BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP.HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

-----o0o----

BÁO CÁO HỌC PHẦN: TRÍ TUỆ NHÂN TẠO

TÊN ĐỀ TÀI: TÌM HIỂU THUẬT TOÁN NAIVE BAYES (ỨNG DỤNG PHÂN LOẠI EMAIL SPAM)

NHÓM: 11

Thành phố Hồ Chí Minh, 20 tháng 12 năm 2024

BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP.HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

-----o0o----

TÊN ĐỀ TÀI: TÌM HIỂU THUẬT TOÁN NAIVE BAYES (ỨNG DỤNG PHÂN LOẠI EMAIL SPAM)

Nhóm: 11 Trưởng nhóm: Trần Công Minh Thành viên: 1. Lê Đức Trung	**Giảng viên hướng dẫn:** **Nguyễn Thị Thùy Trang**

Thành phố Hồ Chí Minh, 20 tháng 11 năm 2024

LỜI CẢM ƠN

Lời đầu tiên, em xin được gửi lời cảm ơn chân thành nhất đến thầy Trần Thanh Nhã Trong quá trình học tập và tìm hiểu môn **Trí tuệ nhân tạo**, chúng em đã nhận được rất nhiều sự quan tâm, giúp đỡ, hướng dẫn tâm huyết và tận tình của thầy. Thầy đã giúp chúng em tích lũy thêm nhiều kiến thức về môn học này để có thể hoàn thành được bài tiểu luận về đề tài: **Tìm hiểu thuật toán Naive Bayes (ứng dụng phân loại email spam)**

Trong quá trình làm bài chắc chắn khó tránh khỏi những thiếu sót. Do đó, chúng em kính mong nhận được những lời góp ý của thầy để bài tiểu luận của chúng em ngày càng hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

BẢNG PHÂN CÔNG VÀ ĐÁNH GIÁ

MSSV	Họ tên	Đóng góp tỷ lệ %	Nhiệm vụ được phân công	Mức độ hoàn thành công việc được phân công
2001222641	Trần Công Minh	100	- Power Point thuật toán - Báo cáo (phần 1,2,3 + format) - Ứng dụng (Các form chính)	Hoàn thành tốt, đúng hạn
2001225676	Lê Đức Trung	100	- Dữ liệu huấn luyện - Báo cáo (phần 4,5,6) - Ứng dụng (phần Gửi email)	Hoàn thành tốt, đúng hạn

MỤC LỤC

LỜI MỞ ĐẦU

Trong thời đại công nghệ số hiện nay, email đã trở thành một phương tiện giao tiếp không thể thiếu trong cuộc sống hàng ngày và trong kinh doanh. Tuy nhiên, cùng với sự gia tăng sử dụng email, vấn đề spam email cũng ngày càng trở nên nghiêm trọng. Spam email không chỉ làm giảm hiệu quả của việc giao tiếp mà còn có thể chứa các mối đe dọa bảo mật tiềm ẩn, như virus hoặc lừa đảo trực tuyến. Do đó, việc phát triển các phương pháp phân loại email nhằm xác định và loại bỏ spam trở nên vô cùng cần thiết.

Thuật toán **Naive Bayes** là một trong những phương pháp được sử dụng rộng rãi trong lĩnh vực phân loại văn bản, đặc biệt là trong việc phân loại email spam. Với cơ sở lý thuyết dựa trên định lý Bayes và giả định độc lập giữa các đặc trưng, Naive Bayes cung cấp một cách tiếp cận đơn giản nhưng hiệu quả để phân loại dữ liệu. Sự dễ dàng trong việc triển khai và khả năng xử lý lượng dữ liệu lớn đã khiến cho **Naive Bayes** trở thành một trong những lựa chọn phổ biến trong việc phát hiện spam.

Bài báo cáo này sẽ đi vào tìm hiểu chi tiết về thuật toán Naive Bayes, cách thức hoạt động của nó, cũng như những ứng dụng thực tiễn trong việc phân loại email spam. Bên cạnh đó, chúng tôi sẽ thảo luận về những ưu điểm và nhược điểm của phương pháp này, đồng thời so sánh với các thuật toán phân loại khác để làm nổi bật tính hiệu quả của **Naive Bayes** trong lĩnh vực này.

PHẦN 1. NỘI DUNG

1. Giới thiệu về thuật toán Naive Bayes

1.1. Đặt vấn đề

Trong thời đại công nghệ thông tin hiện nay, email đã trở thành một công cụ giao tiếp không thể thiếu trong cuộc sống hàng ngày cũng như trong kinh doanh. Tuy nhiên, sự gia tăng nhanh chóng của các email spam (thư rác) đã trở thành một vấn đề lớn, làm ảnh hưởng đến hiệu suất làm việc và chất lượng thông tin. Email spam không chỉ gây rối cho người dùng mà còn có thể chứa nội dung độc hại hoặc lừa đảo. Do đó, việc phát triển các hệ thống phân loại email tự động để phân biệt giữa email hợp lệ và email spam trở nên cần thiết. Thuật toán Naive Bayes là một trong những phương pháp phổ biến để giải quyết bài toán này nhờ vào tính đơn giản, hiệu quả và khả năng xử lý nhanh chóng.

1.2. Mục tiêu nghiên cứu

Mục tiêu của nghiên cứu này là áp dụng thuật toán Naive Bayes để phân loại email thành hai loại: spam và không spam (ham). Cụ thể, báo cáo sẽ:

- Trình bày nguyên lý hoạt động của thuật toán Naive Bayes.
- Mô tả quy trình thu thập và tiền xử lý dữ liệu email.
- Triển khai thuật toán Naive Bayes trong C# để xây dựng một bộ phân loại email.
- Đánh giá hiệu suất của bộ phân loại dựa trên các chỉ số như độ chính xác, độ nhạy, và độ đặc hiệu.

1.3. Ý nghĩa của nghiên cứu

Nghiên cứu này không chỉ giúp nâng cao khả năng phân loại email spam mà còn đóng góp vào lĩnh vực học máy và xử lý ngôn ngữ tự nhiên. Bằng việc ứng dụng Naive Bayes – một trong những thuật toán phân loại đơn giản nhưng hiệu quả, nghiên cứu có thể làm rõ cách thức hoạt động của thuật toán này trong thực tế và cung cấp những kiến thức hữu ích cho các lập trình viên và nhà nghiên cứu trong lĩnh vực tương tự.

Hơn nữa, việc phát triển một hệ thống phân loại spam hiệu quả có thể giúp giảm thiểu số lượng thư rác mà người dùng phải đối mặt hàng ngày, từ đó nâng cao năng suất làm việc và trải nghiệm người dùng trong việc sử dụng email.

2. Cơ sở lí thuyết

2.1. Giới thiệu về thuật toán Naive Bayes

Thuật toán Naive Bayes là một trong những phương pháp phân loại xác suất dựa trên Định lý Bayes, được ứng dụng rộng rãi trong nhiều bài toán phân loại văn bản, đặc biệt là trong lĩnh vực phát hiện spam. Đây là thuật toán có tính đơn giản và hiệu quả cao, đặc biệt phù hợp với dữ liệu lớn và có nhiều đặc điểm độc lập.

2.1.1. Nguyên tắc hoạt động của Naive Bayes

Naive Bayes Classifiers (NBC) là một trong những thuật toán tiêu biểu cho bài toán phân lớp dựa trên lý thuyết xác suất áp dụng định lý Bayes. Định lý Bayes cho phép chúng ta có thể tính toán một xác suất chưa biết dựa vào các xác suất có điều kiện khác. Với công thức tổng quát tính xác suất của biến cố A với điều kiện biến cố B_k xảy ra trước (hay được gọi là xác suất hậu nghiệm): Với P(A) > 0 và $\{B_1,B_2,\ldots, B_n\}$ là một hệ đầy đủ các biến cố thỏa mãn tổng xác suất của hệ bằng 1 ($\sum_{k=1}^{n} P(B_k) = 1$) và từng đôi một xung khắc ($P(B_i \cap B_j) = 0$). Khi đó ta có:

$$p(B_k|A) = \frac{p(A|B_k)p(B_k)}{p(A)} = \frac{p(A|B_k)p(B_k)}{\sum_{i=1}^{n} p(A|B_i)p(B_i)} \qquad (1)$$

Bộ phân lớp Naive bayes hoạt động như sau:

- Gọi D là tập dữ liệu huấn luyện, trong đó mỗi phần tử dữ liệu A chứa n giá trị thuộc tính B_1, B_2,...,B_n được biểu diễn bằng một vector n thành phần $\{x_1,x_2,...,x_n\}$
- Giả sử có m lớp C_1, C_2,..,C_m. Cho một phần tử dữ liệu A, bộ phân lớp sẽ gán nhãn cho A là lớp có xác suất hậu nghiệm lớn nhất. Cụ thể, bộ phân lớp Bayes sẽ dự đoán A thuộc vào lớp C_i nếu và chỉ nếu:

$$P(C_i|A) > P(C_j|A) \quad (i \le j, j \le m \; i \neq j) \qquad (2)$$

Giá trị này sẽ tính dựa trên định lý Bayes.

- Để tìm xác suất lớn nhất, ta nhận thấy các giá trị P(A) là giống nhau với mọi lớp nên không cần tính. Do đó ta chỉ cần tìm giá trị lớn nhất của $P(A|C_i) * P(C_i)$. Chú ý rằng $P(C_i)$ được ước lượng bằng $|D_i|/|D|$, trong đó D_i là tập các phần tử dữ liệu thuộc lớp C_i. Nếu xác suất tiền nghiệm $P(C_i)$ cũng không xác định được thì ta coi chúng bằng nhau $P(C_1) = P(C_2) = ... = P(C_m)$, khi đó ta chỉ cần tìm giá trị $P(A| C_i)$ lớn nhất.
- Khi số lượng các thuộc tính mô tả dữ liệu là lớn thì chi phí tính toán P(A|Ci) là rất lớn, do đó có thể giảm độ phức tạp của thuật toán Naive Bayes nếu giả thiết các thuộc tính độc lập nhau. Khi đó ta có thể tính:

$$p(A|C_i) = p(x_1|C_i) \dots p(x_n|C_i) \qquad (3)$$

NBC có thể hoạt động với các vector đặc trưng mà một phần là liên tục (sử dụng Gaussian Naive Bayes), phần còn lại ở dạng rời rạc (sử dụng Multinomial hoặc Bernoulli). Trong phần thực nghiệm, tác giả sử dụng MultinomialNB để xây dựng mô hình. Mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển. Giá trị của thành phần thứ i trong mỗi vector chính là số lần từ thứ i xuất hiện trong văn bản đó.

Khi đó, $P(xi |C_j)$ tỉ lệ với tần suất từ thứ i (hay thuộc tính thứ i cho trường hợp tổng quát) xuất hiện trong các văn bản của lớp Cj. Giá trị này có thể được tính bằng cách

$$P(x_i|C_j) = \frac{N_{ci}}{N_c} \qquad (4)$$

Trong đó:

- N_{ci} là tổng số lần từ thứ i xuất hiện trong các văn bản của lớp C_j, nó được tính bằng tổng của tất cả các thành phần thứ i của các vector thuộc tính ứng với lớp C_j
- N_c là tổng số từ (kể cả lặp) xuất hiện trong lớp C_j. Nói cách khác, nó bằng tổng độ dài của toàn bộ các văn bản thuộc vào lớp C_j.

Cách tính này có một hạn chế là nếu có một từ mới chưa bao giờ xuất hiện trong lớp C_j thì biểu thức (1) sẽ bằng 0, điều này dẫn $P(A|C_i) = 0$ bất kể các giá trị còn lại có lớn thế nào. Việc này sẽ dẫn đến kết quả không chính xác. Để giải quyết việc này, một kỹ thuật được gọi là Laplace smoothing được áp dụng như trong biểu thức (5):

$$P(x_i|C_j) = \frac{N_{ci}+\alpha}{N_c+\alpha} \qquad (5)$$

Với α là một số dương, thường bằng 1, để tránh trường hợp tử số bằng 0. Mẫu số được cộng với dα để đảm bảo tổng xác suất $\sum_{i=1}^{d} P(x_i|Cj) = 1$

2.1.2. Các loại Naive Bayes

Thuật toán Naive Bayes có một số biến thể chính, bao gồm:

Loại Naive Bayes	Mô Tả	Cách Sử Dụng
Gaussian Naive Bayes	Sử dụng phân phối Gaussian (chuẩn) để mô hình hóa các đặc trưng liên tục. Thích hợp cho dữ liệu có các đặc trưng liên tục.	Dữ liệu có đặc trưng liên tục
Multinomial Naive Bayes	Thích hợp cho dữ liệu phân loại văn bản và có thể xử lý các đặc trưng rời rạc (ví dụ: số lần xuất hiện của từ trong một tài liệu).	Phân loại văn bản
Bernoulli Naive Bayes	Dùng cho các biến nhị phân, mô hình hóa sự xuất hiện hoặc không xuất hiện của các đặc trưng. Phù hợp cho dữ liệu nhị phân hoặc "có/không".	Dữ liệu nhị phân
Complement Naive Bayes	Một biến thể của Multinomial Naive Bayes, được thiết kế để cải thiện độ chính xác cho các tập dữ liệu không cân bằng.	Dữ liệu không cân bằng
Categorical Naive Bayes	Sử dụng cho dữ liệu phân loại có đặc trưng rời rạc. Mô hình hóa xác suất cho các đặc trưng phân loại.	Dữ liệu phân loại

Bảng 1. Bảng các loại Naïve Bayes

2.2. Phân loại email spam

Spam là các email không mong muốn, thường chứa quảng cáo hoặc nội dung không hữu ích cho người nhận. Đối với người dùng, spam có thể gây mất thời gian và làm phiền, đôi khi còn chứa các phần mềm độc hại hoặc các liên kết lừa đảo.

2.2.1. Định nghĩa spam

Spam, hay còn gọi là thư rác, là các email được gửi hàng loạt mà không được sự cho phép của người nhận. Các email này thường chứa quảng cáo, nội dung không liên quan hoặc thậm chí có thể là các liên kết độc hại nhằm lừa người dùng truy cập vào các trang web giả mạo.

2.2.2. Tại sao cần phân loại spam?

Việc phân loại spam mang lại nhiều lợi ích thiết thực:

- Tăng hiệu suất làm việc: Người dùng không phải mất thời gian lọc hoặc xóa các email không mong muốn.
- Bảo vệ an ninh: Giảm nguy cơ truy cập vào các liên kết độc hại hoặc tải xuống các tệp đính kèm chứa phần mềm độc hại.
- Cải thiện trải nghiệm người dùng: Hộp thư đến của người dùng sẽ chứa những email có giá trị và cần thiết, giúp họ tập trung vào những thông tin quan trọng.

Phân loại spam là một bài toán phân loại nhị phân điển hình, và Naive Bayes là một phương pháp phù hợp để giải quyết bài toán này nhờ khả năng xử lý văn bản hiệu quả và tốc độ tính toán nhanh chóng. Trong nghiên cứu này, thuật toán Naive Bayes sẽ được triển khai để tự động nhận diện và phân loại các email thành hai nhóm: spam và không spam, giúp người dùng tiết kiệm thời gian và tăng cường tính bảo mật cho hệ thống email của họ.

3. Ứng dụng của Naive Bayes trong phân loại email spam

3.1. Quy trình phân loại email

Quy trình phân loại email spam bằng thuật toán Naive Bayes bao gồm các bước chính như sau:

3.1.1. Thu thập dữ liệu

Để xây dựng mô hình phân loại spam, bước đầu tiên là thu thập dữ liệu email. Dữ liệu này bao gồm các email đã được phân loại thành hai loại chính: spam và ham (không phải spam). Các nguồn dữ liệu bao gồm:

- Email thực tế từ người dùng (được phân loại bởi người dùng).
- Tập dữ liệu đào tạo (email_dataset)
- Sử dụng API như Gmail API để thu thập email từ tài khoản Gmail.

Trong mã code đã cung cấp, việc thu thập email thực hiện thông qua các phương thức của ***GmailServiceHelper***, nơi chúng ta có thể lấy danh sách các email từ tài khoản Gmail.

3.1.2. Tiền xử lý dữ liệu

Sau khi thu thập dữ liệu, bước tiếp theo là tiền xử lý dữ liệu để chuẩn bị cho mô hình. Việc này bao gồm:

- Làm sạch dữ liệu: Loại bỏ các ký tự không cần thiết, mã hóa không hợp lệ, và xử lý các trường hợp email rỗng.
- Chuyển đổi định dạng: Chuyển đổi tất cả văn bản thành chữ thường để giảm thiểu độ phức tạp.
- Tách từ: Tách nội dung email thành các từ riêng lẻ để dễ dàng tính toán tần suất xuất hiện. Mã code sử dụng phương thức ***Split*** để thực hiện điều này.

```
private static void AddToCount(Dictionary<string, int> countDict, string content)
{
    var words = content.ToLower().Split(new[] { ' ', '.', ',', '!', '?' }, StringSplitOptions.RemoveEmptyEntries);
    foreach (var word in words)
    {
        if (countDict.ContainsKey(word))
            countDict[word]++;
        else
            countDict[word] = 1;
    }
}
```

Hình 1.Hình ảnh tiền xử lí dữ liệu

3.1.3. Huấn luyện mô hình

Trong bước này, mô hình Naive Bayes được huấn luyện với tập dữ liệu đã được tiền xử lý. Mô hình sẽ tính toán xác suất cho từng từ xuất hiện trong các email spam và không spam. Điều này được thực hiện thông qua phương thức **Train** trong mã code, nơi chúng ta lưu trữ tần suất xuất hiện của từng từ trong hai từ điển riêng biệt: ***_spamCount*** và ***_hamCount***.

```
1 reference
public void Train(Email email)
{
    _emails.Add(email);
    if (email.IsSpam)
    {
        _totalSpam++;
        AddToCount(_spamCount, email.Content);
        _totalSpamWords += email.Content.Split(new[] { ' ', '.', ',', '!', '?' }, StringSplitOptions.RemoveEmptyEntries).Length;
    }
    else
    {
        _totalHam++;
        AddToCount(_hamCount, email.Content);
        _totalHamWords += email.Content.Split(new[] { ' ', '.', ',', '!', '?' }, StringSplitOptions.RemoveEmptyEntries).Length;
    }
}
```

Hình 2. Huấn luyện mô hình

3.1.4. Phân loại email mới

Khi mô hình đã được huấn luyện, nó có thể phân loại các email mới. Điều này được thực hiện thông qua phương thức ***Classify***, trong đó mô hình sẽ tính toán xác suất email mới là spam hoặc không spam dựa trên tần suất từ đã được lưu trữ trong mô hình.

Khi một email mới đến, nội dung email sẽ được xử lý tương tự như trong bước tiền xử lý, và sau đó mô hình sẽ sử dụng các xác suất đã được tính toán để phân loại email. Kết quả cuối cùng sẽ cho biết liệu email đó có khả năng cao là spam hay không.

```
1 reference
public (string result, double spamProbability, double notSpamProbabilit) ClassifyWithWordProbabilities(string content)
{
    // Loại bỏ ký tự xuống dòng và khoảng trắng thừa
    content = string.Join(" ", content.Split(new[] { '\r', '\n' }, StringSplitOptions.RemoveEmptyEntries)).Trim();
    content = System.Text.RegularExpressions.Regex.Replace(content, @"\s+", " "); // Loại bỏ khoảng trắng thừa

    var words = content.ToLower().Split(new[] { ' ', '.', ',', '!', '?' }, StringSplitOptions.RemoveEmptyEntries);

    //xác suất tiên nghiệm
    double priorSpamProbability = _totalSpam > 0 ? (double)_totalSpam / (_totalSpam + _totalHam) : 0.0;
    double priorNotSpamProbability = _totalHam > 0 ? (double)_totalHam / (_totalSpam + _totalHam) : 0.0;


    //sử dụng log để tránh tràn số
    double logSpamProbability = Math.Log(priorSpamProbability);
    double logNotSpamProbability = Math.Log(priorNotSpamProbability);

    foreach (var word in words)
    {
        var spamWordProbability = GetWordProbability(word, _spamCount, _totalSpamWords);
        var notSpamWordProbability = GetWordProbability(word, _hamCount, _totalHamWords);

        logSpamProbability += Math.Log(spamWordProbability);
        logNotSpamProbability += Math.Log(notSpamWordProbability);
    }

    string result = logSpamProbability > logNotSpamProbability ? "Spam" : "Not Spam";

    return (result, Math.Exp(logSpamProbability), Math.Exp(logNotSpamProbability));
}
```

Hình 3. Thuật toán phân loại email

3.2. Các yếu tố ảnh hưởng đến hiệu suất

Khi xây dựng mô hình phân loại email spam, có nhiều yếu tố có thể ảnh hưởng đến hiệu suất của mô hình. Dưới đây là một số yếu tố quan trọng:

3.2.1. Chất lượng dữ liệu

Chất lượng của tập dữ liệu huấn luyện là rất quan trọng đối với hiệu suất của mô hình Naive Bayes. Nếu dữ liệu bị nhiễu hoặc chứa nhiều lỗi, mô hình sẽ khó khăn hơn trong việc học và phân loại chính xác. Một số yếu tố cần chú ý:

- Độ chính xác của nhãn: Nếu các email không được phân loại chính xác (spam hoặc ham), mô hình sẽ học sai và đưa ra các dự đoán không chính xác.

- Độ đa dạng của dữ liệu: Một tập dữ liệu đa dạng với nhiều loại email khác nhau sẽ giúp mô hình học tốt hơn về các đặc điểm khác nhau của spam và ham.

3.2.2. Các đặc trưng của email

Các đặc trưng của email cũng đóng vai trò quan trọng trong việc xác định khả năng spam. Những đặc trưng này bao gồm:

- Nội dung văn bản: Các từ xuất hiện trong email là chỉ số quan trọng để phân loại. Một số từ có thể xuất hiện thường xuyên trong email spam, trong khi những từ khác thường gặp trong email hợp lệ.
- Địa chỉ người gửi: Một số địa chỉ email có thể có khả năng cao là spam. Việc phân tích địa chỉ người gửi có thể giúp cải thiện độ chính xác của mô hình.
- Tiêu đề email: Tiêu đề cũng là một yếu tố quan trọng. Nhiều email spam thường có tiêu đề gây chú ý hoặc mang tính lừa đảo.

4. Phân tích mã nguồn

4.1. Lớp Email và EmailData

- **Lớp Email**: Lớp này định nghĩa cấu trúc của một email trong hệ thống. Nó bao gồm các thuộc tính như ***Id, Content, IsSpam,*** và các thuộc tính khác để lưu trữ thông tin liên quan đến email. Đặc biệt, nó lưu trữ xác suất phân loại spam và không spam, cho phép mô hình đưa ra quyết định phân loại dựa trên những xác suất này.

```
29 references
public class Email
{
    5 references
    public string Id { get; set; }
    20 references
    public string Content { get; set; }
    4 references
    public bool IsSpam { get; set; }
    0 references
    public double SpamProbability { get; set; } // Xác suất email là spam
    0 references
    public double NotSpamProbability { get; set; } // Xác suất email không phải spam

    12 references
    public string Sender { get; set; }
    2 references
    public string Recipient { get; set; }
    14 references
    public string Subject { get; set; }
    17 references
    public DateTime DateTime { get; set; }
}
```

Hình 4. Lớp Email

- **Lớp EmailData**: Lớp này chứa thông tin về nhãn phân loại (spam hay không spam) và nội dung của email. Nó được sử dụng để lưu trữ dữ liệu gốc trước khi được xử lý để huấn luyện mô hình.

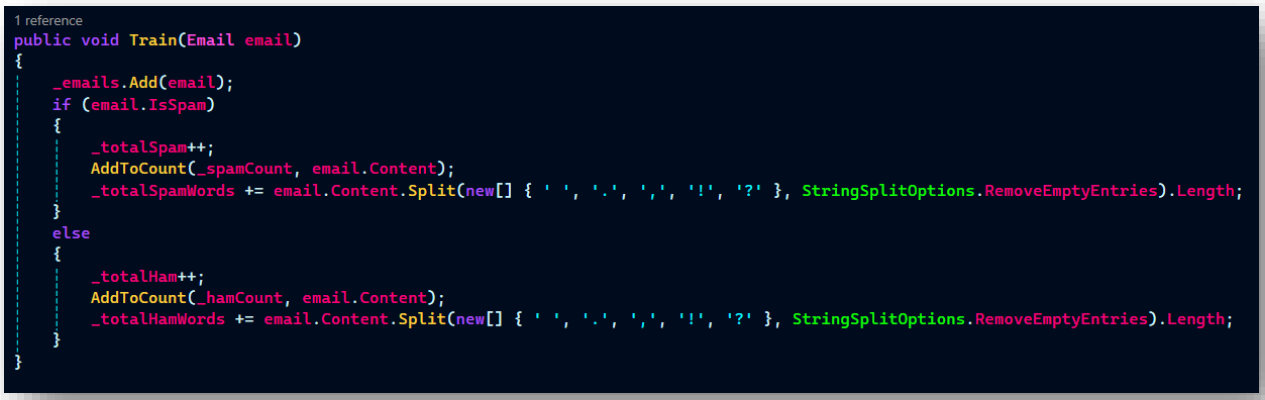
```
public class EmailData
{

    public string label { get; set; }  // Nhãn spam hay không spam
    4 references
    public string content { get; set; } // Nội dung email
}
```

Hình 5. Lớp Email Data

4.2. Lớp NaiveBayesClassifier

Lớp **NaiveBayesClassifier** là cốt lõi của ứng dụng, thực hiện chức năng phân loại email dựa trên thuật toán Naive Bayes. Lớp này thực hiện các bước quan trọng trong quy trình phân loại, bao gồm huấn luyện mô hình từ tập dữ liệu email và sử dụng mô hình đã huấn luyện để dự đoán xác suất một email mới là spam hay không.

4.2.1. Phương thức Train



```
1 reference
public void Train(Email email)
{
    _emails.Add(email);
    if (email.IsSpam)
    {
        _totalSpam++;
        AddToCount(_spamCount, email.Content);
        _totalSpamWords += email.Content.Split(new[] { ' ', '.', ',', '!', '?' }, StringSplitOptions.RemoveEmptyEntries).Length;
    }
    else
    {
        _totalHam++;
        AddToCount(_hamCount, email.Content);
        _totalHamWords += email.Content.Split(new[] { ' ', '.', ',', '!', '?' }, StringSplitOptions.RemoveEmptyEntries).Length;
    }
}
```

Hình 6. Phương thức Train

Chức năng: Huấn luyện mô hình với email mới.

Mô tả chi tiết:

- Phương thức này nhận một đối tượng Email và cập nhật các thống kê liên quan đến số lượng email spam và không spam.
- Nếu email là spam (*IsSpam* là true), nó tăng biến *_totalSpam*, cập nhật từ điển *_spamCount*, và tổng số từ trong email spam (*_totalSpamWords*).
- Ngược lại, nếu email không phải là spam, phương thức cũng thực hiện các cập nhật tương tự cho các biến *_totalHam*, *_hamCount*, và *_totalHamWords*.

Ý nghĩa: Đây là bước thiết yếu trong quá trình học của mô hình, cho phép nó xây dựng cơ sở dữ liệu cần thiết để dự đoán cho các email mới.

4.2.2. Phương thức AddToCount

```
2 references
private static void AddToCount(Dictionary<string, int> countDict, string content)
{
    var words = content.ToLower().Split(new[] { ' ', '.', ',', '!', '?' }, StringSplitOptions.RemoveEmptyEntries);
    foreach (var word in words)
    {
        if (countDict.ContainsKey(word))
            countDict[word]++;
        else
            countDict[word] = 1;
    }
}
```

Hình 7. Phương thức AddToCount()

Chức năng: Cập nhật số lần xuất hiện của các từ trong email vào từ điển.

Mô tả chi tiết:

- Nhận vào một từ điển và nội dung email, phương thức tách nội dung thành từng từ.
- Với mỗi từ, nó sẽ kiểm tra xem từ đó đã có trong từ điển chưa. Nếu có, nó sẽ tăng số đếm; nếu chưa, nó sẽ thêm từ mới với giá trị đếm là 1.

Ý nghĩa: Phương thức này giúp duy trì thống kê số lần xuất hiện của các từ trong email, là yếu tố quan trọng để tính toán xác suất.

4.2.3. Phương thức ClassifyWithWordProbabilities

```
1 reference
public (string result, double spamProbability, double notSpamProbabilit) ClassifyWithWordProbabilities(string content)
{
    // Loại bỏ ký tự xuống dòng và khoảng trắng thừa
    content = string.Join(" ", content.Split(new[] { '\r', '\n' }, StringSplitOptions.RemoveEmptyEntries)).Trim();
    content = System.Text.RegularExpressions.Regex.Replace(content, @"\s+", " "); // Loại bỏ khoảng trắng thừa

    var words = content.ToLower().Split(new[] { ' ', '.', ',', '!', '?' }, StringSplitOptions.RemoveEmptyEntries);

    //xác suất tiên nghiệm
    double priorSpamProbability = _totalSpam > 0 ? (double)_totalSpam / (_totalSpam + _totalHam) : 0.0;
    double priorNotSpamProbability = _totalHam > 0 ? (double)_totalHam / (_totalSpam + _totalHam) : 0.0;


    //sử dụng log để tránh tràn số
    double logSpamProbability = Math.Log(priorSpamProbability);
    double logNotSpamProbability = Math.Log(priorNotSpamProbability);

    foreach (var word in words)
    {
        var spamWordProbability = GetWordProbability(word, _spamCount, _totalSpamWords);
        var notSpamWordProbability = GetWordProbability(word, _hamCount, _totalHamWords);

        logSpamProbability += Math.Log(spamWordProbability);
        logNotSpamProbability += Math.Log(notSpamWordProbability);
    }

    string result = logSpamProbability > logNotSpamProbability ? "Spam" : "Not Spam";

    return (result, Math.Exp(logSpamProbability), Math.Exp(logNotSpamProbability));
}
```

Hình 8. Phương thức ClassifyWithWordProbabilities()

Chức năng: Phân loại email dựa trên nội dung và xác suất từ.

Mô tả chi tiết:

- Nhận vào nội dung email, phương thức này sẽ loại bỏ các ký tự xuống dòng và khoảng trắng thừa, sau đó chuyển đổi nội dung thành chữ thường và tách thành các từ.
- Tính toán xác suất tiên nghiệm cho cả hai lớp spam và không spam dựa trên tổng số email đã được huấn luyện.
- Sử dụng logarit để tránh vấn đề tràn số khi tính toán xác suất, phương thức sẽ tính xác suất log của email là spam hoặc không spam dựa trên các từ trong nội dung email.
- Cuối cùng, nó so sánh giá trị log và trả về kết quả phân loại cùng với xác suất tương ứng cho cả hai lớp.

Ý nghĩa: Đây là phương thức chính cho việc phân loại, cho phép mô hình đưa ra dự đoán dựa trên nội dung email.

4.2.4. Phương thức GetWordProbability

```
2 references
public static double GetWordProbability(string word, Dictionary<string, int> wordCounts, int total)
{
    // Tính toán xác suất của từ dựa trên số lần xuất hiện và tổng số từ
    if (wordCounts.TryGetValue(word, out int count))
    {
        return (double)(count + 1) / (total + wordCounts.Count); // Sử dụng Laplace smoothing
    }

    // Trả về giá trị rất nhỏ cho từ không tồn tại trong dữ liệu huấn luyện
    return 1.0 / (total + wordCounts.Count); // Xử lý trường hợp từ không xuất hiện trong dữ liệu huấn luyện
}
```

Hình 9. Phương thức GetWordProbability()

Chức năng: Tính toán xác suất của một từ dựa trên tần suất xuất hiện và tổng số từ.

Mô tả chi tiết:

- Phương thức này kiểm tra xem từ có tồn tại trong từ điển không. Nếu có, nó sẽ tính toán xác suất dựa trên số lần xuất hiện và tổng số từ, sử dụng kỹ thuật làm trơn Laplace.
- Nếu từ không xuất hiện trong dữ liệu huấn luyện, nó sẽ trả về một giá trị rất nhỏ để đảm bảo rằng xác suất không bao giờ bằng 0.

Ý nghĩa: Phương thức này đảm bảo rằng mọi từ đều có một xác suất nhất định, ngay cả khi nó chưa xuất hiện trong dữ liệu huấn luyện, giúp cải thiện độ chính xác của mô hình.

4.3. Lớp GmailServiceHelper

Lớp **GmailServiceHelper** đóng vai trò là một lớp trợ giúp (helper) để tương tác với dịch vụ Gmail thông qua API. Lớp này cung cấp các phương thức để truy xuất email từ tài khoản Gmail, từ đó hỗ trợ quá trình thu thập dữ liệu email phục vụ cho mô hình Naive Bayes.

4.3.1. Phương thức GetGmailService()

```
2 references
public static GmailService GetGmailService()
{
    UserCredential credential;
    string credPath = @"C:\CongNghe.Net\NaiveBayes\SpamEmailFilter\SpamFilterApp\credentials.json";

    using (var stream = new FileStream(credPath, FileMode.Open, FileAccess.Read))
    {
        string tokenPath = "token.json";
        var clientSecrets = GoogleClientSecrets.FromStream(stream);
        credential = GoogleWebAuthorizationBroker.AuthorizeAsync(
            clientSecrets.Secrets,
            Scopes,
            "user",
            CancellationToken.None,
            new FileDataStore(tokenPath, true)).Result;
    }

    // Tạo Gmail service
    return new GmailService(new BaseClientService.Initializer()
    {
        HttpClientInitializer = credential,
        ApplicationName = ApplicationName,
    });
}
```

Hình 10. Phương thức GetGmailService()

- Chức năng: Khởi tạo và trả về một đối tượng GmailService được cấu hình sẵn với các quyền truy cập xác định trong Scopes.
- Mô tả chi tiết: Phương thức này yêu cầu xác thực người dùng thông qua tệp credentials.json. Khi người dùng xác thực, mã thông báo sẽ được lưu trong token.json để có thể truy cập vào Gmail mà không cần xác thực lại trong các phiên sau.
- Ứng dụng: Phương thức này là điểm khởi đầu để tạo một kết nối API, cần thiết cho các tác vụ truy xuất, đọc và gửi email.

4.3.2. Phương thức GetEmails()

```
2 references
public static List<Message> GetEmails(GmailService service, string query, int maxResults = 10)
{
    var request = service.Users.Messages.List("me");
    request.Q = query;
    request.MaxResults = maxResults;

    var response = request.Execute();
    return response.Messages?.ToList() ?? new List<Message>();
}
```

Hình 11. Phương thức GetEmail()

- Chức năng: Lấy danh sách các email từ tài khoản Gmail dựa trên từ khóa truy vấn (query) và giới hạn số lượng email trả về (maxResults).
- Mô tả chi tiết: Phương thức gửi yêu cầu tới Gmail API và trả về danh sách các email khớp với từ khóa. Danh sách này giúp dễ dàng lọc ra các email quan trọng hoặc những email đáp ứng yêu cầu cụ thể.
- Ứng dụng: Được dùng để thu thập tập hợp email để huấn luyện và kiểm thử mô hình phân loại spam.

4.3.3. Phương thức GetEmailDetails()

```
2 references
public static Email GetEmailDetails(GmailService service, string messageId)
{
    var email = service.Users.Messages.Get("me", messageId).Execute();
    var headers = email.Payload.Headers;

    string sender = headers.FirstOrDefault(h => h.Name == "From")?.Value;
    string recipient = headers.FirstOrDefault(h => h.Name == "To")?.Value;
    string subject = headers.FirstOrDefault(h => h.Name == "Subject")?.Value;
    string date = headers.FirstOrDefault(h => h.Name == "Date")?.Value;

    DateTime emailDateTime;
    if (!DateTime.TryParse(date, out emailDateTime))
    {
        emailDateTime = DateTime.MinValue;
    }

    // Lấy nội dung email
    string emailContent = GetEmailContent(email);

    return new Email
    {
        Sender = sender,
        Recipient = recipient,
        Subject = subject,
        DateTime = emailDateTime,
        Content = emailContent
    };
}
```

Hình 12. Phương thức GetEmailDetails()

- Chức năng: Lấy chi tiết của một email bao gồm người gửi, người nhận, chủ đề, ngày giờ và nội dung.
- Mô tả chi tiết: Phương thức gửi yêu cầu đến API để truy xuất email dựa trên messageId. Các header của email được lấy ra để thu thập thông tin người gửi, người nhận, chủ đề và ngày giờ. Sau đó, nội dung email được lấy và chuyển mã từ Base64.
- Ứng dụng: Hữu ích cho việc lấy thông tin chi tiết về email và sử dụng các thông tin này để phân loại email.

4.3.4. Phương thức GetEmailContent()

```
1 reference
public static string GetEmailContent(Message email)
{
    if (email?.Payload?.Parts == null)
        return email?.Payload?.Body?.Data;

    var part = email.Payload.Parts.FirstOrDefault(p => !string.IsNullOrEmpty(p.Body?.Data));
    if (part == null) return string.Empty;

    string encodedBody = part.Body.Data.Replace('-', '+').Replace('_', '/');
    byte[] data = Convert.FromBase64String(encodedBody);
    return Encoding.UTF8.GetString(data);
}
```

Hình 13. Phương thức GetEmaiContent()

- Chức năng: Lấy và giải mã nội dung email.
- Mô tả chi tiết: Phương thức kiểm tra và giải mã nội dung email từ định dạng Base64. Nếu email có phần thân (body) mã hóa, phương thức sẽ chuyển đổi sang chuỗi UTF-8 để dễ đọc.
- Ứng dụng: Hỗ trợ thu thập dữ liệu thô để huấn luyện mô hình Naive Bayes phân loại email spam.

4.3.5. Phương thức SendEmail()

```
1 reference
public static void SendEmail(GmailService service, string to, string subject, string body)
{
    var emailMessage = new AE.Net.Mail.MailMessage
    {
        Subject = subject,
        Body = body,
        From = new MailAddress("tcongminh1604@gmail.com")
    };
    emailMessage.To.Add(new MailAddress(to));
    emailMessage.ReplyTo.Add(emailMessage.From);

    var msgString = new StringWriter();
    emailMessage.Save(msgString);

    var msg = new Message
    {
        Raw = Base64UrlEncode(msgString.ToString())
    };

    service.Users.Messages.Send(msg, "me").Execute();
}
```

Hình 14. Phương thức SendEmail()

- Chức năng: Gửi một email từ tài khoản Gmail đã xác thực.
- Mô tả chi tiết: Phương thức tạo một đối tượng MailMessage, thêm người nhận, chủ đề và nội dung email, sau đó mã hóa email và gửi qua API.
- Ứng dụng: Được dùng khi cần gửi các thông báo từ ứng dụng tới người dùng, chẳng hạn như thông báo về email bị phát hiện là spam.

4.3.6. Phương thức Base64UrlEncode()

```
1 reference
private static string Base64UrlEncode(string input)
{
    var inputBytes = System.Text.Encoding.UTF8.GetBytes(input);
    return Convert.ToBase64String(inputBytes)
        .Replace('+', '-')
        .Replace('/', '_')
        .Replace("=", "");
}
```

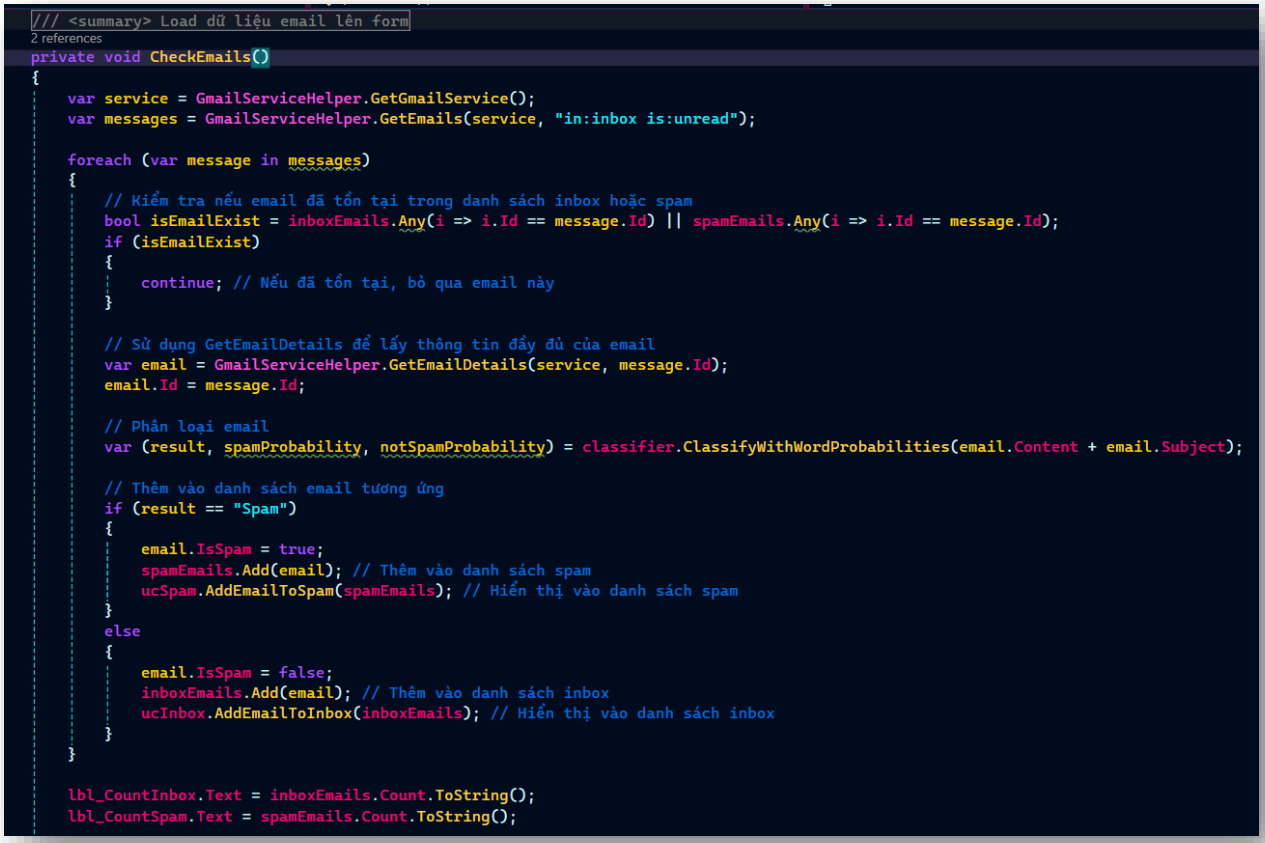
Hình 15. Phương thức Base64UrlEncode()

- Chức năng: Mã hóa chuỗi đầu vào sang định dạng Base64 với ký hiệu an toàn cho URL.
- Mô tả chi tiết: Mã hóa email theo chuẩn Base64 và thực hiện thay thế một số ký tự để phù hợp với URL.
- Ứng dụng: Được dùng trong SendEmail() để định dạng nội dung email trước khi gửi đi qua Gmail API.

4.4. Các phương thức quan trọng khác

4.4.1. Phương thức CheckEmails()

Phương thức **CheckEmails()** có nhiệm vụ kiểm tra và phân loại các email trong hộp thư đến. Nó sẽ lấy tất cả các email mới và sử dụng mô hình Naive Bayes để phân tích nội dung của từng email, sau đó xác định xem email đó có phải là spam hay không. Kết quả phân loại sẽ được lưu trữ lại trong thuộc tính *IsSpam* của lớp Email.



```
/// <summary> Load dữ liệu email lên form
2 references
private void CheckEmails()
{
    var service = GmailServiceHelper.GetGmailService();
    var messages = GmailServiceHelper.GetEmails(service, "in:inbox is:unread");

    foreach (var message in messages)
    {
        // Kiểm tra nếu email đã tồn tại trong danh sách inbox hoặc spam
        bool isEmailExist = inboxEmails.Any(i => i.Id == message.Id) || spamEmails.Any(i => i.Id == message.Id);
        if (isEmailExist)
        {
            continue; // Nếu đã tồn tại, bỏ qua email này
        }

        // Sử dụng GetEmailDetails để lấy thông tin đầy đủ của email
        var email = GmailServiceHelper.GetEmailDetails(service, message.Id);
        email.Id = message.Id;

        // Phân loại email
        var (result, spamProbability, notSpamProbability) = classifier.ClassifyWithWordProbabilities(email.Content + email.Subject);

        // Thêm vào danh sách email tương ứng
        if (result == "Spam")
        {
            email.IsSpam = true;
            spamEmails.Add(email); // Thêm vào danh sách spam
            ucSpam.AddEmailToSpam(spamEmails); // Hiển thị vào danh sách spam
        }
        else
        {
            email.IsSpam = false;
            inboxEmails.Add(email); // Thêm vào danh sách inbox
            ucInbox.AddEmailToInbox(inboxEmails); // Hiển thị vào danh sách inbox
        }
    }

    lbl_CountInbox.Text = inboxEmails.Count.ToString();
    lbl_CountSpam.Text = spamEmails.Count.ToString();
```

Hình 16. Phương thức CheckEmail()

4.4.2. Phương thức LoadData

Phương thức **LoadData** chịu trách nhiệm tải dữ liệu email từ một nguồn dữ liệu (có thể là file, cơ sở dữ liệu, hoặc API). Dữ liệu này sẽ được sử dụng để xây dựng mô hình Naive Bayes. Nó sẽ tạo ra các đối tượng **EmailData**, sau đó phân chia dữ liệu thành tập huấn luyện và tập kiểm tra.

```
/// <summary> Load dữ liệu từ email_dataset để lọc email
2 references
private void LoadData(string filePath)
{
    using (var reader = new StreamReader(filePath))
    using (var csv = new CsvReader(reader, CultureInfo.InvariantCulture))
    {
        // Đọc các bản ghi từ file CSV
        var records = csv.GetRecords<EmailData>().ToList();

        foreach (var record in records)
        {
            // Nhãn spam được xác định bởi label (spam là "Spam", không spam là "Not Spam")
            bool isSpam = record.label.Trim().ToLower() == "spam";

            // Huấn luyện bộ phân loại với thông tin email
            classifier.Train(new Email
            {
                Content = record.content.Trim(),
                IsSpam = isSpam
            });
        }
    }
}
```

Hình 17. Phương thức LoadData()

5. Thực nghiệm và kết quả

5.1. Thiết lập môi trường thử nghiệm

Để thực hiện các thử nghiệm cho ứng dụng phân loại email spam sử dụng thuật toán Naive Bayes, một môi trường thử nghiệm đã được thiết lập với các thành phần chính sau:

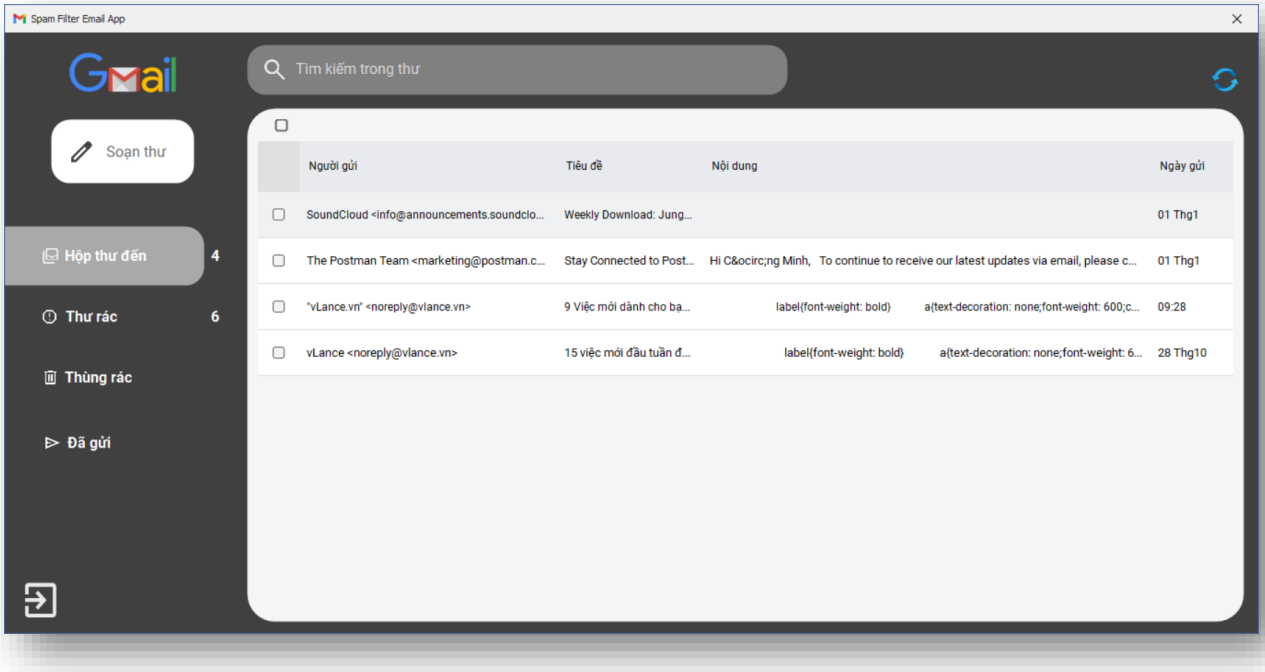
- Ngôn ngữ lập trình: C#.
- Môi trường phát triển: Visual Studio (phiên bản 2022).
- Thư viện: Sử dụng các thư viện cơ bản của .NET Framework cho xử lý dữ liệu và giao diện người dùng.
- Dữ liệu thử nghiệm: Bộ dữ liệu email chứa các email đã được gán nhãn là spam và không spam, được thu thập từ các nguồn khác nhau để đảm bảo tính đa dạng và độ chính xác. Số lượng email trong bộ dữ liệu là khoảng 100 email, trong đó có 50 email được đánh dấu là spam và 50 email không phải spam.

	A	B
1	label	content
2	Spam	Nhận ưu đãi miễn phí cho sản phẩm mới!
3	Spam	Giảm giá 50% cho tất cả các mặt hàng!
4	Spam	Khuyến mãi lớn, nhấp vào đây để nhận!
5	Spam	Bạn đã thắng giải thưởng trị giá 1000 đô la!
6	Spam	Đừng bỏ lỡ cơ hội này, nhấp vào liên kết!
7	Spam	Mua một, tặng một miễn phí trong hôm nay!
8	Spam	Cơ hội đầu tư tuyệt vời, hãy tham gia ngay!
9	Spam	Nhấn vào đây để nhận quà tặng!
10	Spam	Hãy kiểm tra tài khoản của bạn để nhận phần thưởng!
11	Spam	Giảm giá đặc biệt chỉ trong hôm nay!
12	Not Spam	Thông báo từ ngân hàng về tài khoản của bạn.
13	Not Spam	Cập nhật dịch vụ từ nhà cung cấp của bạn.
14	Not Spam	Thông tin về đơn hàng mà bạn đã đặt.
15	Not Spam	Cảm ơn bạn đã đăng ký nhận bản tin của chúng tôi.
16	Not Spam	Đừng quên thanh toán hóa đơn của bạn vào cuối tháng này.
17	Not Spam	Hướng dẫn sử dụng dịch vụ mới đã được cập nhật.
18	Not Spam	Mời bạn tham gia hội thảo trực tuyến về công nghệ mới.
19	Not Spam	Chúng tôi đã nhận được yêu cầu hỗ trợ của bạn và đang xử lý.
20	Not Spam	Lịch trình cuộc họp của bạn vào thứ Hai tới đã được xác nhận.
21	Not Spam	Bạn có một tin nhắn mới từ người bạn của bạn.
22	Spam	Chúc mừng bạn đã thắng giải thưởng trị giá 500 đô la! Nhấp vào liên kết dưới đây để nhận phần thưởng của bạn ngay bây giờ. Đừng bỏ lỡ cơ hội này!
23	Not Spam	Xin chào, đây là thông báo từ ngân hàng của bạn. Bạn có một giao dịch mới trong tài khoản. Vui lòng kiểm tra tài khoản của bạn để biết thêm chi tiết.
24	Spam	Nhận ưu đãi 70% cho tất cả sản phẩm trong tuần này! Nhấp vào đây để biết thêm chi tiết.

Hình 18. Dữ liệu huấn luyện (100 email)

5.2. Trang hiển thị (UC_Inbox)

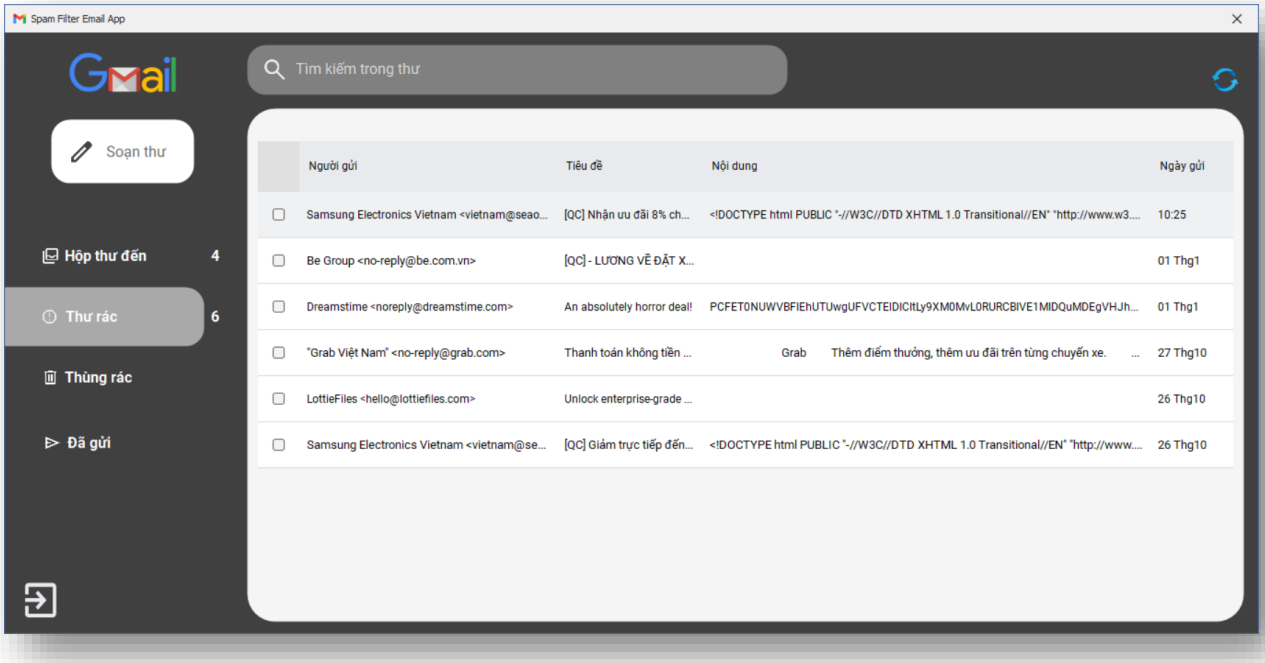
Các email được tính là không spam sẽ hiển thị tại đây:



Hình 19. Trang hiển thị email đến (không spam)

5.3. Trang Email Spam (UC_Spam)

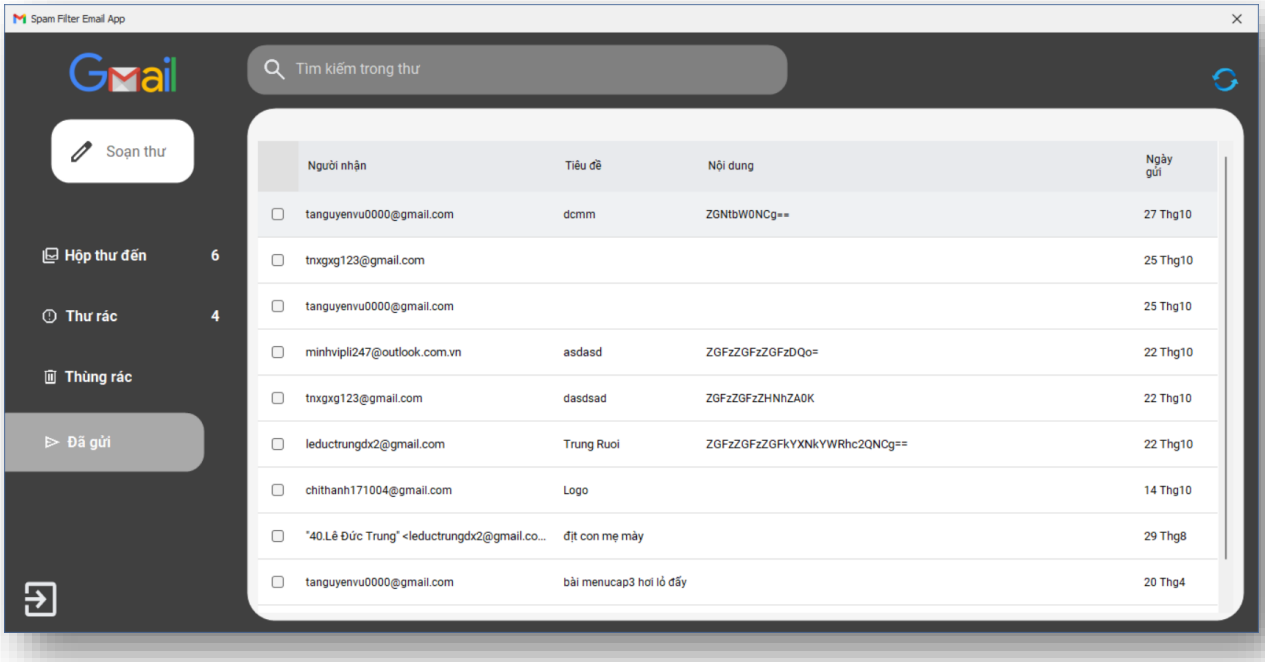
Các email được tính là spam sẽ hiển thị tại đây



Hình 20. Trang hiển thị email spam

5.4. Trang Email đã gửi (UC_Sent)

Trang hiển thị các email đã được gửi



Hình 21. Trang hiển thị email đã gửi

5.5. So sánh với các thuật toán khác

Để đánh giá hiệu quả của thuật toán Naive Bayes trong phân loại email spam, mô hình này đã được so sánh với một số thuật toán phân loại khác như:

- Support Vector Machines (SVM): Mô hình SVM cho kết quả tỷ lệ chính xác là 88%, với độ nhạy đạt 85% và độ chính xác là 80%. Mặc dù SVM có thể cung cấp kết quả tốt, nhưng thời gian huấn luyện và dự đoán của nó thường lâu hơn so với Naive Bayes.
- Decision Trees: Mô hình Decision Trees đạt tỷ lệ chính xác khoảng 87%, nhưng có xu hướng bị ảnh hưởng bởi overfitting, dẫn đến hiệu suất kém khi xử lý các tập dữ liệu lớn hơn.
- Random Forest: Mô hình Random Forest có tỷ lệ chính xác cao hơn, khoảng 91%, tuy nhiên, mô hình này yêu cầu nhiều tài nguyên tính toán hơn và thời gian xử lý lâu hơn.

Kết quả cho thấy, thuật toán Naive Bayes mặc dù đơn giản nhưng lại đạt được hiệu suất tương đối tốt trong việc phân loại email spam, với tốc độ xử lý nhanh và yêu cầu tài

nguyên thấp. Điều này làm cho Naive Bayes trở thành một lựa chọn phổ biến trong các ứng dụng thực tế.

6. Kết luận và hướng phát triển

6.1. Kết luận

Thuật toán **Naïve Bayes** đem lại kết quả bất ngờ, tỉ lệ nhận dạng cao. Kết quả phân loại cao nhưng chỉ nằm trong một điều kiện, bài toán cụ thể chứ không phải đúng trong tất cả mô hình khác.

Mô hình trên chỉ quan tâm tới nội dung của email. Email còn có các đặc điểm để phân loại nữa là tiêu đề, địa chỉ người gửi. Để xây dựng được mô hình với đầy đủ các đặc điểm trên là bài toán lớn với em và các nhà khoa học trên thế giới.

Một trong những lý do nghĩ đến có thể là trong bức thư đó những từ hay xuất hiện ở thư rác và những từ hay xuất hiện thư hợp lệ có tỷ lệ gần bằng nhau dẫn đến việc tính xác suất bị sai. Khi đó mô hình sẽ bị phân loại sai.

6.2. Hướng phát triển

Mô hình phân loại thư rác trên chỉ sử dụng thuật toán Naïve Bayes, chúng ta còn có thể sử dụng các thuật toán khác như SVM, mạng Neutron… cũng cho kết quả khá tốt. Không có thuật toán nào là tối ưu nên việc chọn thuật toán cho phù hợp với bài toán là rất quan trọng.

Việc thu thập thêm dữ liệu từ cơ sở dữ liệu trên cũng là vấn đề em quan tâm. Dựa vào cơ sở dữ liệu trên ta có thể tìm thêm các dữ liệu rồi gán nhãn cho chúng là thư rác hoặc thư hợp lệ. Việc dữ liệu càng nhiều thì mô hình phân loại sẽ chính xác hơn.

Xây dựng một mô hình thư rác hoàn chỉnh, hiệu quả cao, tạo ra một ứng dụng để người dùng email có thể sử dụng nó.

PHẦN 2. TÀI LIỆU THAM KHẢO

[1] **Viblo. (n.d.)**. *Xây dựng mô hình lọc thư rác bằng Naive Bayes*. Truy cập từ https://viblo.asia/p/xay-dung-mo-hinh-loc-thu-rac-bang-naive-bayes-Ljy5Vqxklra

[2] **Đặng Văn Nam. (2020).** *NLP (Natural Language Processing)*. Truy cập từ https://qlkh.humg.edu.vn/CongBo/Download/4413?FileName=ERSD2020_DangVanNam_NLP.pdf

[3] **OpenAI. (n.d.).** ***ChatGPT*****. Retrieved October 31, 2024,** Truy cập từ *https://chatgpt.com/*

PHẦN 3. PHỤ LỤC

Danh mục hình ảnh

Danh mục bảng