.3 Đặc trưng dữ liệu . . . 13

5 Mô hình đề xuất **14**
5.1 Các phương pháp thông dụng . . . 14
5.1.1 Support Vector Machine (SVM) . . . 14
5.1.2 Naive Bayes . . . 17
5.1.3 Logistic Regression . . . 20
5.1.4 XGBoost . . . 24
5.1.5 Đánh giá kết quả . . . 26
5.2 Phương pháp đề xuất . . . 27
5.2.1 Voting Classifier . . . 27
5.2.2 Stacking Classifier . . . 28
5.2.3 Grid Search . . . 30
5.3 Kết quả thử nghiệm . . . 31
5.3.1 Các metric đánh giá . . . 32

6 Thử nghiệm thực tế **33**
6.1 Đường dẫn liên quan . . . 33
6.2 Thử nghiệm chức năng . . . 33

6.2.1 Cách cài đặt và chạy chương trình 33
6.2.2 Giao diện chương trình 33
6.2.3 Chức năng 1: Nhập vào một email và dự đoán 34
6.2.4 Chức năng 2: Đọc file csv, thực hiện dự đoán và đánh giá 34
6.3 Hạn chế và hướng phát triển 35

Tài liệu tham khảo **36**

1 Thông tin chung

MSSV	Họ và tên	Đóng góp (tối đa 100%)
21120107	Nguyễn Minh Nhật	50%
21120056	Nguyễn Đặng Tường Duy	50%

1.1 Đánh giá kết quả

STT	Đặc tả yêu cầu	Mức độ hoàn thành
1	Tải và xuất dữ liệu Enron-Spam	Hoàn Thành
2	Tiền xử lý dữ liệu	Hoàn Thành
3	Train mô hình và đánh giá kết quả	Hoàn Thành
4	Viết chương trình kiểm thử tính năng	Hoàn Thành

2 Giới thiệu bài toán

Bài toán phân loại email rác là một trong những ứng dụng quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên và học máy. Trong bài báo cáo này, chúng tôi sẽ giới thiệu về bộ dữ liệu Enron-Spam, một nguồn tài liệu hữu ích cho việc nghiên cứu và phát triển các phương pháp phân loại thư điện tử.

2.1 Bộ dữ liệu Enron-Spam

- Bộ dữ liệu Enron-Spam được thu thập bởi V. Metsis, I. Androutsopoulos và G. Paliouras và được mô tả trong ấn phẩm của họ "Spam Filtering with Naive Bayes - Which Naive Bayes?". Bộ dữ liệu này chứa tổng cộng 17.171 thư rác và 16.545 thư không phải thư rác (hay còn gọi là "ham"), với tổng số lượng thư điện tử lên đến 33.716.
- Mỗi thư (1 dòng) trong tập dữ liệu có các đặc điểm như sau:
  - **Subject:** Tên tiêu đề của thư.
  - **Message:** Nội dung của email. Có thể chứa chuỗi rỗng nếu tin nhắn chỉ có dòng tiêu đề và không có nội dung. Trong trường hợp chuyển tiếp email hoặc trả lời, điều này cũng chứa tin nhắn gốc với các dòng tiêu đề như "Từ:", "Đến:", v.v.
  - **Spam/Ham:** Có giá trị "spam" hoặc "ham". Đây là nhãn của thư được phân loại, cho biết liệu tin nhắn có phải là spam hay không.

3 Tiền xử lý dữ liệu

3.1 Làm sạch dữ liệu

3.1.1 Xử lý trên cấp độ dòng dữ liệu

Trong bộ dữ liệu Enron-Spam được cung cấp bao gồm tổng cộng 27.284 email trong tập dữ liệu huấn luyện (train set) và 3084 email trong tập dữ liệu đánh giá lại (val set).
Như đã có giới thiệu ở phần 2.1, mỗi dòng trong tập dữ liệu tương ứng với một email. Và trong tập train cùng với tập val cũng có chứa một số lượng không ít những dòng gây nhiễu, chúng em tiến hành xóa bỏ những dòng dữ liệu gây nhiễu này để làm sạch dữ liệu:

- Những dòng bị lặp lại.
- Những dòng mà không có cả tiêu đề và nội dung email (giá trị ở hai cột Subject và Message đều là rỗng).
- Những dòng mà hoặc không có tiêu đề, hoặc không có nội dung email (giá trị ở một trong hai cột Subject hoặc Message là rỗng) thì vẫn được giữ lại, do thông thường email được gửi đi cũng có thể không có tiêu đề hoặc không có nội dung.

Sau khi đã xử lý xong, số lượng dòng dữ liệu (email) trong tập train và tập val lần lượt còn lại là 25.021 email và 3031 email. Ta thấy được số lượng dòng dữ liệu nhiễu trong tập train chiếm 7.635% và trong tập val chiếm 1.72%.

3.1.2 Xử lý trong từng dòng dữ liệu

Ta thấy rằng tiêu đề và nội dung của từng email là dạng văn bản chứa nhiều thành phần như chữ cái, chữ số, các dấu câu và một số loại kí tự khác.
Trong bài toán phát hiện thư rác, các từ trong email sẽ được quan tâm nhiều hơn, đồng thời vì cách trích xuất đặc trưng dữ liệu của nhóm chúng em nên:

- Các dấu câu và các kí tự khác sẽ trở thành các thành phần gây nhiễu và cần được loại bỏ.
- Các từ dừng (stopword) như (“the”, “is”, . . .) trong tiêu đề và nội dung email cũng được loại bỏ, vì các từ này xuất hiện nhiều trong các email nhưng không mang nhiều thông tin phân biệt giữa các email đó.
- Chúng em còn chuẩn hóa các từ trong tiêu đề và nội dung email bằng cách bổ đề hóa (lemmatize) các từ đó, mục đích là để chuyển các từ tiếng Anh về dạng gốc của nó thay vì xử lý với nhiều loại khác nhau của cùng một từ vựng.

Tổng hợp cả hai cách xử lý làm sạch dữ liệu vừa trình bày ở trên, có thể giúp cho phương pháp đề xuất của chúng em trở nên hiệu quả hơn, accuracy của mô hình trên tập val tăng lên khoảng 0.08%. Kết quả chi tiết được trình bày ở phần sau.

3.1.3 Xử lý giá trị outlier

Sau khi đã rút trích được đặc trưng từ văn bản của bộ dữ liệu email (sẽ được trình bày ở phần 3.3), sẽ có những mẫu dữ liệu mang đặc trưng khác xa so với phần còn lại của bộ dữ liệu. Nếu bộ dữ liệu Enron-Spam là tập hợp các email của nhân viên trong công ty Enron thì những mẫu dữ liệu outlier có thể có nội dung khác hoàn toàn, không liên quan gì đến công việc, hay

công ty Enron.
Và như vậy, chúng em sử dụng Rừng biệt lập (Isolation Forest) là một phương pháp tách biệt các mẫu dữ liệu outlier bằng cách sử dụng cây quyết định:

- Phương pháp này xây dựng nhiều cây quyết định bằng cách chọn ngẫu nhiên một đặc trưng và sau đó chọn ngẫu nhiên một giá trị phân tách trong khoảng giữa cực đại và cực tiểu của thuộc tính đó. Quy trình này tiếp tục cho đến khi mỗi mẫu được tách biệt.
- Đường đi từ gốc cây đến nút cuối cùng, hay còn gọi là độ dài đường đi, thể hiện số lần phân tách cần thiết để cô lập một mẫu. Các mẫu bình thường thường cần nhiều lần phân tách hơn, trong khi các mẫu bất thường thường được tách biệt dễ dàng hơn, tức là có độ dài đường đi ngắn hơn.
- Sau khi tạo ra nhiều cây quyết định, rừng biệt lập tính toán độ dài đường đi trung bình cho từng mẫu trong toàn bộ rừng. Các mẫu có độ dài đường đi ngắn hơn trung bình có khả năng cao là outlier.

Tuy vậy, số lượng các mẫu dữ liệu outlier trong tập huấn luyện là không đáng kể, nên biện pháp này làm cải thiện độ hiệu quả của mô hình đề xuất lên khá ít, kết quả chi tiết được trình bày ở phần sau.

3.2 Tăng cường dữ liệu

Ngoài phương pháp làm sạch dữ liệu, chúng em còn áp dụng tăng cường dữ liệu (data augmentation) đối với tập huấn luyện, như là một phương pháp giúp tăng độ đa dạng, phong phú cho dữ liệu email, với kỳ vọng mô hình đề xuất của chúng em có khả năng học tốt hơn.
Các phương pháp tăng cường dữ liệu được chúng em áp dụng là:

- Chọn ra một số từ trong email một cách ngẫu nhiên, sau đó chèn một từ đồng nghĩa với từ đó (nếu có thể) vào trong email.
- Chọn ngẫu nhiên một số từ trong email và tiến hành đổi vị trí của chúng trong email.

Nhìn chung các phương pháp tăng cường dữ liệu có thể đem lại sự đa dạng cho tập dữ liệu nhưng chưa giúp cải thiện được độ chính xác của mô hình đề xuất của chúng em, gần như không đáng kể. Chi tiết về kết quả được mô tả ở phần sau.

3.3 Trích xuất đặc trưng dữ liệu

Sau khi trải qua bước làm sạch dữ liệu, ta cần phải chọn ra phương pháp rút trích đặc trưng phù hợp, mạnh mẽ cho từng email nhằm mục đích dùng đặc trưng đó để phân loại cho chính email.
Để huấn luyện dữ liệu văn bản cho các mô hình học máy, ta cần phải lượng hóa dữ liệu văn bản và nếu làm như vậy, dữ liệu sau khi lượng hóa có thể có rất nhiều chiều và thưa thớt.
Khi này, ta có thể sử dụng Word Embedding, nó là kỹ thuật biểu diễn mỗi từ bằng một vector có số chiều nhỏ, cố định và có dạng dày đặc hơn (dense), các vector biểu diễn cho các từ thuộc cùng một nhóm hoặc liên quan với nhau sẽ nằm gần nhau trong không gian [1].

3.3.1 Trích xuất đặc trưng TF-IDF

Để vector hóa dữ liệu văn bản, một cách tiếp cận đơn giản chính là sử dụng Bag of Words (BOW):

- Ta tạo một tập từ điển gồm tất cả các từ có trong các văn bản.
- Với mỗi một văn bản, ta đếm tần suất xuất hiện trong từ điển của các từ trong văn bản đó.
- Mỗi văn bản sẽ được biểu diễn dưới dạng vector có số chiều bằng với kích thước của từ điển, các phần từ của một vector là tần suất xuất hiện của các từ trong câu đó.

Trong phương pháp của chúng em đã đề xuất việc sử dụng đặc trưng TF-IDF (Term Frequency - Inverse Document Frequency) để làm công cụ trích xuất đặc trưng chủ yếu. Nghĩa là áp dụng giá trị TF-IDF lên các vector đã trích xuất được từ BOW.
TF-IDF là một phương pháp thống kê (hay cách tính trọng số) được sử dụng để đánh giá mức độ quan trọng của một từ đối với một văn bản (document) trong một tập hợp các văn bản (corpus) [2, 3]. Nó là sự kết hợp của hai thành phần:

- TF hay Term Frequency: Tần suất từ đo lường Đo lường mức độ thường xuyên một từ xuất hiện trong một văn bản. Với ý nghĩa rằng một từ xuất hiện nhiều trong một văn bản sẽ có xu hướng đại diện cho nội dung văn bản đó.
  TF được tính bằng số lần xuất hiện của từ chia cho độ dài của văn bản (tổng số từ trong văn bản đó) hay:
  $$TF_{t,d} = \frac{Count(t,d)}{NumWords(d)}.$$
  Trong đó:
  - $Count(t,d)$ là số lần xuất hiện của từ (term) t trong văn bản (document) d.
  - $NumWords(d)$ là tổng số lượng từ trong văn bản d.

  Đôi khi ta còn dùng một công thức khác để tính tần suất từ:
  $$TF = \begin{cases} 1 + \log_{10} Count(t,d), & Count(t,d) > 0 \\ 0, & Count(t,d) = 0 \end{cases}.$$
  Ở đây việc sử dụng logarithm cơ số 10 trong công thức giúp giảm sự ảnh hưởng của những từ xuất hiện quá nhiều lần trong một văn bản.
- IDF hay Inverse Document Frequency: Tần suất nghịch đảo của văn bản đo lường mức độ quan trọng của một từ thông qua việc xem xét mức độ phổ biến (số lần xuất hiện) của từ đó trên toàn bộ tập các văn bản.
  Ở đây, IDF là nghịch đảo (Inverse) của số lượng văn bản mà một từ xuất hiện bên trong (Document Frequency) và được tính bằng công thức:
  $$IDF_t = \log_{10}\left(\frac{N}{1 + DF_t}\right).$$
  Trong đó:
  - N là tổng số lượng văn bản trong corpus.
  - DF_t là số lượng văn bản mà từ t xuất hiện bên trong. Ở đây, $1 + DF_t$ để tránh trường hợp từ t không xuất hiện trong văn bản nào.

  Logarithm cơ số 10 cũng được sử dụng để giảm mức độ quan trọng của những từ xuất hiện thường xuyên trong nhiều tài liệu, đồng thời tăng cường mức độ quan trọng của các từ mà xuất hiện ít hơn.

Khi đó, giá trị TF-IDF đại diện cho mức độ quan trọng của từ đang xét với văn bản và được tính bằng công thức:

$$TF\text{-}IDF_{t,d} = TF_{t,d} \times IDF_t.$$

Trong đó:

- $TF_{t,d}$ là tần suất xuất hiện của từ t trong văn bản d.
- IDF_t là tần suất nghịch đảo văn bản của từ t.

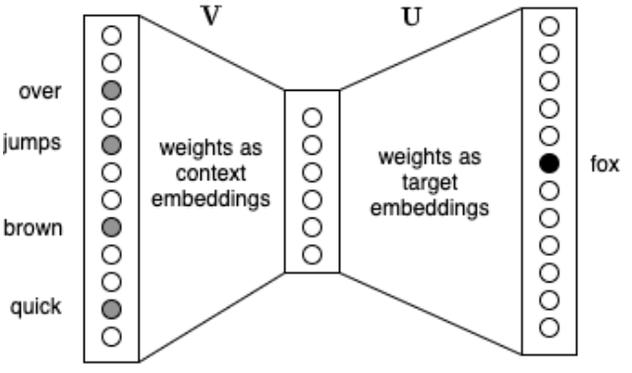
Sau khi biến đổi các vector từ BOW bằng TF-IDF, ta sẽ được các vector biểu diễn cho dữ liệu văn bản với đặc trưng tốt hơn, bám sát ngữ nghĩa hơn.

3.3.2 Trích xuất đặc trưng word2vec

word2vec là một công cụ cung cấp cách trích xuất đặc trưng của dữ liệu văn bản dưới dạng vector mà ở đó, chúng biểu diễn mối quan hệ tương đồng về ngữ nghĩa một cách tốt hơn giữa các từ [4]. word2vec bao gồm hai cách xây dựng mô hình: Skip–gram và Continuous Bag of Words (CBOW).
Để biểu diễn ngữ nghĩa của dữ liệu văn, cả hai phương pháp này đều dựa trên xác suất có điều kiện, dùng để dự đoán một số từ sử dụng một số từ khác nằm trong cùng tập ngữ liệu (corpus):

- Skip–gram: Giả sử ta có một target word, từ nó ta có thể dự đoán được các context word (từ xung quanh). Skip-gram có khả năng biểu diễn tốt những từ có tần suất thấp vì ta xây dựng được nhiều mẫu huấn luyện xung quanh từ có tần suất thấp này.
- CBOW: Giả sử rằng từ các context word, ta có thể dự đoán được target word. Khác với Skip–gram, CBOW có khả năng biểu diễn tốt hơn các từ xảy ra thường xuyên.

Chúng em sử dụng word2vec với cách xây dựng mô hình là CBOW. Trong đó ta chọn ra một từ "trung bình" (trung bình embedding của các context word) để dự đoán cho target word. Ta



Hình 1: Minh họa cho CBOW dưới dạng Mạng Neural một lớp ẩn

có thể được sức mạnh của word2vec so với TF-IDF trong bài toán có liên quan đến dữ liệu văn bản, do có khả năng biểu diễn mối quan hệ ngữ nghĩa tốt hơn cách tính toán thống kê có phần thô sơ của TF-IDF.
Tuy vậy, đối với bài toán phân loại email trên bộ dữ liệu Enron-Spam, phương pháp tính embedding vector sử dụng TF-IDF lại cho thấy được hiệu quả tốt hơn so với phương pháp word2vec. Đó là bởi vì:

- Trong email có nhiều trường hợp nội dung của chúng chứa các từ thông dụng, thường hay xuất hiện với tần suất cao như "investment", "free", "offer", hoặc đôi khi các từ trong đó bị viết sai chính tả, dẫn đến khả năng nắm bắt ngữ cảnh của word2vec bị ảnh hưởng và hoạt động không tốt [5].

- TF-IDF đánh giá độ quan trọng của một từ trong văn bản dựa trên tần suất xuất hiện, giúp biểu diễn tốt các từ xuất hiện nhiều lần hoặc mang tính chất đặc biệt.
- Bên cạnh đó, việc sử dụng word2vec trên dữ liệu huấn luyện có kích thước không quá lớn khiến cho mô hình word2vec được khớp chưa đủ tốt. Trong khi TF-IDF vẫn có thể hoạt động khá hiệu quả đối với tập dữ liệu có kích vừa và nhỏ như tập Enron-Spam [6].

Kết quả phân loại của mô hình khi sử dụng các phương pháp trích xuất đặc trưng mà chúng em vừa đề cập ở trên sẽ được trình bày trong phần sau.

4 Phân tích dữ liệu Enron-Spam

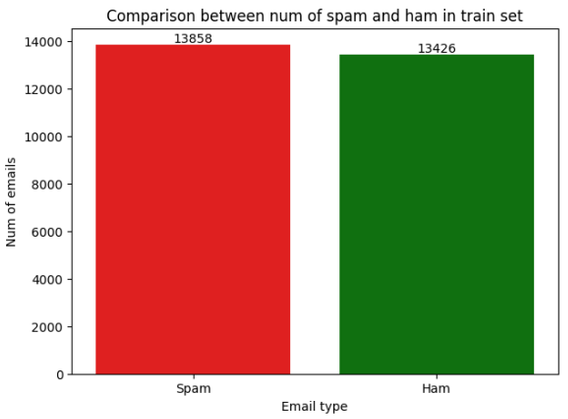
4.1 Phân bố nhãn dữ liệu

Số lượng các email được gán nhãn là spam hoặc ham trong tập huấn luyện và tập đánh giá lại, là khá cân bằng và không có sự chênh lệch quá đáng kể, cả trước và sau khi tiến hành làm sạch dữ liệu:

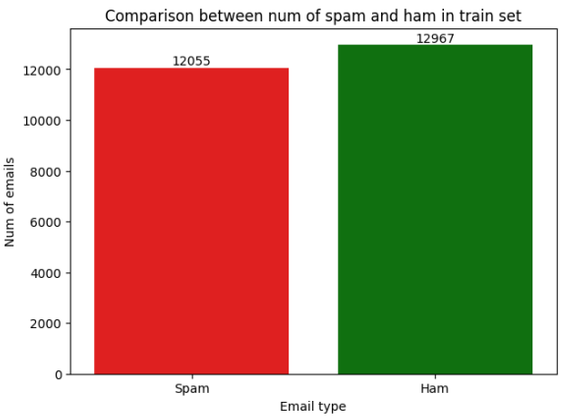
Tập train (đv: email)	**Spam**	**Ham**	**Tổng cộng**
Trước khi làm sạch	13.858	13.426	27.284
Sau khi làm sạch	12.055	12.967	25.022

Bảng 1: Số lượng email và phân bố nhãn trong tập huấn luyện

Ta có thể thấy rằng sau khi được làm sạch thì số lượng chênh lệch giữa spam email và ham email đã tăng lên. Lý do là vì những email có cả tiêu đề và nội dung đều là rỗng, đa phần được gán nhãn là spam email. Những email như vậy chiếm số lượng không ít và khiến cho số lượng email được gán nhãn là spam bị giảm đi. Nhưng nhìn chung, ở hình 2 sự chênh lệch giữa số lượng spam email và ham email là không quá đáng kể, tập huấn luyện có thể xem là cân bằng.



Phân bố nhãn trước làm sạch



Phân bố nhãn sau làm sạch

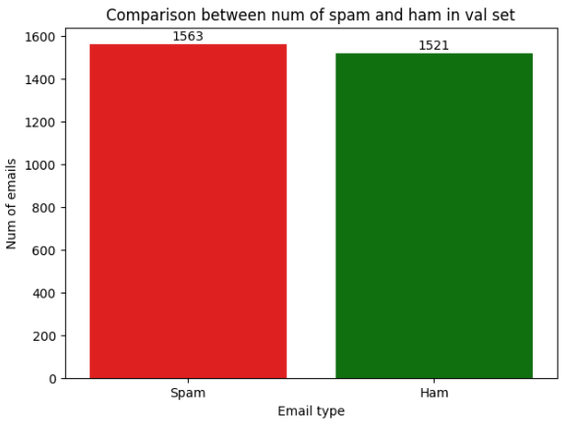
Hình 2: Phân bố nhãn trong tập huấn luyện

Tương tự như vậy, số lượng email trong tập val trước và sau khi làm sạch dữ liệu được trình bày ở bảng sau:

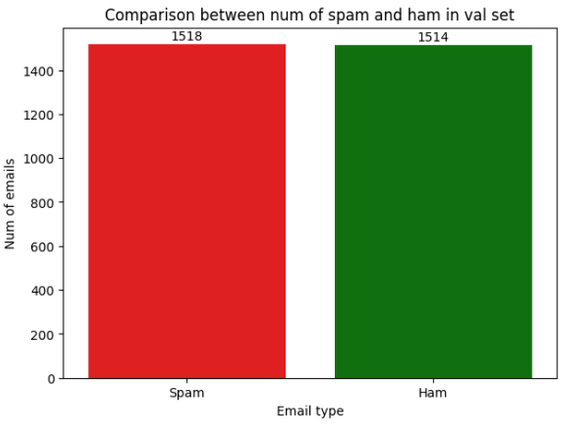
Tập val (đv: email)	**Spam**	**Ham**	**Tổng cộng**
Trước khi làm sạch	1563	1521	3084
Sau khi làm sạch	1518	1514	3032

Bảng 2: Số lượng email và phân bố nhãn trong tập kiểm trả lại

Dữ liệu email trong tập val phân bố đồng đều và cũng chứa ít mẫu dữ liệu nhiễu, do đó, chênh lệch giữa số lượng hai loại spam và ham sau khi làm sạch cũng không đáng kể.



Phân bố nhãn trước làm sạch



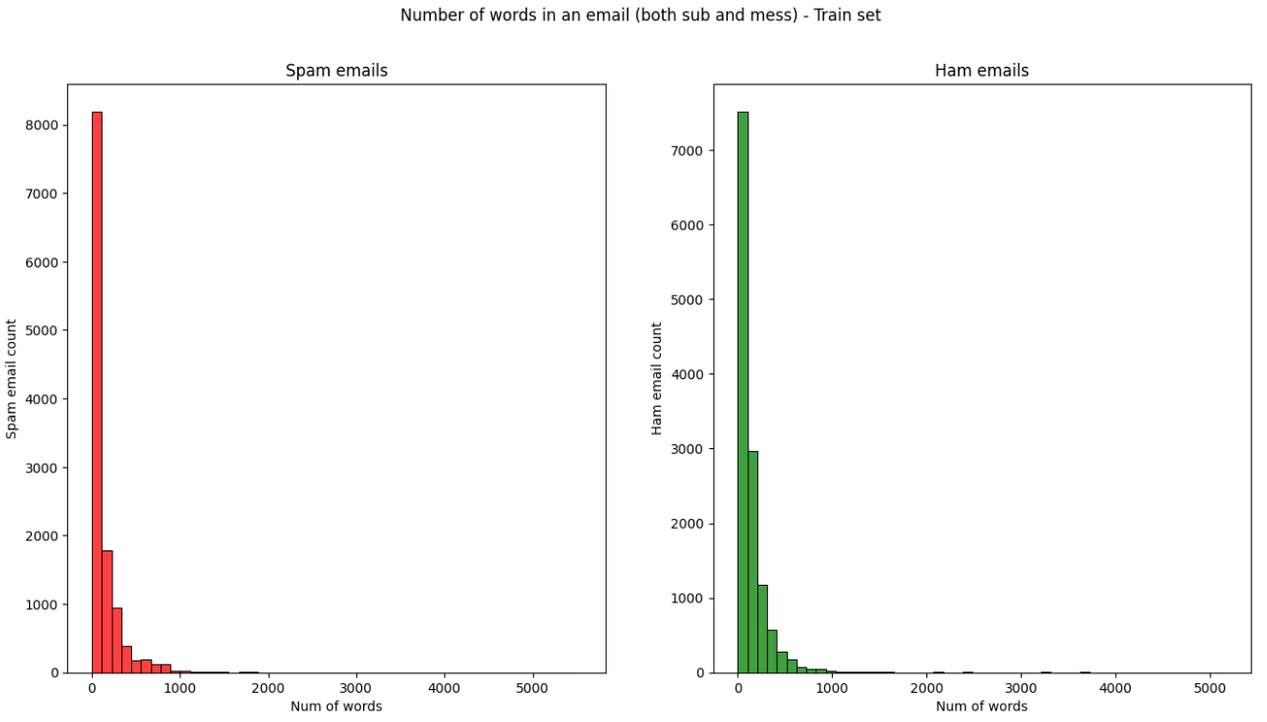
Phân bố nhãn sau làm sạch

Hình 3: Phân bố nhãn trong tập kiểm tra lại

4.2 Một số phân bố dữ liệu khác

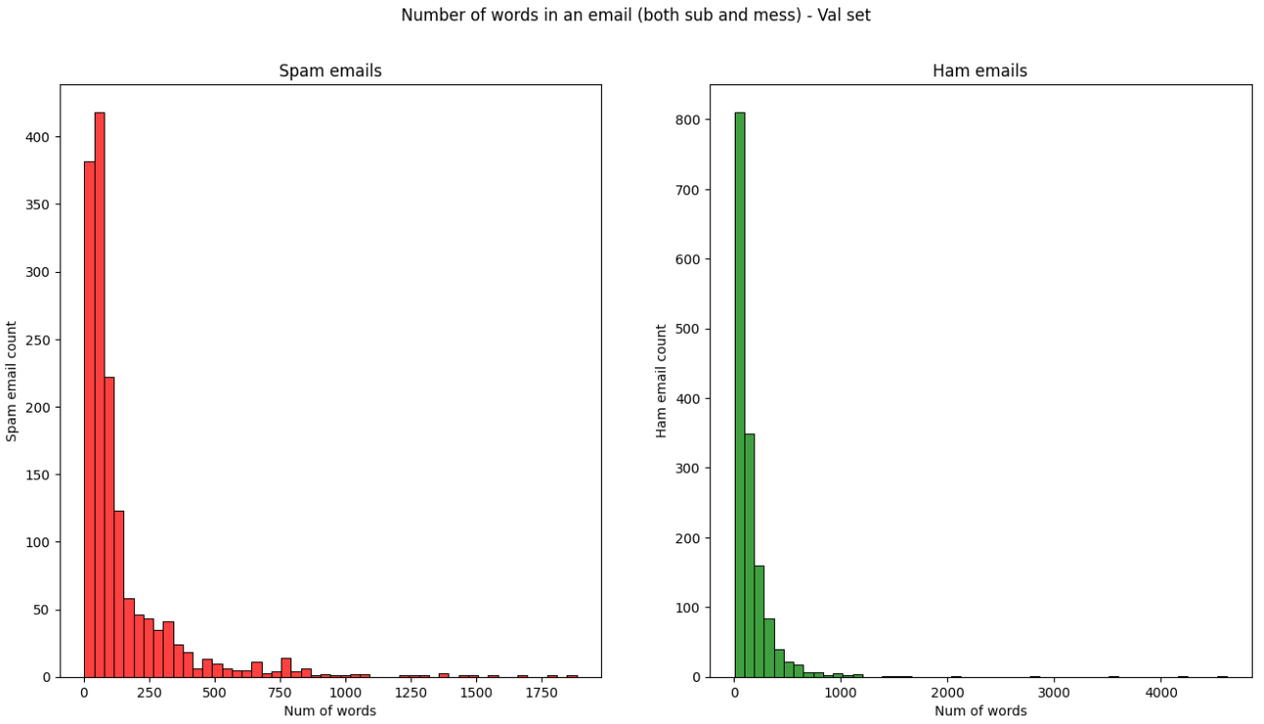
Chúng em tiến hành đếm số lượng từ trong một email nhằm mục đích trực quan hóa phân bố số lượng từ (khác nhau) trong các spam email và các ham email. Kết quả cho thấy:

- Đối với tập train: Số lượng từ trong email (cả spam và ham) đa số nằm trong khoảng 0 đến 200 từ. Trong đó, số từ ít nhất và nhiều nhất trong một spam email lần lượt là 2 và 5551 từ. Với ham email lần lượt là 2 và 5179 từ.



Hình 4: Số lượng từ trong email và số lượng email có số lượng từ đó - Tập train

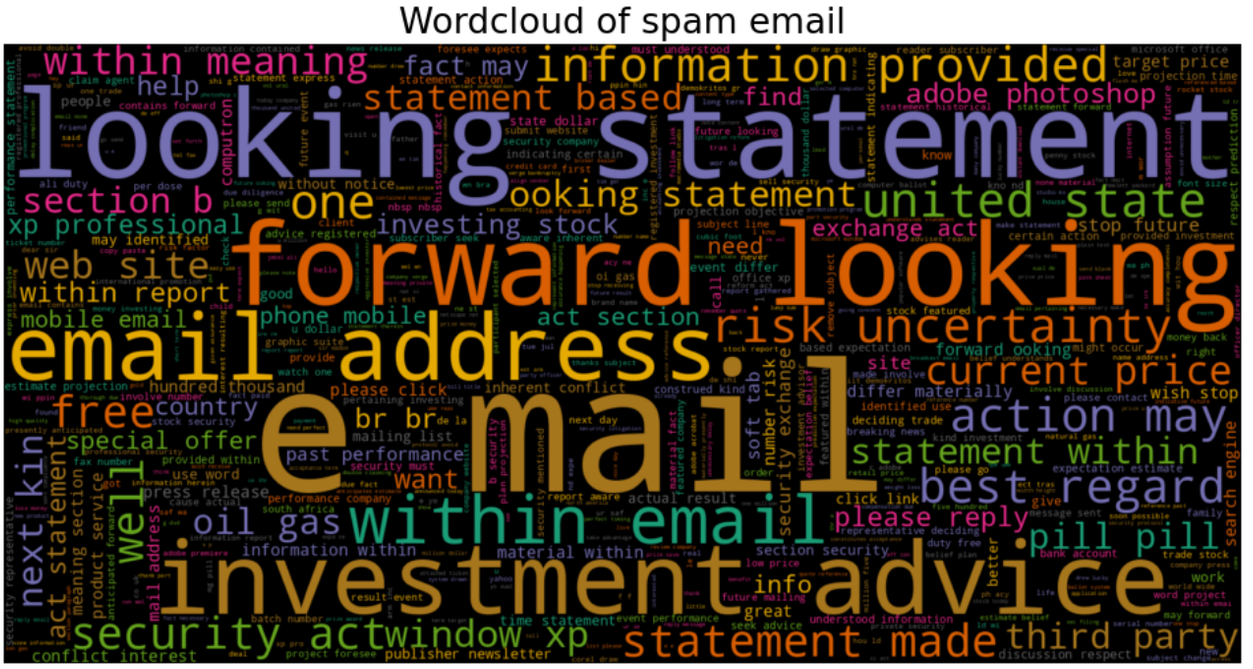
- Đối với tập val: Phân bố số lượng từ của tập val tương đối giống với phân bố của tập train, số từ trong email thường ở khoảng 0 đến 200 từ. Số từ ít nhất trong một spam email và ham email đều là 2 từ. Trong khi đó, số từ nhiều nhất trong một spam email và ham email lần lượt là 1885 từ và 4618 từ.



Hình 5: Số lượng từ trong email và số lượng email có số lượng từ đó - Tập val

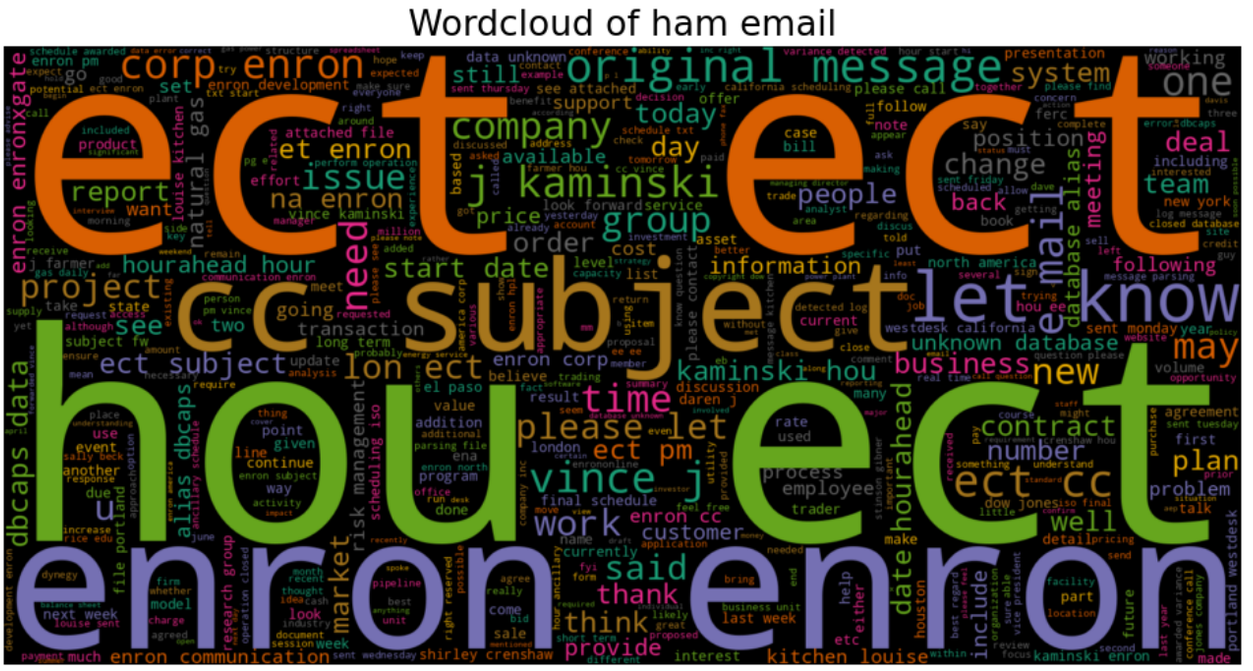
Chúng em còn thống kê xem các từ nào xuất hiện nhiều trong spam email và ham email, cho kết quả như sau:

- Trong spam email, các từ thường xuất hiện nhất (không kể đến các từ thông dụng trong email như “company”, “email”, . . .) là “information”, “please”, “price”, "money", “free”, “investment”, . . .



Hình 6: Word Cloud thể hiện tần suất xuất hiện của từ trong các spam email

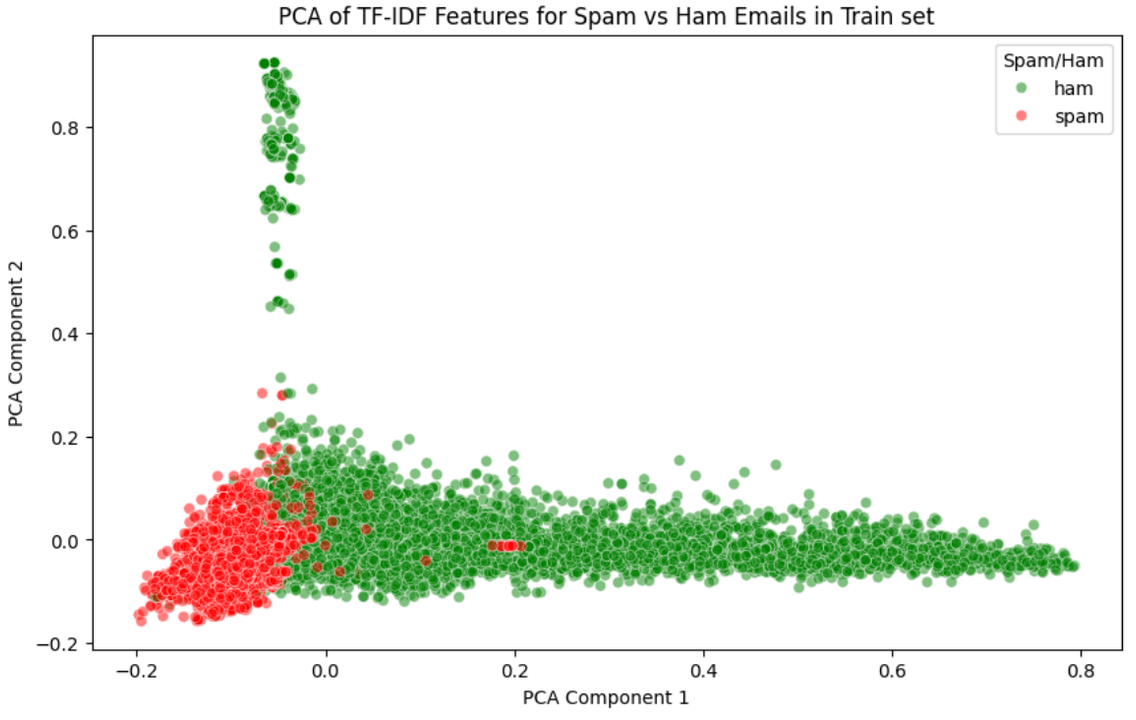
- Còn đối với ham email, các từ thường xuất hiện là “meeting”, “report”, "attached", "forwarded", "work", "management", . . .



Hình 7: Word Cloud thể hiện tần suất xuất hiện của từ trong các ham email

4.3 Đặc trưng dữ liệu

Dưới đây là minh họa cho sự phân tách dữ liệu trên tập huấn luyện, dựa trên đặc trưng trích xuất được từ phương pháp TF-IDF. Để thuận tiện cho việc minh họa này, chúng em còn sử dụng thêm kĩ thuật giảm chiều dữ liệu PCA (Principal Component Analysis) [7]:



Hình 8: Sự phân tách về nhãn của tập train dựa trên đặc trưng dữ liệu

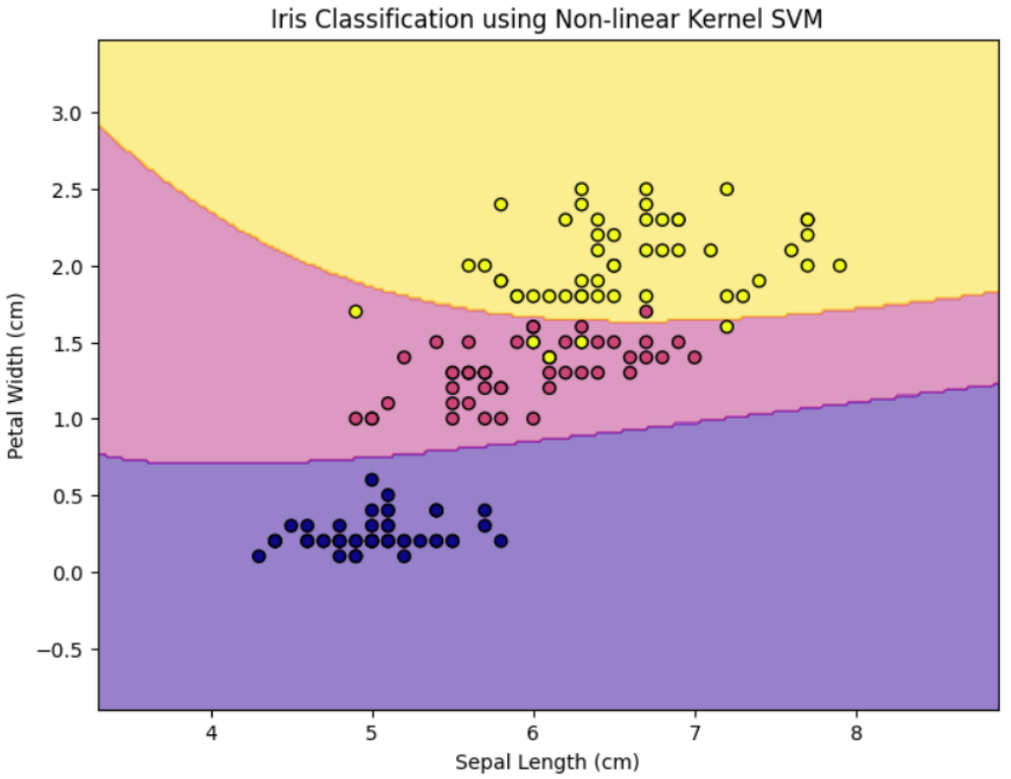
Ta có thể thấy là đặc trưng dữ liệu văn bản được trích xuất ra đã rất phân tách, chỉ có một số mẫu spam email là nằm sai ở vùng ham email.

5 Mô hình đề xuất

5.1 Các phương pháp thông dụng

5.1.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) là một trong những phương pháp học máy mạnh mẽ và phổ biến trong các bài toán phân loại. SVM tìm kiếm một siêu phẳng tối ưu để phân chia các lớp dữ liệu trong không gian nhiều chiều. Phương pháp này có khả năng xử lý tốt cả dữ liệu tuyến tính và phi tuyến, nhờ vào việc sử dụng các hàm nhân (kernel functions) [8].



Hình 9: SVM Kernel phi tuyến tính

SVM tối đa hóa khoảng cách giữa các lớp bằng cách tìm một siêu phẳng phân tách. Công thức của SVM để phân lớp hai lớp dữ liệu như sau:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

Trong đó:

- $f(\mathbf{x})$: Hàm quyết định (decision function).
- $\mathbf{w}$: Vector trọng số.
- b: Hệ số chặn (bias).

Mục tiêu của SVM là tối thiểu hóa:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

với điều kiện:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \forall i$$

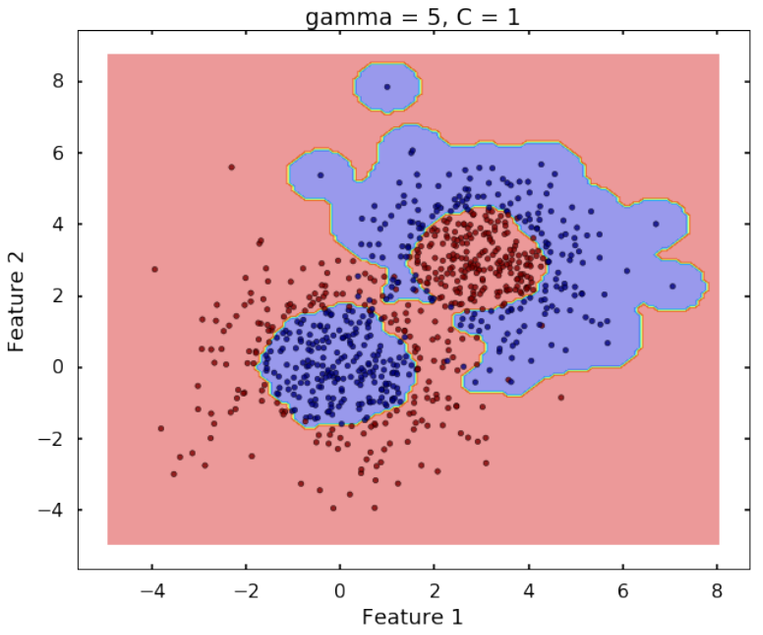
SVM được lựa chọn trong bài toán phân loại email rác vì khả năng tạo ra các quyết định phân loại rõ ràng và chính xác, đặc biệt trong các bài toán có số lượng đặc trưng lớn như phân loại văn bản. Khả năng chống overfitting cũng là một điểm mạnh của SVM, làm cho nó trở thành một lựa chọn lý tưởng cho bài toán này.

Và sau một số thử nghiệm trên mô hình cơ bản chưa được tối ưu tham số kết quả đều xấp xỉ 99%, nên nhóm quyết định sử dụng Grid Search để tối ưu tham số cho mô hình.

```
param_grid = {
    'kernel': ['linear', 'rbf'],
    'gamma': ['scale', 'auto']
}
svm = SVC()
grid_search = GridSearchCV(svm, param_grid, cv=5, n_jobs=-1, verbose=1)
```

Kết quả của quá trình tối ưu như sau:

```
Best Parameters: {'gamma': 'scale', 'kernel': 'rbf'}
```



Hình 10: Tham số gamma và kernel của mô hình

1. Gamma (γ)

   γ quyết định mức độ ảnh hưởng của một điểm dữ liệu cụ thể trong không gian đặc trưng. Khi γ lớn, mô hình sẽ cố gắng "fit" gần với từng điểm dữ liệu, dẫn đến khả năng *overfitting*, còn khi γ nhỏ, ảnh hưởng của từng điểm dữ liệu sẽ mở rộng hơn, tạo ra biên quyết định mượt hơn.

   Với tham số tối ưu:

   $$\gamma = \text{'scale'}$$

   Giá trị γ được tính toán tự động dựa trên số lượng đặc trưng và biến thiên của dữ liệu. Cụ thể, với 'scale', γ được xác định theo công thức:

   $$\gamma = \frac{1}{n_{\text{features}} \cdot \sigma^2}$$

   trong đó n_{features} là số lượng đặc trưng và σ^2 là phương sai của các đặc trưng.

2. Kernel (K)

   Kernel quyết định cách chuyển đổi dữ liệu vào không gian đặc trưng cao hơn để tìm biên quyết định tuyến tính trong không gian mới. Với **SVM**, kernel có nhiệm vụ biến đổi dữ liệu phi tuyến để mô hình hóa các quan hệ phức tạp hơn.

   Với tham số tối ưu:

   $$K(x_i, x_j) = \text{'rbf'}$$

   Kernel **RBF (Radial Basis Function)** là một kernel phi tuyến, giúp SVM tìm ra các biên quyết định phi tuyến tính. Hàm RBF có dạng:

   $$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$$

   Kernel này dựa trên khoảng cách giữa các điểm dữ liệu x_i và x_j, giúp mô hình hóa các quan hệ không tuyến tính trong dữ liệu.

Kết quả chạy model cũng tương đối khả quan, cụ thể như sau:

- Với tập dữ liệu train đã được làm sạch:

```
SVM Accuracy: 0.9906
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1521
           1       0.99      0.99      0.99      1563

    accuracy                           0.99      3084
   macro avg       0.99      0.99      0.99      3084
weighted avg       0.99      0.99      0.99      3084
```

Hình 11: Kết quả đánh giá trên tập Val

- Với tập dữ liệu train đã được làm sạch và tăng cường bằng cách hoán đổi vài cặp từ hoặc thêm ngẫu nhiên một số từ trong email ban đầu:

```
SVM Accuracy: 0.9906
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1521
           1       0.99      0.99      0.99      1563

    accuracy                           0.99      3084
   macro avg       0.99      0.99      0.99      3084
weighted avg       0.99      0.99      0.99      3084
```

Hình 12: Kết quả đánh giá trên tập Val

- Với tập dữ liệu train đã được làm sạch và lọc bớt một số dữ liệu gây nhiễu bằng kỹ thuật Isolation forest:

```
SVM Accuracy: 0.9906
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1521
           1       0.99      0.99      0.99      1563

    accuracy                           0.99      3084
   macro avg       0.99      0.99      0.99      3084
weighted avg       0.99      0.99      0.99      3084
```

Hình 13: Kết quả đánh giá trên tập Val

- Với tập dữ liệu train đã được làm sạch và sử dụng mô hình Word2Vec để lấy ra feature thay cho TF-IDF:

```
SVM Accuracy: 0.9767
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.96      0.98      1521
           1       0.97      0.99      0.98      1563

    accuracy                           0.98      3084
   macro avg       0.98      0.98      0.98      3084
weighted avg       0.98      0.98      0.98      3084
```

Hình 14: Kết quả đánh giá trên tập Val

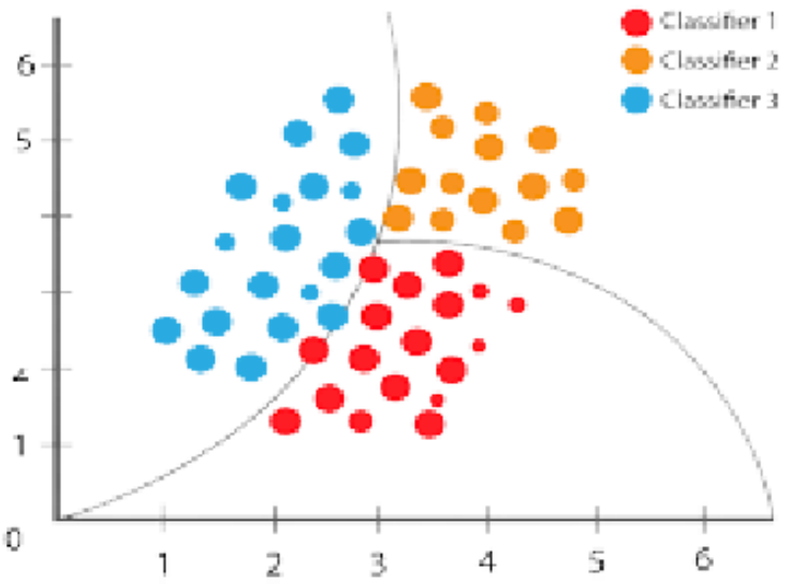
Từ kết quả mô hình SVM trên các tập dữ liệu được xử lý khác nhau, ta thấy rằng mô hình SVM đạt độ chính xác rất cao (99.06%) trên ba tập dữ liệu đã được làm sạch, tăng cường dữ liệu và loại bỏ dữ liệu nhiễu bằng Isolation Forest. Tuy nhiên, khi thay thế phương pháp trích xuất đặc trưng từ TF-IDF sang Word2Vec, độ chính xác có giảm nhẹ xuống 97.67%. Điều này cho thấy rằng với bài toán phân loại email rác, phương pháp TF-IDF vẫn là lựa chọn phù hợp hơn trong việc trích xuất đặc trưng.

Phương pháp xử lý dữ liệu	**Dữ liệu sạch**	**Dữ liệu tăng cường**	**Dữ liệu không nhiễu**	**Dữ liệu Word2Vec**
Độ chính xác (Accuracy)	**0.9906**	**0.9906**	**0.9906**	0.9767

Bảng 3: So sánh độ chính xác của mô hình SVM với các phương pháp xử lý dữ liệu khác nhau

5.1.2 Naive Bayes

Naive Bayes là một phương pháp phân loại dựa trên định lý Bayes, với giả định rằng các đặc trưng của dữ liệu là độc lập với nhau. Mặc dù giả định này không hoàn toàn chính xác trong thực tế, nhưng Naive Bayes vẫn thường cho kết quả tốt trong các bài toán phân loại văn bản, đặc biệt là trong việc phân loại thư rác [9].



Hình 15: Mô hình phân loại Naive Bayes

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

$$P(C_k|\mathbf{x}) = \frac{P(C_k)P(\mathbf{x}|C_k)}{P(\mathbf{x})}$$

Trong đó:

- $P(C_k|\mathbf{x})$: Xác suất một điểm dữ liệu $\mathbf{x}$ thuộc lớp C_k.
- $P(C_k)$: Xác suất tiên nghiệm (prior probability) của lớp C_k.
- $P(\mathbf{x}|C_k)$: Xác suất có điều kiện (likelihood), là xác suất quan sát thấy đặc trưng $\mathbf{x}$ với điều kiện thuộc lớp C_k.
- $P(\mathbf{x})$: Xác suất quan sát đặc trưng $\mathbf{x}$ trên toàn bộ các lớp.

Với giả định rằng các đặc trưng trong $\mathbf{x}$ là độc lập với nhau, xác suất có điều kiện $P(\mathbf{x}|C_k)$ có thể được tính như tích của các xác suất riêng lẻ:

$$P(\mathbf{x}|C_k) = \prod_{i=1}^{n} P(x_i|C_k)$$

Với x_i là các đặc trưng của dữ liệu $\mathbf{x}$. Cuối cùng, ta có thể dự đoán lớp C_k dựa trên xác suất lớn nhất:

$$C_{\text{predict}} = \arg\max_{C_k} P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$

Naive Bayes được lựa chọn trong nghiên cứu này vì tính đơn giản và hiệu quả của nó trong việc xử lý dữ liệu lớn. Phương pháp này cũng có thể xử lý tốt các vấn đề liên quan đến dữ liệu không cân bằng, điều này rất phù hợp với bộ dữ liệu Enron-Spam, nơi có sự chênh lệch giữa số lượng email rác và email hợp lệ (ham).

Cụ thể, kết quả chạy của mô hình như sau:

- Với tập dữ liệu train đã được làm sạch:

Naive Bayes Accuracy: 0.9802
Naive Bayes Classification Report:

	precision	recall	f1-score	support
0	0.99	0.97	0.98	1521
1	0.98	0.99	0.98	1563
accuracy			0.98	3084
macro avg	0.98	0.98	0.98	3084
weighted avg	0.98	0.98	0.98	3084

Hình 16: Kết quả đánh giá trên tập Val

- Với tập dữ liệu train đã được làm sạch và tăng cường bằng cách hoán đổi vài cặp từ hoặc thêm ngẫu nhiên một số từ trong email ban đầu:

Naive Bayes Accuracy: 0.9802
Naive Bayes Classification Report:

	precision	recall	f1-score	support
0	0.99	0.97	0.98	1521
1	0.98	0.99	0.98	1563
accuracy			0.98	3084
macro avg	0.98	0.98	0.98	3084
weighted avg	0.98	0.98	0.98	3084

Hình 17: Kết quả đánh giá trên tập Val

- Với tập dữ liệu train đã được làm sạch và lọc bớt một số dữ liệu gây nhiễu bằng kỹ thuật Isolation forest:

Naive Bayes Accuracy: 0.9802
Naive Bayes Classification Report:

	precision	recall	f1-score	support
0	0.99	0.97	0.98	1521
1	0.98	0.99	0.98	1563
accuracy			0.98	3084
macro avg	0.98	0.98	0.98	3084
weighted avg	0.98	0.98	0.98	3084

Hình 18: Kết quả đánh giá trên tập Val

- Với tập dữ liệu train đã được làm sạch và sử dụng mô hình Word2Vec để lấy ra feature thay cho TF-IDF:

```
Naive Bayes Accuracy: 0.8476
Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.84      0.85      1521
           1       0.85      0.85      0.85      1563

    accuracy                           0.85      3084
   macro avg       0.85      0.85      0.85      3084
weighted avg       0.85      0.85      0.85      3084
```

Hình 19: Kết quả đánh giá trên tập Val

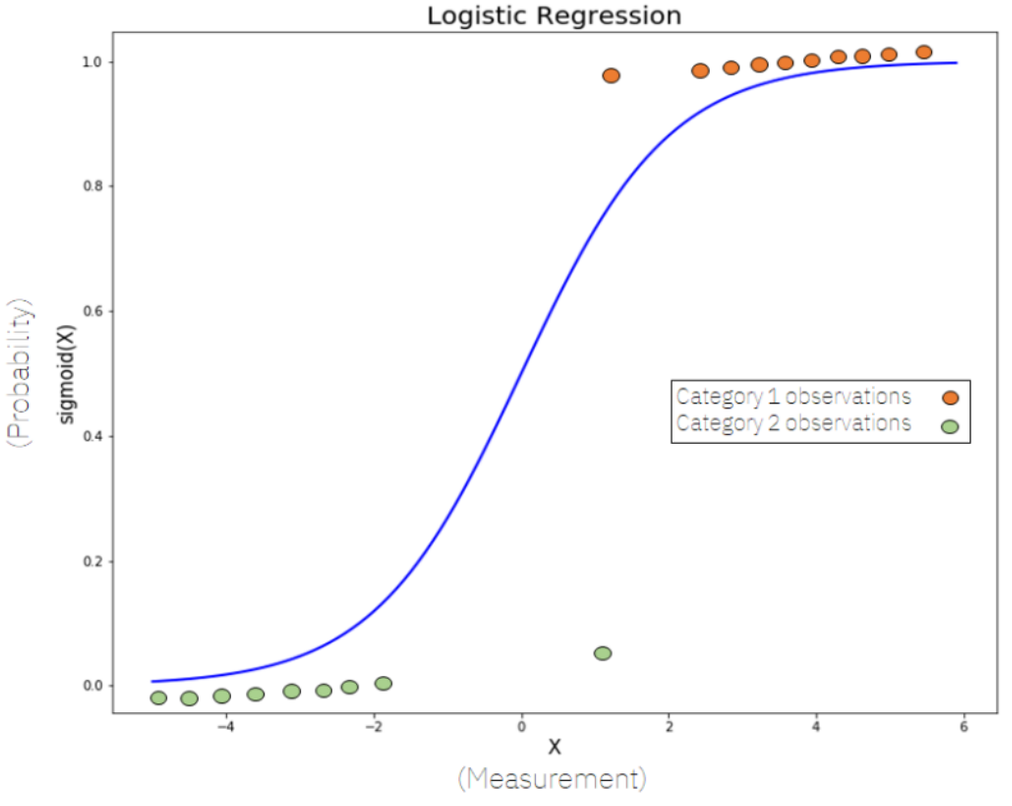
Qua các thử nghiệm với các phương pháp xử lý dữ liệu khác nhau, mô hình cho kết quả chính xác cao khi xử lý dữ liệu đã được làm sạch và tăng cường, tuy nhiên, khi sử dụng Word2Vec để thay thế cho TF-IDF, độ chính xác giảm xuống rõ rệt. Điều này cho thấy việc lựa chọn kỹ thuật xử lý đặc trưng (feature engineering) ảnh hưởng đáng kể đến kết quả của mô hình. Ngoài ra khi so sánh với các thuật toán như SVM và Logistic Regression cũng có phần yếu hơn đôi chút.

Bảng 4: So sánh độ chính xác giữa các phương pháp xử lý dữ liệu cho Naive Bayes

Phương pháp xử lý dữ liệu	**Dữ liệu sạch**	**Dữ liệu tăng cường**	**Dữ liệu không nhiễu**	**Dữ liệu Word2Vec**
Độ chính xác (Accuracy)	**0.9802**	**0.9802**	**0.9802**	0.8476

5.1.3 Logistic Regression

Logistic Regression là một phương pháp phân loại tuyến tính, được sử dụng rộng rãi trong các bài toán phân loại nhị phân. Phương pháp này tính toán xác suất của các lớp và sử dụng hàm sigmoid để đưa ra quyết định phân loại [10].



Hình 20: Logistic regression cho phân loại dữ liệu

Logistic Regression sử dụng hàm sigmoid để ánh xạ đầu ra của hàm tuyến tính về giá trị xác suất trong khoảng từ 0 đến 1. Công thức của Logistic Regression như sau:

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+b)}}$$

Trong đó:

- $P(y = 1|\mathbf{x})$: Xác suất của lớp 1 với điều kiện đầu vào $\mathbf{x}$.
- $\mathbf{w}^T\mathbf{x}$: Tích vô hướng của vector trọng số $\mathbf{w}$ và vector đầu vào $\mathbf{x}$.
- b: Hệ số chặn (bias).
- e: Hằng số Euler.

Hàm sigmoid:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

được sử dụng để ánh xạ giá trị tuyến tính về khoảng xác suất [0, 1]. Mục tiêu của Logistic Regression là tối ưu hàm mất mát:

$$\mathcal{L}(\mathbf{w}, b) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

Logistic Regression được chọn vì tính trực quan và khả năng giải thích cao của nó. Hơn nữa, Logistic Regression có thể cung cấp các trọng số cho từng đặc trưng, giúp hiểu rõ hơn về tầm quan trọng của từng yếu tố trong quyết định phân loại. Điều này rất hữu ích trong các bài toán phân loại văn bản, nơi mà việc hiểu rõ các đặc trưng là rất quan trọng.

Và sau một số thử nghiệm trên mô hình cơ bản chưa được tối ưu tham số kết quả đều xấp xỉ 99%, tốc độ train mô hình lại rất nahnh, nên nhóm quyết định sử dụng Grid Search để tối ưu tham số.

```
param_grid = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
    'solver': ['liblinear', 'lbfgs', 'saga'],
    'penalty': ['l2'],
    'max_iter': [100, 200, 300, 400]
}

grid_search = GridSearchCV(LogisticRegression(), param_grid, cv=5,
    scoring='accuracy', n_jobs=-1, verbose=1)
```

Kết quả của quá trình tối ưu như sau:

```
Best Parameters: {'C': 10, 'max_iter': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
```

1. C

   Tham số C điều chỉnh mức độ phạt (regularization) đối với các hệ số hồi quy trong mô hình. Đây là nghịch đảo của cường độ điều chuẩn. Giá trị C nhỏ hơn sẽ áp dụng nhiều điều chuẩn hơn, giúp giảm khả năng *overfitting*. Ngược lại, giá trị C lớn hơn sẽ giảm bớt điều chuẩn, cho phép mô hình "fit" sát hơn với dữ liệu huấn luyện.

2. max_iter

   max_iter là số lượng vòng lặp tối đa mà thuật toán sẽ chạy để hội tụ đến nghiệm tối ưu. Nếu mô hình không hội tụ sau số vòng lặp này, quá trình huấn luyện sẽ dừng lại.

   Với kết quả tối ưu:

3. penalty

   Tham số penalty chỉ định loại điều chuẩn được sử dụng. Điều chuẩn giúp tránh overfitting bằng cách áp dụng một hình phạt lên các hệ số hồi quy lớn.

   Với kết quả tối ưu:

   $$\text{penalty} = \text{'l2'}$$

   Ở đây, điều chuẩn L_2 (Ridge) được sử dụng, phạt theo bình phương của các hệ số hồi quy. Điều này giúp làm giảm độ lớn của các hệ số hồi quy để mô hình tổng quát hóa tốt hơn.

4. solver

   solver là thuật toán tối ưu hóa được sử dụng để tìm các hệ số hồi quy tốt nhất.

   Với kết quả tối ưu:

   $$\text{solver} = \text{'lbfgs'}$$

   lbfgs là một thuật toán tối ưu hóa dựa trên gradient (Limited-memory Broyden–Fletcher Goldfarb–Shanno). Nó phù hợp cho các vấn đề hồi quy logistic với kích thước nhỏ và trung bình, và có hiệu quả trong việc xử lý các vấn đề với điều chuẩn L_2.

Kết quả chạy model cũng tương đối khả quan, cụ thể như sau:

- Với tập dữ liệu train đã được làm sạch:

```
Logistic Regression Accuracy: 0.9896
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99      1521
           1       0.98      1.00      0.99      1563

    accuracy                           0.99      3084
   macro avg       0.99      0.99      0.99      3084
weighted avg       0.99      0.99      0.99      3084
```

Hình 21: Kết quả đánh giá trên tập Val

- Với tập dữ liệu train đã được làm sạch và tăng cường bằng cách hoán đổi vài cặp từ hoặc thêm ngẫu nhiên một số từ trong email ban đầu, kết quả tốt nhất trong các mô hình đơn lẻ:

```
Logistic Regression Accuracy: 0.9909
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      1521
           1       0.99      1.00      0.99      1563

    accuracy                           0.99      3084
   macro avg       0.99      0.99      0.99      3084
weighted avg       0.99      0.99      0.99      3084
```

Hình 22: Kết quả đánh giá trên tập Val

- Với tập dữ liệu train đã được làm sạch và lọc bớt một số dữ liệu gây nhiễu bằng kỹ thuật Isolation forest:

```
Logistic Regression Accuracy: 0.9896
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99      1521
           1       0.98      1.00      0.99      1563

    accuracy                           0.99      3084
   macro avg       0.99      0.99      0.99      3084
weighted avg       0.99      0.99      0.99      3084
```

Hình 23: Kết quả đánh giá trên tập Val

- Với tập dữ liệu train đã được làm sạch và sử dụng mô hình Word2Vec để lấy ra feature thay cho TF-IDF:

```
Logistic Regression Accuracy: 0.9643
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.96      0.96      1521
           1       0.96      0.97      0.96      1563

    accuracy                           0.96      3084
   macro avg       0.96      0.96      0.96      3084
weighted avg       0.96      0.96      0.96      3084
```

Hình 24: Kết quả đánh giá trên tập Val

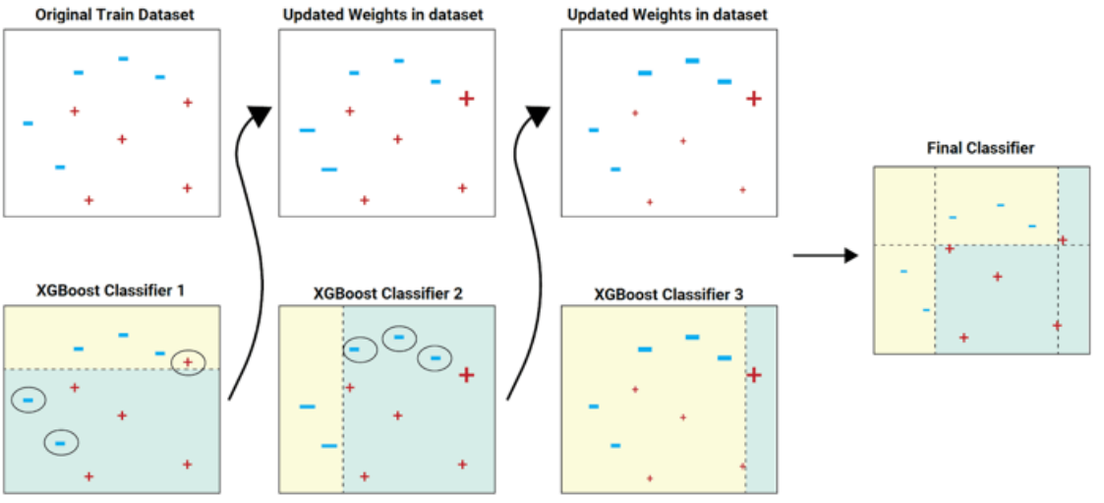
Mô hình Logistic Regression đạt được độ chính xác cao trên tất cả các phương pháp xử lý dữ liệu. Trong đó, phương pháp tăng cường dữ liệu đạt kết quả cao nhất với độ chính xác là 99.09%. Điều này cho thấy việc tăng cường dữ liệu giúp cải thiện hiệu suất của mô hình. Phương pháp Word2Vec tuy mang lại sự khác biệt trong việc xử lý đặc trưng văn bản, nhưng kết quả đạt được chỉ là 96.43%.

Bảng 5: So sánh độ chính xác của Logistic Regression trên các phương pháp xử lý dữ liệu khác nhau

Phương pháp xử lý dữ liệu	Dữ liệu sạch	Dữ liệu tăng cường	Dữ liệu không nhiễu	Dữ liệu Word2Vec
Độ chính xác (Accuracy)	0.9896	0.9909	0.9896	0.9643

5.1.4 XGBoost

XGBoost (Extreme Gradient Boosting) là một thuật toán học máy dựa trên cây quyết định, nổi bật với khả năng xử lý dữ liệu lớn và độ chính xác cao. XGBoost sử dụng kỹ thuật boosting để kết hợp nhiều cây quyết định, từ đó cải thiện khả năng phân loại [11].



Hình 25: XGBoost cho phân loại dữ liệu

Gradient Boosting là một phương pháp kết hợp nhiều mô hình yếu (weak learners) để tạo ra một mô hình mạnh hơn. Mỗi cây quyết định (decision tree) trong XGBoost được huấn luyện để sửa các lỗi mà cây trước đó mắc phải, bằng cách sử dụng gradient descent để tối thiểu hóa hàm mất mát (loss function). XGBoost không chỉ mạnh mẽ trong việc phân loại mà còn có khả năng khái quát hóa tốt khi xử lý các bài toán hồi quy.

Công thức cập nhật dự đoán trong quá trình boosting có dạng:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(x_i)$$

Trong đó:

- $\hat{y}_i^{(t)}$ là dự đoán của mô hình tại bước t,
- η là learning rate,
- $f_t(x_i)$ là cây quyết định mới được thêm vào mô hình tại bước t.

Các kỹ thuật chính giúp XGBoost đạt được hiệu suất cao bao gồm:

- **Shrinkage**: Hệ số điều chỉnh η giúp kiểm soát mức độ đóng góp của mỗi cây mới vào dự đoán cuối cùng, làm cho quá trình học dần dần hơn và ít bị overfitting.
- **Column Subsampling**: Lựa chọn ngẫu nhiên một tập con các đặc trưng để huấn luyện mỗi cây, giúp giảm phương sai và cải thiện tốc độ huấn luyện.
- **Regularization**: Các thành phần L1 và L2 trong hàm mất mát giúp kiểm soát sự phức tạp của mô hình, tránh overfitting.

Việc chọn XGBoost nhờ vào hiệu suất cao, khả năng xử lý tốt dữ liệu không cân bằng, và xử lý đặc trưng nhiều chiều. Thuật toán này tối ưu hóa quá trình học thông qua boosting,

kiểm soát overfitting với các cơ chế regularization, đồng thời cung cấp khả năng diễn giải tốt qua feature importance.

Kết quả chạy model cũng tương đối khả quan và rất ổn định mặc dù có nhiều thay đổi, cụ thể như sau:

- Với tập dữ liệu train đã được làm sạch:

```
XGBoost Accuracy: 0.9841
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1521
           1       0.97      0.99      0.98      1563

    accuracy                           0.98      3084
   macro avg       0.98      0.98      0.98      3084
weighted avg       0.98      0.98      0.98      3084
```

Hình 26: Kết quả đánh giá trên tập Val

- Với tập dữ liệu train đã được làm sạch và tăng cường bằng cách hoán đổi vài cặp từ hoặc thêm ngẫu nhiên một số từ trong email ban đầu, kết quả tốt nhất trong các mô hình đơn lẻ:

```
XGBoost Accuracy: 0.9822
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1521
           1       0.97      0.99      0.98      1563

    accuracy                           0.98      3084
   macro avg       0.98      0.98      0.98      3084
weighted avg       0.98      0.98      0.98      3084
```

Hình 27: Kết quả đánh giá trên tập Val

- Với tập dữ liệu train đã được làm sạch và lọc bớt một số dữ liệu gây nhiễu bằng kỹ thuật Isolation forest:

```
XGBoost Accuracy: 0.9841
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1521
           1       0.97      0.99      0.98      1563

    accuracy                           0.98      3084
   macro avg       0.98      0.98      0.98      3084
weighted avg       0.98      0.98      0.98      3084
```

Hình 28: Kết quả đánh giá trên tập Val

- Với tập dữ liệu train đã được làm sạch và sử dụng mô hình Word2Vec để lấy ra feature thay cho TF-IDF:

```
XGBoost Accuracy: 0.9831
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      1521
           1       0.98      0.99      0.98      1563

    accuracy                           0.98      3084
   macro avg       0.98      0.98      0.98      3084
weighted avg       0.98      0.98      0.98      3084
```

Hình 29: Kết quả đánh giá trên tập Val

Bảng 6: So sánh độ chính xác giữa các phương pháp xử lý dữ liệu cho XGBoost

Phương pháp xử lý dữ liệu	Dữ liệu sạch	Dữ liệu tăng cường	Dữ liệu không nhiễu	Dữ liệu Word2Vec
Độ chính xác (Accuracy)	**0.9841**	0.9822	**0.9841**	0.9831

Kết quả chạy XGBoost cho thấy mô hình hoạt động rất ổn định trên các bộ dữ liệu khác nhau. Mô hình đạt độ chính xác cao nhất với dữ liệu sạch và dữ liệu đã lọc nhiễu (**0.9841**). Điều này cho thấy việc loại bỏ các điểm nhiễu có thể giúp cải thiện đáng kể hiệu suất của mô hình. Tuy nhiên, dữ liệu Word2Vec mặc dù có kết quả khá tốt (**0.9831**) nhưng vẫn kém hơn so với các phương pháp khác sử dụng TF-IDF. Phương pháp tăng cường dữ liệu mang lại sự cải thiện đáng kể, nhưng không vượt qua được hiệu suất của mô hình với dữ liệu sạch. Điều này có thể chỉ ra rằng chất lượng dữ liệu ban đầu đã đủ tốt và việc tăng cường dữ liệu không mang lại nhiều lợi ích hơn trong trường hợp này.

5.1.5 Đánh giá kết quả

Có thể thấy hiệu suất của bốn mô hình khác nhau (SVM, Naive Bayes, Logistic Regression, và XGBoost) trên bốn loại dữ liệu xử lý khác nhau:

- **SVM:** Mô hình SVM đạt được độ chính xác cao nhất với kết quả 99.06% trên tất cả các loại dữ liệu, cho thấy khả năng phân loại rất mạnh mẽ của mô hình này.
- **Naive Bayes:** Độ chính xác của Naive Bayes tương đối thấp hơn so với các mô hình khác, với kết quả tốt nhất là 98.02%.
- **Logistic Regression:** Mô hình Logistic Regression cho thấy độ chính xác ấn tượng với 99.09% cho dữ liệu tăng cường, chứng tỏ rằng việc tăng cường dữ liệu có tác động tích cực đến hiệu suất mô hình. Mặc dù không đạt được độ chính xác tối đa như SVM, nhưng Logistic Regression vẫn là một phương pháp mạnh mẽ cho bài toán phân loại này.
- **XGBoost:** Đối với mô hình XGBoost, mặc dù không đạt được độ chính xác cao nhất, nhưng kết quả 98.41% với dữ liệu sạch và không nhiễu cho thấy khả năng của nó trong việc xử lý các vấn đề phức tạp với dữ liệu lớn. Đặc biệt, việc sử dụng Word2Vec cũng mang lại độ chính xác tốt 0.9831, cho thấy rằng việc trích xuất đặc trưng bằng mô hình Word2Vec có thể là một chiến lược hiệu quả cho bài toán này.

Mô hình	Dữ liệu sạch	Dữ liệu tăng cường	Dữ liệu không nhiễu	Dữ liệu Word2Vec
SVM	**0.9906**	0.9906	**0.9906**	0.9767
Naive Bayes	0.9802	0.9802	0.9802	0.8476
Logistic Regression	0.9896	**0.9909**	0.9896	0.9643
XGBoost	0.9841	0.9822	0.9841	**0.9831**

Bảng 7: Đánh giá độ chính xác của các mô hình phân loại với các phương pháp xử lý dữ liệu khác nhau

5.2 Phương pháp đề xuất

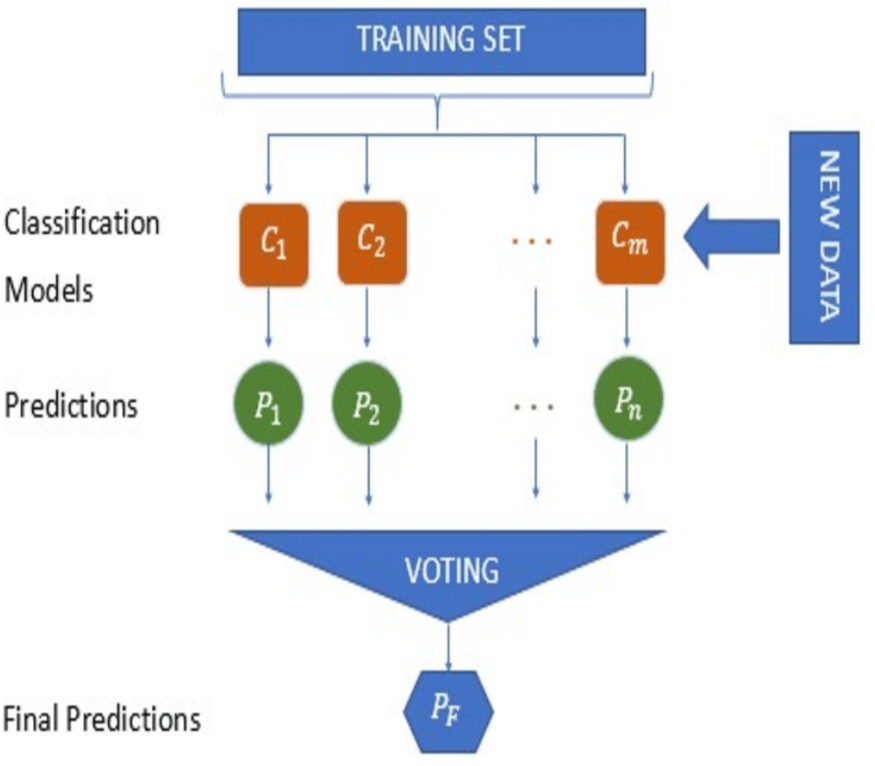
Để cải thiện độ chính xác của mô hình phân loại, chúng em đã thử nghiệm thêm hai phương pháp ensemble và như đã đề cập là Grid Search để tối ưu tham số:

5.2.1 Voting Classifier

Voting Classifier là một phương pháp ensemble, tức là kết hợp dự đoán của nhiều mô hình để cải thiện độ chính xác và độ ổn định của quá trình phân loại. Trong Voting Classifier, mỗi mô hình cơ bản sẽ đưa ra dự đoán riêng lẻ, sau đó các dự đoán này được kết hợp lại để đưa ra quyết định cuối cùng. Có hai loại voting chính:

- **Voting cứng (hard voting):** Mỗi mô hình sẽ bỏ một phiếu cho một lớp dự đoán và lớp nào có số phiếu nhiều nhất sẽ được chọn làm kết quả cuối cùng.
- **Voting mềm (soft voting):** Thay vì chỉ dựa vào dự đoán cuối cùng của mỗi mô hình, Voting mềm sẽ tính toán xác suất dự đoán cho từng lớp từ mỗi mô hình và lấy trung bình của các xác suất này để đưa ra quyết định phân loại.

Voting mềm thường cho kết quả tốt hơn, đặc biệt trong các bài toán phân loại văn bản, vì nó tận dụng được thông tin xác suất từ nhiều mô hình và giúp giảm thiểu rủi ro do quyết định sai lệch của một mô hình đơn lẻ [12].



Hình 30: Voting Classifier cho phân loại dữ liệu

Nhóm lựa chọn Voting Classifier vì nó giúp tận dụng ưu điểm của nhiều mô hình khác nhau, từ đó cải thiện hiệu suất dự đoán tổng thể. Hơn nữa, sự kết hợp này còn giảm thiểu rủi ro của việc phụ thuộc quá mức vào một mô hình đơn lẻ, giúp mô hình trở nên ổn định và linh hoạt hơn trong việc xử lý các dữ liệu không đồng nhất.

Kết quả đạt được như sau:

```
Model: Voting Classifier
Accuracy: 0.9891160949868074
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1514
           1       0.99      0.99      0.99      1518

    accuracy                           0.99      3032
   macro avg       0.99      0.99      0.99      3032
weighted avg       0.99      0.99      0.99      3032
```

Hình 31: Kết quả đánh giá trên tập Val

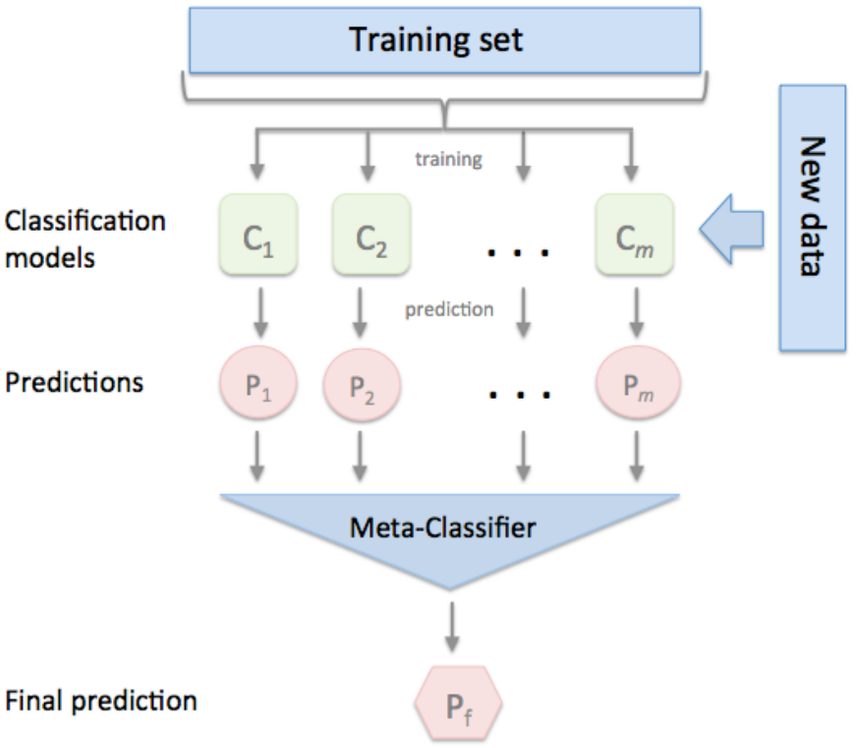
Mặc dù Voting Classifier chỉ đạt được độ chính xác là 98.91% trên tập val, thấp hơn một chút so với SVM (99.06%) và tương đương với Logistic Regression (98.96%) khi sử dụng cùng một tập dữ liệu đã được làm sạch nhưng chưa qua các kỹ thuật tăng cường. Tuy nhiên, Voting Classifier mang lại sự ổn định hơn và giảm thiểu rủi ro quá khớp (overfitting) nhờ vào việc kết hợp nhiều mô hình, giúp mô hình hoạt động tốt hơn trên dữ liệu đa dạng. Trong bối cảnh dữ liệu ngày càng phức tạp, phương pháp này cũng tạo điều kiện mở rộng, tận dụng sức mạnh từ nhiều mô hình để cải thiện khả năng dự đoán.

Dù vậy, Voting Classifier vẫn có một số điểm hạn chế. Ví dụ, nếu các mô hình cơ sở không đủ mạnh hoặc không bổ trợ lẫn nhau, việc kết hợp chúng có thể không mang lại lợi ích đáng kể. Ngoài ra, Voting Classifier thường yêu cầu nhiều tài nguyên tính toán hơn so với việc sử dụng một mô hình đơn lẻ. Do đó, trong tương lai, việc áp dụng các kỹ thuật ensemble khác như Bagging hoặc Boosting có thể mang lại kết quả tốt hơn nhờ khả năng xử lý hiệu quả các điểm yếu này.

Nhóm tin rằng kết quả này sẽ còn được cải thiện khi dữ liệu được tăng cường hoặc khi áp dụng các kỹ thuật xử lý tiên tiến hơn, từ đó giúp Voting Classifier phát huy tối đa tiềm năng của nó trong các bài toán phân loại phức tạp.

5.2.2 Stacking Classifier

Stacking Classifier là một phương pháp kết hợp các mô hình khác nhau thông qua một mô hình meta (hoặc mô hình cuối). Mô hình này học cách kết hợp các dự đoán của các mô hình cơ sở để cải thiện độ chính xác tổng thể [13].



Hình 32: Stacking Classifier cho phân loại dữ liệu

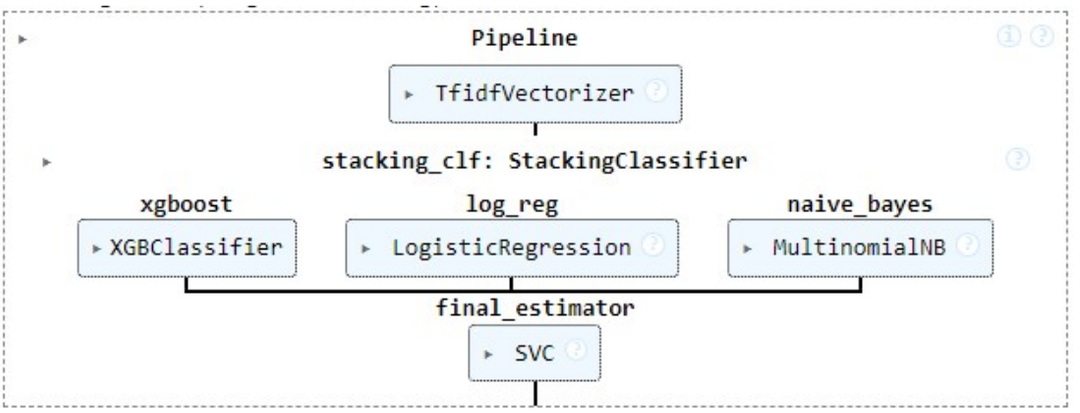
Nhóm quyết định sử dụng Stacking Classifier cho bốn mô hình: SVM, Naive Bayes, Logistic Regression và XGBoost. Mô hình meta được chọn là SVM, do đây là mô hình có hiệu suất tốt nhất trong các mô hình cơ sở. Việc sử dụng SVM làm mô hình meta cho phép tận dụng sức mạnh của các mô hình cơ sở và cải thiện kết quả phân loại.

```
Model: Stacking Classifier
Accuracy: 0.9894459102902374
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1514
           1       0.99      0.99      0.99      1518

    accuracy                           0.99      3032
   macro avg       0.99      0.99      0.99      3032
weighted avg       0.99      0.99      0.99      3032
```

Hình 33: Kết quả đánh giá trên tập Val

Kết quả mô hình Stacking Classifier cho thấy hiệu suất cao hơn so với Voting Classifier, mặc dù vẫn chưa đạt được kết quả tốt nhất như SVM. Điều này cho thấy rằng Stacking Classifier có thể phát huy tối đa tiềm năng của các mô hình cơ sở khi áp dụng vào các bài toán phân loại phức tạp. Nhóm tin rằng mô hình này sẽ thích hợp hơn khi xem xét bối cảnh dữ liệu ngày càng đa dạng, khi mà việc mở rộng và cải tiến mô hình là rất cần thiết.

Pipeline của mô hình được thể hiện dưới đây, cho thấy cách các mô hình cơ sở tương tác với mô hình meta để đưa ra dự đoán cuối cùng.

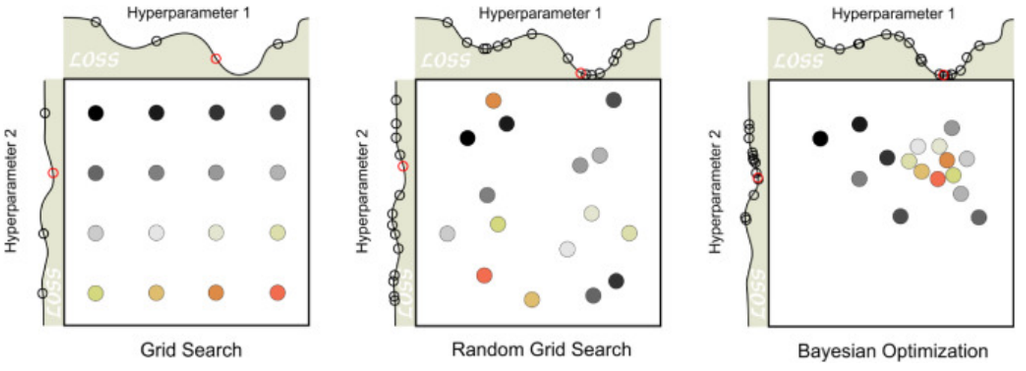


Hình 34: Pipeline của mô hình

Stacking Classifier không chỉ cải thiện độ chính xác mà còn mang lại khả năng tổng quát tốt hơn cho mô hình, giảm thiểu rủi ro quá khớp (overfitting) so với việc sử dụng các mô hình đơn lẻ. Điều này rất quan trọng trong bối cảnh mà dữ liệu có thể thay đổi theo thời gian và mô hình cần phải thích ứng với các đặc điểm mới. Trong tương lai, việc áp dụng thêm các mô hình cơ sở mạnh mẽ và kỹ thuật chọn lọc mô hình có thể mang lại sự cải thiện đáng kể cho Stacking Classifier.

5.2.3 Grid Search

Grid Search là một phương pháp tối ưu hóa hyperparameter trong quá trình huấn luyện mô hình học máy. Hyperparameters là các tham số không được học trong quá trình huấn luyện mà được định nghĩa trước khi mô hình bắt đầu học. Việc lựa chọn hyperparameters phù hợp là rất quan trọng, vì chúng có thể ảnh hưởng đáng kể đến hiệu suất và khả năng tổng quát của mô hình [14].



Hình 35: Tối ưu hóa Hyperparameter cho mô hình

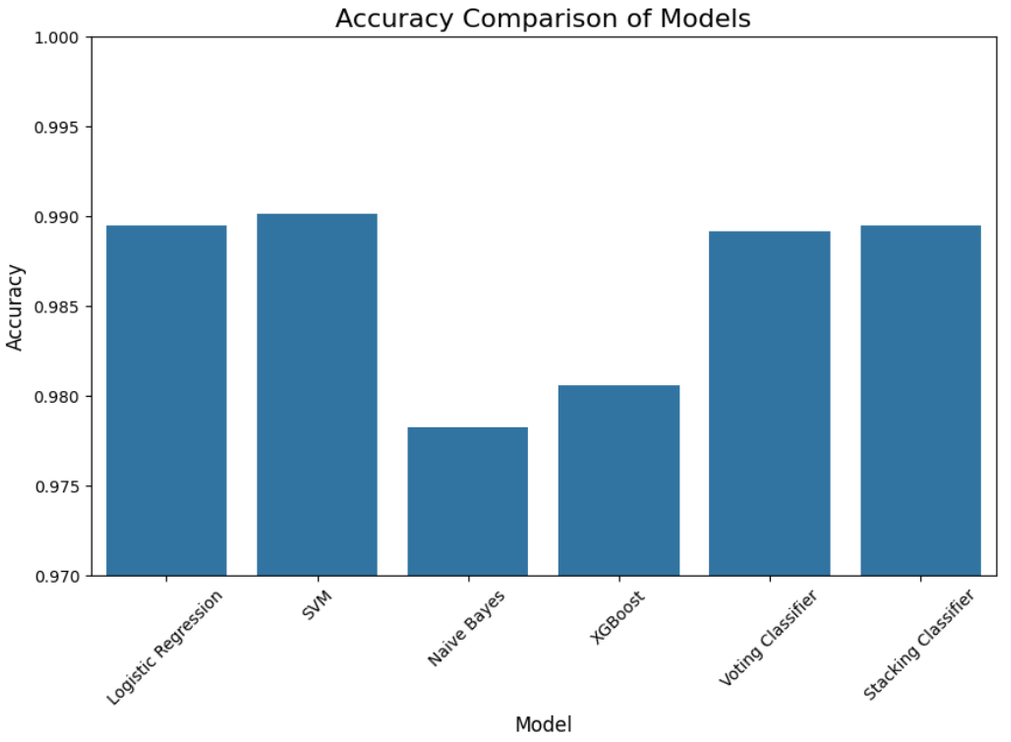
Grid Search thực hiện việc tìm kiếm các giá trị tốt nhất cho hyperparameters bằng cách kiểm tra tất cả các kết hợp có thể có trong một không gian hyperparameter đã được xác định trước. Quy trình này thường bao gồm các bước sau:

1. Xác định không gian tìm kiếm: Người dùng sẽ định nghĩa các hyperparameters cần tối ưu và các giá trị khả thi cho từng hyperparameter.
2. Tạo lưới tìm kiếm: Grid Search tạo ra một "lưới" các kết hợp của các giá trị hyperparameters từ không gian tìm kiếm đã định nghĩa.
3. Đánh giá mô hình: Mỗi kết hợp trong lưới được thử nghiệm bằng cách huấn luyện mô hình và đánh giá hiệu suất của nó trên tập dữ liệu kiểm tra hoặc tập xác thực.

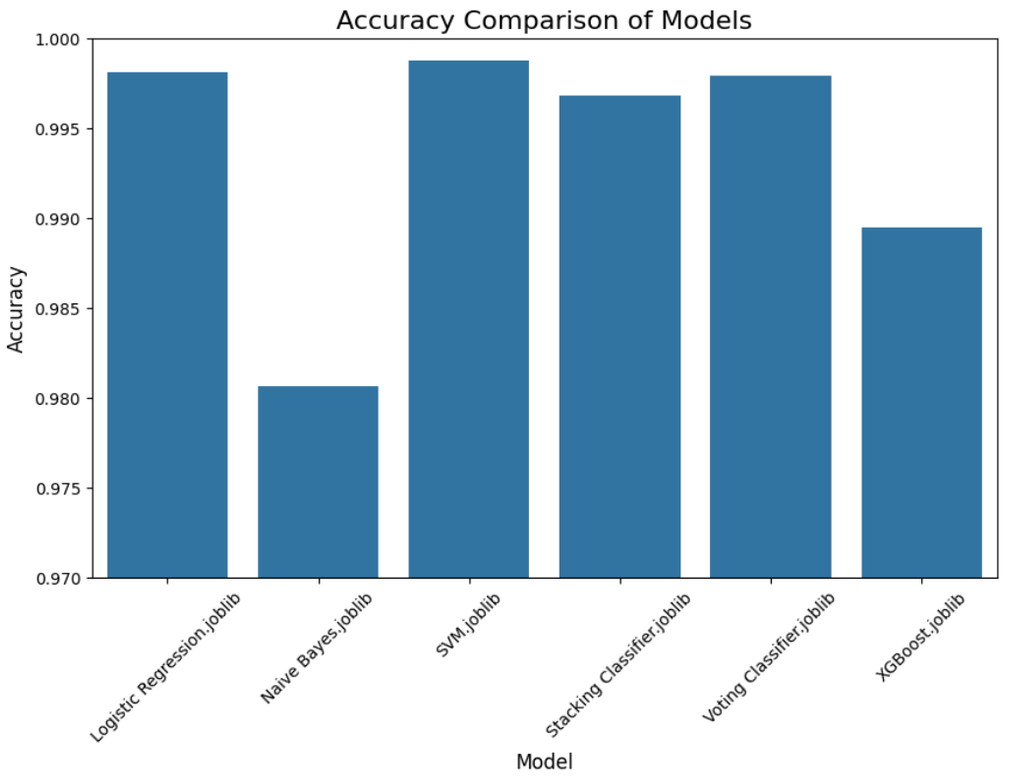
4. Chọn mô hình tốt nhất: Sau khi tất cả các kết hợp đã được kiểm tra, Grid Search sẽ chọn mô hình với hiệu suất tốt nhất (thường là độ chính xác cao nhất) dựa trên các tiêu chí đã định trước.

Phương pháp này có một số nhược điểm. Grid Search có thể tốn kém về thời gian và tài nguyên, đặc biệt khi không gian tìm kiếm rất lớn hoặc khi mô hình cần thời gian huấn luyện dài. Để khắc phục nhược điểm này, người ta có thể áp dụng các kỹ thuật như Random Search hoặc Bayesian Optimization, giúp giảm thiểu số lượng kết hợp cần thử nghiệm mà vẫn đạt được kết quả tốt.

5.3 Kết quả thử nghiệm



Hình 36: So sánh độ hiệu quả các phương pháp trên tập val



Hình 37: So sánh độ hiệu quả các phương pháp trên tập train

5.3.1 Các metric đánh giá

Các metric được sử dụng [15]:

- **Accuracy (Độ chính xác)**: Tỷ lệ số lượng dự đoán đúng trên tổng số lượng dự đoán.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

  Trong đó:

  - TP (True Positive): Số lượng dự đoán đúng cho lớp dương tính.
  - TN (True Negative): Số lượng dự đoán đúng cho lớp âm tính.
  - FP (False Positive): Số lượng dự đoán sai cho lớp dương tính.
  - FN (False Negative): Số lượng dự đoán sai cho lớp âm tính.

- **Precision (Độ chính xác dương tính)**: Tỷ lệ giữa số lượng dự đoán đúng trong các dự đoán dương tính.

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

- **Recall (Độ nhạy)**: Tỷ lệ giữa số lượng dự đoán đúng trong tổng số thực sự dương tính.

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

- **F1-Score**: Là trung bình điều hòa của Precision và Recall, F1-Score cho phép cân bằng giữa Precision và Recall, đặc biệt hữu ích khi tập dữ liệu không cân bằng.

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

6 Thử nghiệm thực tế

6.1 Đường dẫn liên quan

- Đường dẫn tới Colab: Colab Lab 1: Enron Spam
- Đường dẫn tới GitHub: Github Lab 1: Enron Spam

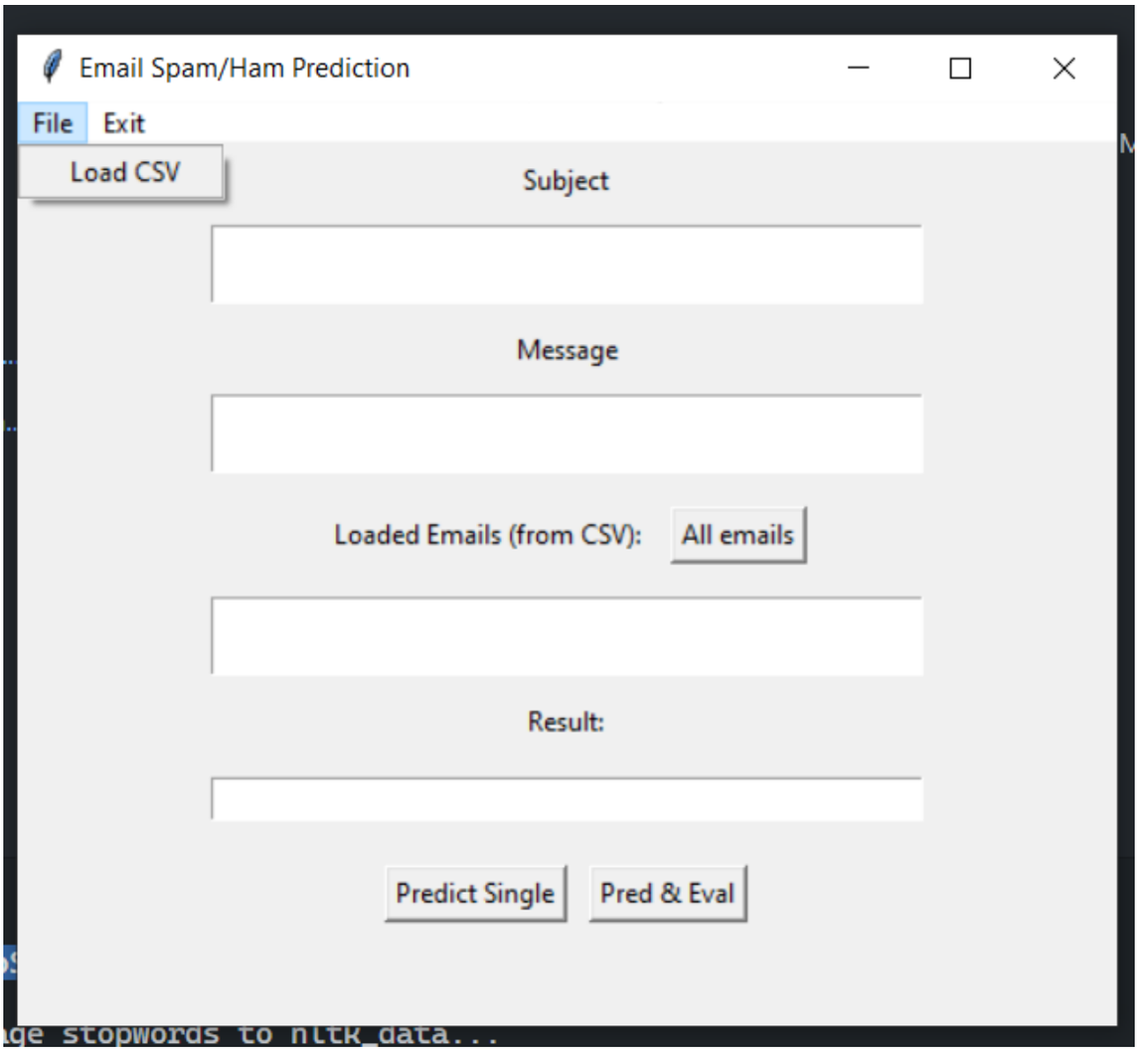
6.2 Thử nghiệm chức năng

6.2.1 Cách cài đặt và chạy chương trình

- Trước hết, vào đường dẫn tới project GitHub, tải mã nguồn của chương trình về theo như hướng dẫn.
- Ta cũng cần cài đặt các thư viện hỗ trợ (đã có hướng dẫn đi kèm project GitHub).
- Cuối cùng là chạy chương trình.

6.2.2 Giao diện chương trình

Giao diện chính của chương trình thử nghiệm mô hình của chúng em như sau:



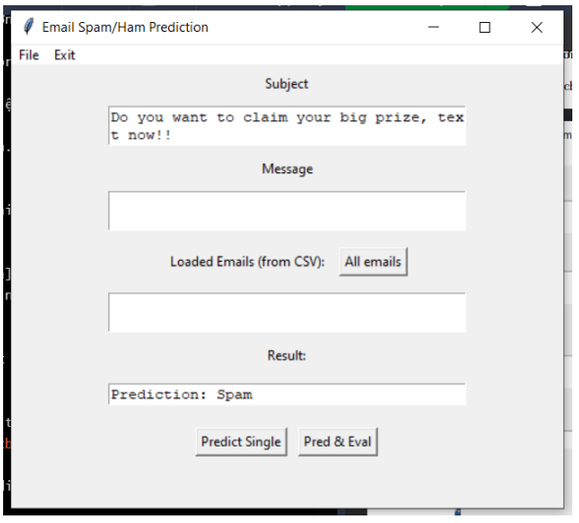
Hình 38: Giao diện chương trình thử nghiệm mô hình

Chương trình của chúng em sử dụng lập trình giao diện cơ bản của Python, với từng chức năng, chương trình sẽ gọi đến mô hình với tham số đã lưu sẵn từ trước để dự đoán trên một hoặc nhiều email.

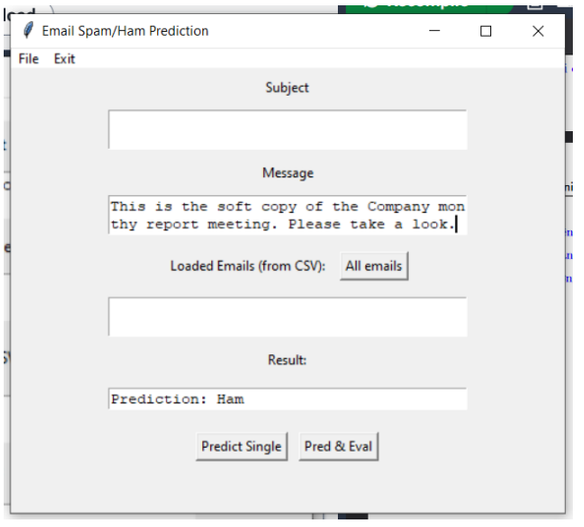
6.2.3 Chức năng 1: Nhập vào một email và dự đoán

Các bước thực hiện bao gồm:

- Ta nhập tiêu đề của email vào trong khung **Subject** và nội dung của email vào trong khung **Message** ở trên giao diện chương trình.
- Tiếp theo ta nhấn **Predict Single** để thực hiện dự đoán xem email vừa nhập vào là spam hay ham.
- Kết quả dự đoán được trả về là nhãn: *Spam* hoặc *Ham*. Và sẽ được hiển thị trong khung **Result**.



Dự đoán ra spam email



Dự đoán ra ham email

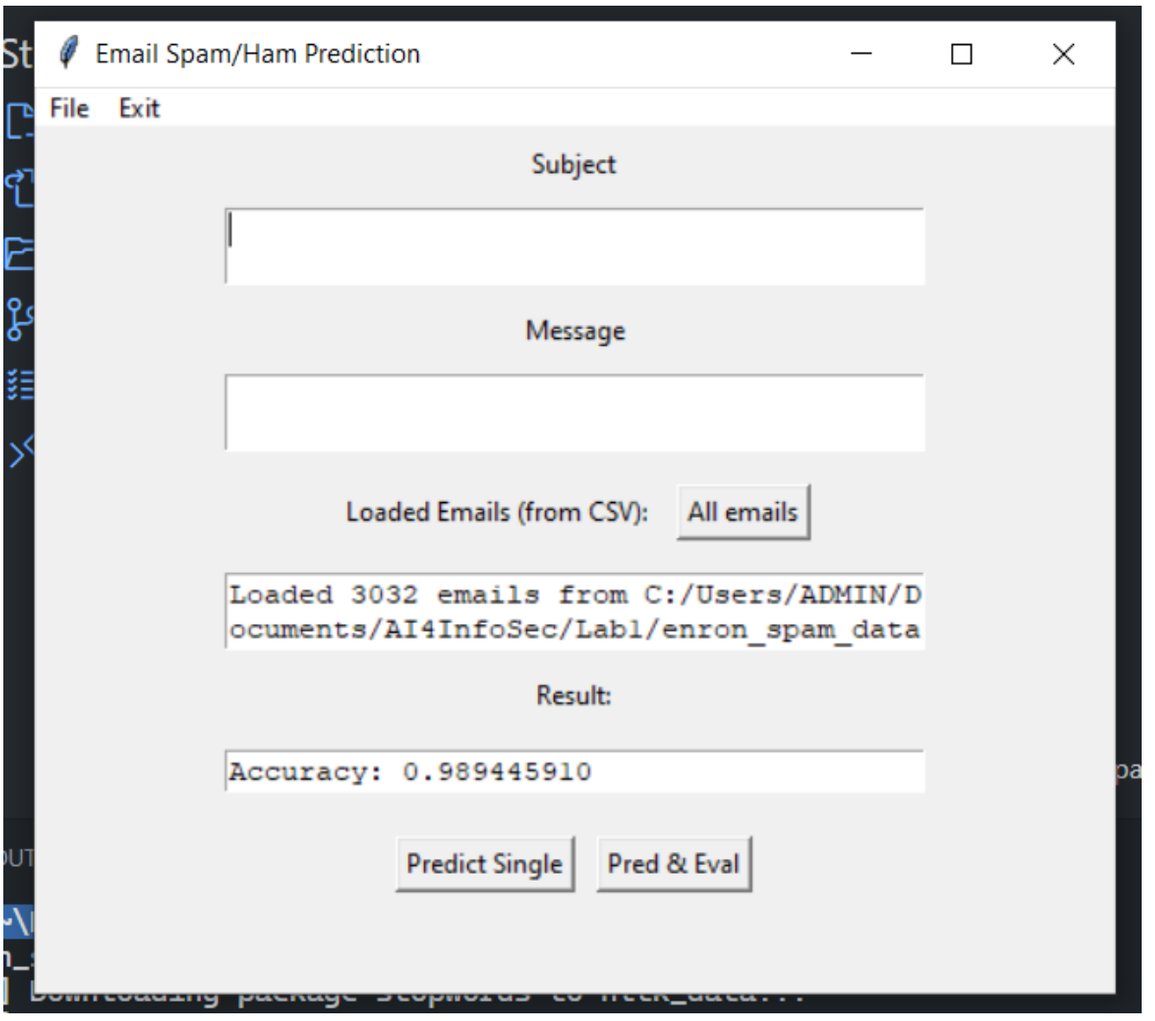
Hình 39: Chức năng dự đoán trên một email đơn

Ở đây, ta có thể nhập đầy đủ cả hai trường **Subject** và **Message** cho email hoặc cũng có thể để trống một trong hai mục. Tuy nhiên nếu bỏ trống cả hai mục thì chương trình sẽ báo lỗi.

6.2.4 Chức năng 2: Đọc file csv, thực hiện dự đoán và đánh giá

Các bước thực hiện bao gồm:

- Trên thanh công cụ, ta chọn **File**, sau đó chọn **Load CSV**.
- Một cửa số giúp ta chọn file sẽ hiện ra, ta cần tìm đến file csv đã chuẩn bị trước.
- Chương trình sẽ nạp dữ liệu từ file csv ta vừa chọn và tiến hành tiền xử lý dữ liệu, để cho mô hình dự đoán được tốt hơn.
- Ta cũng có thể xem lại nội dung của file csv bằng cách nhấn vào nút **All emails**.
- Cuối cùng, ta nhấn nút **Pred & Eval** để dự đoán cho file csv và trả về kết quả đánh giá. Kết quả đánh giá sẽ được hiển thị trong khung **Result**. Còn về kết quả dự đoán, ta có thể nhấn vào nút **All emails** một lần nữa để xem.



Hình 40: Chức năng 2: Dự đoán trên file csv và đánh giá hiệu quả

6.3 Hạn chế và hướng phát triển

Chương trình thử nghiệm mô hình của chúng em chỉ là bản thô sơ, nhằm giúp cho cho việc nhập xuất dữ liệu email của hai chức năng được dễ dàng và có tính minh họa.
Còn đối với phương pháp trích xuất đặc trưng và xây dựng mô hình:

- Chúng em còn nhiều thiếu sót như chưa thể trình bày được đa dạng các cách trích xuất đặc trưng dữ liệu hơn, ví dụ như một phương pháp cải tiến so với TF-IDF là phương pháp BM25 (Best match 25) chúng em vẫn chưa có thời gian để thử nghiệm.
- Cũng như là cần phải thử nghiệm nhiều mô hình học máy hơn, tìm ra cách ensemble mô hình tốt hơn nữa.

Tài liệu tham khảo

[1] Tiep Vu. *Embedding*. URL: https://machinelearningcoban.com/tabml_book/ch_embedding/embedding.html (visited on 10/10/2024).

[2] ProtonX. *Information Retrieval – Truy xuất thông tin*. https://drive.google.com/file/d/1urJAdygJrMdQUymmi5ANgz-B8-C1A9Od/view. Sept. 2024.

[3] scikit-learn. *TfidfTransformer*. URL: https://scikit-learn.org/1.5/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html (visited on 10/11/2024).

[4] Tiep Vu. *Word2vec*. URL: https://machinelearningcoban.com/tabml_book/ch_embedding/word2vec.html (visited on 10/14/2024).

[5] Abhijeet Talaulikar. *Spam or Ham: Word Embedding vs TF-IDF*. URL: https://www.kaggle.com/code/abhijeetstalaulikar/spam-or-ham-word-embedding-vs-tf-idf-85-recall/ (visited on 10/16/2024).

[6] Denis Cahyani and Irene Patasik. "Performance comparison of TF-IDF and Word2Vec models for emotion text classification". In: *Bulletin of Electrical Engineering and Informatics* 10 (Oct. 2021), pp. 2780–2788. DOI: 10.11591/eei.v10i5.3157.

[7] Thầy Vũ Quốc Hoàng. *Bài giảng: Các kĩ thuật phân cụm và giảm chiều*. Oct. 2024.

[8] *ML | Non-Linear SVM*. https://www.geeksforgeeks.org/ml-non-linear-svm/. Aug. 2023.

[9] Dewi Widyawati and Amaliah Faradibah. "Comparison Analysis of Classification Model Performance in Lung Cancer Prediction Using Decision Tree, Naive Bayes, and Support Vector Machine". In: *Indonesian Journal of Data and Science* 4 (July 2023), pp. 80–89. DOI: 10.56705/ijodas.v4i2.76.

[10] Sparsh Gupta. *Why Is Logistic Regression a Classification Algorithm?* https://builtin.com/machine-learning/logistic-regression-classification-algorithm. June 2022.

[11] *Introduction to XGBoost in Python*. https://blog.quantinsti.com/xgboost-python/. Dec. 2023.

[12] Navdeep Shakya, Rahul Dubey, and Laxmi Shrivastava. "Stress Detection using EEG Signal Based on Fast Walsh Hadamard transform and Voting Classifier". In: (Aug. 2021). DOI: 10.21203/rs.3.rs-782483/v1.

[13] *StackingClassifier: Simple stacking*. https://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/.

[14] Dário Passos and Puneet Mishra. "A tutorial on automatic hyperparameter tuning of deep spectral modelling for regression and classification tasks". In: *Chemometrics and Intelligent Laboratory Systems* 223 (2022), p. 104520. ISSN: 0169-7439. DOI: https://doi.org/10.1016/j.chemolab.2022.104520. URL: https://www.sciencedirect.com/science/article/pii/S0169743922000314.

[15] Trung Đức. *Đánh giá các mô hình học máy*. https://viblo.asia/p/danh-gia-cac-mo-hinh-hoc-may-RnB5pp4D5PG. Aug. 2021.