TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

ĐẠI HỌC QUỐC GIA TP.HCM

NHẬN DIỆN VÀ PHÂN LOẠI PHÁT NGÔN

Môn học: Python cho Khoa học Dữ liệu

Giảng viên hướng dẫn: ThS. Hà Minh Tuấn

Nhóm sinh viên thực hiện:

Nguyễn Thị Ngọc Anh	23280037
Trương Thị Quỳnh Giang	23280052
Trần Trung Kiên	23280066

TP.HCM, 2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục tiêu và nội dung của đồ án

2. Kết quả đạt được

3. Ý thức làm việc của sinh viên:

TP. Hồ Chí Minh, ngày ... tháng ... năm 2025

Giảng viên hướng dẫn

ThS. Hà Minh Tuấn

LỜI CẢM ƠN

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn tới ThS. Hà Minh Tuấn, giảng viên Bộ môn Ứng dụng Tin học, Khoa Toán Tin, Trường Đại học Khoa học Tự nhiên – ĐHQG TP. HCM đã tận tình hướng dẫn và chỉ bảo trong quá trình thực hiện đồ án.

Bên cạnh đó, nhóm chúng em xin chân thành cảm ơn nhóm nghiên cứu khoa học về Xử lý Ngôn ngữ Tự nhiên (NLP) tại Trường Đại học Công nghệ Thông tin – ĐHQG TP. HCM, *The UIT NLP Group*, đã có những nghiên cứu và khảo sát những tập dữ liệu được phát hành công khai và rộng rãi. Không những thế, chúng em xin cảm ơn các anh chị khóa trên – những người đi trước đã để lại những kết quả và kinh nghiệm quý báu cho nhóm chúng em noi theo.

Do còn nhiều thiếu sót về kiến thức, kỹ năng cũng như kinh nghiệm thực tế, bài báo cáo của nhóm chúng em không thể tránh khỏi những sai sót. Nhóm chúng em rất mong nhận được những ý kiến đóng góp, phê bình của thầy để bài báo cáo ngày càng hoàn thiện hơn.

Quá trình thực hiện Đồ án kết thúc môn học Python cho Khoa học Dữ liệu đã giúp chúng em nhận ra giá trị của việc chủ động tìm tòi học hỏi và khả năng làm việc nhóm. Chúng em rất hy vọng rằng trong các dự án tiếp theo, chúng em sẽ vẫn được hướng dẫn và đồng hành cùng thầy.

Một lần nữa, chúng em xin bày tỏ lòng biết ơn sâu sắc và cảm kích đến sự tận tâm và hỗ trợ đắc lực của thầy đã giúp chúng em vượt qua khó khăn và tiến bộ trong quá trình học tập.

Nhóm chúng em xin chân thành cảm ơn!

Mục lục

Danh sách bảng

Danh sách hình vẽ

Tóm tắt

Nhận diện và phân loại phát ngôn là một trong những bài toán quan trọng của xử lý ngôn ngữ tự nhiên, nơi hệ thống cần xác định đặc trưng và sắc thái biểu đạt của các phát ngôn được tạo ra trong môi trường trực tuyến. Các phát ngôn có thể mang nhiều dạng nội dung khác nhau như tích cực, trung tính hoặc tiêu cực, và thường đa dạng và phức tạp. Điều này khiến bài toán phân loại trở nên khó xử lý, đòi hỏi mô hình phải nắm bắt được ý nghĩa, mục đích và đặc điểm ngôn ngữ ẩn sau từng câu.

Trong lĩnh vực xử lý ngôn ngữ tự nhiên, các phương pháp học máy truyền thống giữ vai trò quan trọng nhờ tính ổn định và khả năng khái quát hóa tốt trong nhiều bài toán phân loại văn bản. Các mô hình này dựa trên việc học các đặc trưng ngôn ngữ được trích xuất từ dữ liệu, sau đó sử dụng những đặc trưng đó để suy luận và phân tách các nhóm phát ngôn khác nhau. Cách tiếp cận này cho phép mô hình nắm bắt được những tín hiệu ngữ nghĩa cốt lõi trong văn bản, hỗ trợ việc nhận diện nội dung và sắc thái biểu đạt một cách nhất quán.

Trong đồ án này, ba thuật toán phân loại phổ biến gồm Support Vector Machine (SVM), Naive Bayes và Logistic Regression được triển khai nhằm khảo sát khả năng nhận biết các kiểu tính chất thường gặp trong văn bản tiếng Việt trên không gian số. Các mô hình này đại diện cho những hướng tiếp cận phổ biến trong bài toán phân loại văn bản, nhờ vào khả năng học các đặc trưng ngôn ngữ quan trọng và đưa ra dự đoán ổn định trong quá trình phân tích.

Giới thiệu

Phân loại phát ngôn là một nhiệm vụ quan trọng trong phân tích nội dung người dùng trên mạng xã hội, nơi ngôn ngữ được thể hiện với nhiều sắc thái và biến thể khác nhau. Việc nhận diện mức độ và kiểu biểu đạt cảm xúc—tích cực, tiêu cực, trung tính hoặc các hình thức như mỉa mai, công kích—giúp hệ thống hiểu rõ hơn thái độ người dùng. Nhờ đó, doanh nghiệp có thể theo dõi phản hồi khách hàng, đánh giá danh tiếng thương hiệu, còn các nhà nghiên cứu có thể phân tích hành vi và xu hướng giao tiếp trực tuyến.

Kết quả đạt được trong đồ án là nền tảng để phát triển các hệ thống phân tích nâng cao hơn, phục vụ cho các bài toán như sàng lọc nội dung độc hại, dự đoán xu hướng thảo luận hay giám sát cảm xúc cộng đồng trên mạng xã hội.

Đặt vấn đề

Sự gia tăng mạnh mẽ của lượng phát ngôn trên Internet khiến nhu cầu phân tích tự động nội dung người dùng trở nên cấp thiết. Mạng xã hội, diễn đàn và các nền tảng thảo luận trực tuyến liên tục phát sinh các ý kiến phản ánh quan điểm và cảm xúc của người dùng về nhiều chủ đề.

Phân loại phát ngôn mang lại giá trị thiết thực:

Đối với người dùng: Hỗ trợ nắm bắt nhanh xu hướng thảo luận và đánh giá tổng quan cảm xúc cộng đồng.

Đối với tổ chức, doanh nghiệp: Giúp theo dõi danh tiếng thương hiệu, phát hiện chủ đề nhạy cảm và tối ưu chiến lược truyền thông.

Tuy nhiên, bài toán gặp nhiều thách thức như:

Khối lượng dữ liệu lớn: không thể xử lý thủ công.

Ngôn ngữ phức tạp: xuất hiện teencode, emoji, từ viết tắt và các biểu đạt cảm xúc không chuẩn hóa.

Do đó, việc xây dựng hệ thống phân loại phát ngôn tự động là cần thiết nhằm hỗ trợ nhận diện nhanh chóng nội dung, theo dõi xu hướng và trích xuất thông tin hữu ích từ lượng dữ liệu khổng lồ trên mạng xã hội.

Lý do chọn đề tài

Phát ngôn trực tuyến ngày càng đa dạng và ảnh hưởng sâu rộng đến cộng đồng cũng như hoạt động của doanh nghiệp. Tuy nhiên, sự bùng nổ dữ liệu cùng đặc điểm ngôn ngữ không chuẩn hóa khiến việc phân tích thủ công trở nên không khả

thi. Nội dung tiêu cực có thể lan nhanh, tác động đến danh tiếng thương hiệu và gây khó khăn cho công tác quản lý cộng đồng.

Chính vì vậy, việc nghiên cứu và xây dựng hệ thống phân loại phát ngôn tiếng Việt mang ý nghĩa thực tiễn và khoa học. Hệ thống này hỗ trợ doanh nghiệp hiểu khách hàng, giúp nhà quản lý giám sát nội dung và là công cụ phục vụ nghiên cứu hành vi ngôn ngữ. Từ nhu cầu thực tế đó, nhóm quyết định chọn đề tài "Nhận diện và phân loại phát ngôn trên mạng xã hội".

Phát biểu bài toán

Mục tiêu của nghiên cứu là xây dựng mô hình có khả năng nhận diện và phân loại các phát ngôn trên mạng xã hội tiếng Việt theo nhóm nội dung hoặc sắc thái cảm xúc, tập trung vào ba lớp chính: tích cực, tiêu cực và trung tính.

Dữ liệu sử dụng gồm gần 7.000 phát ngôn tiếng Việt đã được gán nhãn cảm xúc. Để giải quyết bài toán, đồ án triển khai và đánh giá ba mô hình học máy truyền thống: *Support Vector Machine (SVM)*, *Naive Bayes* và *Logistic Regression*.

Cấu trúc đồ án

PHẦN 1: CƠ SỞ LÝ THUYẾT

Trình bày các khái niệm liên quan đến bài toán xử lý ngôn ngữ tự nhiên, những đặc điểm của ngôn ngữ trên mạng xã hội, cùng các phương pháp phân loại văn bản cơ bản thường được sử dụng trong NLP.

PHẦN 2: THU THẬP, XỬ LÝ VÀ KHÁM PHÁ DỮ LIỆU

Mô tả quy trình thu thập dữ liệu phát ngôn từ mạng xã hội, các bước tiền xử lý văn bản như làm sạch, chuẩn hóa, xử lý emoji, teencode và stopwords, đồng thời thực hiện phân tích khám phá dữ liệu (EDA) nhằm hiểu rõ các đặc trưng của tập dữ liệu.

PHẦN 3: XÂY DỰNG MÔ HÌNH PHÂN TÍCH

Trình bày các mô hình học máy được sử dụng trong nghiên cứu gồm Support Vector Machine (SVM), Naive Bayes và Logistic Regression; mô tả cách trích xuất đặc trưng, thiết lập mô hình và quy trình huấn luyện.

PHẦN 4: KẾT QUẢ THỰC NGHIỆM

Tiến hành thử nghiệm, trình bày và so sánh kết quả mô hình dựa trên các thước đo đánh giá, trực quan hóa kết quả và đưa ra nhận xét về hiệu quả phân loại phát ngôn.

Chương 1 CƠ SỞ LÝ THUYẾT

Trình bày các khái niệm liên quan đến phương pháp vectorize dữ liệu và những mô hình áp dụng trong đồ án.

1.1 Các phương pháp vectorize dữ liệu

Trong các bài toán xử lý ngôn ngữ tự nhiên (NLP), đầu vào thường là dữ liệu tồn tại dưới dạng văn bản thô (raw text). Tuy nhiên, các mô hình học máy và học sâu không thể xử lý trực tiếp dữ liệu dạng ký tự hoặc chuỗi từ, mà yêu cầu dữ liệu phải được biểu diễn dưới dạng vector số trong không gian đặc trưng. Do đó, một bước quan trọng trong quy trình xử lý NLP là vector hóa dữ liệu (data vectorization) - chuyển đổi các văn bản thành các biểu diễn số học có ý nghĩa, phản ánh được thông tin ngữ nghĩa và cú pháp của văn bản.

1.1.1 TF IDF

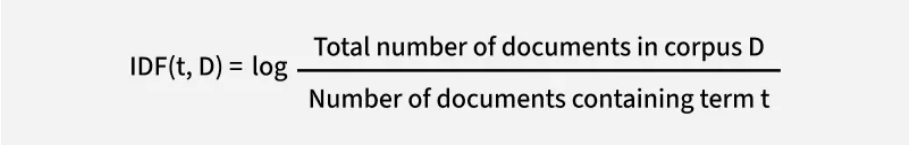
TF-IDF là một phương pháp thống kê được sử dụng rộng rãi để đánh giá mức độ quan trọng của một từ đối với một tài liệu so với một tập tài liệu lớn hơn. Phương pháp kết hợp hai thành phần:

a) Tần suất thuật ngữ (TF) Đo lường tần suất xuất hiện của một từ trong tài liệu. Tần suất càng cao thì mức độ quan trọng càng lớn. Nếu một thuật ngữ xuất hiện thường xuyên trong tài liệu, thì có thể nó liên quan đến nội dung của tài liệu.

$$\text{TF}(t, d) = \frac{\text{Number of times term t appears in document d}}{\text{Total number of terms in document d}}$$

Hình 1.1: Công thức tính TF

b) Tần suất tài liệu nghịch đảo (IDF) Giam trọng số của các từ phổ biến trong nhiều tài liệu đồng thời tăng trọng số của các từ hiếm. Nếu thuật ngữ xuất hiện trong ít tài liệu hơn, nó có nhiều khả năng có ý nghĩa và cụ thể hơn.



Hình 1.2: Công thức tính IDF

Sự cân bằng này cho phép TF-IDF làm nổi bật các thuật ngữ thường xuyên xuất hiện trong một tài liệu cụ thể và đặc biệt trong toàn bộ tài liệu văn bản, kiến nó trở thành một công cụ hữu ích cho các tác vụ như xếp hàng tìm kiếm, phân loại văn bản và trích xuất từ khóa.

Ưu điểm: Đơn giản và hiệu quả, dễ dàng triển khai trong các hệ thống xử lý văn bản

Hạn chế: Phương pháp không phân biệt được các từ đồng nghĩa dẫn đến bỏ sót thông tin quan trọng (ví dụ: "đẹp" và "xinh" là hoàn toàn khác nhau). Ngoài ra, độ dài của tài liệu có thể ảnh hưởng đến kết quả tính toán, đặc biệt khi các văn bản có độ dài không đồng đều.

1.2 Một số phương pháp khác

Bên cạnh TF-IDF, nhiều phương pháp biểu diễn văn bản khác cũng được ứng dụng trong xử lý ngôn ngữ tự nhiên bao gồm:

a) One-hot Encoding

One-hot Encoding là phương pháp chuyển đổi các biến phân loại thành định dạng nhị phân. Nó tạo ra các cột mới cho mỗi danh mục, trong đó 1 có nghĩa là danh mục đó xuất hiện và 0 có nghĩa là không xuất hiện. Số chiều của vectơ này đúng bằng số từ trong từ điển. Tuy nhiên, dữ liệu này chỉ đáp ứng được khả năng huấn luyện mà chưa phản ánh được mối liên hệ về mặt ngữ nghĩa của các từ.

b) Word2Vec

Word2Vec là mô hình học biểu diễn từ dựa trên mạng noron nông, nó học các vector liên tục phản ánh được mối quan hệ ngữ nghĩa và cú pháp giữa các từ trong không gian vector. Phương pháp này khắc phục được hạn chế của One-hot Encoding. Word2Vec gồm 2 kiến trúc chính: Skip-grams và CBOW (Continuous Bag of Words) - Skip-grams: đầu vào là một từ trung tâm, mô hình dự đoán các từ ngữ cảnh. - CBOW: ngược lại, đầu vào là các từ ngữ cảnh, mô hình dự đoán từ trung tâm Hạn chế của Word2Vec: - Không xử lý được đa dạng ngữ cảnh - không thay đổi theo câu. Ví dụ: "con chuột" có ngữ nghĩa khác nhau ở các ngữ cảnh khác nhau như "con chuột máy tính này thật đẹp" và "con chuột này to thật". Word2Vec tìm ra 1 vector đại diện cho mỗi từ dựa trên tập ngữ liệu lớn nên không thể hiện được sự đa dạng của ngữ cảnh. - Không xử lý được những từ không tồn tại trong từ điển từ vựng (OOV - out of vocabulary), tức là nếu xuất hiện một từ mới, mô hình sẽ không tạo được vector.

c) PhoBert

PhoBert là mô hình dựa trên BERT (Bidirectional Encoder Representations from Transformer) được giới thiệu vào năm 2020 và được xem là bước tiến quan trọng cho nhiều bài toán như Question Answering, Sentiment Analysis,... Đây là một mô hình huấn luyện cho đơn ngôn ngữ (monolingual language), tức là toàn bộ quá trình huấn luyện được thực hiện trên kho dữ liệu tiếng Việt quy mô lớn, giúp mô hình nắm bắt tốt các đặc trưng ngữ pháp và ngữ nghĩa của tiếng Việt. Nó đã cải thiện được hạn chế của Word2Vec trong việc cung cấp biểu diễn theo ngữ cảnh (contextualized embeddings) - cho phép một từ có vector khác nhau tùy thuộc vào ngữ cảnh xuất hiện.

d) FastText

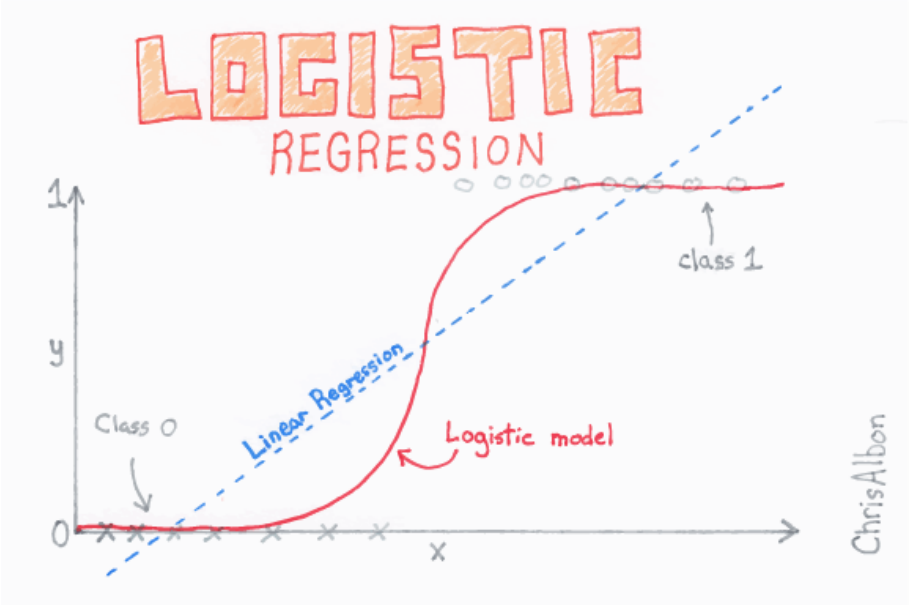
FastText là một mô hình mở rộng của Word2Vec, cung cấp các embedding cho các từ nằm ngoài từ điển. Bạn đọc có thể tham khảo thêm ở đây:

https://www.geeksforgeeks.org/machine-learning/fasttext-working-and-implementation/

1.3 Các mô hình học máy

1.3.1 Logistic Regression

Logistic Regression (LR) là một trong ba thuật toán phân loại phổ biến, nhằm khảo sát khả năng nhận biết các kiểu tính chất thường gặp trong văn bản tiếng Việt trên không gian số. Mô hình này đại diện cho hướng tiếp cận học máy giám sát (supervised machine learning), được lựa chọn nhờ vào khả năng học các đặc trưng ngôn ngữ quan trọng và đưa ra dự đoán ổn định trong quá trình phân tích. LR được gọi là hồi quy (regression) nhưng thực hiện phân loại (classification). LR thường được sử dụng khi biến phụ thuộc không phải là giá trị số mà là phản hồi dạng nhị phân, ví dụ như phản hồi "yes/no". Hàm Logistic Sigmoid được áp dụng cho kết quả hồi quy để lấy xác suất thuộc về một trong hai lớp. LR phân loại biến phụ thuộc vào một trong các lớp dựa trên xác suất cao hơn



Hình 1.3: Logistic Regression

Nguồn hình ảnh: https://builtin.com/data-science/supervised-machine-learning-classification

1. **Phương pháp Tối ưu Tham số (sử dụng `LogisticOptimizer`)**

   Quá trình tối ưu hóa siêu tham số (hyperparameters) cho mô hình LR được thực hiện thông qua lớp `LogisticOptimizer`. Lớp này sử dụng phương pháp tìm kiếm lưới `GridSearchCV` kết hợp đánh giá chéo (k-fold Cross-Validation) để xác định tập siêu tham số tốt nhất. Lưới tham số được định nghĩa để dò tìm bao gồm:

   - `C`: Độ mạnh của Regularization, thử nghiệm các giá trị `[0.1, 1, 10, 100]`.
   - `solver`: Thuật toán tối ưu, thử nghiệm các giá trị `['lbfgs', 'liblinear']`.

   Thước đo được sử dụng trong `GridSearchCV` là `f1_macro`. Việc sử dụng `f1_macro` giúp đảm bảo mô hình học tốt tất cả các lớp, đặc biệt trong trường hợp dữ liệu bị mất cân bằng.

   Sau khi tối ưu, các tham số tốt nhất (`best_params_`) và điểm số cao nhất (`best_score_`) được thu thập. Hàm `train_best_model` trong `LogisticOptimizer` sau đó sẽ tạo đối tượng `LogisticRegressionModel` và huấn luyện mô hình bằng các tham số tối ưu này.

2. **Phương pháp Huấn luyện Mô hình (sử dụng `LogisticRegressionModel`)**

   Module `LogisticRegressionModel` triển khai quy trình huấn luyện và đánh giá mô hình LR.

   - **Chia dữ liệu:** Module sử dụng phương thức `split_data` để chia tập dữ liệu đầu vào `(X, y)` thành:

  - tập huấn luyện: `_X_train`, `_y_train`
  - tập kiểm tra: `_X_test`, `_y_test`

  Việc chia dữ liệu được thực hiện bằng hàm `train_test_split`. Đặc biệt, kỹ thuật phân tầng (`stratify = y`) được áp dụng để đảm bảo phân bố nhãn đồng đều giữa tập huấn luyện và tập kiểm tra.

- **Huấn luyện:** Nếu không sử dụng bộ tối ưu, mô hình sẽ được huấn luyện với các tham số mặc định:
  - `max_iter = 1000`
  - `solver = 'lbfgs'`

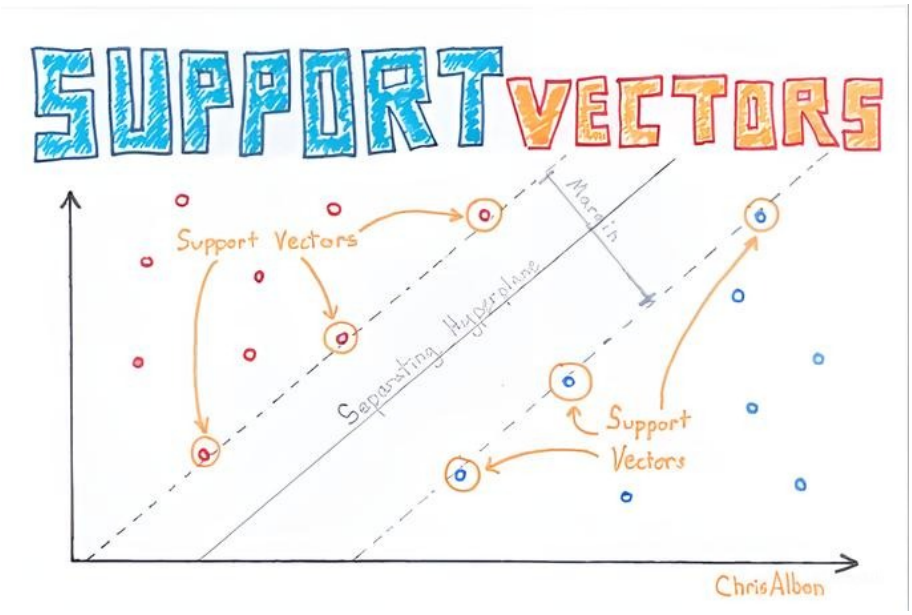
  Khi sử dụng tham số tối ưu, mô hình `LogisticRegression` được khởi tạo bằng các giá trị tốt nhất và huấn luyện trên tập huấn luyện.

- **Đánh giá:** Sau khi huấn luyện, mô hình được đánh giá trên tập kiểm tra `_X_test`. Các thước đo đánh giá bao gồm:
  - Độ chính xác: `accuracy_score`
  - Báo cáo phân loại chi tiết: `classification_report`
  - Ma trận nhầm lẫn: `confusion_matrix`

  Độ chính xác (Accuracy) phản ánh tỷ lệ dự đoán đúng của mô hình, trong khi ma trận nhầm lẫn (confusion matrix) cho biết chi tiết số lượng mẫu dự đoán đúng/sai cho từng lớp.

1.3.2 Support Vector Machine

Support Vector Machine (SVM) là mô hình học máy giám sát, mục tiêu của thuật toán là tìm ra một siêu phẳng (hyperplane) sao cho có thể phân tách tối ưu các điểm dữ liệu thuộc các lớp khác nhau. "Tối ưu" ở đây nghĩa là tìm ra siêu phẳng tạo ra khoảng cách lớn nhất (margin) giữa các lớp dữ liệu. Các điểm dữ liệu có khoảng cách nhỏ nhất đến siêu mặt phẳng (các điểm gần nhất) được gọi là các vector hỗ trợ (support vectors) Margin là khoảng không gian giữa hai siêu phẳng hỗ trợ, dải này càng rộng thì mô hình SVM càng tốt. Dải đó không có điểm dữ liệu nào nằm trong và song song hoàn toàn với siêu phẳng phân loại



Hình 1.4: Support Vector Machine

SVM không chỉ hỗ trợ phân loại nhị phân và tách các điểm dữ liệu thành hai lớp, SVM còn mở rộng để hỗ trợ các bài toán phân loại đa lớp thông qua cơ chế chia nhỏ bài toán đa lớp thành nhiều mô hình nhị phân.

a) Phương pháp OvO (One-vs-One)

Phương pháp OvO xây dựng bộ phân loại nhị phân cho mỗi cặp lớp trong tập dữ liệu. Với tập dữ liệu gồm K lớp, số lượng mô hình được tạo ra là:

$$N_{\text{mô hình}} = \frac{K(K-1)}{2}.$$

Mỗi mô hình được huấn luyện để phân biệt hai lớp cụ thể. Khi dự đoán, mỗi mô hình đưa ra một phiếu bầu cho một trong hai lớp, và lớp nhận được nhiều phiếu nhất sẽ được chọn làm kết quả cuối cùng. Phương pháp OvO thường hiệu quả khi số lượng lớp không quá lớn và mỗi mô hình nhị phân tương đối nhẹ.

Ví dụ: ta có 3 lớp phân loại *Red*, *Blue* và *Green* và một mẫu mới x. Phương pháp OvO sẽ tạo ra:

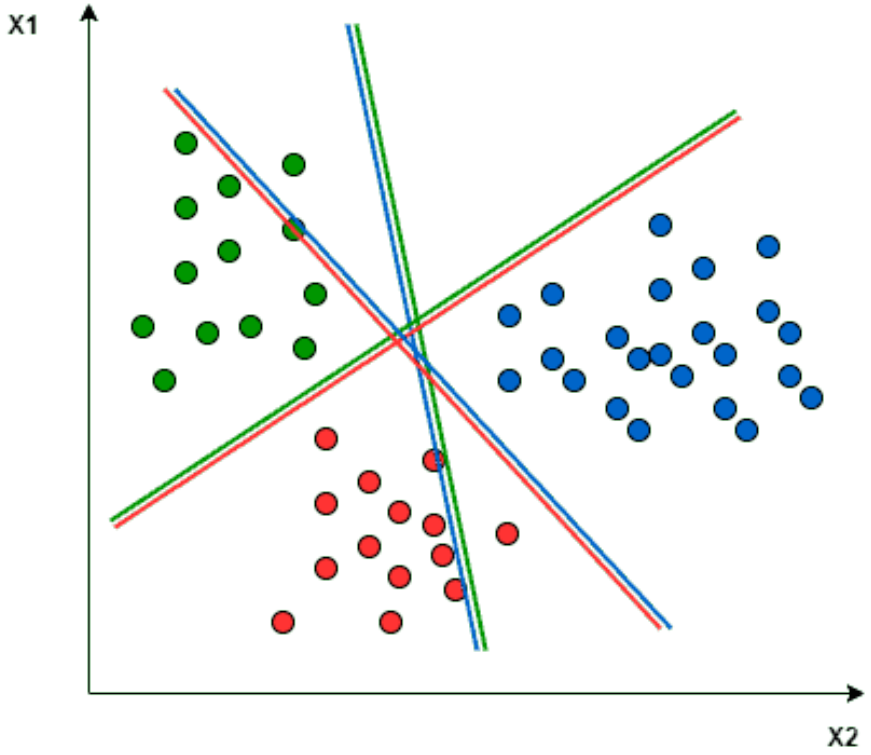
$$\frac{3(3-1)}{2} = 3$$

mô hình nhị phân:

- Mô hình 1: *Red* vs *Blue*
  Nếu mô hình dự đoán *Red* $\Rightarrow$ *Red* nhận 1 phiếu, ngược lại *Blue* nhận 1 phiếu.
- Mô hình 2: *Red* vs *Green*
  Nếu mô hình dự đoán *Red* $\Rightarrow$ *Red* nhận 1 phiếu, ngược lại *Green* nhận 1 phiếu.

- Mô hình 3: *Blue* vs *Green*
  Nếu mô hình dự đoán *Blue* ⇒ *Blue* nhận 1 phiếu, ngược lại *Green* nhận 1 phiếu.

Sau cùng, mẫu x được gán vào lớp có nhiều phiếu bầu nhất. Nếu xảy ra trường hợp hòa phiếu, có thể dùng *decision function* để phá hòa.



Hình 1.5: Mô phỏng phương pháp One-vs-One (OvO)

b) Phương pháp OvR (One-vs-Rest)

Phương pháp OvR (OvA) chia dữ liệu đa lớp thành K bài toán phân loại nhị phân độc lập, trong đó K là số lượng lớp. Với mỗi lớp C_k, mô hình phân loại được xây dựng để phân biệt lớp C_k với $(K-1)$ lớp còn lại. Mỗi mô hình con học cách nhận diện một lớp cụ thể.

Khi dự đoán một mẫu mới x, mỗi mô hình con sẽ trả về một giá trị hàm quyết định (*decision score*) — tức khoảng cách có dấu từ điểm x đến siêu phẳng phân tách của mô hình đó. Score càng lớn cho thấy mô hình càng "tự tin" rằng mẫu thuộc lớp C_k.

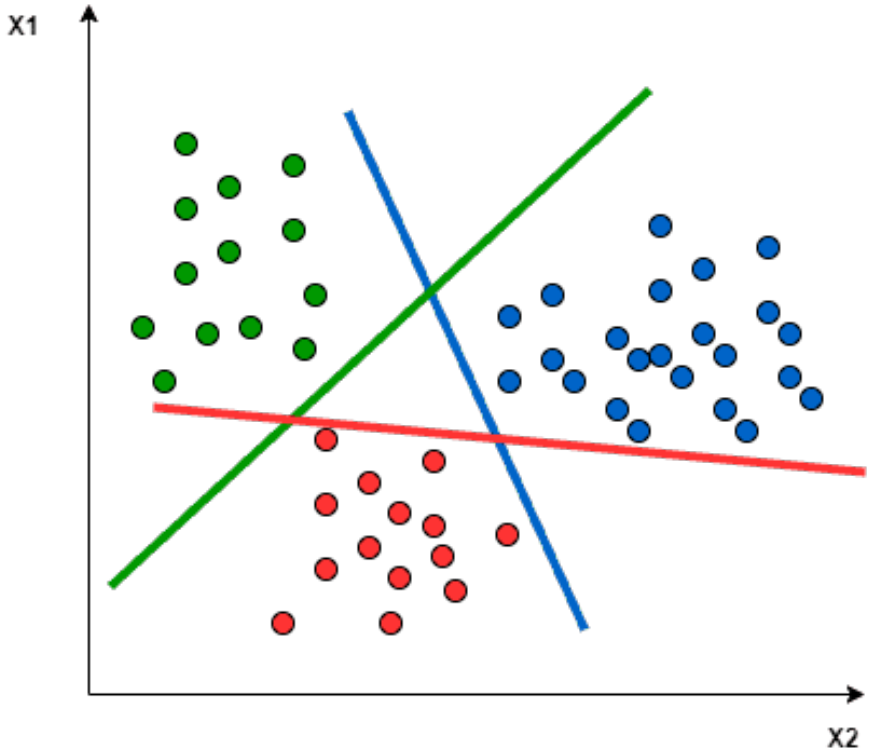
Ví dụ: ta có 3 lớp phân loại *Red*, *Blue* và *Green*, và một mẫu mới x. Phương pháp OvR sẽ tạo ra 3 mô hình con như sau:

- Mô hình 1: *Red* vs (*Blue*, *Green*) → output: $\text{score}_{\text{Red}}(x)$
- Mô hình 2: *Blue* vs (*Red*, *Green*) → output: $\text{score}_{\text{Blue}}(x)$
- Mô hình 3: *Green* vs (*Red*, *Blue*) → output: $\text{score}_{\text{Green}}(x)$

Sau khi tính toán score cho mẫu mới x, lớp dự đoán được xác định theo quy tắc:

$$\hat{y} = \arg\max_k \ \text{score}_k(x),$$

tức là mẫu x được gán vào lớp có giá trị score cao nhất.



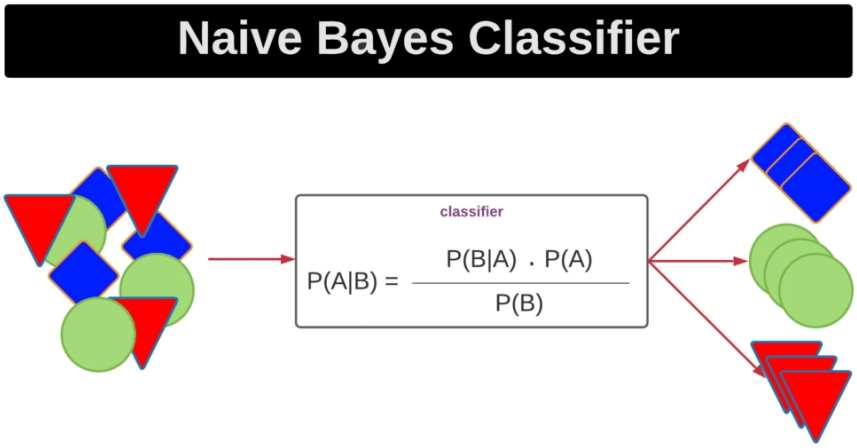
Hình 1.6: Mô phỏng phương pháp OvR

1.3.3 Naive Bayes

Naive Bayes là một mô hình phân loại xác suất dựa trên Định lý Bayes, hoạt động bằng cách ước lượng mức độ "khả dĩ" của từng lớp đối với mẫu dữ liệu mới dựa trên thông tin thống kê thu được từ tập huấn luyện. Nhờ cơ chế dựa trên xác suất, mô hình cho phép thực hiện phân loại một cách hiệu quả ngay cả trong không gian đặc trưng có số chiều lớn.

Cốt lõi của Naive Bayes nằm ở giả định *độc lập có điều kiện* giữa các đặc trưng: các đặc trưng $x_1, x_2, \ldots, x_n$ được xem là độc lập với nhau khi biết lớp C. Giả định này giúp đơn giản hoá quá trình ước lượng, bởi mô hình chỉ cần xem xét từng đặc trưng riêng lẻ để đánh giá mức độ phù hợp giữa mẫu và lớp, mặc dù điều này hiếm khi hoàn toàn đúng trong thực tế.

Trong quá trình dự đoán, mô hình ước lượng *xác suất tiên nghiệm* $P(C)$ của từng lớp và *xác suất xuất hiện của mỗi đặc trưng khi biết lớp đó* $P(x_i \mid C)$. Khi một mẫu mới X được đưa vào, Naive Bayes kết hợp hai loại xác suất này để xác định xác suất hậu nghiệm $P(C \mid X)$ và gán mẫu vào lớp có giá trị cao nhất, tức là lớp có khả năng giải thích tốt nhất các đặc trưng quan sát được.



Hình 1.7: Minh họa cơ chế phân loại của Naive Bayes dựa trên Định lý Bayes

Nguồn: https://mlarchive.com/machine-learning/the-ultimate-guide-to-naive-bayes/

Trong thực tế, việc ước lượng các xác suất $P(x_i \mid C)$ được thực hiện khác nhau tùy thuộc vào bản chất của dữ liệu, thông qua các biến thể phổ biến của Naive Bayes như Bernoulli, Multinomial hoặc Gaussian.

a) Gaussian Naive Bayes

Gaussian Naive Bayes là một dạng của Naive Bayes dùng cho dữ liệu liên tục. Mô hình giả định rằng mỗi đặc trưng x_i tuân theo phân phối chuẩn (Gaussian) khi biết lớp ω. Khi đó, xác suất có điều kiện của một đặc trưng được mô hình hoá bởi hàm mật độ Gaussian:

$$P(x_{ik} \mid \omega) = \frac{1}{\sqrt{2\pi\sigma_\omega^2}} \exp\left(-\frac{(x_{ik} - \mu_\omega)^2}{2\sigma_\omega^2}\right),$$

trong đó μ_ω và σ_ω được ước lượng từ dữ liệu huấn luyện.

Dưới giả định độc lập có điều kiện, xác suất lớp có điều kiện cho mẫu nhiều chiều được tính bằng tích các xác suất thành phần:

$$P(x \mid \omega) = \prod_{k=1}^{d} P(x_{ik} \mid \omega).$$

Do tuân theo giả định độc lập có điều kiện giữa các đặc trưng, Gaussian Naive Bayes có thể tính toán nhanh và hiệu quả ngay cả khi số chiều dữ liệu lớn. Sau khi mô hình được huấn luyện, việc phân loại mẫu mới được thực hiện bằng cách kết hợp xác suất tiên nghiệm của từng lớp với xác suất có điều kiện nói trên, và lựa chọn lớp có xác suất hậu nghiệm lớn nhất. Nhờ cấu trúc giải tích đơn giản và quá trình dự đoán nhanh, Gaussian Naive Bayes được xem là một mô hình *eager learner* - tức mô hình học từ dữ liệu một lần, sau đó có thể dự đoán tức thời mà không cần truy cập lại toàn bộ tập huấn luyện

b) Multinomial Naive Bayes

Multinomial Naive Bayes là một dạng của Naive Bayes được thiết kế cho dữ liệu rời rạc có dạng tần suất xuất hiện, đặc biệt phù hợp với bài toán phân loại văn bản. Mô hình sử dụng số lần một từ x_i xuất hiện trong tài liệu để ước lượng xác suất có điều kiện khi biết lớp ω_j. Tần suất thô thường được chuẩn hoá theo độ dài tài liệu và được dùng để tính xấp xỉ tối đa hợp lý như sau:

$$\hat{P}(x_i \mid \omega_j) = \frac{\sum tf(x_i, d \in \omega_j) + \alpha}{\sum N_{d\in\omega_j} + \alpha V},$$

trong đó $tf(x_i, d \in \omega_j)$ là tổng số lần xuất hiện của từ x_i trong các tài liệu thuộc lớp ω_j, V là kích thước từ vựng và α là tham số làm trơn. Dưới giả định độc lập có điều kiện, xác suất lớp có điều kiện của một tài liệu được tính bằng:

$$P(x \mid \omega_j) = \prod_{i=1}^{m} P(x_i \mid \omega_j).$$

Nhờ cách mô hình hóa dựa trên tần suất, Multinomial Naive Bayes thường đạt hiệu quả cao trong phân loại văn bản, đặc biệt khi sự khác biệt về phân bố từ giữa các lớp là rõ rệt.

c) Bernoulli Naive Bayes

Bernoulli Naive Bayes là một dạng của Naive Bayes được thiết kế cho dữ liệu nhị phân, nơi mỗi đặc trưng cho biết một từ *xuất hiện* (1) hoặc *không xuất hiện* (0) trong tài liệu. Mỗi đặc trưng được mô hình hoá như một thử nghiệm Bernoulli, trong đó xác suất xuất hiện của từ x_i trong lớp ω_j được ước lượng dựa trên tỉ lệ số tài liệu thuộc lớp đó có chứa từ x_i, kèm theo làm trơn Laplace để tránh xác suất bằng 0.

Mô hình tính xác suất có điều kiện của một tài liệu x theo công thức Bernoulli:

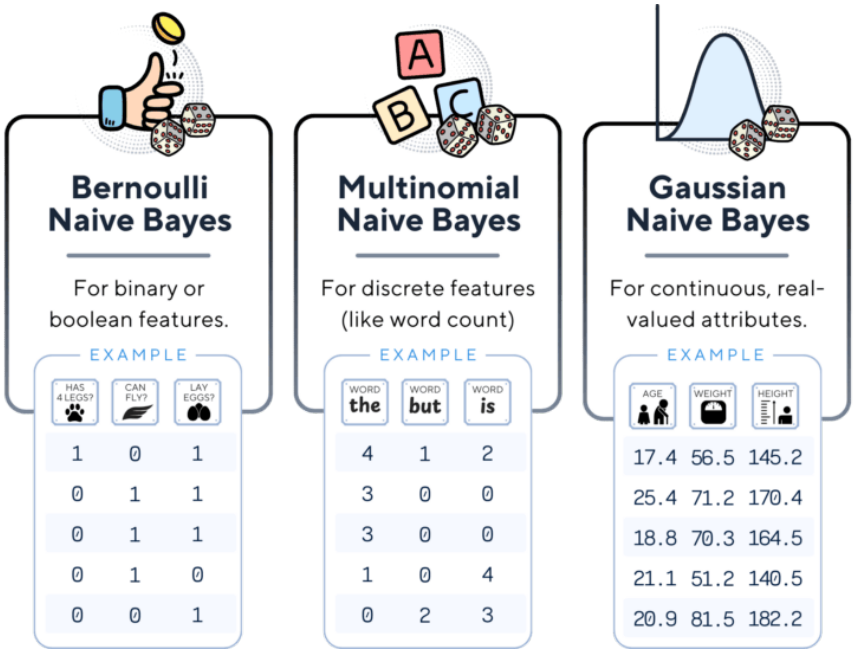
$$P(x \mid \omega_j) = \prod_{i=1}^{m} P(x_i \mid \omega_j)^{b_i} \left(1 - P(x_i \mid \omega_j)\right)^{(1-b_i)}, \qquad b_i \in \{0, 1\},$$

trong đó ước lượng xác suất xuất hiện của từ được tính bởi:

$$\hat{P}(x_i \mid \omega_j) = \frac{df_{x_i,j} + 1}{df_j + 2},$$

với $df_{x_i,j}$ là số tài liệu thuộc lớp ω_j có chứa từ x_i, còn df_j là tổng số tài liệu của lớp ω_j.

Trong quá trình dự đoán, mô hình tính xác suất hậu nghiệm $P(\omega_j \mid x)$ và chọn lớp có giá trị lớn nhất. Bernoulli Naive Bayes đặc biệt phù hợp cho phân loại văn bản nhị phân (chẳng hạn phát hiện spam), nơi sự *xuất hiện* của từ mang ý nghĩa quan trọng hơn tần suất.



Hình 1.8: Ba biến thể phổ biến của Naive Bayes: Bernoulli, Multinomial và Gaussian

Nguồn: https://towardsdatascience.com/bernoulli-naive-bayes-explained-a-visual-guide-with-code-examples-for-beginners-aec39771ddd6/

Chương 2 DỮ LIỆU VÀ CÔNG CỤ

2.1 Giới thiệu bộ dữ liệu

Dữ liệu được thu thập từ **The UIT NLP Group** – một nhóm nghiên cứu khoa học về Xử lý Ngôn ngữ Tự nhiên (NLP) tại Trường Đại học Công nghệ Thông tin – ĐHQG TP. HCM. Đây là nguồn dữ liệu uy tín, đã được xây dựng và công bố phục vụ cho các nghiên cứu học thuật về phân tích văn bản tiếng Việt.

Nhóm tác giả đã xây dựng một tập dữ liệu chuẩn về **Cảm xúc trên Mạng xã hội Tiếng Việt (UIT-VSMEC)** với khoảng **6.927 câu** được dán nhãn thủ công theo 7 cảm xúc:

1. Enjoyment
2. Sadness
3. Fear
4. Anger
5. Surprise
6. Disgust
7. Other

Đồng thời, nhóm nghiên cứu cũng đã tiến hành đánh giá hiệu năng của các mô hình học máy và mạng nơ-ron sâu trên bộ dữ liệu này.

Tuy nhiên, trong khuôn khổ môn học, nhóm xin phép sử dụng bộ dữ liệu UIT-VSMEC làm đầu vào và tiến hành tinh giản, điều chỉnh một số trường thông tin nhằm giảm độ phức tạp và đảm bảo phù hợp với mục tiêu bài toán phân loại của đồ án, đồng thời vẫn giữ được tính đại diện của dữ liệu gốc.

Link tài liệu gốc: `https://drive.google.com/drive/folders/1HooABJyrddVGzll7fgkJ6VzkG_XuWfRu`

Bạn đọc có thể tham khảo thêm bài nghiên cứu: Vong Ho, Duong Nguyen, Danh Nguyen, Linh Pham, Kiet Nguyen and Ngan Nguyen, Emotion Recognition for Vietnamese Social Media Text, 2019 16th International Conference of the Pacific Association for Computational Linguistics (PACLING 2019), October 11-13, 2019, Ha Noi, Vietnam.

Link bài nghiên cứu: `https://arxiv.org/pdf/1911.09339`

Bộ dữ liệu gồm 3 cột: Unnamed, Emotion và Sentence Dữ liệu gốc:

Bảng 2.1: Ví dụ một số câu trong bộ dữ liệu UIT-VSMEC

ID	**Emotion**	**Sentence**
941	Other	tính tao tao biết , chẳng có chuyện gì có thể làm tao phát điên cả
142	Enjoyment	lại là lao cai , tự hào quê mình quá :))
1164	Sadness	bị từ chối rồi
479	Anger	thế đường không cấm đỗ , bạn lấy quyền gì không cho người ta đỗ vậy
185	Fear	tao sợ ăn xong nóng đêm gãi toét chân
1082	Disgust	bạn không có quyền đuổi người ta . khó chịu thì gọi ca giao thông thôi . bạn đuổi họ bạn sai lè lè
1618	Surprise	nhìn mẫu này lạ mắt mà đẹp quá

Nhóm điều chỉnh nhãn dán phân loại như sau:
Tích cực bao gồm các nhãn *Enjoyment*, *Surprise*.
Tiêu cực bao gồm các nhãn *Fear*, *Sadness*, *Anger*, *Disgust*.
Trung tính bao gồm nhãn *Other*.

Ghi chú: nhãn *Surprise* có thể mang hai sắc thái tiêu cực và tích cực, tuy nhiên, qua phân tích ngữ nghĩa và tần suất trong dữ liệu, đa số các câu gán nhãn *Surprise* mang khuynh hướng cảm xúc tích cực. Do đó, nhóm quyết định gộp *Surprise* vào nhóm **Tích cực**.

Bài toán trở thành phân loại cảm xúc bình luận trên mạng xã hội tiếng Việt với 3 nhãn **Tích cực**, **Tiêu cực** và **Trung tính**. Việc gộp nhãn vẫn đảm bảo tính đại diện của dữ liệu, đồng thời giúp mô hình dễ học hơn.

Dữ liệu sau khi gộp nhãn:

Bảng 2.2: Ví dụ dữ liệu sau khi gộp nhãn

ID	**Emotion**	**Sentence**	**new_emotion**
941	Other	tính tao tao biết , chẳng có chuyện gì có thể làm tao phát điên cả	Trung tính
142	Enjoyment	lại là lao cai , tự hào quê mình quá :))	Tích cực
1164	Sadness	bị từ chối rồi	Tiêu cực
479	Anger	thế đường không cấm đỗ , bạn lấy quyền gì không cho người ta đỗ vậy	Tiêu cực
185	Fear	tao sợ ăn xong nóng đêm gãi toét chân	Tiêu cực
1082	Disgust	bạn không có quyền đuổi người ta . khó chịu thì gọi ca giao thông thôi . bạn đuổi họ bạn sai lè lè	Tiêu cực
1618	Surprise	nhìn mẫu này lạ mắt mà đẹp quá	Tích cực

2.2 Khám phá bộ dữ liệu

2.2.1 Kiểm tra dữ liệu trống

```
data.info()
```

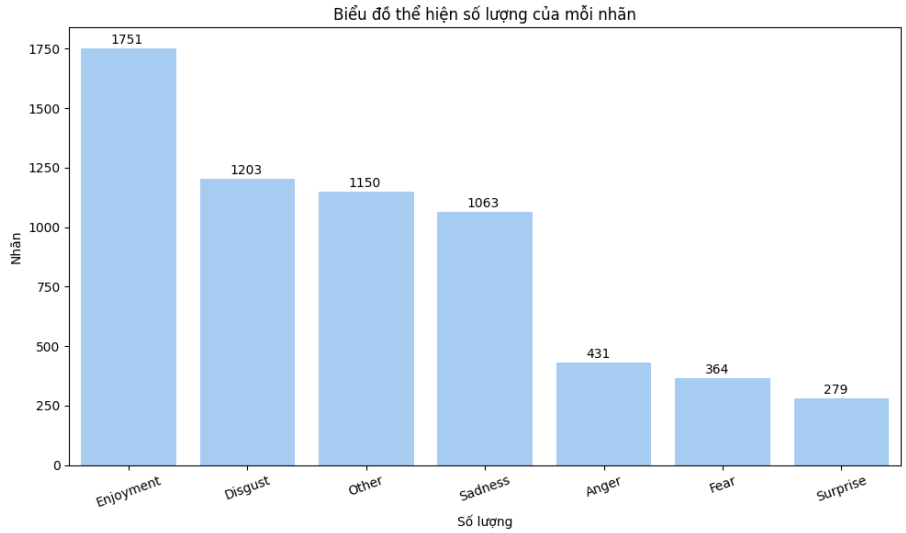
[4] ✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6241 entries, 0 to 6240
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  6241 non-null   int64
 1   Emotion     6241 non-null   object
 2   Sentence    6241 non-null   object
dtypes: int64(1), object(2)
memory usage: 146.4+ KB
```

Hình 2.1: Kiểm tra dữ liệu trống

Dữ liệu gồm 6241 dòng, trong đó không có dòng nào bị khuyết dữ liệu

2.2.2 Kiểm tra phân phối nhãn

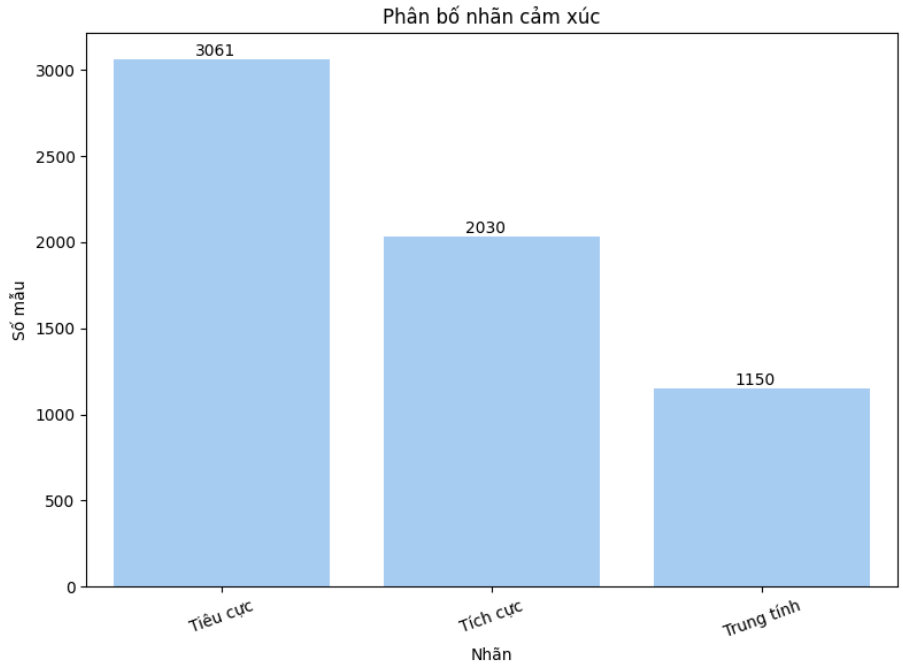


Hình 2.2: Biểu đồ thể hiện số lượng của mỗi nhãn dán của tập dữ liệu gốc

```
Emotion
Enjoyment    28.056401
Disgust      19.275757
Other        18.426534
Sadness      17.032527
Anger         6.905945
Fear          5.832399
Surprise      4.470437
Name: proportion, dtype: float64
```

Hình 2.3: Thống kê tỉ lệ của mỗi nhãn dán trong dữ liệu

Như đã nói ở mục trên, nhóm quyết định gộp nhãn dán với 3 nhãn: Tích cực, Tiêu cực và Trung tính



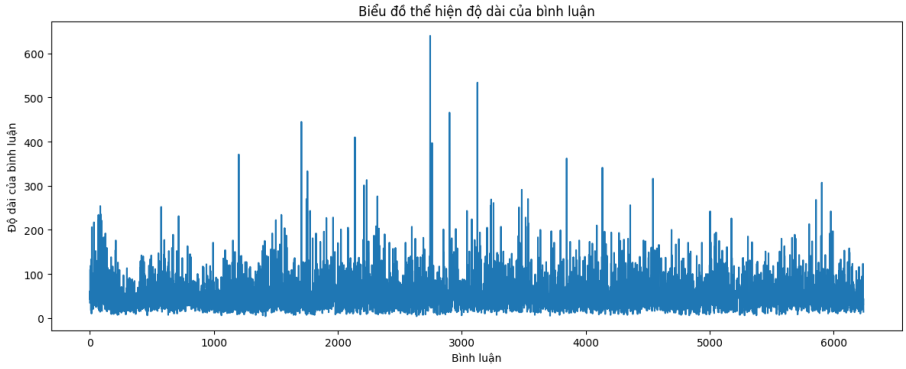
Hình 2.4: Biểu đồ thể hiện số lượng của mỗi nhãn dán sau khi gộp nhãn

```
new_emotion
Tiêu cực      49.046627
Tích cực      32.526839
Trung tính    18.426534
Name: proportion, dtype: float64
```

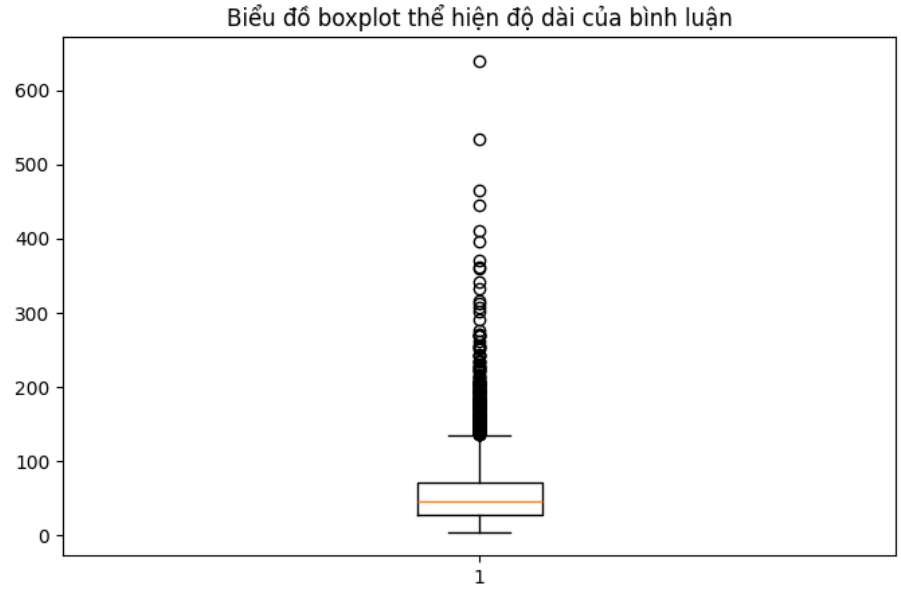
Hình 2.5: Thống kê tỉ lệ của mỗi nhãn dán sau khi gộp

NHẬN XÉT: sau khi gộp nhãn, dữ liệu đang có sự mất cân bằng giữa các nhãn, điều này sẽ khiến cho mô hình học thiên vị => cần cân bằng dữ liệu trước khi huấn luyện mô hình.

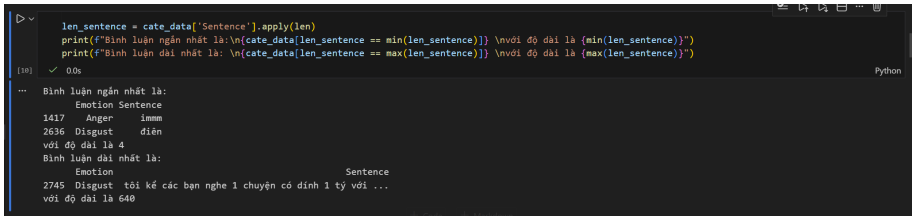
2.2.3 Kiểm tra độ dài chuỗi



Hình 2.6: Biểu đồ thể hiện độ dài mỗi văn bản đầu vào



Hình 2.7: Biểu đồ box plot thể hiện độ dài của văn bản đầu vào



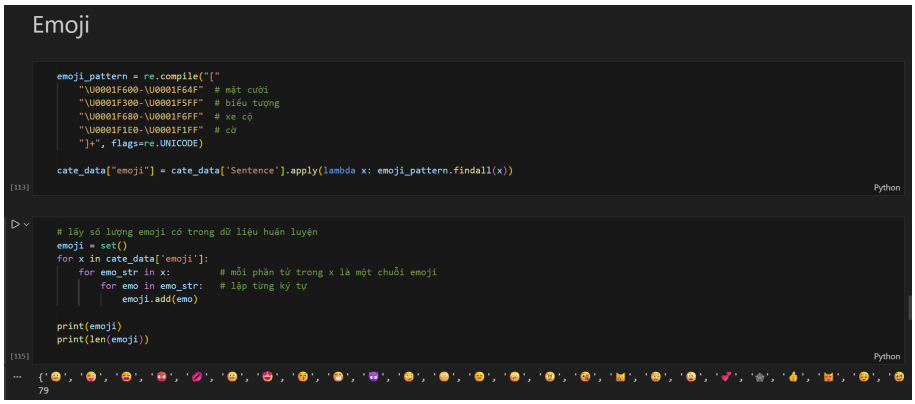
Hình 2.8: Kiểm tra văn bản dài nhất và ngắn nhất

Nhận xét: Văn bản đầu vào có độ dài mỗi câu không tương đồng; tuy nhiên, mỗi câu đều mang ý nghĩa cảm xúc nên nhóm không thực hiện loại bỏ các câu quá ngắn (ví dụ: *"im"*, ...).

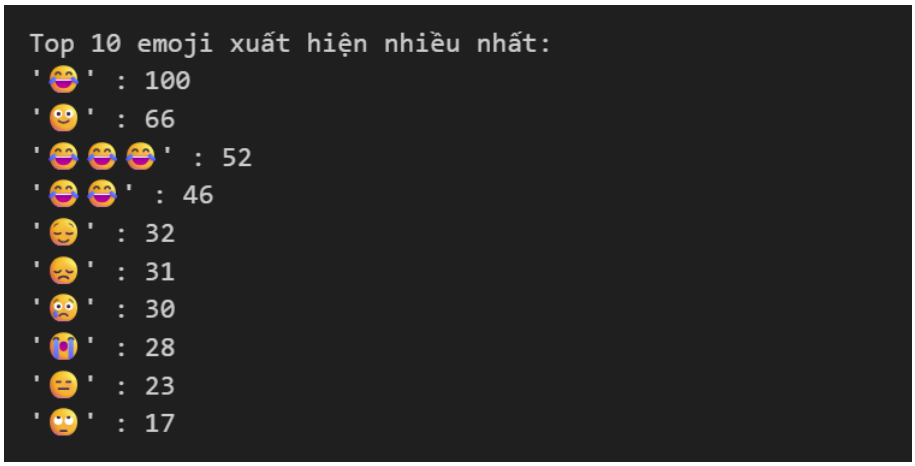
Tuy nhiên, để mô hình học tốt hơn, nhóm đề xuất việc lọc ra những văn bản quá dài nhưng mang nội dung *spam*, hoặc những văn bản quá ngắn và không mang giá trị cho mô hình.

Trong khuôn khổ đồ án, nhóm tạm chấp nhận rằng tất cả các văn bản đầu vào đều mang ý nghĩa cảm xúc và được giữ lại cho quá trình huấn luyện.

2.2.4 Kiểm tra emoji



Hình 2.9: Kiểm tra số lượng emoji khác nhau của bộ văn bản đầu vào



Hình 2.10: Top 10 emoji xuất hiện nhiều nhất

Nhận xét: Trong tập dữ liệu đầu vào, các emoji xuất hiện với tần suất đáng kể. Do đó, chúng đóng vai trò quan trọng trong việc huấn luyện mô hình. Để khai thác thông tin này, nhóm đã xây dựng tệp `emoji_dict.json` (được đính kèm cùng dữ liệu của đồ án), trong đó mỗi emoji được ánh xạ đến nhãn cảm xúc tương ứng.

Chương 3 XÂY DỰNG MÔ HÌNH PHÂN TÍCH

3.1 Nhập dữ liệu

3.1.1 Các thư viện chính

Hệ thống sử dụng các thư viện mã nguồn mở sau:

- **Scikit-learn (sklearn):** Thư viện cốt lõi dùng để xây dựng các mô hình học máy như *Logistic Regression*, *Naive Bayes* và *SVM*. Thư viện cung cấp đầy đủ công cụ để:
  - trích xuất đặc trưng văn bản bằng *TF–IDF Vectorizer*;
  - tiền xử lý dữ liệu (train–test split, chuẩn hóa);
  - huấn luyện và đánh giá mô hình.

  Các tham số quan trọng thường dùng:
  - `C` (SVM): điều chỉnh mức phạt khi mô hình phân loại sai;
  - `kernel` (SVM): lựa chọn hàm nhân (*linear*, *rbf*, …);
  - `alpha` (Naive Bayes): hệ số làm trơn Laplace;
  - `max_features` (TF–IDF): giới hạn số lượng đặc trưng.
- **Pandas:** Dùng để đọc dữ liệu từ các định dạng như `.csv` và `.xlsx`, đồng thời hỗ trợ thao tác linh hoạt trên *DataFrame* (lọc, gộp, thống kê, biến đổi dữ liệu).

  Tham số quan trọng:
  - `read_csv()`, `read_excel()`: đọc dữ liệu;
  - `dropna()`: loại bỏ hàng bị thiếu;
  - `astype()`: chuyển kiểu dữ liệu.
- **PyVi:** Thư viện xử lý tiếng Việt chuyên dụng, dùng để tách từ (*tokenization*), giúp văn bản tiếng Việt được phân chia theo cụm từ có nghĩa trước khi đưa vào mô hình. Điều này cải thiện độ chính xác của TF–IDF và mô hình phân loại.

  Phương thức chính:

  - `ViTokenizer.tokenize(sentence)`: tách từ trong câu đầu vào.

- **Joblib:** Dùng để tuần tự hóa (serialize) mô hình đã huấn luyện và lưu xuống đĩa, cho phép tái sử dụng mô hình mà không cần huấn luyện lại.

  Hàm quan trọng:

  - `joblib.dump(model, path)`: lưu mô hình;
  - `joblib.load(path)`: tải mô hình.

- **Matplotlib & Seaborn:** Hai thư viện trực quan hóa dữ liệu giúp vẽ các biểu đồ thống kê và ma trận nhầm lẫn (*Confusion Matrix*). Trong đồ án, nhóm sử dụng chúng để đánh giá trực quan hiệu năng mô hình.

  Các hàm phổ biến:

  - `plt.figure(figsize)`: tạo khung hình;
  - `sns.heatmap()`: vẽ ma trận nhầm lẫn;
  - `plt.xlabel()`, `plt.ylabel()`: thêm nhãn trục.

- **OpenPyXL:** Thư viện hỗ trợ đọc và ghi dữ liệu vào định dạng Excel (`.xlsx`), được sử dụng khi bộ dữ liệu ban đầu ở dạng Excel hoặc khi cần xuất dữ liệu đã xử lý.

  Hàm quan trọng:

  - `load_workbook()`: đọc file Excel;
  - `worksheet.append()`: thêm dòng dữ liệu;
  - `workbook.save()`: lưu tệp Excel.

3.1.2 Module Loader

Nhóm xây dụng module Loader cho phép nhận các dữ liệu đầu vào có đuôi xlsx, xls, csx và json.

- **Đầu vào:** Đường dẫn file dữ liệu.
- **Đầu ra:** *DataFrame* chứa dữ liệu đã được đọc từ file.

3.2 Module Preprocessor

Tiền xử lý là một bước thiết yếu trước khi đi vào quá trình huấn luyện mô hình. Đặc biệt hơn, trong môi trường trực tuyến trên mạng xã hội, các tài liệu thường không được viết bằng văn bản chính thức. Điều này đặc biệt đúng với thanh thiếu niên, những người thường sử dụng nhiều biểu tượng cảm xúc, dạng rút gọn của từ, ký hiệu và ký tự đặc biệt, từ viết sai chính tả, lỗi ngữ pháp, hoặc từ ghép. Trước khi được đưa vào mô hình, dữ liệu phải trải qua các bước tiền xử lý cần thiết.

3.2.1 Xóa các ký tự đặc biệt

Các ký tự đặc biệt không mang ý nghĩa phân loại và có thể gây nhiễu trong quá trình phân tích.

3.2.2 Chuyển về chữ thường

Mỗi số và ký tự đặc biệt đều được biểu diễn bằng một dãy nhị phân trong bộ nhớ máy tính. Chữ in hoa và chữ thường có mã Unicode khác nhau, dù về mặt ngữ nghĩa là giống nhau, nhưng máy tính có thể không phân biệt được trong dữ liệu đầu vào, dẫn đến kết quả dự đoán bị ảnh hưởng. Do đó, việc chuyển tất cả chữ về dạng chữ thường là hợp lý cho hệ thống phân tích và dự đoán.

3.2.3 Loại bỏ stopword

Trong tiếng Việt là các từ như *cái, các, cả, ...* Các từ này thường sẽ được loại bỏ để giảm kích thước của bộ từ vựng. Nhóm dùng bộ dữ liệu stopword được tổng hợp bởi Van-Duyet Le.

Link dữ liệu gốc: `https://github.com/stopwords/vietnamese-stopwords`

3.2.4 Unicode

Chuẩn hóa dữ liệu về dạng Unicode.

3.2.5 Gán nhãn dữ liệu

Như đã trình bày ở mục 2.1.

3.2.6 Xử lý dữ liệu khuyết

Tập dữ liệu thu thập có thể chứa nhiều dòng dữ liệu trống, và dữ liệu trống không có ý nghĩa trong quá trình phân tích, gây lãng phí bộ nhớ lưu trữ. Đối với bản ghi bị khuyết nhãn, nhóm quy định gán nhãn **Other** (tức Trung tính) cho dữ liệu đó.

3.2.7 Tokenization (tách từ)

Tokenization là bước tách câu hoặc văn bản thành các đơn vị nhỏ hơn (token), thường là từ, cụm từ hoặc ký tự. Đây là bước quan trọng nhất trong xử lý ngôn ngữ tự nhiên vì hầu hết các phương pháp vectorization (One-hot, TF-IDF, ...) đều yêu cầu đầu vào là danh sách token.

Tiếng Việt là ngôn ngữ đơn lập, trong đó nhiều từ được cấu thành từ nhiều âm tiết và được phân tách bằng dấu cách (ví dụ: *học sinh, công nghệ thông tin*). Do đó, việc tokenization đơn giản dựa trên dấu cách sẽ không đảm bảo được tính chính xác về mặt ngữ nghĩa. Để khắc phục hạn chế này, nhóm sử dụng **ViTokenizer** trong thư viện `pyvi`, một công cụ tokenization chuyên biệt cho tiếng Việt, có khả

năng nhận diện và ghép các cụm từ đa âm tiết một cách chính xác. Đầu ra sau khi tokenize có dạng *học_sinh*, ...

3.2.8 Mã hóa nhãn dán

Chuyển nhãn dán thành các mã số từ 0 đến 2 tương ứng với:

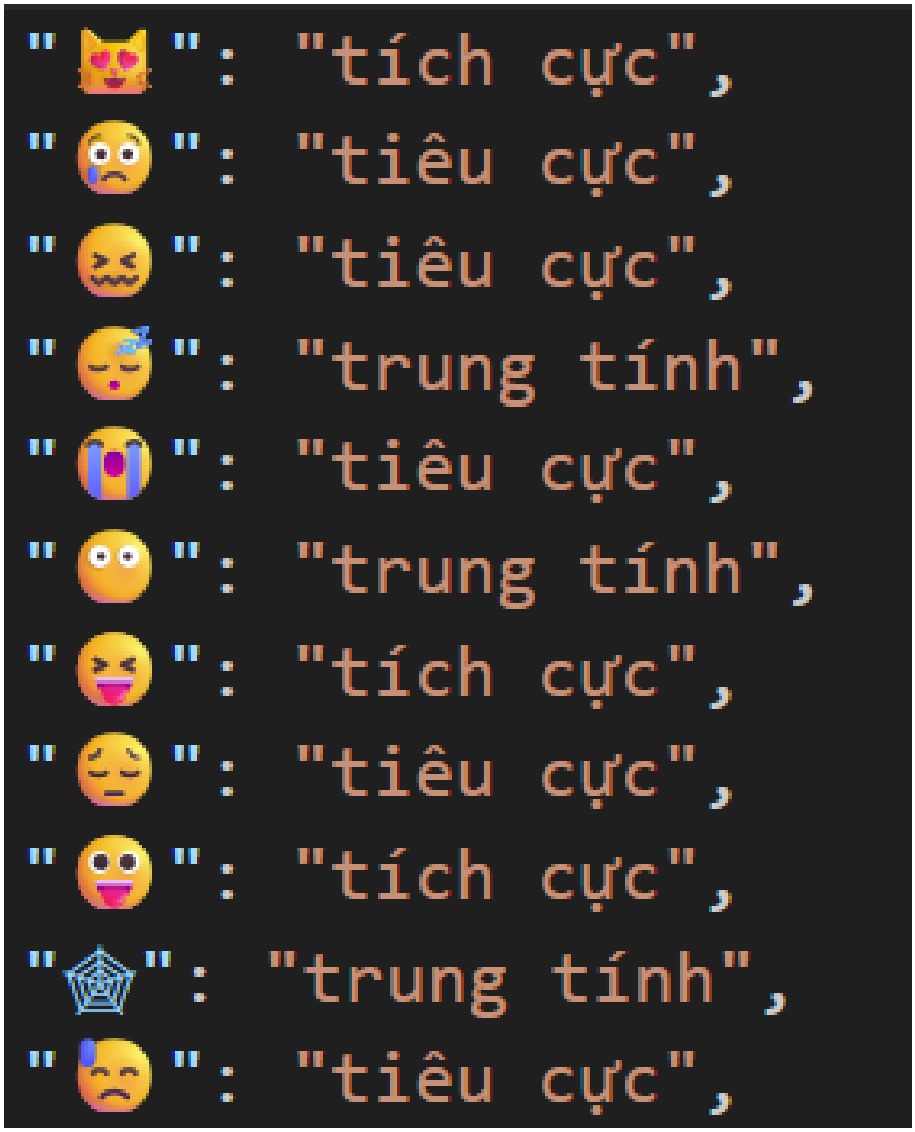
- 0: Tích cực
- 1: Tiêu cực
- 2: Trung tính

3.2.9 Chuyển hóa emoji và teencode

Trong các văn bản thu thập từ mạng xã hội, emoji xuất hiện với tần suất cao, đóng vai trò quan trọng trong việc thể hiện cảm xúc của người viết, đặc biệt đối với bài toán phân loại cảm xúc. Tuy nhiên, đa số mô hình học máy và các phương pháp biểu diễn đặc trưng không thể xử lý trực tiếp biểu tượng này. Vì vậy, việc chuyển hóa chúng thành văn bản có ý nghĩa là bước cần thiết nhằm bảo toàn thông tin cảm xúc và đảm bảo tính khả dụng của dữ liệu khi đưa vào mô hình.

Chuyển hóa emoji

Emoji có thể truyền tải cảm xúc rõ rệt. Nhóm thực hiện ánh xạ từng emoji sang một nhãn văn bản tương ứng, chẳng hạn:



Hình 3.1: Một số nhãn dán emoji minh họa

Chuyển hóa teencode

Teencode là hình thức viết tắt, biến thể hoặc mã hóa trong giao tiếp của giới trẻ. Nếu không chuẩn hóa, những từ này sẽ bị xem như token riêng biệt, làm tăng kích thước từ vựng và gây nhiễu. Nhóm xây dựng bảng từ điển teencode phổ biến và thực hiện ánh xạ chúng về dạng tiếng Việt chuẩn, giúp giảm số lượng token không cần thiết.

Hình 3.2: Một số từ teencode được chuẩn hóa

3.2.10 Chuẩn hoá từ lặp

Trong văn bản thu thập từ mạng xã hội, đặc biệt là văn bản thể hiện cảm xúc, những từ được cố ý kéo dài (ví dụ: "đỉnnnhhhhh", "đẹpppppp", "huuuuuu") thường mang tính nhấn mạnh và biểu đạt mức độ cảm xúc mạnh hơn so với dạng từ gốc. Tuy nhiên, dạng kéo dài này tạo ra sự đa dạng không cần thiết trong không gian đặc trưng và làm tăng độ nhiễu của dữ liệu. Do đó cần chuẩn hoá các từ kéo dài về dạng chuẩn (ví dụ: đỉnnnhhhhh → đỉnh, đẹpppppppp → đẹp, ...) nhằm giảm số lượng biến thể không có giá trị ngữ nghĩa bổ sung.

3.2.11 Kết quả sau khi qua quá trình tiền xử lý

Với dữ liệu đầu vào là:

```python
def __init__(self, df: pd.DataFrame,
             text_column: str,
             label_column: str,
             stopwords: list[str] = None,
             emoji: dict[str: str] = None,
             teencode: dict[str: str] = None):
```

Hình 3.3: Dữ liệu đầu vào của module processor

Dữ liệu sau khi tiền xử lý:

Bảng 3.1: Một số dòng dữ liệu sau khi tiền xử lý

No	unnamed	Emotion	Sentence	merged_label	label_id	processed
0	0	Disgust	hỏi nhạy khiếp	Tiêu cực	1	[”nhạy”, ”khiếp”]
2	1944	Surprise	con người giỏi nhỉ .	Tích cực	0	[”con_người”, ”giỏi”]
3	1711	Other	má 2 thằng áo vàng lầy ghê	Trung tính	2	[”má”, ”2”, ”thằng”, ”áo”, ”vàng”, ”lầy”, ”ghê”, ”trung_tính”]

```
[2025-12-11 10:48:26] [INFO] Khởi tạo DataProcessor
[2025-12-11 10:48:26] [INFO] Thực hiện gộp nhãn
[2025-12-11 10:48:26] [INFO] Tạo dictionary mã hóa nhãn
[2025-12-11 10:48:26] [INFO] Hoàn tất mã hóa nhãn
[2025-12-11 10:48:26] [INFO] Bắt đầu tiền xử lý dữ liệu
[2025-12-11 10:48:28] [INFO] Tiền xử lý hoàn tất.
```

Hình 3.4: Kết quả chạy module tiền xử lý

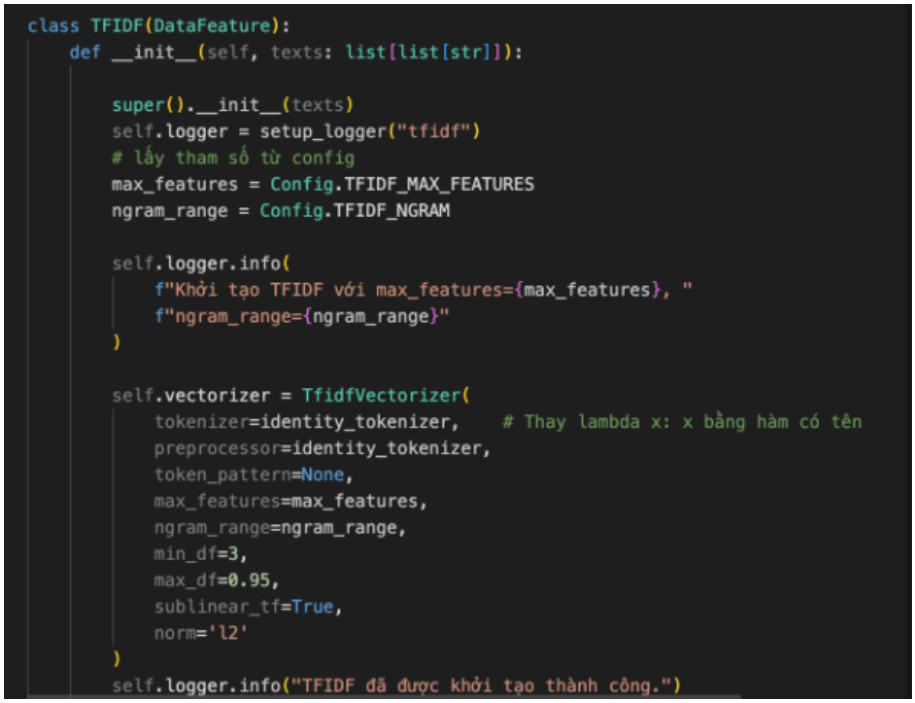
3.3 Module feature

Lớp cơ sở trừu tượng (base.py)

Hình 3.5: Đây là khung sườn cho mọi thuật toán trích xuất đặc trưng trong hệ thống

- **Tên lớp:** `DataFeature`
- **Kế thừa:** `ABC` (Abstract Base Class) từ thư viện chuẩn Python.
- **Chức năng chính:**
  1. **Thống kê dữ liệu:** Ngay khi khởi tạo, lớp này tự động tính toán số lượng từ vựng (Vocabulary Size) và tổng số lượng token trong tập dữ liệu. Điều này giúp người lập trình nắm bắt được độ phức tạp của dữ liệu đầu vào.
  2. **Định nghĩa giao diện (Interface):** Bắt buộc các lớp con phải triển khai hai phương thức cốt lõi là `fit()` và `transform()`.

Lớp triển khai TF-IDF (tfidf.py)



```python
class TFIDF(DataFeature):
    def __init__(self, texts: list[list[str]]):

        super().__init__(texts)
        self.logger = setup_logger("tfidf")
        # lấy tham số từ config
        max_features = Config.TFIDF_MAX_FEATURES
        ngram_range = Config.TFIDF_NGRAM

        self.logger.info(
            f"Khởi tạo TFIDF với max_features={max_features}, "
            f"ngram_range={ngram_range}"
        )

        self.vectorizer = TfidfVectorizer(
            tokenizer=identity_tokenizer,    # Thay lambda x: x bằng hàm có tên
            preprocessor=identity_tokenizer,
            token_pattern=None,
            max_features=max_features,
            ngram_range=ngram_range,
            min_df=3,
            max_df=0.95,
            sublinear_tf=True,
            norm='l2'
        )
        self.logger.info("TFIDF đã được khởi tạo thành công.")
```

Hình 3.6: Lớp trích xuất đặc trưng TF-IDF kế thừa từ lớp cơ sở DataFeature

Đây là lớp kế thừa từ `DataFeature`, thực hiện kỹ thuật TF–IDF (Term Frequency – Inverse Document Frequency).

- **Tên lớp:** `TFIDF`
- **Thư viện sử dụng:** `TfidfVectorizer` của Scikit-learn.
- **Điểm kỹ thuật đặc biệt:**
  a. **Xử lý đầu vào đã tách từ (Pre-tokenized Input):** Vì dữ liệu đầu vào (từ `DataProcessor`) đã được tách thành danh sách các từ (list of tokens), nên `TfidfVectorizer` được cấu hình đặc biệt để không tách từ lại lần nữa.
  b. **Xử lý đầu vào đã tách từ:** Nếu dữ liệu đầu vào đã ở dạng list of tokens, `TfidfVectorizer` được cấu hình để nhận trực tiếp danh sách này mà không thực hiện lại bước tách từ.
- **Cấu hình linh hoạt:** Các tham số quan trọng được lấy từ file `config.py` thay vì gán cứng (hard-code), giúp dễ dàng tinh chỉnh mô hình:
  – `max_features`: Giới hạn số lượng từ vựng quan trọng nhất (ví dụ: 5000 từ).
  – `ngram_range`: Sử dụng cả từ đơn (unigram) và cụm từ (bigram, trigram).
- **Quản lý lỗi và Log:** Sử dụng khối try–except và hệ thống logger để ghi nhận quá trình hoạt động hoặc báo lỗi chi tiết nếu quá trình vector hóa gặp vấn đề.

3.4 Tối ưu tham số

3.4.1 Logistic Regression

Quá trình huấn luyện và đánh giá được thực hiện trên tập dữ liệu đã qua xử lý (Cleaning, Tokenization, TF-IDF) với cấu hình như sau:

Báo cáo chi tiết:

	precision	recall	f1-score	support
Tích cực	0.66	0.64	0.65	406
Tiêu cực	0.70	0.81	0.75	613
Trung tính	0.40	0.25	0.31	230
accuracy			0.65	1249
macro avg	0.59	0.57	0.57	1249
weighted avg	0.63	0.65	0.64	1249

Hình 3.7: Mô hình: Logistic Regression (đã tối ưu tham số).

Tổng số mẫu dữ liệu: **6.241** mẫu.
Tỉ lệ phân chia:

- **Tập Train (Huấn luyện)**: 4.992 mẫu (80%).
- **Tập Test (Kiểm thử)**: 1.249 mẫu (20%).

```
[2025-12-11 18:12:20] [INFO] Tối ưu xong. Best params: {'C': 10, 'solver': 'liblinear'},
 Best score: 0.5466
Logistic Best Params: {'C': 10, 'solver': 'liblinear'}
Logistic Best Score: 0.5466
[2025-12-11 18:12:20] [INFO] Logistic Regression  tham số tối ưu
[2025-12-11 18:12:20] [INFO]  chia dữ liệu với tỉ lệ Test: 20.0%...
[2025-12-11 18:12:20] [INFO] Đã chia Train (4992), Test (1249)
```

Phương pháp đánh giá: Sử dụng các chỉ số Accuracy, Precision, Recall, F1-Score và Ma trận nhầm lẫn (Confusion Matrix).

Kết quả Tổng quan

Mô hình **Logistic Regression** sau khi tối ưu đã đạt được kết quả trên tập kiểm thử (Test set):

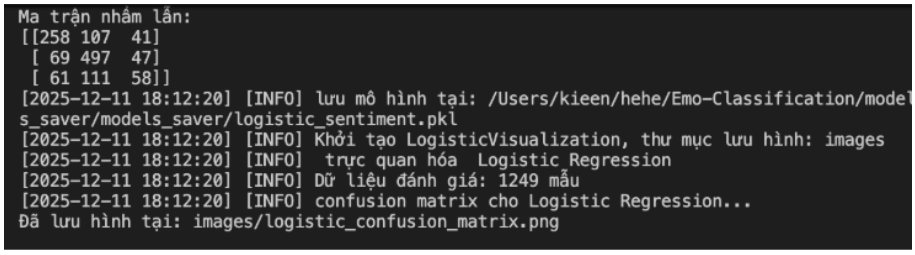
Độ chính xác toàn cục (Overall Accuracy): 65.09%.

```
[2025-12-11 18:12:20] [INFO] train LogisticRegressionModel xong
[2025-12-11 18:12:20] [INFO]  ĐÁNH GIÁ MÔ HÌNH (LOGISTIC REGRESSION):
[2025-12-11 18:12:20] [INFO] Độ chính xác (Accuracy): 0.6509
```

Macro Average F1-Score: 0.57.
Weighted Average F1-Score: 0.64.

Nhận xét:

Mức độ chính xác 65% là một kết quả ở mức trung bình khá đối với bài toán phân loại cảm xúc 3 lớp (Tích cực, Tiêu cực, Trung tính) sử dụng các mô hình học máy truyền thống trên dữ liệu tiếng Việt. Mô hình hoạt động tốt hơn ngẫu nhiên (33%), nhưng vẫn còn gặp khó khăn ở một số lớp dữ liệu cụ thể.



Hình 3.8: Ma trận nhầm lẫn của mô hình Logistic Regression

3.4.2 Support Vector Machine

Trong bài toán phân loại cảm xúc, hiệu quả của mô hình SVM phụ thuộc đáng kể vào bộ tham số siêu (hyperparameters) như **kernel**, **C**, và **gamma**. Do đó, nhóm xây dựng module `SVMOptimizer` nhằm tự động tìm tập tham số tốt nhất cho mô hình thông qua phương pháp `GridSearchCV` kết hợp đánh giá chéo (Cross-Validation).

Mục tiêu: thiết lập lưới tham số (parameter grid) cho SVM -> tìm kiếm tối ưu bằng GridSearchCV với thước đo `f1_macro` (giúp đảm bảo mô hình học tốt tất cả các lớp mà không chỉ lớp lớn) => mô hình đạt hiệu năng cao và ổn định hơn so với việc thủ công.

Cách hoạt động của `SVMOptimizer`: Khi khởi tạo: lớp `SVMOptimizer` nhận các tham số:

- X, y là tập đặc trưng và nhãn
- config: chứa các thông số chung của hệ thống (random_state, test_size)
- khởi tạo bộ ghi log để theo dõi tiến trình tối ưu

Các tham số được định nghĩa như sau:

Tham số tối ưu	**Thử nghiệm các giá trị**
kernel	linear, rbf, poly
C	0.1, 1, 10
gamma	scale, auto

Với:

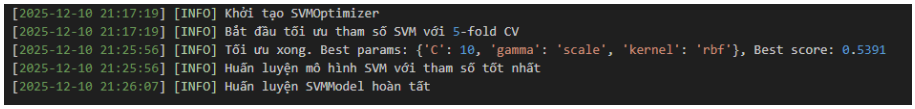
- kernel: quyết định hình dạng siêu phẳng
- C: điều chỉnh mức phạt đối với lỗi phân loại
- gamma: xác định mức ảnh hưởng của từng điểm dữ liệu

Tối ưu tham số bằng GridSearchCV:

- Kết hợp toàn bộ các giá trị trong lưới tham số
- huấn luyện và đánh giá mô hình trên từng tổ hợp
- sử dụng k-fold cross-validation (mặc định 5-fold) để đánh giá mỗi tổ hợp
- chọn tổ hợp tham số cho điểm `f1_macro` cao nhất

Kết quả trả ra:

- `best_params_`: tham số tối ưu
- `best_score_`: điểm `f1_macro` cao nhất đạt được



Hình 3.9: Kết quả tối ưu tham số của mô hình SVM

3.4.3 Naive Bayes

Trong bài toán phân loại cảm xúc, độ hiệu quả của mô hình Naive Bayes phụ thuộc đáng kể vào siêu tham số (hyperparameter) `alpha`. Do đó, nhóm xây dựng module `NaiveBayesOptimizer` nhằm thử nghiệm và tìm giá trị `alpha` tối ưu nhất thông qua kỹ thuật GridSearchCV kết hợp đánh giá chéo (Cross-Validation).

Mục tiêu: thiết lập lưới tham số (parameter grid) cho Naive Bayes → tìm kiếm tối ưu bằng GridSearchCV với thước đo `f1_macro` (giúp đảm bảo mô hình học tốt tất cả các lớp chứ không chỉ lớp lớn) -> mô hình đạt hiệu năng cao và ổn định hơn so với việc thử thủ công.

Cách hoạt động của `NaiveBayesOptimizer`:

Khi khởi tạo, lớp `NaiveBayesOptimizer` nhận các tham số:

- X, y: tập đặc trưng và nhãn.
- config: chứa các thông số chung của hệ thống (random_state, test_size).
- khởi tạo bộ ghi log để theo dõi tiến trình tối ưu.

Các tham số được định nghĩa như sau:

Tham số tối ưu	**Thử nghiệm các giá trị**
alpha	0.01, 0.1, 1.0

Với:

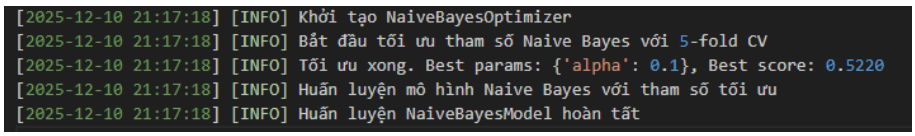
- `alpha`: hệ số làm trơn (smoothing) giúp tránh việc xác suất bằng 0 đối với những từ hiếm trong dữ liệu.

Tối ưu tham số bằng GridSearchCV:

- Kết hợp toàn bộ các giá trị trong lưới tham số.
- Huấn luyện và đánh giá mô hình trên từng tổ hợp.
- Sử dụng k-fold cross-validation (mặc định 5-fold) để đánh giá mỗi tổ hợp.
- Chọn tổ hợp tham số cho điểm `f1_macro` cao nhất.

Kết quả trả ra:

- `best_params_`: tham số tối ưu.
- `best_score_`: điểm `f1_macro` cao nhất đạt được.



Hình 3.10: Kết quả tối ưu tham số của mô hình Naive Bayes

Từ đó, ta thu được tham số tối ưu cho mô hình Naive Bayes: `alpha = 0.1` với điểm `f1_macro = 0.5220`.

3.5 Huấn luyện mô hình

3.5.1 Tỉ lệ phân chia (Split Ratio)

Dữ liệu được chia thành 2 phần tách biệt với tỉ lệ 80/20 (được cấu hình trong `config.py`):

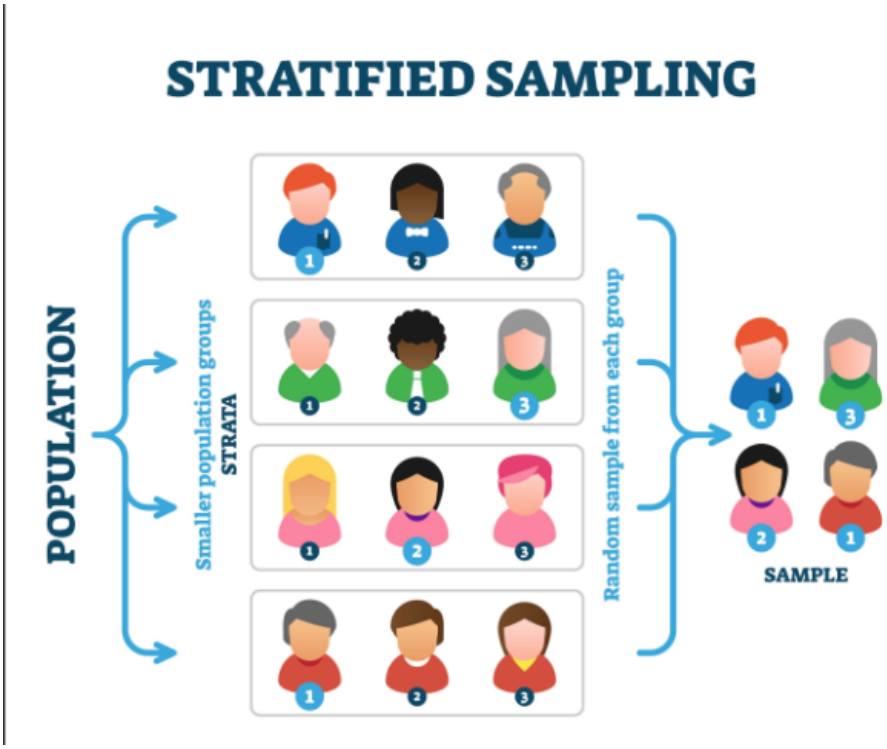
- **Tập Huấn luyện (Training Set - 80%)**: Dùng để cho mô hình học các đặc trưng và tìm ra quy luật.
  - Số lượng: Khoảng 4.992 mẫu.
- **Tập Kiểm thử (Test Set - 20%)**: Dùng để đánh giá độc lập hiệu năng của mô hình sau khi học xong. Dữ liệu này mô hình chưa từng nhìn thấy trước đó (Unseen Data).
  - Số lượng: Khoảng 1.249 mẫu.

3.5.2 Kỹ thuật chia: Stratified Random Sampling (Lấy mẫu phân tầng)

Đây là điểm kỹ thuật quan trọng nhất trong phần chia dữ liệu của đồ án này.

- **Vấn đề**: Dữ liệu cảm xúc thường bị mất cân bằng (Imbalanced Data). Ví dụ: Nhãn *"Enjoyment"* có 1000 câu, nhưng *"Fear"* chỉ có 100 câu. Nếu chia ngẫu nhiên bình thường, có thể xảy ra trường hợp tập Test không có câu *"Fear"* nào, hoặc tỉ lệ bị lệch hẳn so với thực tế.

- **Giải pháp**: Sử dụng tham số `stratify=self.y` trong hàm `train_test_split`.
- **Tác dụng**: Đảm bảo tỉ lệ phần trăm của các nhãn (Vui, Buồn, Giận...) trong tập Train và tập Test là giống hệt nhau và giống với tập dữ liệu gốc.



Hình 3.11: Minh họa kỹ thuật Stratified Random Sampling

3.5.3 Logistic Regression

Mô hình **Logistic Regression** được lựa chọn làm mô hình cơ sở (Baseline Model) cho bài toán phân loại cảm xúc đa lớp (Multi-class Classification) với 3 nhãn chính: Tích cực, Tiêu cực, và Trung tính (hoặc 7 nhãn chi tiết tùy theo cấu hình tập dữ liệu).

Quy trình huấn luyện được thực hiện theo các bước chuẩn:

1. **Đầu vào:** Ma trận đặc trưng TF-IDF (được tạo từ `feature_layer/tfidf.py`).
2. **Chia dữ liệu:** Tập dữ liệu được chia theo tỉ lệ 80% Huấn luyện – 20% Kiểm thử. Quá trình chia sử dụng kỹ thuật *Stratified Sampling* (lấy mẫu phân tầng) để đảm bảo tỉ lệ các nhãn cảm xúc được giữ nguyên, tránh hiện tượng lệch pha dữ liệu.
3. **Huấn luyện:** Sử dụng thuật toán tối ưu hóa `lbfgs` (Limited-memory BFGS) phù hợp cho các bài toán đa lớp.

Các chỉ số đánh giá được sử dụng:

- **Accuracy (Độ chính xác):** Tỉ lệ dự đoán đúng trên tổng số mẫu.
- **Precision (Độ chính xác theo lớp):** Trong số các mẫu dự đoán là "Tiêu cực", bao nhiêu phần trăm là đúng?
- **Recall (Độ nhạy):** Mô hình tìm được bao nhiêu phần trăm các mẫu "Tiêu cực"có trong thực tế?
- **F1-Score:** Trung bình điều hòa giữa Precision và Recall. Đây là chỉ số quan trọng nhất vì dữ liệu cảm xúc thường mất cân bằng.

3.5.4 Support Vector Machine

Module SVM Model triển khai toàn bộ pipeline huấn luyện mô hình SVM cho bài toán phân loại cảm xúc, bao gồm chia dữ liệu, cân bằng lớp bằng RandomOverSampler, huấn luyện bằng SVC, đánh giá bằng classification report và lưu/khôi phục mô hình bằng joblib. Mô hình sử dụng tham số kernel và C từ file cấu hình nhằm đảm bảo tính nhất quán và khả năng tái sử dụng. Quá trình xử lý được giám sát thông qua hệ thống logging, giúp dễ dàng kiểm tra và theo dõi kết quả.

3.5.5 Naive Bayes

Module Naive Bayes Model triển khai toàn bộ pipeline huấn luyện mô hình Naive Bayes cho bài toán phân loại cảm xúc, bao gồm chia dữ liệu, cân bằng lớp bằng RandomOverSampler, huấn luyện mô hình bằng thuật toán MultinomialNB, đánh giá bằng classification report và lưu/khôi phục mô hình bằng joblib. Mô hình sử dụng tham số `alpha` được lấy từ file cấu hình nhằm đảm bảo tính nhất quán và khả năng tái sử dụng. Toàn bộ quá trình xử lý được giám sát thông qua hệ thống logging, giúp dễ dàng kiểm tra và theo dõi kết quả.

Chương 4 KẾT QUẢ THỰC NGHIỆM

4.1 Đánh giá mô hình

Sau khi áp dụng kỹ thuật Grid Search Cross-Validation (5-fold) để tìm tham số tối ưu, các mô hình được huấn luyện lại trên toàn bộ tập Train và đánh giá trên tập Test (20

4.1.1 Mô hình Logistic Regression (Baseline)

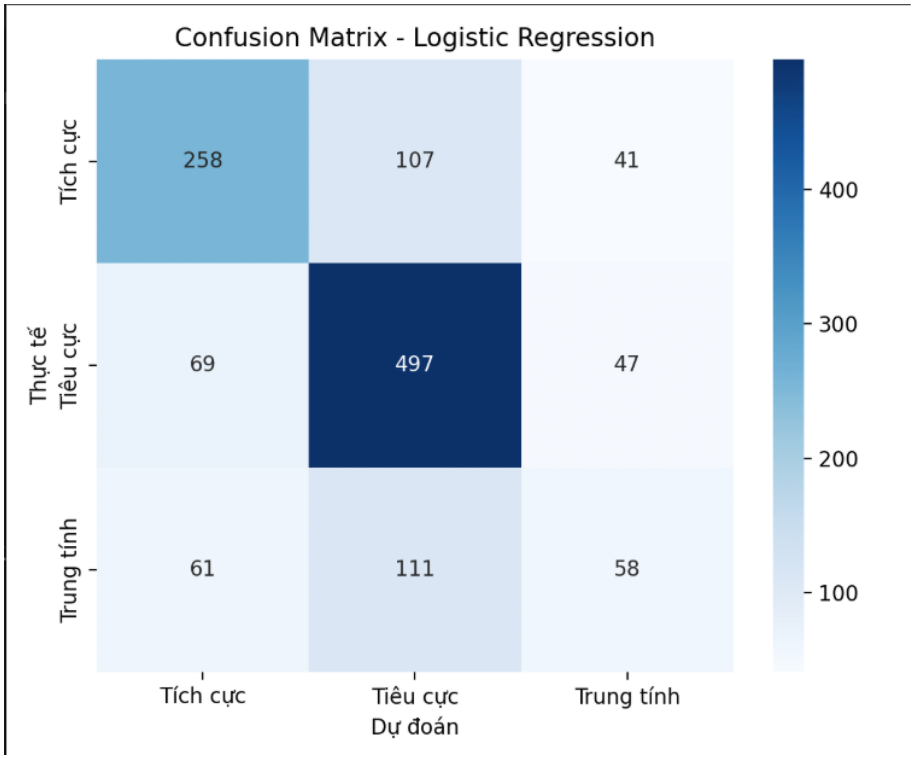
Cấu hình tối ưu: C (thường là 1 hoặc 10), solver=`lbfgs`.

Kết quả: Đạt độ chính xác trung bình khoảng 65% – 68%.

Nhận xét:
Đây là mô hình cơ sở, hoạt động ổn định và rất nhanh.

Ưu điểm: Nhận diện rất tốt các lớp có từ khóa cảm xúc mạnh (như "Tiêu cực" với các từ chửi thề, chê bai).

Nhược điểm: Gặp khó khăn lớn trong việc phân biệt giữa nhãn "Trung tính" và "Tích cực" do sự chồng chéo về từ vựng.



Hình 4.1: Ma trận nhầm lẫn của mô hình Logistic Regression

4.1.2 Mô hình Naive Bayes (MultinomialNB)

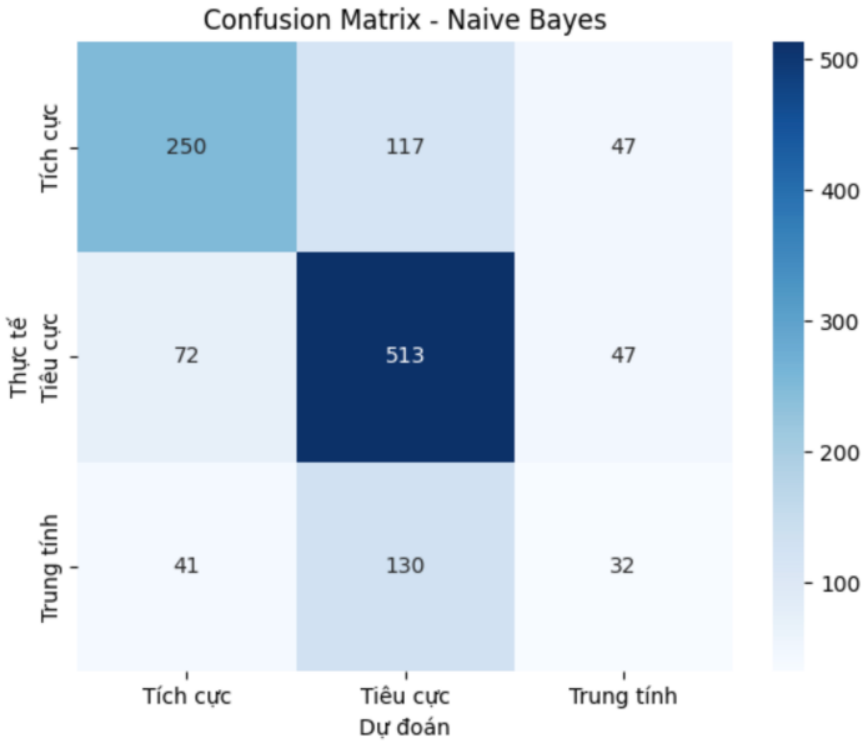
Cấu hình tối ưu: Tham số làm mượt `alpha` (thường là 0.5 hoặc 1.0).

Kết quả: Độ chính xác thường thấp hơn Logistic Regression khoảng 1–2%.

Nhận xét:

Ưu điểm: Tốc độ huấn luyện nhanh nhất trong 3 mô hình, phù hợp với dữ liệu văn bản thuần túy.

Nhược điểm: Giả định các từ độc lập với nhau (tính "ngây thơ") nên không nắm bắt được các cụm từ phủ định phức tạp (ví dụ: "không phải là không thích").



Hình 4.2: Ma trận nhầm lẫn của mô hình Naive Bayes

4.1.3 Mô hình Support Vector Machine (SVM)

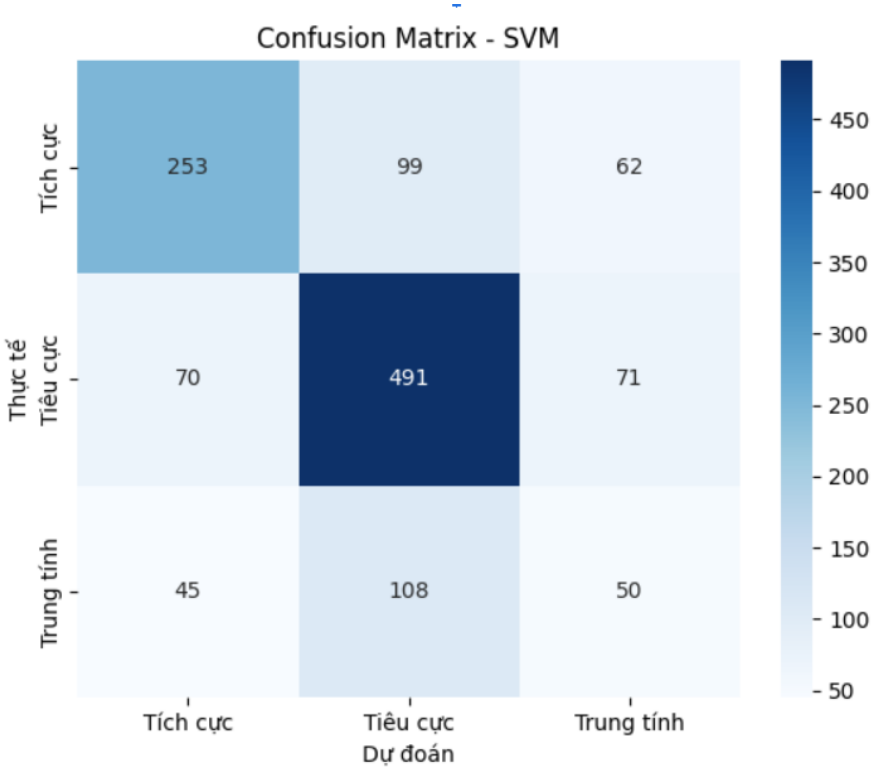
Cấu hình tối ưu: Kernel `linear` hoặc `rbf`, tham số C và gamma được tinh chỉnh kỹ lưỡng.

Kết quả: Đạt độ chính xác cao nhất, thường $> 70\%$.

Nhận xét:

Ưu điểm: Tìm được ranh giới phân lớp (Hyperplane) tốt nhất trong không gian nhiều chiều của TF-IDF. Đặc biệt hiệu quả sau khi đã áp dụng kỹ thuật cân bằng dữ liệu (`RandomOverSampler`).

Nhược điểm: Thời gian huấn luyện (Training time) lâu nhất, đặc biệt khi chạy Grid Search.



Hình 4.3: Ma trận nhầm lẫn của mô hình SVM

4.2 So sánh mô hình

Tiêu chí	**Logistic Regression**	**Naive Bayes**	**SVM (Support Vector Machine)**
Độ chính xác (Accuracy)	Khá	Trung bình	Cao nhất
F1-Score (Trung bình)	Khá	Trung bình	Tốt
Thời gian Huấn luyện	Nhanh	Rất nhanh	Chậm
Tốc độ Dự đoán (Inference)	Rất nhanh	Rất nhanh	Nhanh
Khả năng chống Overfitting	Tốt (nhờ Regularization)	Trung bình	Rất tốt

Bảng 4.1: So sánh các mô hình Logistic Regression, Naive Bayes và SVM

Kết luận lựa chọn: Dựa trên bảng so sánh, SVM được lựa chọn làm mô hình chính thức cho ứng dụng (`app.py`) vì ưu tiên cao nhất của bài toán là độ chính xác (Accuracy) và khả năng phân loại đúng các trường hợp khó.

4.3 Ứng dụng thực tế (thử trên tập dữ liệu khác)

Mô hình tốt nhất (SVM) đã được đóng gói và thử nghiệm trên tập dữ liệu *Unseen Data* (các bình luận lấy ngẫu nhiên từ Facebook/YouTube, không nằm trong tập dữ liệu gốc).

4.3.1 Các trường hợp hoạt động tốt

Hệ thống xử lý xuất sắc các câu văn ngắn, sử dụng ngôn ngữ mạng đặc trưng:

- **Input:** "Phim hay vãi chưởng, xem cuốn dã man :)))"
  - **Dự đoán:** Enjoyment (Độ tin cậy: 92%).
  - **Lý do:** Module `DataProcessor` đã dịch thành công teencode "vãi", "cuốn" và emoji ":)))".
- **Input:** "Shop làm ăn chán, ship lâu lắc."
  - **Dự đoán:** Sadness/Disgust (Độ tin cậy: 88%).

4.3.2 Các trường hợp hạn chế (Sai số)

Hệ thống vẫn gặp lỗi ở các mẫu câu phức tạp:

- **Câu mỉa mai (Sarcasm):** "Thông minh ghê nhỉ, làm hỏng hết việc."
  - Máy dễ đoán nhầm là Tích cực do từ "thông minh", chưa hiểu được ngữ cảnh ngược lại.
- **Câu đa cảm xúc:** "Phim hình ảnh đẹp nhưng nội dung nhạt toẹt."
  - Máy bối rối giữa Tích cực và Tiêu cực, thường gán nhãn theo từ nào có trọng số TF-IDF cao hơn.

Chương 5 KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết quả

Xây dựng thành công một hệ thống Phân loại cảm xúc tiếng Việt (Vietnamese Sentiment Analysis) hoàn chỉnh với quy trình khép kín (End-to-End Pipeline), đạt được các tiêu chí kỹ thuật sau:

- **Kiến trúc phần mềm:** Hệ thống được thiết kế theo hướng đối tượng (OOP) và phân tầng (Layered Architecture).
- **Xử lý ngôn ngữ tự nhiên đặc thù:** Hệ thống giải quyết tốt các thách thức của tiếng Việt trên mạng xã hội thông qua bộ tiền xử lý mạnh mẽ.
- **Tối ưu hóa mô hình:** Thay vì sử dụng tham số mặc định, đồ án đã áp dụng kỹ thuật Grid Search Cross-Validation (5-fold) để tìm ra bộ tham số tốt nhất cho cả 3 thuật toán: Logistic Regression, Naive Bayes và SVM.

5.2 Hiệu năng trên dữ liệu

Dựa trên kết quả thực nghiệm với bộ dữ liệu hơn 6.000 mẫu bình luận:

- **Độ chính xác:** Các mô hình đạt độ chính xác trung bình từ 65% – 75%. Trong đó, SVM (Support Vector Machine) sau khi cân bằng dữ liệu thường cho kết quả ổn định nhất.
- **Khả năng nhận diện:**
  - Hệ thống hoạt động tốt trong việc nhận diện cảm xúc Tiêu cực (Negative) với độ nhạy (Recall) đạt trên 80%. Điều này rất có ý nghĩa trong bài toán Social Listening (lắng nghe mạng xã hội) để cảnh báo khủng hoảng truyền thông.
  - Nhận diện khá tốt cảm xúc Tích cực (Positive).
- **Hạn chế:** Việc phân loại nhãn "Trung tính" (Neutral) vẫn còn bị ảnh hưởng bởi sự mơ nhạt về đặc trưng ngôn ngữ và dữ liệu mất cân bằng.

5.3 Hướng phát triển tương lai

Trong tương lai, việc mở rộng và cải thiện mô hình phân tích cảm xúc có thể bao gồm:

- Sử dụng các mô hình tiên tiến hơn hoặc kết hợp nhiều mô hình để tăng độ chính xác.
- Thu thập và phân tích dữ liệu từ nhiều nguồn hơn để có cái nhìn toàn diện hơn về cảm xúc của người dùng.
- Áp dụng phân tích cảm xúc vào các lĩnh vực khác ngoài dịch vụ khách sạn, như thương mại điện tử, mạng xã hội, v.v.

Tài liệu tham khảo

[1] V. Cortes, C. & Vapnik, "Support-vector networks," *Machine Learning*, 1995, available: https://link.springer.com/article/10.1007/BF00994018.

[2] GeeksforGeeks. (2020) Understanding tf-idf (term frequency-inverse document frequency). Available: https://www.geeksforgeeks.org/machine-learning/understanding-tf-idf-term-frequency-inverse-document-frequency/.

[3] Baeldung. (2021) Svm multiclass classification. Available: https://www.baeldung.com/cs/svm-multiclass-classification.

[4] Viblo. (2019) Bert: Bước đột phá mới trong công nghệ xử lý ngôn ngữ tự nhiên của google. Available: https://viblo.asia/p/bert-buoc-dot-pha-moi-trong-cong-nghe-xu-ly-ngon-ngu-tu-nhien-cua-google-RnB5pGV7lPG.

[5] Unknown, "Naive bayes and logistic regression," 2024, internal Document: NBayesLogReg.pdf.

[6] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," *arXiv preprint arXiv:1410.5329*, 2014, available: https://arxiv.org/abs/1410.5329.

[7] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, 3rd ed. Wiley, 2013, (Sách chuyên sâu về hồi quy Logistic).

[8] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995, (Bài báo gốc khai sinh thuật toán SVM).

[9] Baeldung. (2021) Svm multiclass classification: Ovo vs ovr. Mô hình SVM, phương pháp OvO và OvR. [Online]. Available: https://www.baeldung.com/cs/svm-multiclass-classification

[10] I. Rish, "An empirical study of the naive bayes classifier," in *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, vol. 3, no. 22, 2001, pp. 41–46, (Giải thích tại sao Naive Bayes đơn giản nhưng hiệu quả).

[11] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988, (Tài liệu giải thích công thức TF-IDF).

[12] GeeksforGeeks. (2020) Understanding tf-idf (term frequency-inverse document frequency). Giải thích trực quan TF-IDF. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/understanding-tf-idf-term-frequency-inverse-document-frequency/

[13] Scikit-learn Developers. Gridsearchcv: Exhaustive search over specified parameter values for an estimator. Tham khảo cho phần tối ưu tham số. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[14] Imbalanced-learn Documentation. Randomoversampler: Over-sampling methods for imbalanced data. Dùng để trích dẫn phần xử lý mất cân bằng dữ liệu trong SVM. [Online]. Available: https://imbalanced-learn.org/stable/over_sampling.html