ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO LAB 3

PHÂN LOẠI THƯ RÁC BẰNG NAIVE BAYES

Môn học: Phương pháp Toán cho Trí tuệ Nhân Tạo

Sinh viên:
Bàng Mỹ Linh - 23122009
Huỳnh Trung Kiệt - 23122039
Dào Sỹ Duy Minh - 23122041
Nguyễn Lâm Phú Quý - 23122048

GVHD:
TS. Cấn Trần Thành Trung
ThS. Nguyễn Ngọc Toàn

Ngày 14 tháng 6 năm 2025

Mục lục

1 Giới thiệu

Báo cáo này trình bày kết quả thực hiện bài lab về xây dựng mô hình phân loại thư rác (spam classification) sử dụng thuật toán Naive Bayes. Nghiên cứu tập trung vào việc áp dụng các kiến thức về Maximum Likelihood Estimation (MLE) và Maximum A Posteriori estimation (MAP) để xây dựng mô hình thống kê phân loại email. Nhóm đề xuất và đánh giá mô hình Naive Bayes được cài đặt từ đầu, so sánh hiệu suất thông qua các độ đo accuracy, precision, recall và F1-score.

Phân công

MSSV	Họ và Tên	Phân công
23122009	Bàng Mỹ Linh	• Xây dựng mô hình Naive Bayes • Tính năng đánh giá mô hình
23122039	Huỳnh Trung Kiệt	• Tiền xử lý dữ liệu • Cài đặt vector hóa (Bag of Words) • Làm sạch và stop words
23122041	Đào Sỹ Duy Minh	• Khám phá, phân tích dữ liệu • Thử nghiệm, đánh giá mô hình • Viết tính năng phân loại email thủ công
23122048	Nguyễn Lâm Phú Quý	• Viết tính năng phân loại email từ file .csv • Visualize dữ liệu và phân tích • Viết báo cáo và tổng hợp kết quả

Bảng 1: Phân công nhiệm vụ cho các thành viên nhóm

2 Tập dữ liệu Enron-Spam

2.1 Giới thiệu

Bộ dữ liệu Enron-Spam là một nguồn tài liệu tuyệt vời 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 chứa tổng cộng 17.171 thư rác và 16.545 thư không phải thư rác ("ham") (tổng cộng 33.716 thư điện tử).
Mỗi thư (1 dòng) trong tập dữ liệu có đặc điểm như sau:

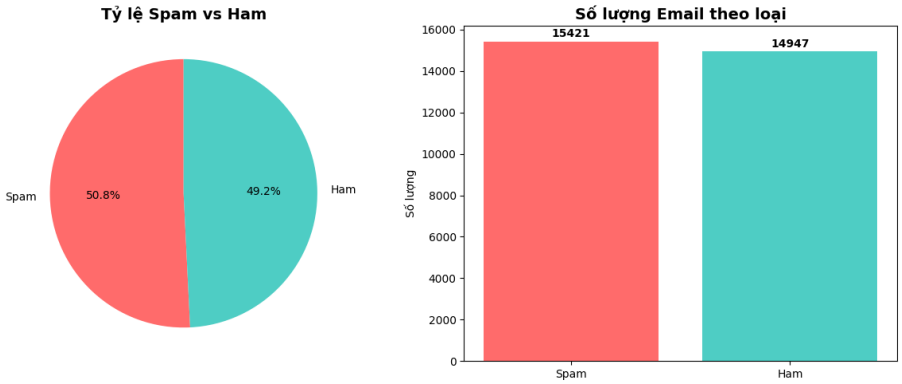
- **Subject:** tên tiêu đề của thư.
- **Message:** Nội dung của email. Có thể chứa chuỗi rỗng nếu tin nhắn chỉ có dòng tiêu đề và không có nội dung. Trong trường hợp chuyển tiếp email hoặc trả lời, điều này cũng chứa tin nhắn gốc với dòng tiêu đề, "từ:", "đến:", v.v.
- **Spam/Ham:** Có giá trị "spam" hoặc "ham". Nhãn của thư được phân loại có là tin nhắn spam hay không.

2.2 Mô tả dữ liệu

Thống kê mô tả

- Tổng số email: 30,360
- Số email spam: 15,421 (50.8%)
- Số email ham: 14,947 (49.2%)

Nhận xét: Tỷ lệ cân bằng giữa spam và ham



Hình 1: Tỷ lệ email Spam và Ham

2.2.1 5 dòng đầu tiên của tập dữ liệu

	Unnamed: 0	Message ID	Subject	Message	Spam/Ham	split
0	0	0	christmas tree farm pictures	NaN	ham	0.038415
1	1	1	vastar resources , inc .	gary , production from the high island larger ...	ham	0.696509
2	2	2	calpine daily gas nomination	- calpine daily gas nomination 1 . doc	ham	0.587792
3	3	3	re : issue	fyi - see note below - already done .\nstella\...	ham	-0.055438
4	5	5	mcmullen gas for 11 / 99	jackie ,\nsince the inlet to 3 river plant is ...	ham	-0.419658

Hình 2: Minh họa 5 dòng đầu tiên của tập dữ liệu

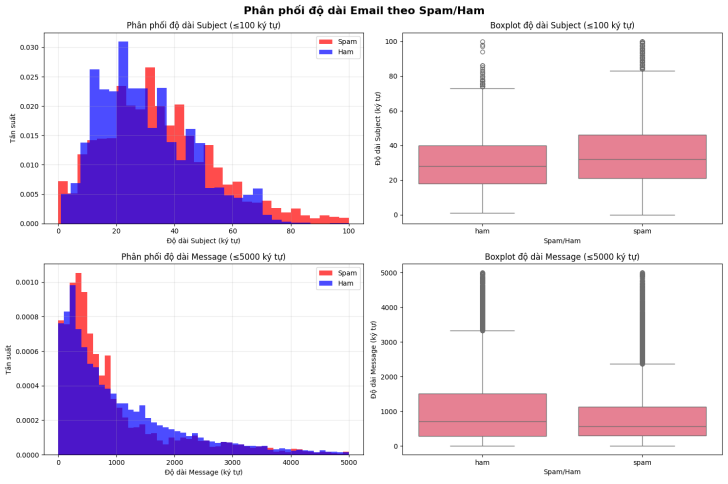
2.2.2 Phân tích độ dài email

Thống kê độ dài Subject:

- Trung bình: 34.15
- Spam: 37.02
- Ham: 31.18

Thống kê độ dài Message:

- Trung bình: 1341.84
- Spam: 1273.50
- Ham: 1412.34



Hình 3: Phân phối độ dài theo loại spam/ham

Nhận xét: Độ dài email Spam và Ham không cho thấy quá rõ sự khác biệt giữa 2 class.

3 Tiền xử lý dữ liệu

3.1 Kiểm tra dữ liệu

Tập Train:

- Số dòng: 27284
- Số cột: 6
- Số giá trị thiếu:
  - Subject: 229
  - Message: 352
  - Spam/Ham: 0

Tập Val:

- Số dòng: 3084
- Số cột: 6
- Số giá trị thiếu:
  - Subject: 29
  - Message: 35
  - Spam/Ham: 0

Nhóm đã điền các giá trị thiếu ở Subject và Message bằng chuỗi rỗng. Đồng thời loại bỏ cột `split` vì không cần thiết.

3.2 Làm sạch văn bản

Các bước tiền xử lý văn bản bao gồm:

1. Chuyển văn bản về chữ thường
2. Loại bỏ URL, email address, số điện thoại
3. Loại bỏ ký tự đặc biệt và dấu câu
4. Loại bỏ khoảng trắng thừa
5. Xử lý stop words

3.2.1 Nối Subject và Message

Kết hợp nội dung từ Subject và Message thành một trường Content duy nhất để vectorization.

3.3 Vectorization: Bag of Words

Trước khi huấn luyện, mỗi email cần được mã hóa thành một feature vector. Có nhiều phương pháp mã hóa khác nhau tùy vào loại bài toán cần xử lý, ở đây nhóm ưu tiên sử dụng Bag of words (BoW) để mã hóa bởi tính đơn giản, và cách mã hóa này cũng cho kết quả khá tốt.

Bag of Words là cách vector hóa một văn bản bằng cách:

- Tạo danh sách từ điển là tất cả các từ phân biệt trong tập dữ liệu
- Lưu tần số của mỗi từ trong văn bản đó (nếu từ đó xuất hiện trong từ điển) và không quan tâm thứ tự của chúng. Mỗi văn bản $x^{(i)}$ sẽ được mã hóa thành một vector có độ dài n là số từ trong từ điển, và giá trị $x_j^{(i)}$ ứng với số lần từ ở vị trí j trong từ điển xuất hiện trong văn bản $x^{(i)}$.

  Ví dụ: Ta có câu `s = ‘‘I love this movie, I would recommend it.’’` và từ điển `dict = {‘‘a’’ : 0, ‘‘aardvark’’ : 1, ‘‘buy’’ : 2, ‘‘catch’’ : 3, ..., ‘‘i’’ : 15, ..., ‘‘recommend’’ : 67, ...}`

  Thì dạng vector hóa của s là x_s = [0 0 0 ... 2 ... 1 ...]

Trong **implementation**, nhóm sử dụng thư viện `CountVectorizer` của scikit-learn với các tham số được tinh chỉnh cụ thể:

- **ngram_range = (1,2)**: Tạo n-grams từ 1-gram (từ đơn) đến 2-gram (cặp từ liền kề).
  - 1-gram: "spam", "money", "free"
  - 2-gram: "free money", "click here", "limited time"
  - *Lý do*: 2-grams giúp capture context và phrases đặc trưng của spam. Tăng dimensionality nhưng cải thiện accuracy
- **stop_words = "english"**: Loại bỏ các stop words tiếng Anh.
  - Loại bỏ: "the", "and", "is", "in", etc.
  - *Lý do*: Stop words ít mang thông tin phân biệt spam/ham
- **min_df = 2**: Loại bỏ terms xuất hiện trong ít hơn 2 documents.
  - *Lý do*: Terms hiếm (xuất hiện 1 lần) thường là typos hoặc noise → Giảm overfitting, tăng generalization
- **max_df = 0.8**: Loại bỏ terms xuất hiện trong >80% documents.
  - *Lý do*: Terms quá phổ biến ít có khả năng phân biệt

  Kết quả sau khi áp dụng tham số:
  - Vocabulary size: 495,881 features (bao gồm 1-grams và 2-grams)
  - Training data shape: (27,284 × 495,881)
  - Validation data shape: (3,084 × 495,881)
  - Sparse matrix format để tiết kiệm memory

4 Ý tưởng Naive Bayes và Multinomial Naive Bayes

Naive Bayes là một mô hình phân loại bằng xác suất. Đối với bài toán phân loại văn bản, biến thể tiêu biểu thường được sử dụng là mô hình Multinomial Naive Bayes. Với mỗi văn bản d cần phân loại vào các class c ∈ C, mô hình sẽ trả về kết quả là class $\hat{c}$ có xác suất hậu nghiệm $P(c|d)$ lớn nhất (Maximum A Posteriori). Cụ thể:

$$\hat{c} = \arg\max_{c \in C} P(c|d)$$

Theo công thức xác suất Bayes, ta có:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

Do đó:

$$\hat{c} = \arg\max_{c \in C} P(c|d) = \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)}$$

Vì P(d) không đổi với mọi c nên để đơn giản ta có thể viết:

$$\hat{c} = \arg\max_{c \in C} \underbrace{P(d \mid c)}_{\text{likelihood}} \underbrace{P(c)}_{\text{prior}}$$

Trong đó xác suất P(c) là tỉ lệ của các văn bản thuộc class c trong tổng số văn bản:

$$P(c) = \frac{N_c}{N}$$

Gọi f_i là đặc trưng thứ i trong cách mã hóa Bag of Words, tương ứng với số lần từ w_i xuất hiện trong văn bản đang xét (w_i nằm trong từ điển gồm n từ V = [w_1 ... w_n]) Ta có:

$$\hat{c} = \arg\max_{c \in C} P(w_1, w_2, \ldots, w_n|c)P(c)$$

Đến đây, với một giả định "Naive" rằng các w_i độc lập cùng phân phối, tức là sự xuất hiện của w_i không ảnh hưởng đến khả năng xuất hiện của w_j trong câu. Do đó xác suất trên có thể được tách thành:

$$\hat{c} = \arg\max_{c \in C} P(w_1, w_2, \ldots, w_n|c)P(c) = P(w_1|c).P(w_2|c) \ldots P(w_n|c)$$

Vậy ta có kết quả của mô hình cho văn bản d với $d_{vectorized} = [f_1 \ldots f_n]$

$$\hat{c} = \arg\max_{c \in C} P(c) \prod_{j \leftarrow \text{all positions in d}} P(w_j|c) = \prod_{i=1}^{n} P(w_i|c)^{f_i}$$

trong đó $P(w_i|c) = \frac{N_{ci}}{N_c}$ là xác suất từ thứ i xuất hiện trong văn bản thuộc class c - N_{ci} là tổng số lần xuất hiện của từ w_i trong các văn bản thuộc class c - N_c là tổng số từ trong class c và bằng tổng $\sum_{i=1}^{n} N_{ci}$. Suy ra tổng xác suất $\sum_{i=1}^{n} P(w_i|c) = 1$

Mặt khác, nếu tồn tại một từ w nào đó trong văn bản d không có trong từ điển thì sẽ kéo theo xác suất hậu nghiệm bằng 0 bất kể các giá trị xác suất còn lại có lớn thế nào. Điều này dẫn đến kết quả không hợp lý. Để giải quyết trường hợp này, có thể áp dụng kĩ thuật **Laplace smoothing**: cộng thêm vào tử số của $P(w_i|c)$ một đại lượng α (thường là 1) và mẫu số cũng cộng thêm đại lượng $n.\alpha$ để đảm bảo tổng xác suất bằng 1.

Như vậy ta có

$$P(w_i|c) = \frac{N_{ci} + \alpha}{N_c + n\alpha}$$

Do tích của các xác suất là một giá trị rất nhỏ dẫn đến khó khăn trong tính toán, nên để đơn giản ta có thể tối ưu log của xác suất (hàm logarit là hàm đồng biến và giá trị các xác suất luôn không âm) Do đó:

$$\hat{c} = \arg\max_{c \in C} \left(\log(P(c)) + \sum_{i=1}^{n} f_i . \log(P(w_i|c)) \right)$$

$\hat{c}$ là kết quả phân loại của văn bản d.

5 Áp dụng mô hình Multinomial Naive Bayes trên dữ liệu Enron-Spam

5.1 Kết quả trên tập train

Metric	Giá trị
Accuracy	99.5785%
Precision	99.6460%
Recall	99.5237%
F1-Score	99.5848%

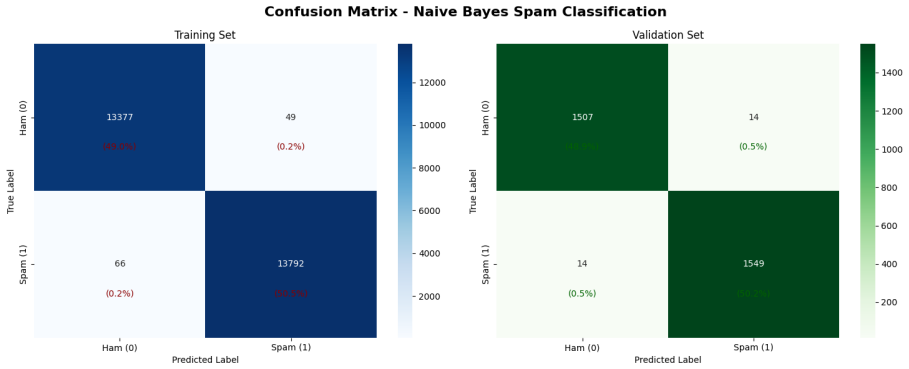
Bảng 2: Hiệu suất trên tập huấn luyện

5.2 Kết quả trên tập validation

Metric	Giá trị
Accuracy	99.0921%
Precision	99.1043%
Recall	99.1043%
F1-Score	99.1043%

Bảng 3: Hiệu suất trên tập validation

5.3 Confusion matrices



Hình 4: Confusion matrices

5.4 Đánh giá hiệu suất và ý nghĩa các metrics

1. Accuracy (Độ chính xác):

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

- *Ý nghĩa*: Tỷ lệ dự đoán đúng trên tổng số mẫu
- *Trong spam detection*: Tỷ lệ email được phân loại đúng
- *Kết quả*: Train: 99.58%, Validation: 99.09%
- *Đánh giá*: Rất tốt - mô hình dự đoán đúng >99% email

2. Precision (Độ chính xác dương):

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

- *Ý nghĩa*: Trong số email được dự đoán là spam, bao nhiêu % thực sự là spam
- *Trong spam detection*: Tỷ lệ "không báo động giả" - tránh chặn nhầm email quan trọng
- *Kết quả*: Train: 99.64%, Validation: 99.10%
- *Đánh giá*: Rất tốt - ít email ham bị nhầm là spam (chỉ 0.9% false alarm)

3. Recall (Độ nhạy):

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

- *Ý nghĩa*: Trong số email thực sự là spam, bao nhiêu % được phát hiện đúng
- *Trong spam detection*: Khả năng "bắt được spam" - tránh để spam lọt vào inbox
- *Kết quả*: Train: 99.52%, Validation: 99.10%
- *Đánh giá*: Rất tốt - phát hiện được >99% spam (chỉ 0.9% spam bị bỏ sót)

4. F1-Score (Điểm F1):

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

- *Ý nghĩa*: Trung bình điều hòa của Precision và Recall
- *Trong spam detection*: Cân bằng giữa việc không bỏ sót spam và không chặn nhầm ham
- *Kết quả*: Train: 99.58%, Validation: 99.10%
- *Đánh giá*: Rất tốt - cân bằng tối ưu giữa precision và recall
- Mô hình đạt độ chính xác rất cao trên cả hai tập dữ liệu
- Chênh lệch nhỏ giữa train và validation cho thấy mô hình ít overfitting
- Precision và Recall cân bằng, cho thấy mô hình phân loại tốt cả hai class

Tài liệu

[1] Metsis, V., Androutsopoulos, I., & Paliouras, G. (2006). Spam filtering with naive bayes-which naive bayes? In CEAS (pp. 27-28).

[2] Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge university press.

[3] Mitchell, T. M. (1997). Machine learning. McGraw-hill.

[4] Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press.

[5] Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830.

[6] Stanford CS229. (2018). CS229 Machine Learning - Naive Bayes [Video]. *YouTube*. Retrieved from `https://www.youtube.com/watch?v=nt63k3bfXS0`

[7] Vũ, H. T. (2017, August 8). Naive Bayes Classifier. *Machine Learning cơ bản*. Retrieved from `https://machinelearningcoban.com/2017/08/08/nbc/`