

PHÂN TÍCH CẢM XÚC CÁC ĐÁNH GIÁ THUỘC NHÀ SÁCH TIKI TRÊN SÀN THƯƠNG MẠI ĐIỆN TỬ TIKI

1. Mục tiêu của dự án

Tiki là một trong những trang thương mại điện tử nổi tiếng và uy tín tại Việt Nam. Trong đó mặt hàng mà được cho là thế mạnh và đặc trưng của Tiki chính là sách. Tiki cho phép người dùng tương tác với các sản phẩm qua các nhận xét, bình luận về sản phẩm, trong đó ý kiến khách hàng là những phản hồi của khách hàng bao gồm cả hai mặt tích cực và tiêu cực. Những bình luận tích cực, tiêu cực và những khía cạnh mà khách hàng đang quan tâm đều giúp doanh nghiệp biết được ưu và nhược điểm của sách cũng như dịch vụ, từ đó quảng bá, truyền thông hoặc quyết định tiếp tục tái bản hay phát hành sách đó hay không. Thị trường cạnh tranh ngày càng một tăng cao, để phục vụ khách hàng thì doanh nghiệp cần nắm bắt đúng nhu cầu của họ.

Mục tiêu của dự án là phân tích cảm xúc của các bình luận, đánh giá thuộc Nhà sách đến từ khách hàng trên sàn thương mại điện tử Tiki, từ đó cung cấp cho doanh nghiệp tài liệu để tham khảo cho các chiến lược tiếp thị tiếp theo thông qua việc điều chỉnh các sản phẩm, dịch vụ sao cho phù hợp với nhu cầu của khách hàng để cải thiện trải nghiệm của khách hàng.

2. Các nghiên cứu liên quan

- Năm 2022, Emil R. Kaburuan và các cộng sự đã phân tích và phân loại quan điểm của các đánh giá sản phẩm thuộc quần áo ở nhà phụ nữ trên sàn thương mại điện tử Shopee bằng dùng thuật toán phân loại Naive Bayes. Nghiên cứu này có độ chính xác là 90.03%. Nghiên cứu này cung cấp bức tranh khách quan về đánh giá của khách hàng, tích cực hay tiêu cực và từ nào xuất hiện nhiều trong mỗi quan điểm để cải thiện.

- Năm 2020, Hồ Trung Thành và cộng sự đã sử dụng các phương pháp học máy và kiểm định Bootstrap để phân tích ý kiến khách hàng trên sàn thương mại điện tử Tiki. Họ thực hiện phân cụm dữ liệu bình luận bằng hai thuật toán SOM và K-Means nhằm khám phá nội dung khách hàng thường trao đổi về sản phẩm và dịch vụ, qua đó đánh giá trải nghiệm khách hàng và xác định ưu, nhược điểm của các sản phẩm sách. Kết quả cho thấy sự khác biệt giữa hai thuật toán, nhưng kiểm định T với phương pháp Bootstrap chỉ ra rằng độ chênh lệch không lớn ($-0,760$), có ý nghĩa thống kê với p-value nhỏ hơn 0,05, và khoảng tin cậy 95% dao động từ $-0,795$ đến $-0,719$.

- Năm 2023, Manal Loukili và các cộng sự đã phân tích quan điểm các đánh giá trên sàn thương mại điện tử và dự đoán xem khách hàng có đề xuất sản phẩm dựa trên các đánh giá hay không bằng việc sử dụng 4 thuật toán là KNN, Random Forest, CatBoost Classifier và Logistic Regression. Kết quả cho thấy mô hình Logistic Regression vượt trội hơn tất cả các mô hình trước đó về các chỉ số hiệu suất (Độ chính xác = 90%, Độ nhạy = 78,6%, F1 = 81,4%).

- Năm 2020, Xiaoxin phân tích quan điểm về đánh giá của khách hàng thương mại điện tử dựa trên xử lý ngôn ngữ tự nhiên sử dụng các thuật toán Naive Bayes, KNN, Random Forest, LightGBM và Logistic Regression, so sánh một cách khách quan và đi đến kết luận rằng mô hình LightGBM cho ra kết quả cao nhất với độ chính xác là 98%, giá trị AUC là 96%.

- Năm 2022, Ketan Gupta và các cộng sự đã phân tích và phân loại các đánh giá từ 3 bộ data là Amazon, Yelp, và IMDB bằng sử dụng các thuật toán Naïve Bayes, Random Forest, K-Nearest Neighbor (KNN), và Support Vector Machine (SVM). Kết quả mô hình Random Forest đạt được kết quả tốt nhất với độ chính xác trên 78%.

- Năm 2022, Priyanshi Kathuria và các cộng sự đã thực hiện phân tích với nhiều thông số khác nhau và nhiều mô hình để tìm hiểu liệu khách hàng có thích 1 sản phẩm nào

đó hay không. Bên cạnh các mô hình học máy, nhóm tác giả còn sử dụng thêm 2 thư viện của python là Vader và Text Blob để phân tích cụ thể hơn về cảm xúc trong một lượt đánh giá của khách hàng. Dự đoán cho thấy với mô hình Logistic Regression cho kết quả có độ chính xác cao nhất ở cả hai nhóm phân tích là bình luận (88,18%) và đánh giá là (80,68%).

3. Mô tả dữ liệu

Bộ dữ liệu thu thập được bao gồm hơn 26000 dòng và bao gồm 7 cột dữ liệu. Dữ liệu được thu thập là dữ liệu thuộc Nhà sách Tiki trên sàn thương mại điện tử Tiki. Các dòng trong bộ dữ liệu này đại diện cho các đánh giá từ người mua đối với các quyển sách đang được bán trên Tiki. Mỗi dòng chứa thông tin về một đánh giá cụ thể, bao gồm:

	id	title	content	thank_count	customer_id	rating	product_id	
0	19906519	Cực kì hài lòng	Sau khi phản ánh việc tác giả đặt tên sách trù...	8	13450478	5	275025208	
1	19968834	Hài lòng	Tạm thời thấy ổn.Đọc xong sẽ review sau.hj.Yh...	0	19821878	4	275025208	
2	19943636	Rất không hài lòng	quyển sách với tựa đề làm hiểu lầm cái khách h...	0	14317039	1	275025208	
3	19919040	Cực kì hài lòng	Giao hàng nhanh, đóng gói cẩn thận	0	13713637	5	275025208	
4	19963835	Hài lòng	Tạm được, phù hợp những bạn chưa đọc nhiều sác...	0	7159313	4	275025208	
...	
26227	19939226	Cực kì hài lòng		NaN	0	13696065	5	73316970
26228	19931668	Cực kì hài lòng		NaN	0	29245947	5	73316970
26229	9198319	Cực kì hài lòng	Chất lượng sách tốt, giấy tốt, Tiki giao hàng ...	0	7562223	5	73316970	
26230	19943620	Cực kì hài lòng		NaN	0	12028298	5	274909433
26231	19900164	Cực kì hài lòng		NaN	0	135809	5	274909433

26232 rows × 7 columns

Hình 1: Dữ liệu được thu thập từ Nhà sách Tiki

Trong đó:

- **id:** Mã đánh giá là giá trị duy nhất được Tiki gán cho mỗi đánh giá. Mã này được dùng để tham chiếu đánh giá trong hệ thống của Shopee.
- **title:** Tiêu đề là mức độ hài lòng của khách hàng đối với sản phẩm bao gồm ‘Cực kì hài lòng’, ‘Hài lòng’, ‘Rất không hài lòng’, ‘Bình thường’, ‘Không hài lòng’.

- content: Nội dung đánh giá là văn bản được người mua viết để đánh giá sản phẩm. Nội dung đánh giá có thể bao gồm các thông tin như chất lượng giấy, chất lượng dịch thuật, màu mực, nội dung cuốn sách, chất lượng dịch vụ như giao hàng hay tư vấn, thời gian giao hàng, ...
- thank_count: Số lượng cảm ơn là số lượng người mua khác vào đọc đánh giá để đánh giá chất lượng sản phẩm và gửi cảm ơn người đã đánh giá trước đó.
- customer_id: Mã khách hàng là giá trị duy nhất mà Tiki gán cho mỗi khách hàng để có thể tham chiếu thông tin khách hàng trong hệ thống của Tiki.
- rating_star: Mức độ đánh giá chất lượng sản phẩm của người mua, được đánh giá từ 1 sao đến 5 sao.
- product_id: Mã sản phẩm là giá trị duy nhất mà Tiki gán cho mỗi sản phẩm được bán ra. Mã sản phẩm giúp tham chiếu thông tin sản phẩm trong hệ thống của Tiki.

Mỗi dòng dữ liệu là mỗi đánh giá của một sản phẩm cụ thể. Cột content là cột chứa nội dung đánh giá.

4. Phương pháp trong dự án

Để giải quyết vấn đề cho dự án “**Phân tích cảm xúc các đánh giá đến từ khách hàng thuộc các sản phẩm thuộc nhà sách Tiki trên sàn thương mại điện tử Tiki**”, mình tiến hành phân tích thị trường sách tại Việt Nam trên sàn thương mại điện tử uy tín về sách và đo lường cảm xúc của người mua thông qua các bình luận về sản phẩm. Các đánh giá sẽ thường phản ánh những vấn đề tốt và chưa tốt mà bên phía Tiki cung cấp. Từ các đánh giá sẽ phát hiện được các vấn đề về sản phẩm và dịch vụ mà được các khách hàng thảo luận nhiều và từ đó sẽ là tài liệu để các bên bộ phận sản phẩm và dịch vụ có thể cải thiện những điểm chưa tốt và phát huy những điểm tốt. Cụ thể giải pháp được thực hiện như sau:

- Thực hiện thu thập dữ liệu bình luận và đánh giá của các quyền sách trên sàn thương mại điện tử Tiki.
- Tập hợp dữ liệu và tiến hành tiền xử lý đối với những dữ liệu bình luận của cuốn sách.
- Tiến hành gán nhãn dữ liệu theo 2 nhãn (Tích cực và Tiêu cực).
- Tiến hành phân tích, thống kê và đưa ra cái nhìn tổng quát nhất đối với bộ dữ liệu.
- Thực hiện huấn luyện mô hình và chọn mô hình tốt nhất.
- Tìm siêu tham số của mô hình đã chọn để đạt kết quả tốt nhất.
- Tiếp tục đánh giá mô hình trên tập huấn luyện và tập kiểm tra để phát hiện ra mô hình có đang bị vấn đề quá khớp (overfitting) hay không.
- Phân tích những đánh giá bị dán sai nhãn ở tập huấn luyện và tiến hành dán lại nhãn cho đến khi đạt được độ chính xác mong muốn.
- Ứng dụng mô hình vào bộ dữ liệu để xác định nhãn của các đánh giá.
- Tiến hành phân tích những vấn đề thường được khách hàng nhắc tới theo từng nhãn và phát hiện ra các điểm mạnh và điểm yếu trong vấn đề sản phẩm và dịch vụ theo từng nhãn đã được xác định bằng mô hình.

5. Mô hình phương pháp

5.1. API

a. Khái niệm về API

API là cụm từ viết tắt của Application Programming Interface hay còn gọi là Giao diện lập trình ứng dụng, là cơ chế cho phép 2 thành phần phần mềm giao tiếp với nhau thông qua một tập hợp các định nghĩa và giao thức. Giao diện có thể xem là một hợp đồng nhằm xác định cách thức hai ứng dụng giao tiếp với nhau thông qua các yêu cầu và phản hồi.

Ví dụ về API: Hệ thống phần mềm của cơ quan thời tiết chứa dữ liệu về thời tiết hàng ngày. Ứng dụng thời tiết trên điện thoại của bạn sẽ lấy dữ liệu với hệ thống trên thông qua API và cập nhật thông tin về thời tiết hàng ngày để hiển thị trên điện thoại của bạn hàng ngày.

b. Cách thức hoạt động của API

Kiến trúc API thường được giải thích dưới dạng máy chủ và máy khách. Ứng dụng gửi yêu cầu gọi là máy khách, còn ứng dụng gửi phản hồi gọi là máy chủ.

c. Web API

Web API là một phương thức được sử dụng để các website hay ứng dụng web khác nhau có thể trao đổi thông tin, dữ liệu qua lại. Mỗi khi thực hiện truy xuất thông tin, Web API sẽ trả lại dữ liệu dưới dạng JSON hoặc XML thông qua giao thức HTTP hoặc HTTPS.

Ví dụ: API REST của Twitter cho phép các website khác truy cập một phần thông tin, để tích hợp các tính năng vào ứng dụng của mình.

Giao thức HTTP (Hypertext Transfer Protocol) là giao thức truyền tải siêu văn bản. Đây là giao thức tiêu chuẩn cho World Wide Web (www) để truyền tải dữ liệu dưới dạng văn bản, âm thanh, hình ảnh, video từ Web Server tới trình duyệt web của người dùng và ngược lại.

HTTPS (Hypertext Transfer Protocol Secure) là giao thức truyền tải siêu văn bản an toàn. Thực chất, đây chính là giao thức HTTP nhưng tích hợp thêm Chứng chỉ bảo mật SSL nhằm mã hóa các thông điệp giao tiếp để tăng tính bảo mật. Có thể hiểu, HTTPS là phiên bản HTTP an toàn, bảo mật hơn.

5.2. Các phương pháp đánh giá mô hình

5.2.1. Classification Report

Classification report là một công cụ đánh giá mô hình học máy, đặc biệt hữu ích trong các bài toán phân loại. Nó tóm tắt các chỉ số hiệu suất chính của mô hình phân loại, cho phép ta nhanh chóng nắm bắt được hiệu quả của mô hình đối với từng lớp dữ liệu. Dưới đây là các thành phần chính của một classification report:

1. Precision (Độ chính xác):

- Đo lường tỷ lệ dự đoán đúng trong số các dự đoán dương tính. Precision cao cho thấy mô hình ít có khả năng dự đoán sai các lớp dương tính.

2. Recall (Độ nhạy):

- Đo lường tỷ lệ phát hiện đúng các trường hợp dương tính trong số tất cả các trường hợp thực tế dương tính. Recall cao cho thấy mô hình có khả năng nhận diện tốt các lớp dương tính.

3. F1 Score:

- Là trung bình điều hòa giữa Precision và Recall, cung cấp một chỉ số duy nhất để đánh giá hiệu suất của mô hình trong trường hợp có sự mất cân đối giữa FP và FN. F1 Score cao cho thấy mô hình hoạt động tốt cả về độ chính xác và độ nhạy.

4. Support:

- Số lượng mẫu thực tế cho mỗi lớp trong tập dữ liệu. Chỉ số này cho biết tần suất xuất hiện của từng lớp trong tập dữ liệu.

5. Accuracy (Độ chính xác tổng thể):

- Mặc dù không phải là một phần riêng biệt trong báo cáo phân loại, nhưng thường được tính toán và hiển thị, cho thấy tỷ lệ đúng cho tất cả các lớp.

Việc sử dụng Classification Report nhằm giúp ta đánh giá tổng quan hiệu suất , cung cấp một cái nhìn tổng quan nhanh chóng về hiệu suất của mô hình trên từng lớp. Bên cạnh đó cũng giúp ta tối ưu hóa mô hình thông qua việc nhận diện các lớp mà mô hình hoạt động kém, từ đó có thể thực hiện các điều chỉnh cần thiết.

5.2.2. K-Fold Cross Validation

a. Cross validation là gì?

Cross validation là phương pháp thống kê được sử dụng để ước lượng hiệu quả của các mô hình học máy, thường được sử dụng để so sánh và chọn ra mô hình tốt nhất.

b. K-Fold Cross Validation là gì?

K-Fold Cross Validation là phương pháp lấy mẫu để đánh giá mô hình học máy trong đó k là tham số vô cùng quan trọng, đại diện cho số nhóm mà dữ liệu được chia ra. Một cách thường được sử dụng là chia tập huấn luyện thành k tập con và không có phần tử chung, có kích thước bằng nhau. Tại mỗi lần thử nghiệm (*run*), một trong k tập con được lấy ra làm tập kiểm thử (*validate set*). Mô hình được xây dựng dựa trên k-1 tập con còn lại. Hiệu suất mô hình được xác định dựa trên trung bình của các độ lỗi trên tập huấn luyện và tập kiểm thử.

5.3. Tìm kiếm siêu tham số để tối ưu hóa mô hình

Siêu tham số là các tham số không được học trong quá trình huấn luyện mà cần được cấu hình trước và ảnh hưởng đến hiệu suất và học tập của mô hình.

Tìm kiếm siêu tham số là quá trình tìm kiếm và lựa chọn giá trị tối ưu cho các siêu tham số trong mô hình học máy.

Có một phương pháp phổ biến để tìm kiếm siêu tham số như GridSearch (Tìm kiếm theo lưới). Random Search (Tìm kiếm ngẫu nhiên), ... Ở đây mình lựa chọn phương pháp GridSearch do sự đơn giản và dễ hiểu của nó.

a. GridSearchCV là gì?

GridSearchCV là một kỹ thuật để tìm kiếm siêu tham số (hyperparameter) tốt nhất cho mô hình học máy. Nó giúp xác định sự kết hợp tối ưu của các tham số để cải thiện hiệu suất của mô hình. Tùy vào mô hình khác nhau mà các tham số được đưa vào sẽ khác nhau.

b. Cách thức GridSearchCV hoạt động

Ta tiến hành truyền các giá trị được xác định trước cho siêu tham số vào hàm GridSearchCV. Sau đó GridSearchCV tiến hành kết hợp các tham số và tiến hành thử từng kết hợp để xem kết hợp các tham số nào hoạt động tốt nhất.

5.4. Xử lý dữ liệu mất cân bằng

Khi dữ liệu bị mất cân bằng, dữ liệu huấn luyện cho nhóm thiểu số trở thành trở ngại trong việc xác định ranh giới để phân loại hiệu quả. Để giải quyết vấn đề này thì phương pháp sao chép các ví dụ từ lớp thiểu số. Tuy nhiên việc này chỉ tăng cường mà không cung cấp thông tin mới cho mô hình. Một cách khác là tổng hợp các ví dụ mới có thể mang lại thông tin phong phú hơn và cải thiện khả năng của mô hình.

SMOTE (Synthetic Minority Over-sampling Technique) là một kỹ thuật phổ biến được sử dụng để xử lý vấn đề mất cân bằng dữ liệu, đặc biệt trong các bài toán phân loại. SMOTE tạo ra các ví dụ tổng hợp bằng cách tìm các điểm gần nhau trong không gian đặc điểm và sinh ra các ví dụ mới nằm giữa các điểm.

5.5. Trích xuất đặc trưng dữ liệu

Trong phân tích ngôn ngữ tự nhiên có các phương pháp thông dụng để trích xuất đặc trưng dữ liệu như TF-IDF, Bag-Of-Words (túi từ), tuy nhiên trong bài toán phân loại cảm xúc, mình lựa chọn TF-IDF thay vì Bag-Of-Words vì

- **Trọng số từ quan trọng:** TF-IDF tính trọng số từ dựa trên tần suất xuất hiện và mức độ quan trọng của từ trong toàn bộ tập dữ liệu, giúp tập trung vào từ liên quan đến cảm xúc hơn là từ phổ biến.
- **Loại bỏ từ không cần thiết:** TF-IDF giảm trọng số của từ phổ biến nhưng ít ý nghĩa, giúp mô hình tập trung vào những từ có ý nghĩa cảm xúc.
- **Cải thiện độ chính xác:** TF-IDF cung cấp thông tin sâu hơn về ngữ cảnh của từ trong tài liệu, giúp mô hình phân loại cảm xúc chính xác hơn.

a. **TF-IDF**

TF-IDF (Term Frequency-Inverse Document Frequency) là một phương pháp phổ biến trong xử lý ngôn ngữ tự nhiên (NLP) được sử dụng để đánh giá tầm quan trọng của một từ trong một văn bản mà bản thân văn bản nằm trong tập hợp văn bản.

b. **TF(Term frequency) là gì?**

Tần suất thuật ngữ (Term frequency : TF) hoạt động bằng cách xem xét tần suất của một thuật ngữ cụ thể so với tài liệu. Có nhiều cách đo lường hoặc định nghĩa tần suất:

- Số lần từ xuất hiện trong một tài liệu (số đếm thô).
- Tần suất thuật ngữ được điều chỉnh theo độ dài của tài liệu (số đếm thô của các lần xuất hiện chia cho số từ trong tài liệu).
- Tần suất được tính theo quy mô logarithmic (ví dụ: $\log(1 + \text{số đếm thô})$).
- Tần suất Boolean (ví dụ: 1 nếu thuật ngữ xuất hiện, hoặc 0 nếu thuật ngữ không xuất hiện trong tài liệu)

c. IDF (inverse document frequency) là gì?

Tần suất ngược tài liệu (Inverse Document Frequency - IDF) xem xét mức độ phổ biến (hoặc không phổ biến) của một từ trong toàn bộ tập tài liệu. IDF được tính như sau, trong đó t là thuật ngữ (từ) mà chúng ta đang muốn đo lường mức độ phổ biến và NNN là số lượng tài liệu (d) trong tập hợp tài liệu (D). Mẫu số đơn giản là số tài liệu mà thuật ngữ t xuất hiện trong đó.

Thuật toán IDF:

$$\text{idf}(t, D) = \log \left(\frac{N}{\text{count}(d \in D : t \in d)} \right)$$

Bằng cách sử dụng tần suất ngược, chúng ta có thể giảm trọng số của các thuật ngữ phổ biến nhưng ít quan trọng và tăng cường thêm tác động của các thuật ngữ ít xuất hiện.

d. TF-IDF

TF-IDF là sự kết hợp của TF và IDF là tầm quan trọng của một thuật ngữ tỉ lệ nghịch với tần suất của nó trong các tài liệu. Điểm số TF-IDF càng cao, thuật ngữ càng quan trọng hoặc liên quan, khi một thuật ngữ trở nên ít liên quan hơn, điểm số TF-IDF của nó sẽ tiến gần đến 0.

Công thức TF-IDF:

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$$

5.6. Mô hình học máy

a. Logistic Regression

Phương trình hồi quy là công cụ thống kê để mô hình hóa và phân tích mối quan hệ giữa một biến phụ thuộc và các biến độc lập.

Mô hình hồi quy logistic dựa trên hàm logistic (hay còn gọi là hàm sigmoid) để biến đổi đầu vào là một giá trị x bất kỳ và trả về đầu ra là một giá trị xác suất nằm trong khoảng từ 0 đến 1 nhằm để phân loại nhị phân, tức là đối tượng thuộc vào một trong hai nhóm. Hàm sigmoid được biểu diễn dưới dạng:

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

Trong đó:

- $\sigma(x)$ là đầu ra trong khoảng từ 0 đến 1 (giá trị xác suất ước lượng)
- z là đầu vào của hàm (giá trị dự đoán của thuật toán, ví dụ như $mx+b$)
- e là hằng số Euler và là cơ số của logarit tự nhiên.

b. Support Machine Vector

SVM(Support Vector Machine) là một thuật toán học máy thuộc loại giám sát, được sử dụng chủ yếu cho các bài toán phân loại và cũng có thể áp dụng cho hồi quy. Ý tưởng chính của SVM là tìm ra một mặt phân cách (hyper-plane) tối ưu giữa các lớp dữ liệu sao cho khoảng cách từ mặt phân cách này đến các điểm dữ liệu gần nhất (các support vectors)

là lớn nhất. Điều này giúp cho mô hình có khả năng phân loại tốt và tăng độ chính xác trên các tập dữ liệu chưa từng thấy trước đó.

c. Random Forest

Random Forest được biết đến là phương pháp thống kê dùng với mục đích phân loại, hội quy và nhiều quyết định khác thông qua việc xây dựng nhiều cây quyết định.

Ở mỗi cây quyết định trong thuật toán thường không dùng toàn bộ dữ liệu hay toàn bộ thuộc tính để xây dựng cây vì mỗi cây có thể đưa ra dự đoán không đúng khiến cho mô hình cây quyết định có thể bị thiên lệch cao. Nhưng kết quả cuối cùng quyết định nên thông tin sẽ bổ sung cho nhau dẫn đến mô hình có dự đoán tốt nhất.

5.7. Overfitting (Quá khớp)

Overfitting là hiện tượng mô hình tìm được quá khớp với dữ liệu huấn luyện. Việc quá khớp này có thể dẫn đến dự đoán bị nhiễu, và chất lượng mô hình không còn tốt trên tập kiểm tra nữa.

6. Quá trình hình thành

6.1. Mô tả cách lấy dữ liệu

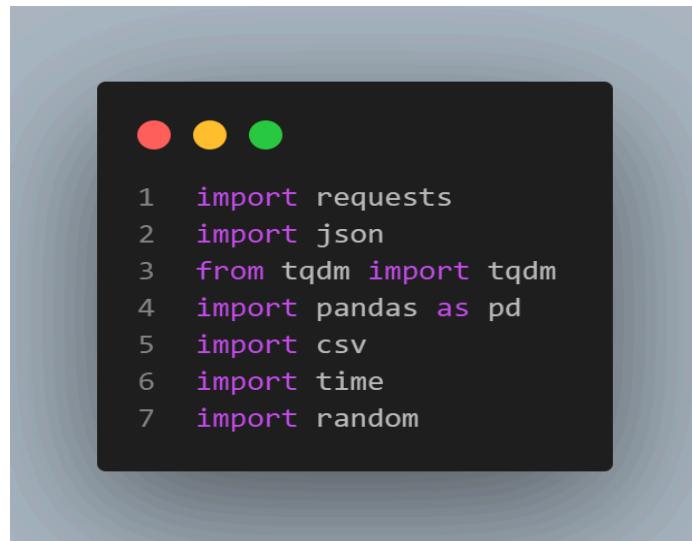
Với mục tiêu phân tích cảm xúc bình luận tại Nhà sách Tiki trên sàn TMĐT Tiki, dữ liệu được thu nhập dựa vào các bình luận, đánh giá trên sàn thương mại điện tử Tiki.vn. Dữ liệu này là công khai và tập trung vào danh mục Nhà sách Tiki.

Do sàn thương mại điện tử Tiki.vn có áp dụng nhiều cơ chế để bảo mật thông tin, vì vậy các thư viện Python thường dùng để truy cập và trích xuất dữ liệu (VD: thư viện

BeautifulSoup, Web Scraper, v.v.) sẽ gặp khó khăn khi lấy dữ liệu từ sàn thương mại điện tử này. Do đó, để thu thập dữ liệu, nhóm đã sử dụng API do Tiki cung cấp qua các bước:

Bước 1: Import các thư viện cần thiết:

- requests: Thư viện giúp gửi các yêu cầu HTTP để lấy dữ liệu từ API của Tiki.
- json: Thư viện dùng để xử lý dữ liệu định dạng JSON.
- tqdm: Thư viện cung cấp tiến trình hiển thị dạng thanh (progress bar) khi chạy vòng lặp.
- pandas: Thư viện dùng để xử lý và lưu trữ dữ liệu trong cấu trúc DataFrame.
- csv: Thư viện hỗ trợ ghi dữ liệu vào file CSV.
- time và random: Dùng để tạo độ trễ ngẫu nhiên giữa các lần gửi yêu cầu đến API để tránh bị chặn bởi server.



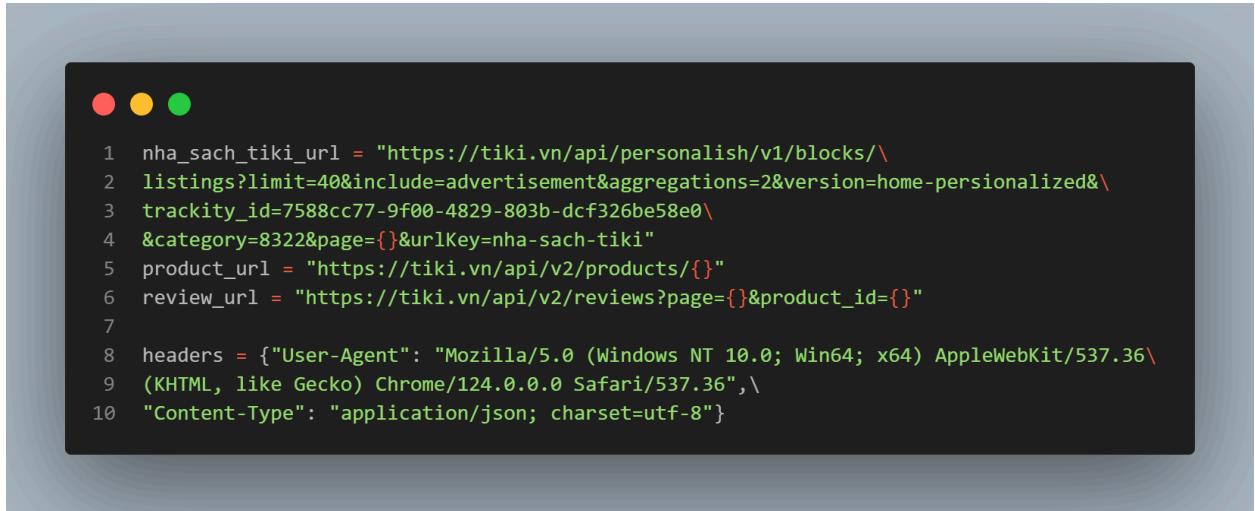
```
● ● ●  
1 import requests  
2 import json  
3 from tqdm import tqdm  
4 import pandas as pd  
5 import csv  
6 import time  
7 import random
```

A screenshot of a Jupyter Notebook cell. The cell contains Python code for importing various libraries: requests, json, tqdm, pandas, csv, time, and random. The code is numbered from 1 to 7. Above the code, there are three colored dots (red, yellow, green) which are part of the Jupyter interface for cell execution status.

Bước 2: Truyền các thông tin về cấu hình URL và tiêu đề yêu cầu (headers):

- nha_sach_tiki_url: URL của API lấy danh sách sản phẩm từ danh mục "nhà sách Tiki".
- product_url: URL của API lấy thông tin chi tiết của từng sản phẩm (dùng product_id).

- review_url: URL của API lấy danh sách các đánh giá (review) của một sản phẩm nhất định (dùng product_id).
- headers: Tiêu đề của yêu cầu HTTP, bao gồm thông tin trình duyệt (User-Agent) và định dạng nội dung (Content-Type).



```
1 nha_sach_tiki_url = "https://tiki.vn/api/personalish/v1/blocks/\
2 listings?limit=40&include=advertisement&aggregations=2&version=home-personalized&\
3 trackity_id=7588cc77-9f00-4829-803b-dcf326be58e0\
4 &category=8322&page={}&urlKey=nha-sach-tiki"
5 product_url = "https://tiki.vn/api/v2/products/{}"
6 review_url = "https://tiki.vn/api/v2/reviews?page={}&product_id={}"
7
8 headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36\
9 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36",\
10 "Content-Type": "application/json; charset=utf-8"}
```

Bước 3: Hàm crawl_product_id()

- Mục đích: Thu thập danh sách product_id của các sản phẩm trong danh mục "nhà sách Tiki".
- Hoạt động:
 1. Khởi tạo danh sách product_list để lưu các ID sản phẩm và biến i để theo dõi số trang.
 2. Gửi yêu cầu đến API Tiki để lấy danh sách sản phẩm cho từng trang.
 3. Với mỗi sản phẩm trong kết quả trả về, lấy id và lưu vào danh sách product_list.
 4. Tăng số trang ($i += 1$) và tiếp tục lặp cho đến khi không còn sản phẩm hoặc có lỗi xảy ra.

```
● ● ●

1 def crawl_product_id():
2     product_list = []
3     i = 1
4     while True:
5         print("Crawl page: ", i)
6         response = requests.get(nha_sach_tiki_url.format(i), headers=headers)
7         if(response.status_code != 200):
8             break
9         products = json.loads(response.text)[ "data" ]
10        if(len(products) == 0):
11            break
12        for product in products:
13            product_id = str(product[ "id" ])
14            product_list.append(product_id)
15        i += 1
16    return product_list, i
```

Bước 4: Hàm crawl_review(product_list):

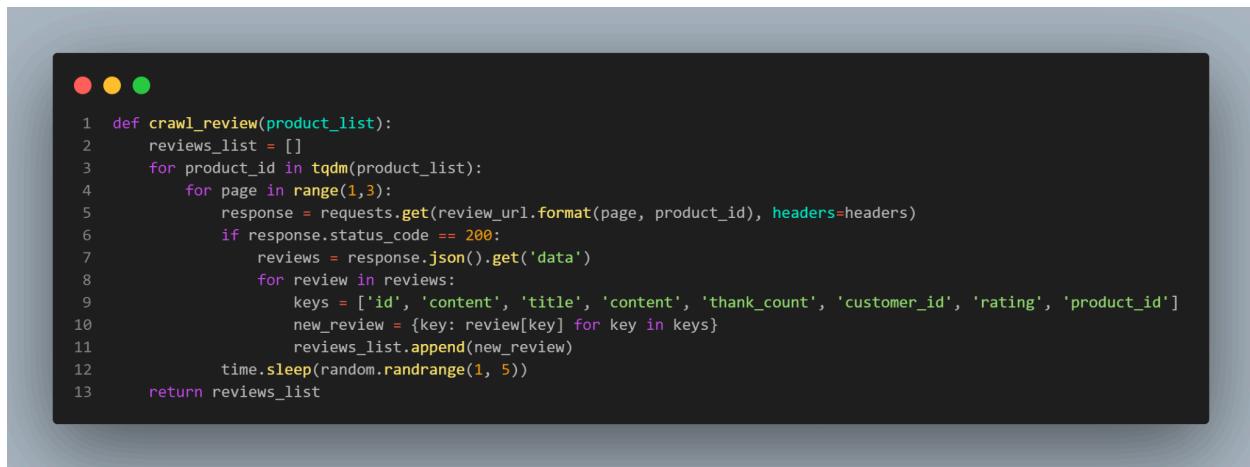
- Mục đích: Thu thập danh sách các đánh giá cho từng sản phẩm.
- Hoạt động:
 1. Khởi tạo danh sách reviews_list để lưu trữ các đánh giá.
 2. Lặp qua từng product_id trong danh sách product_list.
 3. Với mỗi sản phẩm, gửi yêu cầu đến API Tiki để lấy các đánh giá cho sản phẩm đó (lấy 2 trang đầu tiên).
 4. Với mỗi đánh giá (review) trong danh sách reviews, chọn các trường cần thiết:
 - id: ID của đánh giá.
 - title: Tiêu đề của đánh giá.
 - content: Nội dung của đánh giá.
 - thank_count: Số lượt cảm ơn cho đánh giá đó.
 - customer_id: ID của người dùng đã viết đánh giá.

- rating: Số sao mà người dùng đánh giá sản phẩm (thường từ 1-5).
- product_id: ID của sản phẩm mà đánh giá đó thuộc về.

Dữ liệu của mỗi đánh giá sẽ được lưu vào một dictionary new_review, trong đó chỉ chứa các trường được liệt kê (keys).

Dictionary này được thêm vào danh sách reviews_list để lưu tất cả các đánh giá.

5. Thêm độ trễ ngẫu nhiên (1-5 giây) giữa các lần yêu cầu để tránh bị chặn bởi server.



```

 1 def crawl_review(product_list):
 2     reviews_list = []
 3     for product_id in tqdm(product_list):
 4         for page in range(1,3):
 5             response = requests.get(review_url.format(page, product_id), headers=headers)
 6             if response.status_code == 200:
 7                 reviews = response.json().get('data')
 8                 for review in reviews:
 9                     keys = ['id', 'content', 'title', 'content', 'thank_count', 'customer_id', 'rating', 'product_id']
10                     new_review = {key: review[key] for key in keys}
11                     reviews_list.append(new_review)
12                     time.sleep(random.randrange(1, 5))
13     return reviews_list

```

Bước 5. Hàm save_to_csv(reviews_list, path)

- Mục đích: Lưu danh sách đánh giá (reviews_list) vào file CSV.
- Hoạt động:
 1. Lấy dữ liệu của phần tử đầu tiên trong reviews_list và lưu vào file CSV với chế độ ghi (mode=w), có bao gồm header.
 2. Với mỗi đánh giá còn lại, tiếp tục ghi dữ liệu vào file CSV nhưng bỏ qua phần header (mode=a để thêm vào file, không ghi đè).

```
● ● ●  
1 def save_to_csv(reviews_list, path):  
2     df = pd.DataFrame(reviews_list[:1])  
3     df.to_csv(path, index=False, encoding='utf-8', mode='w')  
4  
5     for review in reviews_list[1:]:  
6         df = pd.DataFrame([review])  
7         df.to_csv(path, index=False, encoding='utf-8', mode='a', header=False)
```

Bước 6: Chạy các hàm

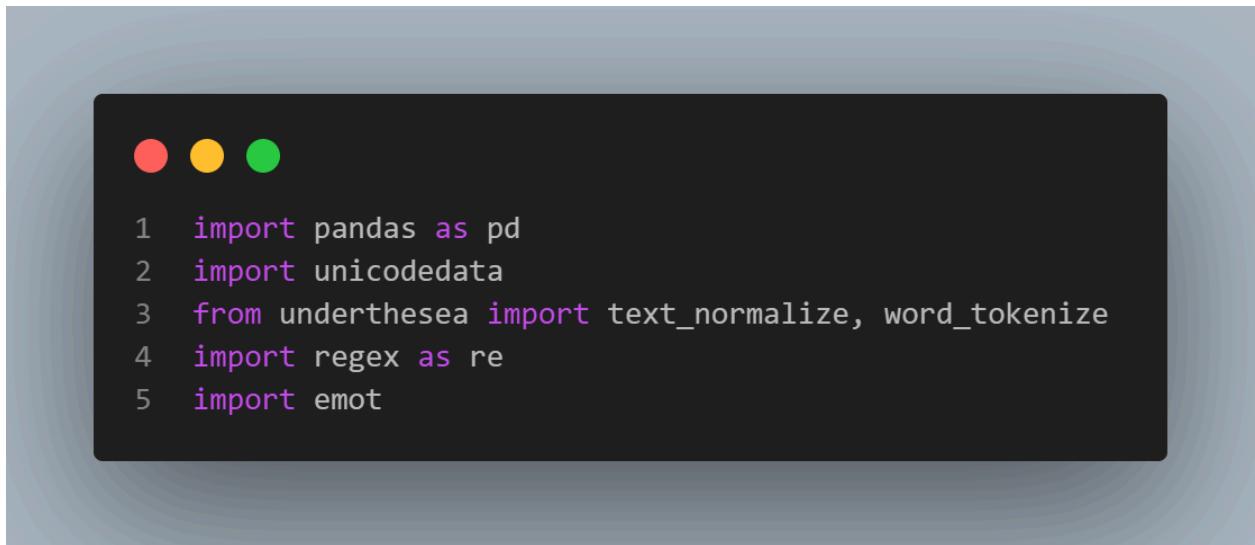
- product_list, page = crawl_product_id(): Thu thập danh sách product_id của các sản phẩm.
- reviews_list = crawl_review(product_list): Thu thập các đánh giá của từng sản phẩm từ danh sách.
- save_to_csv(reviews_list, "./data/review.csv"): Lưu dữ liệu đánh giá vào file CSV có đường dẫn ./data/review.csv.

7. Tiền xử lý dữ liệu

Bước 1: Import các thư viện cần thiết cho quá trình tiền xử lý dữ liệu

- pandas: Thư viện dùng để xử lý và phân tích dữ liệu dưới dạng bảng (DataFrame).
- unicodedata : Thư viện này được sử dụng để chuẩn hóa các ký tự Unicode.
- underthesea: Thư viện hỗ trợ xử lý ngôn ngữ tự nhiên cho tiếng Việt, bao gồm các chức năng chuẩn hóa dấu câu (text_normalize) và tách từ tiếng Việt (word_tokenize).
- regex: Thư viện nâng cao để xử lý các biểu thức chính quy, giúp tìm kiếm và thay thế các chuỗi ký tự trong văn bản phức tạp.

- emot: Thư viện này dùng để nhận diện và loại bỏ emoji và emoticon khỏi văn bản.



```
1 import pandas as pd
2 import unicodedata
3 from underthesea import text_normalize, word_tokenize
4 import regex as re
5 import emot
```

Bước 2: Đọc các file có sẵn để tiền xử lý dữ liệu

- teencode.txt: Chứa danh sách từ lóng (teencode) và các từ chính thống tương ứng. Tạo một dictionary từ các cặp từ này để chuyển đổi từ teencode thành từ đúng.
- wrong-word.txt: Chứa danh sách các từ bị sai chính tả cần loại bỏ.
- vietnamese-stopwords.txt: Chứa danh sách stopwords (các từ dừng) trong tiếng Việt cần loại bỏ.

```
● ● ●
```

```
1 #1. Đọc file teencode
2 with open('files/teencode.txt', 'r', encoding='utf8') as file:
3     teencode_list = file.read().split('\n')
4     teencode_dict = {}
5     for line in teencode_list:
6         key, value = line.split('\t')
7         teencode_dict[key] = str(value)
8 file.close()
9
10 #2. Đọc file cá từ bị sai
11 with open('files/wrong-word.txt', 'r', encoding='utf8') as file:
12     wrongword_list = file.read().split('\n')
13 file.close()
14
15 #3. Đọc file các từ stopword trong tiếng việt
16 with open('files/vietnamese-stopwords.txt', 'r', encoding='utf8') as file:
17     stopword_list = file.read().split('\n')
18 file.close()
```

Bước 3: Xóa emoji và emoticon

- Khởi tạo đối tượng emot_obj: Tạo đối tượng từ thư viện emot để phát hiện emoji và emoticon.
- Phát hiện emoji và emoticon bằng các phương thức emoji() và emoticons().
- Duyệt qua emoji và emoticon và xử lý các ký tự đặc biệt như [(), ., +, *] bằng cách thêm dấu \ nhằm tránh bị lỗi khi sử dụng biểu thức chính quy.
- Tạo biểu thức để tìm emoji/emoticon có khoảng trắng trước và sau và thay thế bằng khoảng trắng.

```

● ● ●

1 #Xóa emoji và emoticon
2 emot_obj = emot.core.emot()
3 def xoa_emoji_va_emoticon(text):
4     emojis = emot_obj.emoji(text)
5     emoticons = emot_obj.emoticons(text)
6     for i in emojis['value'] + emoticons['value']:
7         for punc in [')', '.', '+', '*']:
8             i = i.replace(punc, '\\\\' + punc)
9             pattern = r'\s' + i + r'\s'
10            text = re.sub(pattern, ' ', text)
11    return text
12 df['clean_content'] = df['content'].apply(lambda x: xoa_emoji_va_emoticon(x))

```

Bước 4: Chuyển đổi teencode thành các từ chuẩn

- Chia chuỗi văn bản thành một danh sách các từ, sử dụng khoảng trắng làm dấu phân cách
- Duyệt qua danh sách các từ text, nếu từ nào nằm trong từ điển teencode thì đổi thành từ đúng, nếu không thì giữ nguyên từ đó. Sau đó gộp các từ lại thành văn bản.

```

● ● ●

1 def chuyen_teencode(text):
2     text = ' '.join(teencode_dict[word] if word in teencode_dict else word for word in text.split())
3     return text
4 df['clean_content'] = df['clean_content'].apply(lambda x: chuyen_teencode(x))

```

Bước 5: Xóa các từ bị sai

- Chia chuỗi văn bản thành một danh sách các từ, sử dụng khoảng trắng làm dấu phân cách

- Duyệt qua danh sách các từ text và chỉ gộp các từ không nằm trong danh sách các từ sai lại thành văn bản.

```
1 def xoa_wrongword(text):
2     text = ''.join(['' if word in wrongword_list else word for word in text.split()])
3     return text
4 df['clean_content'] = df['clean_content'].apply(lambda x: xoa_wrongword(x))
```

Bước 6: Xóa dấu câu và số văn bản

- Khai báo một biến pattern (mẫu) chứa các chữ cái Tiếng Việt và các chữ cái Latinh cơ bản
 - Tìm tất cả các ký tự hợp lệ với pattern (mẫu) trong text (văn bản) và kết hợp thành một chuỗi các nhau bởi khoảng trắng.
 - Chuyển văn bản thành chữ thường và xóa các khoảng trắng bị thừa.

Bước 7: Chuẩn hóa unicode Tiếng Việt

- Tạo hàm loaddicchar tạo dic giúp chuẩn hóa và chuyển đổi các ký tự tiếng Việt từ mã hóa Windows-1252 sang UTF-8

- + Tạo 2 chuỗi là uniChars là chuỗi chứa tất cả các ký tự có dấu Tiếng Việt và unsignChars là chuỗi chứa các ký tự không dấu tương ứng với các ký tự trong uniChars.
 - + Tạo 2 chuỗi là char1252 chứa các ký tự đặc biệt tiếng Việt trong mã hóa Windows-1252, được ngăn cách bởi dấu | và charutf8 chứa các ký tự tương ứng trong mã hóa UTF-8, cũng được ngăn cách bởi dấu |.
 - + Gắn mỗi giá trị của chuỗi char1252 tương ứng với ký tự trong chuỗi charutf8 và lưu vào 1 từ điển.
 - Tạo hàm chuyen_unicode dùng hàm sub trong biểu thức chính quy để tìm và thay thế các ký tự trong văn bản bằng các ký tự tương ứng trong từ điển đã tạo trước đó.

Bước 8: Chuẩn hóa dấu Tiếng Việt

- Dùng hàm `text_normalize` của thư viện `underthesea` dùng để chuẩn hóa dấu Tiếng Việt về dạng chuẩn để đồng nhất các từ. Ví dụ, nó sẽ chuyển các trường hợp có dấu không chuẩn (chẳng hạn như “òà” hoặc “oà”) thành dạng chính xác.

```
● ● ●  
1 def chuan_hoa_dau_tieng_viet(text):  
2     text = text_normalize(text)  
3     return text  
4 df['clean_content'] = df['clean_content'].apply(lambda x: chuan_hoa_dau_tieng_viet(x))
```

Bước 9: Tách từ Tiếng Việt

- Sử dụng hàm word_tokenize để tách văn bản thành các từ và cụm từ riêng biệt. Đối với tiếng Việt, các từ ghép và các cụm từ chỉ có ý nghĩa khi ở cạnh nhau, word_tokenize để đảm bảo cho chúng không bị tách rời thành các từ riêng lẻ.

```
● ● ●  
1 def tach_tu_tieng_viet(text):  
2     text = word_tokenize(text, format="text")  
3     return text
```

Bước 10: Xóa stopwords (từ dừng) trong văn bản

- Từ dừng là những từ phổ biến trong ngôn ngữ nhưng không mang nhiều ý nghĩa trong việc phân tích như "và", "nhưng", "thì", "là", "ở" trong tiếng Việt. Những từ này thường được loại bỏ để giảm nhiễu trong văn bản và tăng độ chính xác cho các mô hình NLP.
- Chia chuỗi văn bản thành một danh sách các từ, sử dụng khoảng trắng làm dấu phân cách

- Duyệt qua danh sách các từ text và chỉ gộp các từ không nằm trong danh sách các từ dừng lại thành văn bản.

```
● ● ●  
1 def xoa_stopword(text):  
2     text = ' '.join('' if word in stopword_list else word for word in text.split())  
3     text = re.sub(r'\s+', ' ', text).strip()  
4     return text  
5 df['clean_content'] = df['clean_content'].apply(lambda x: xoa_stopword(x))
```

7. EDA

Với mục đích phân loại cảm xúc của đánh giá của sách trên sàn thương mại điện tử Tiki, mình sẽ chú trọng vào việc phân tích 3 cột dữ liệu là rating, thank_count và clean_content cũng như mối liên hệ giữa 3 cột này.

Bước 1: Import các thư viện cần thiết

```
● ● ●  
1 import pandas as pd  
2 import regex as re  
3 import seaborn as sns  
4 import numpy as np  
5 from wordcloud import WordCloud  
6 import matplotlib.pyplot as plt
```

1. pandas: Đọc, ghi, thao tác và tính toán thống kê trên dữ liệu dạng bảng.

2. regex: Giúp phân tích và xử lý văn bản phức tạp bằng cách xác định các mẫu chuỗi.
3. seaborn: Tạo các biểu đồ đẹp mắt và dễ hiểu từ DataFrame, hỗ trợ phân tích thống kê.
4. numpy (import numpy as np): Thực hiện các phép toán số học và đại số tuyến tính trên dữ liệu lớn.
5. wordcloud : Phân tích tần suất từ trong văn bản để trực quan hóa các từ khóa nổi bật.
6. matplotlib.pyplot : Dùng vẽ các loại biểu đồ khác nhau như biểu đồ đường, cột, tán xạ, và nhiều loại khác.

Bước 2: Đọc dữ liệu



```

1 df = pd.read_csv('data/clean_review.csv', encoding= 'utf-8')
2 df.head()

```

	id	title	content	thank_count	customer_id	rating	product_id	clean_content
0	19906519	Cực kì hài lòng	Sau khi phản ánh việc tác giả đặt tên sách trù...	8	13450478	5	275025208	phản_ánh tác_giả sách_trùng sách_nước_ngoài nh...
1	19968834	Hài lòng	Tạm thời thấy ổn.Đọc xong sẽ review sau.hj.Yh...	0	19821878	4	275025208	tạm_thời ổn đọc review hj yhanks
2	19943636	Rất không hài lòng	quyển sách với tựa đề làm hiểu lầm cái khách h...	0	14317039	1	275025208	quyển sách tựa đề làm khách_hàng viết không đú...
3	19919040	Cực kì hài lòng	Giao hàng nhanh, đóng gói cẩn thận	0	13713637	5	275025208	giao hàng đóng_gói cẩn_thận
4	19963835	Hài lòng	Tạm được, phù hợp những bạn chưa đọc nhiều sác...	0	7159313	4	275025208	tạm đọc sách tác_giả nội_suu_tâm trùng_lặp

Bước 3. Kiểm tra thông tin sơ bộ



```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15403 entries, 0 to 15402
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   id               15403 non-null   int64  
 1   title             15403 non-null   object  
 2   content            15403 non-null   object  
 3   thank_count        15403 non-null   int64  
 4   customer_id       15403 non-null   int64  
 5   rating             15403 non-null   int64  
 6   product_id         15403 non-null   int64  
 7   clean_content      15402 non-null   object  
dtypes: int64(5), object(3)
memory usage: 962.8+ KB
```

Bộ dữ liệu được sử dụng EDA là bộ dữ liệu đã được tiền xử lý, tuy nhiên cột clean_content bị thiếu 1 dòng sau khi bị xóa hết các từ bên mình tiến hành loại bỏ dòng có clean_content bị thiếu và cả những dòng dữ liệu bị trùng lặp do khi khách hàng mua 2 sản phẩm nhưng chỉ đánh giá 1 lần nên nội dung đánh giá sẽ trùng nhau.



```
1 df.dropna(subset= ['clean_content'], inplace=True)
2 df.drop_duplicates(subset= ['clean_content'], inplace=True)
3 df.shape
```

(15402, 8)

Dữ liệu sau khi loại bỏ dữ liệu bị thiếu và bỏ dữ liệu bị trùng lặp còn 15402 dòng và 8 cột

Bước 4: Mô tả dữ liệu của các cột dữ liệu là số và mối quan hệ giữa rating và thank_count



```
1 df[['rating', 'thank_count']].describe()
```

	rating	thank_count
count	15402.000000	15402.000000
mean	4.474484	1.380989
std	1.096904	7.718660
min	1.000000	0.000000
25%	5.000000	0.000000
50%	5.000000	0.000000
75%	5.000000	1.000000
max	5.000000	551.000000

Dữ liệu trong cột ‘rating’ và ‘thank_count’ cho thấy với 15402 đánh giá. Lượt đánh giá 5 sao là cao nhất, 1 sao là thấp nhất và số sao trung bình cho mỗi đánh giá là xấp xỉ 4.47 sao, tương đối cao chứng tỏ đánh giá 4, 5 sao chiếm đa số. Lượng thank_count trung bình là 1,3 cảm ơn trên một đánh giá, cho thấy người dùng có xu hướng đọc cái đánh giá của khách hàng mua trước để có thêm thông tin để quyết định có mua hàng hay không.

```

● ● ●
1 df_group = df[['id', 'rating', 'thank_count']].groupby('rating').agg({'thank_count': ['sum', 'mean'], 'id': 'count'})
2 print(df_group)

```

rating	thank_count	id	
	sum	mean	count
1	1354	1.585480	854
2	545	1.164530	468
3	646	0.790698	817
4	870	0.530488	1640
5	17855	1.536178	11623

Các đánh giá được cảm ơn nhiều là các đánh giá 1 sao và 5 sao, điều này cho thấy người dùng có xu hướng tránh các review mang tính trung lập không mang tính quyết định cao, lựa chọn những review có xu hướng chi tiết hơn và có giá trị hơn về mặt ưu điểm và nhược điểm của các cuốn sách.

Bước 5: Mối quan hệ của cột clean_content với thank_count và rating

- Tạo thêm một cột word_count để tính số từ của các đánh giá trên Tiki.



```
1 df['word_count'] = df['clean_content'].apply(lambda x: len(re.split('_| ', x)) if isinstance(x, str) else 0)
```

- Dùng ma trận tương quan để phát họa mối quan hệ giữa word_count với thank_count và rating.

```
● ● ●  
1 corr_matrix = df[['rating', 'word_count', 'thank_count']].corr()  
2 sns.heatmap(corr_matrix, annot = True, cmap = 'Blues')  
3 plt.show()
```



Mỗi quan hệ giữa rating và word_count có hệ số tương quan là -0.022693, rất yếu và âm, cho thấy rằng khi số từ trong nhận xét tăng lên, điểm đánh giá có thể giảm nhẹ hoặc không có thay đổi rõ rệt.

Mỗi quan hệ giữa word_count và thank_count có hệ số tương quan là 0.205389 có thể chỉ ra rằng những nhận xét có số từ lớn hơn có xu hướng chứa nhiều từ cảm ơn hơn.

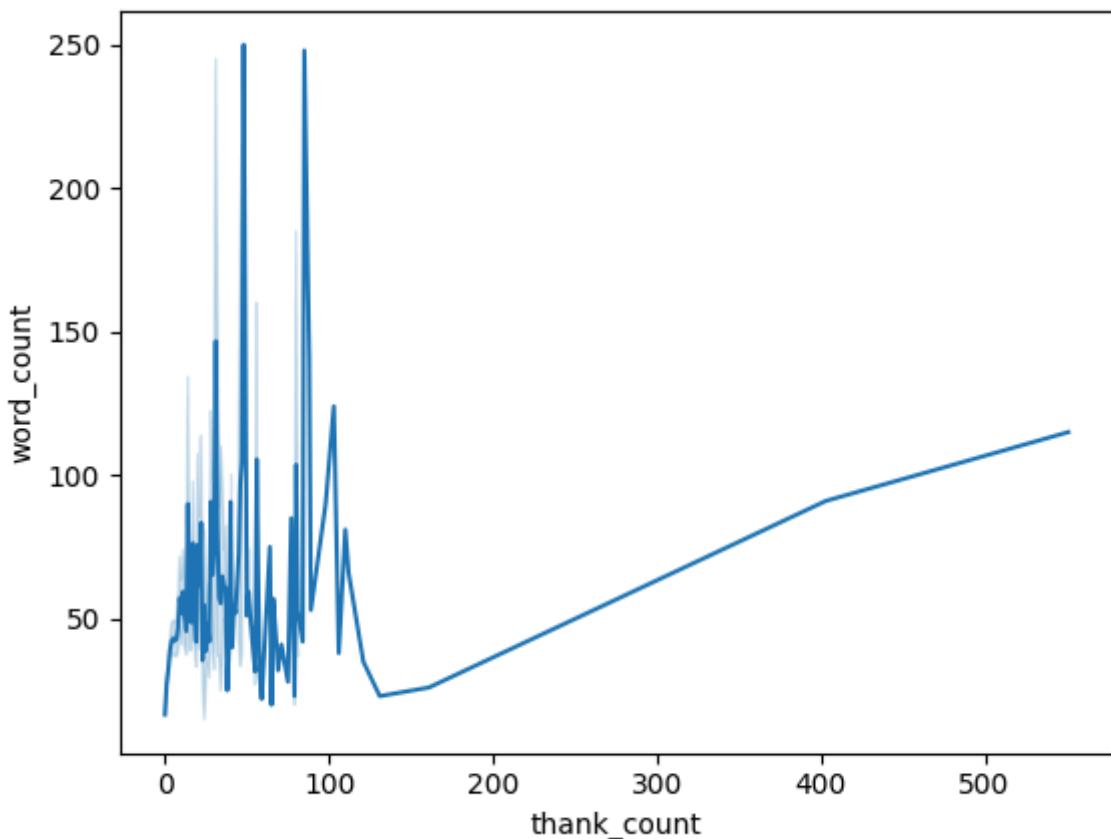
```
● ● ●  
1 df_group = df[['id', 'rating', 'thank_count', 'word_count']].groupby('rating')\n2 .agg({'thank_count': ['sum', 'mean'], 'id': 'count', 'word_count': 'mean'})\n3 print(df_group)
```

rating	thank_count		id word_count	
	sum	mean	count	mean
1	1354	1.585480	854	23.487119
2	545	1.164530	468	23.341880
3	646	0.790698	817	23.332925
4	870	0.530488	1640	23.161585
5	17855	1.536178	11623	21.630474

Trung bình số từ của mỗi rating không có sự khác biệt nhiều, tuy nhiên do lượng rating của mỗi mức lại khác nhau nên những rating có nhiều đánh giá hơn có số lượng từ nhiều hơn so với các mức còn lại.



```
1   sns.lineplot(x='thank_count', y='word_count', data=df)
```



Có độ tương quan thuận giữa word_count và thank_count, điều này cho thấy bình luận nhiều chữ nhận được nhiều lượt cảm ơn hơn, có lẽ do bình luận dài thường chứa nhiều thông tin mang tính quyết định, dễ đánh giá sản phẩm hơn.

Bước 6: Các đánh giá thường nói về các vấn đề gì?

- Tiến hành gán nhãn dữ liệu theo 2 nhãn là positive nếu rating ≥ 4 còn lại là negative.

```
● ● ●  
1 df_final['class'] = df_final.apply(lambda x: 'negative' if x['rating'] < 4 else 'positive', axis=1)  
2 df_final = df_final[['id', 'clean_content','class']]  
3 df_final.reset_index(drop=True, inplace=True)  
4 df_final.head()
```

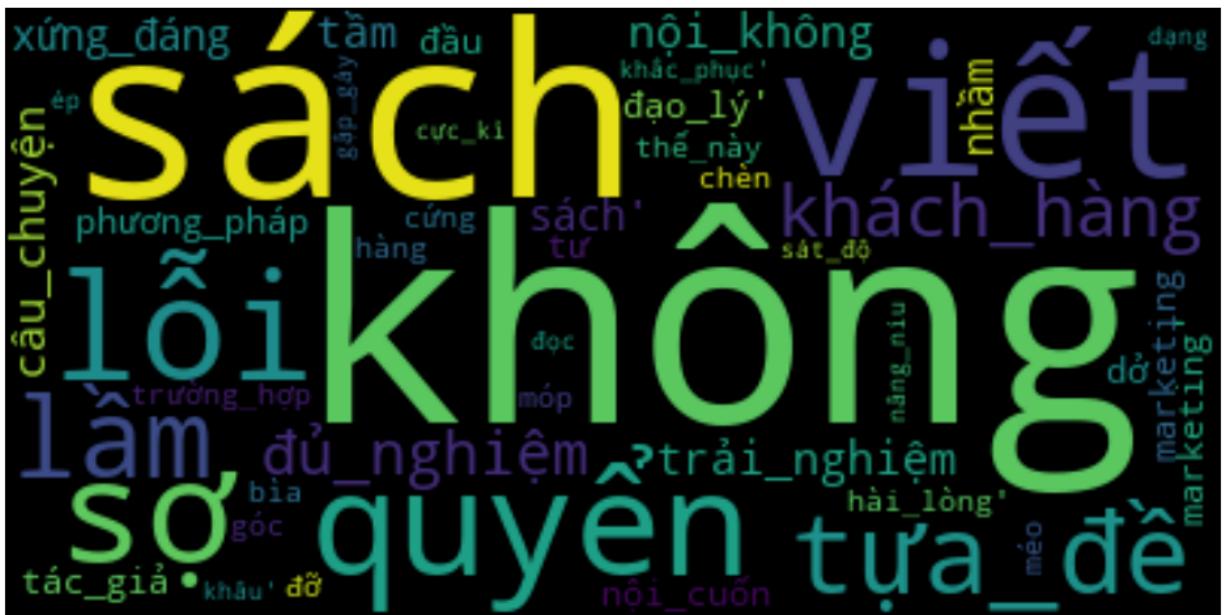
```
class  
positive      12718  
negative      2105  
Name: count, dtype: int64
```

Negative (tiêu cực)

Vẽ chart để minh họa các từ trong các đánh giá được xem là Negative

```
● ● ●  
1 df_final_notlike = df_final[df_final['class'] == 'negative' ]  
2 wc_notlike = WordCloud(  
3     background_color='black',  
4     max_words=500  
5 )  
6 wc_notlike.generate(str(df_final_notlike['clean_content'].values))
```

```
● ● ●  
1 plt.figure(figsize=(12, 12))  
2 plt.imshow(wc_notlike, interpolation='bilinear')  
3 plt.axis('off')  
4 plt.show()
```



Trong các từ phổ biến nhất trong tập dữ liệu văn bản được đánh giá negative, những từ có kích thước là lối, đạo lý, móp, dở, gãy gáy tuy nhiên các từ tiêu cực này chỉ xuất hiện ít so với các từ khác. Những từ có kích thước lớn hơn xuất hiện nhiều hơn trong dữ liệu như "không", "sách", "viết", "quyển", "tựa đề", "sợ" có kích thước lớn, cho thấy đây là những từ xuất hiện thường xuyên trong các nhận xét tuy nhiên nó vẫn là những từ chưa mang tính chất để phân loại đánh giá là tích cực hay tiêu cực.

Positive (tích cực)

Vẽ chart để minh họa biểu đồ về đánh giá positive (tích cực)

```
● ● ●  
1 df_final_like = df_final[df_final['class'] == 'positive' ]  
2 wc_like = WordCloud(  
3     background_color='black',  
4     max_words=500  
5 )  
6 wc_like.generate(str(df_final_like['clean_content'].values))
```

```
● ● ●  
1 plt.figure(figsize=(12, 12))  
2 plt.imshow(wc_like, interpolation='bilinear')  
3 plt.axis('off')  
4 plt.show()
```



Một số từ như "tiki", "sách", "hàng", "giải_quyết", "đọc", "tốt", "hôm", "giao" có kích thước lớn, cho thấy đây là những từ xuất hiện thường xuyên trong các nhận xét. Những từ như "tiki", "sách", và "hàng" có thể liên quan đến trải nghiệm mua sắm trên nền tảng thương mại điện tử Tiki, với sản phẩm chủ yếu là sách. Các từ như "giải_quyết", "giao", và "tốt" có thể liên quan đến chất lượng dịch vụ, trải nghiệm vận chuyển, và sự hài lòng của khách hàng. Tuy nhiên các từ cảm xúc và phản ánh dịch vụ: Từ như "tốt", "thân_thiện", "chất_lượng", "tác_giả" cho thấy rằng nhiều người đánh giá tích cực về sản phẩm hoặc dịch vụ. Một số từ khác như "giải_quyết", "cần_thận", và "phản_ánh" có thể liên quan đến phản hồi của khách hàng đối với các vấn đề cần được giải quyết.

Các từ này chưa thật sự chuẩn với nhãn của đánh giá, nên mình đã tiến hành dán tay hơn 14000 đánh giá.

8. Huấn luyện mô hình

Gán nhãn là một phần quan trọng trong quá trình huấn luyện mô hình, mặc dù có những công cụ hỗ trợ để gán nhãn tự động nhưng vẫn chưa đáng tin cậy, đặc biệt ngôn ngữ được phân tích ở đây là Tiếng Việt. Nên mình đã quyết định gán nhãn tay với 2 nhãn là ‘Positive’ và ‘Negative’ cho hơn 14000 dữ liệu đã qua tiền xử lý và lưu dữ liệu vào file train_data.csv để tiến hành huấn luyện.

Bước 1: Import các thư viện cần thiết để huấn luyện mô hình

1. pandas: Đọc, ghi, thao tác và tính toán thống kê trên dữ liệu dạng bảng (DataFrame).
2. numpy: Thực hiện các phép toán số học và đại số tuyến tính trên dữ liệu lớn.
3. matplotlib.pyplot (plt): Thư viện vẽ đồ thị trực quan phổ biến trong Python. Nó cung cấp các hàm để tạo ra các biểu đồ, đồ thị trực quan từ dữ liệu.
4. seaborn (sns): Thư viện xây dựng trên matplotlib giúp tạo ra các biểu đồ đẹp hơn và trực quan hơn, đặc biệt hữu ích cho việc phân tích và trực quan hóa dữ liệu thống kê.
5. preprocessing: Làm hàm do chính mình tự định nghĩa để tiền xử lý dữ liệu trước khi huấn luyện mô hình nằm trong file preprocessing.py

6. `sklearn.feature_extraction.text (TfidfVectorizer)`: Dùng để chuyển đổi văn bản thành các vector đặc trưng bằng cách sử dụng TF-IDF (Term Frequency-Inverse Document Frequency), một cách tiếp cận phổ biến trong xử lý văn bản.
7. `imblearn (under_sampling, over_sampling, SMOTE)`: Thư viện dùng để xử lý dữ liệu mất cân bằng. SMOTE là phương pháp tạo thêm dữ liệu mẫu thiểu số để cân bằng tập dữ liệu.
8. `sklearn.model_selection (train_test_split, GridSearchCV, cross_val_score)`:
 - `train_test_split`: Chia tập dữ liệu thành hai phần: tập huấn luyện và tập kiểm thử.
 - `GridSearchCV`: Dùng để tìm kiếm các tham số tốt nhất cho mô hình bằng cách thử nghiệm trên nhiều tổ hợp tham số khác nhau.
 - `cross_val_score`: Đánh giá mô hình bằng phương pháp cross-validation (chia nhỏ dữ liệu và kiểm tra hiệu quả mô hình trên các phần chia khác nhau).
9. `sklearn (metrics, accuracy_score, f1_score)`: Các công cụ đo lường hiệu suất của mô hình như độ chính xác (accuracy) và điểm F1 (f1 score), giúp đánh giá kết quả của các mô hình học máy.
10. `sklearn.preprocessing (LabelEncoder)`: Dùng để mã hóa các nhãn phân loại (label) thành các số nguyên, giúp các mô hình học máy dễ dàng xử lý.
11. `sklearn.svm (SVC)`: Mô hình SVM (Support Vector Classifier), một trong những thuật toán phân loại phổ biến.
12. `sklearn.ensemble (RandomForestClassifier)`: Mô hình phân loại dựa trên nhiều cây quyết định (decision trees) để cải thiện độ chính xác và tránh overfitting.
13. `sklearn.linear_model (LogisticRegression)`: Mô hình hồi quy logistic, một thuật toán phân loại dựa trên xác suất, dùng cho các bài toán phân loại nhị phân.



```
1 import pandas as pd
2 import numpy as np
3 from underthesea import word_tokenize
4 from wordcloud import WordCloud
5 import matplotlib.pyplot as plt
6 import seaborn as sns
7 from preprocessing import *
8 from sklearn.feature_extraction.text import TfidfVectorizer
9 from imblearn import under_sampling, over_sampling
10 from imblearn.over_sampling import SMOTE
11 from sklearn.model_selection import train_test_split
12 from sklearn import metrics
13 from sklearn.metrics import accuracy_score, f1_score
14 from sklearn.model_selection import cross_val_score
15 from sklearn.preprocessing import LabelEncoder
16 from sklearn.svm import SVC
17 from sklearn.ensemble import RandomForestClassifier
18 from sklearn.linear_model import LogisticRegression
19 from sklearn.model_selection import GridSearchCV
```

Bước 2: Đọc dữ liệu và loại bỏ dữ liệu bị thiếu và bị trùng lặp



```
1 train_data = pd.read_csv('data/train_data.csv')
2 train_data.dropna(subset='clean_content', inplace=True)
3 train_data.drop_duplicates(subset='clean_content', inplace=True)
```

Bước 3: Mã hóa biến mục tiêu

- Loại bỏ các khoảng trắng dư thừa ở đầu và cuối của các giá trị trong cột class
- Mã hóa nhãn ‘positive’ thành 1 và ‘negative’ thành 0

```

● ● ●

1 train_data['class'] = train_data['class'].str.strip()
2 train_data['class'] = train_data['class'].map({'positive': 1, 'negative': 0})

```

Bước 4: Sử dụng TF-IDF để biến đổi văn bản thành vector đặc trưng (Trích xuất đặc trưng dữ liệu)

Biến đổi văn bản thành các vector số bằng phương pháp TF-IDF.

- ngram_range=(1,5): Sử dụng n-grams từ 1 đến 5 từ.
- max_df=0.5: Bỏ qua các từ xuất hiện trong hơn 50% tài liệu.
- min_df=5: Chỉ giữ lại những từ xuất hiện ít nhất 5 lần trong các tài liệu.
- sublinear_tf=True: Sử dụng sublinear_tf, tức là sử dụng logarit để biến đổi tần số của các từ.
- fit_transform: Học từ dữ liệu và biến đổi văn bản thành vector.

```

● ● ●

1 tfidf_vectorizer = TfidfVectorizer(ngram_range=(1,5), max_df=0.5, min_df=5, sublinear_tf=True, norm='l2')
2 texts = tfidf_vectorizer.fit_transform(train_data['clean_content'].values.astype('U'))
3 #df = pd.DataFrame(texts.toarray(), columns=tfidf_vectorizer.get_feature_names_out())
4 labels = train_data['class']

```

Bước 5: Xử lý dữ liệu mất cân bằng

Sử dụng kỹ thuật SMOTE để tạo thêm mẫu thiểu số (tăng mẫu) để cân bằng tỷ lệ giữa các lớp dữ liệu.

```
● ● ●  
1 smt = SMOTE(random_state=42)  
2 texts_smt, labels_smt = smt.fit_resample(texts,labels)
```

Bước 6: Chia dữ liệu thành tập huấn luyện và tập kiểm thử

Chia dữ liệu thành 70% (huấn luyện) và 30% (kiểm tra).

```
● ● ●  
1 X_train, X_test, y_train, y_test = train_test_split(texts_smt, labels_smt, test_size=0.3, random_state=42)
```

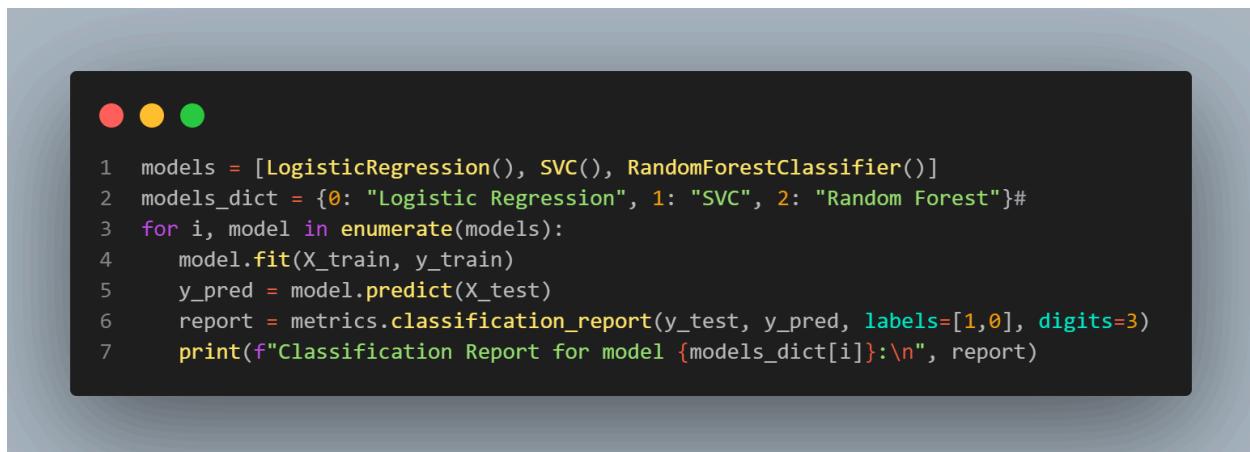
Bước 7: Lựa chọn mô hình

Trong nghiên cứu của Priyanshi Kathuria et al. (2022), Logistic Regression đạt được độ chính xác cao khi áp dụng cho các bình luận và đánh giá trên các sàn thương mại điện tử, chứng minh rằng mô hình này rất hiệu quả trong các bài toán phân loại nhị phân, nhất là khi dữ liệu đã được xử lý tốt. Logistic Regression sử dụng một hàm sigmoid để biến đổi các giá trị dự đoán thành xác suất, từ đó quyết định phân loại, giúp nó trở thành một lựa chọn đơn giản và dễ hiểu.

Trong khi đó, nghiên cứu của Đặng Quang Vinh (2023) cho thấy rằng SVM (Support Vector Machine) có thể mang lại độ chính xác cao trong việc giải quyết các bài toán định giá phức tạp như định giá quyền chọn. Đây là một mô hình linh hoạt, có thể được áp dụng cho cả phân loại và hồi quy, và thường hoạt động tốt với dữ liệu không tuyến tính nhờ vào kernel trick.

Về phần Ketan Gupta et al. (2022), việc sử dụng các bộ dữ liệu từ Amazon, Yelp, và IMDB để phân tích các đánh giá cũng chỉ ra rằng Random Forest hoạt động hiệu quả nhất với độ chính xác vượt qua các thuật toán khác, bao gồm cả Naïve Bayes và KNN. Random Forest là một mô hình dựa trên các cây quyết định, giúp giảm thiểu overfitting và tăng tính chính xác khi làm việc với các tập dữ liệu lớn và phức tạp.

Vì vậy mình quyết định ứng dụng 3 mô hình này vào bài phân tích và chọn ra mô hình nào có hiệu suất tốt hơn để tối ưu.



```
● ● ●  
1 models = [LogisticRegression(), SVC(), RandomForestClassifier()]  
2 models_dict = {0: "Logistic Regression", 1: "SVC", 2: "Random Forest"}#  
3 for i, model in enumerate(models):  
4     model.fit(X_train, y_train)  
5     y_pred = model.predict(X_test)  
6     report = metrics.classification_report(y_test, y_pred, labels=[1,0], digits=3)  
7     print(f"Classification Report for model {models_dict[i]}:\n", report)
```

Classification Report for model Logistic Regression:				
	precision	recall	f1-score	support
1	0.921	0.886	0.903	3380
0	0.891	0.925	0.908	3432
accuracy			0.905	6812
macro avg	0.906	0.905	0.905	6812
weighted avg	0.906	0.905	0.905	6812
Classification Report for model SVC:				
	precision	recall	f1-score	support
1	0.930	0.969	0.949	3380
0	0.968	0.928	0.948	3432
accuracy			0.948	6812
macro avg	0.949	0.948	0.948	6812
weighted avg	0.949	0.948	0.948	6812
Classification Report for model Random Forest:				
	precision	recall	f1-score	support
1	0.948	0.902	0.925	3380
0	0.908	0.952	0.929	3432
accuracy			0.927	6812
macro avg	0.928	0.927	0.927	6812
weighted avg	0.928	0.927	0.927	6812

Dựa trên tất cả các chỉ số, SVC là mô hình có hiệu suất tốt nhất. Mô hình này có precision, recall, f1-score, và accuracy vượt trội so với Logistic Regression và Random Forest, đặc biệt khi đánh giá trên cả hai lớp phân loại.

Bước 8: Tìm kiếm siêu tham số

Tối ưu hóa mô hình SVC bằng cách tìm kiếm các tham số tốt nhất thông qua việc sử dụng GridSearchCV (Thử nghiệm các tổ hợp tham số khác nhau (như C, kernel, gamma) để tìm ra tổ hợp tốt nhất.)

Các siêu tham số của mô hình SVC bao gồm

- Kernel (loại siêu phẳng (hyperplane) nào để phân loại data)
 - linear: dùng siêu phẳng tuyến tính (là đường thẳng trong trường hợp data 2D)
 - rbf và poly: dùng siêu phẳng phi tuyến
- Gamma (parameter cho siêu phẳng phi tuyến). là mức độ ảnh hưởng của các điểm dữ liệu trong việc xác định ranh giới quyết định.
 - Gamma nhỏ: tầm ảnh hưởng của điểm dữ liệu lớn, mô hình sẽ nhìn vào nhiều điểm dữ liệu xa hơn để xác định ranh giới phân loại. → ranh giới phân loại mượt mà hơn
 - Gamma lớn: tầm ảnh hưởng của mỗi điểm dữ liệu nhỏ → những điểm ở gần ranh giới quyết định mới có ảnh hưởng đến việc xác định phân chia các lớp. → ranh giới phân loại phức tạp
 - Giá trị của gamma càng cao thì mô hình càng fit với train data. Tuy nhiên, càng tăng gamma thì càng dễ đến hiện tượng overfitting.
- C là tham số penalty of hàm error, kiểm soát mức độ mô hình phạt các lỗi phân loại.
 - C là việc trade off giữa việc mô hình tạo ra một đường thẳng hoặc mặt phẳng chia các lớp mà không cần uốn cong quá nhiều để phân loại chính xác mọi điểm dữ liệu với việc phân loại đúng các điểm dữ liệu.
 - C nhỏ: mô hình cho phép phân loại sai, tạo ra mặt phẳng ranh giới quyết định không uốn cong và tránh tình trạng overfitting

- C lớn: mô hình cố gắng phân loại nhiều điểm dữ liệu đúng nhất có thể, tạo ra ranh giới quyết định phức tạp, dẫn đến hiện tượng overfitting.

```
● ● ●  
1 parameters = {  
2     'kernel': ('linear', 'rbf'),  
3     'C': [0.125, 0.25, 0.5, 1, 2, 4],  
4     'gamma': [0.125, 0.25, 0.5, 1, 2, 4]  
5 }  
6  
7 grid = GridSearchCV(SVC(), param_grid=parameters, scoring='accuracy', cv=5)  
8 grid_search = grid.fit(X_train, y_train)  
9 print("Best parameters:", grid_search.best_params_)
```

```
Best score: 0.946  
Best estimator: SVC(C=4, gamma=2)  
Best parameters: {'C': 4, 'gamma': 2, 'kernel': 'rbf'}
```

Kết quả nhận được là tổ hợp C=4, gamma=2 và kernel là rbf

Bước 9: Xây dựng mô hình với siêu tham số vừa tìm được

Huấn luyện mô hình SVC với các tham số tối ưu và dự đoán trên tập kiểm tra.

```
● ● ●  
1 model = SVC(C=4, gamma=2, kernel='rbf')  
2 model.fit(X_train, y_train)  
3 y_pred = model.predict(X_test)  
4  
5 report1 = metrics.classification_report(y_test, y_pred, labels=[1, 0], digits=3)  
6 print("Classification Report on Test Set:\n", report1)
```

Classification Report on Test Set:				
	precision	recall	f1-score	support
1	0.941	0.975	0.957	3380
0	0.974	0.940	0.957	3432
accuracy			0.957	6812
macro avg	0.957	0.957	0.957	6812
weighted avg	0.958	0.957	0.957	6812

Kết quả cuối cùng sau nhiều lần kiểm tra overfitting và gán lại nhãn thì mô hình có độ chính xác là 95,7% cao hơn so với giá trị ban đầu là 90%

Bước 10: Kiểm tra overfitting (quá khớp)

Kiểm tra xem mô hình có bị overfitting (phù hợp quá mức với dữ liệu huấn luyện) không bằng cách so sánh kết quả trên tập huấn luyện và tập kiểm tra.

```
● ● ●  
1 model = SVC(C=4, gamma=2, kernel='rbf')  
2 model.fit(X_train, y_train)  
3 y_train_pred = model.predict(X_train)  
4 report2 = metrics.classification_report(y_train, y_train_pred, labels=[1,0], digits=3)  
5 print("Classification Report on Train Set:\n", report2)
```

Nếu mô hình bị overfitting, ta tiến hành đến bước tiếp theo để phân tích các đánh giá bị gán nhầm sai.

Classification Report on Train Set:				
	precision	recall	f1-score	support
1	1.000	0.998	0.999	7973
0	0.998	1.000	0.999	7921
accuracy			0.999	15894
macro avg	0.999	0.999	0.999	15894
weighted avg	0.999	0.999	0.999	15894

Kết quả kiểm định trên tập huấn luyện sau nhiều lần gán lại nhầm là 99%

Bước 11: Phân tích lỗi (Error Analysis)

Phân tích những dự đoán sai của mô hình để tìm hiểu tại sao mô hình mắc lỗi và cải thiện chất lượng mô hình. Ta tìm kiếm các đánh giá trên tập huấn luyện bị đánh giá sai và tiến hành gán lại nhầm và tiếp tục chạy dự đoán kết quả mô hình trên tập huấn luyện cho đến khi đạt được độ chính xác như mong muốn.

```
1 mismatches = []
2 for id, review, true_label, predicted_label in zip(train_data['id'], X_train, y_train, y_train_pred):
3     if true_label != predicted_label:
4         if true_label != 1:
5             mismatches.append(f"{id}, True Label: {true_label}, Predicted Label: {predicted_label}")
6             print(f"{id}, True Label: {true_label}, Predicted Label: {predicted_label}")
7
8 result_length = len(mismatches)
9 print(f"Số lượng bản ghi không khớp: {result_length}")
```

Số lượng bản ghi không khớp: 0

Mình đã gán nhãn lại tất cả các nhãn sai cho đến khi số lượng nhãn sai là 0

Bước 12: Huấn luyện lại bằng Cross-Validation

```
1 cv_results = cross_val_score(SVC(C=4, gamma=2, kernel='rbf'), X_train, y_train, cv=5, scoring='accuracy', n_jobs=-1)
2
3 print(f"Cross-validation Accuracy Scores: {cv_results}")
4 print(f"Mean Accuracy: {cv_results.mean()}")
5 print(f"Standard Deviation: {cv_results.std()}")
```

Sau khi tiến hành phân tích lỗi và cải thiện tới độ chính xác như mong muốn thì mình tiến hành đánh giá mô hình bằng phương pháp Cross Validation để đến lại tính khách quan nhất cho mô hình.

```
Cross-validation Accuracy Scores: [0.94054734 0.94966971 0.94904058 0.94400755 0.94745123]
Mean Accuracy: 0.946143280983151
Standard Deviation: 0.0034180878928392316
```

Kết quả của mô hình đạt được 94%, tốt hơn so với mô hình ban đầu là 89%

Bước 13: Dùng mô hình vừa mới huấn luyện được để áp dụng lên tập kiểm tra mới để xem kết quả

```
1 def doc_lam_sach_du_lieu(filepath):
2     data = pd.read_csv(filepath)
3     data = processing_text(data)
4     data.dropna(subset='clean_content', inplace=True)
5     data.drop_duplicates(subset='clean_content', inplace=True)
6     return data
7
8 test_data = doc_lam_sach_du_lieu('data/test_data.csv')
9 test_data['content'] = test_data['clean_content'].values.astype('U')
10 test_list_vectorized = tfidf_vectorizer.transform(test_data['content'])
11 y_predict = model.predict(test_list_vectorized)
12 test_data['class'] = y_predict
13 test_data = test_data.sort_values(by=['class'])
14 test_data[['id', 'clean_content', 'class']].to_csv('data/submit.csv', index=False)
```

9. Kết quả phân tích wordcloud của mô hình trên tập kiểm thử

9.1. Đánh giá tiêu cực



Các từ khóa cho thấy một bức tranh khá rõ nét về những cảm xúc tiêu cực liên quan đến sản phẩm hoặc dịch vụ.

1. Chất lượng sản phẩm

- Các từ khóa: rách, cũ, xước, trầy, giấy, bìa, lõi, hư hỏng
- Ý nghĩa: Khách hàng không hài lòng với chất lượng sản phẩm, đặc biệt là về mặt vật lý như sách bị rách, bìa hư hỏng. Điều này cho thấy có vấn đề trong quá trình sản xuất hoặc vận chuyển.

2. Dịch vụ khách hàng

- Các từ khóa: giao nhầm, thiếu, chậm, hủy, không đúng, thất vọng
- Ý nghĩa: Khách hàng gặp phải nhiều vấn đề liên quan đến dịch vụ khách hàng như giao hàng chậm, giao nhầm sản phẩm, hủy đơn hàng. Điều này cho thấy quy trình phục vụ khách hàng chưa được hiệu quả.

3. Trải nghiệm đọc

- Các từ khóa: khó đọc, chữ nhỏ, giấy kém, không tập trung
- Ý nghĩa: Chất lượng sách không đáp ứng được nhu cầu của người đọc, gây khó khăn trong việc đọc và trải nghiệm.

4. Cảm xúc chung

- Các từ khóa: thất vọng, tức giận, buồn, tiếc, hối hận
- Ý nghĩa: Những từ khóa này thể hiện rõ ràng cảm xúc tiêu cực của khách hàng khi sử dụng sản phẩm hoặc dịch vụ.

Nhận xét chung: Các từ khóa cho thấy khách hàng đang trải qua những trải nghiệm tiêu cực liên quan đến chất lượng sản phẩm, dịch vụ khách hàng và trải nghiệm đọc. Điều này có thể dẫn đến việc khách hàng mất niềm tin vào thương hiệu và không muốn quay lại mua hàng.

Các vấn đề cần được cải thiện:

- Chất lượng sản phẩm: Cần kiểm soát chặt chẽ chất lượng sản phẩm trước khi giao đến tay khách hàng.
- Dịch vụ khách hàng: Nâng cao chất lượng dịch vụ khách hàng, đảm bảo giao hàng đúng hẹn, đúng sản phẩm và giải quyết nhanh chóng các khiếu nại của khách hàng.
- Trải nghiệm đọc: Cải thiện chất lượng sách, đảm bảo font chữ rõ ràng, giấy tốt để mang đến trải nghiệm đọc tốt nhất cho khách hàng.

Để khắc phục tình hình, bạn có thể:

- Tổ chức khảo sát khách hàng: Thu thập ý kiến phản hồi từ khách hàng để hiểu rõ hơn về vấn đề và đưa ra giải pháp phù hợp.
- Cải thiện quy trình sản xuất và đóng gói: Đảm bảo sản phẩm được sản xuất và đóng gói cẩn thận để tránh hư hỏng.
- Đào tạo nhân viên: Nâng cao kỹ năng và kiến thức cho nhân viên để họ có thể phục vụ khách hàng tốt hơn.
- Xây dựng chương trình chăm sóc khách hàng: Tạo ra các chương trình khuyến mãi, ưu đãi để thu hút khách hàng quay trở lại.

9.2. Đánh giá tích cực



Các đánh giá tích cực thường nói về những điểm mạnh của chất lượng sản phẩm và dịch vụ

1. Chất lượng sản phẩm:

- Bao bì sản phẩm được chăm chút: Từ khóa "gói cẩn thận", "sách đẹp" cho thấy khách hàng hài lòng với bao bì sản phẩm, điều này tạo ấn tượng tốt ngay từ cái nhìn đầu tiên.

2. Dịch vụ khách hàng:

- Quy trình đặt hàng và giao hàng thuận tiện: Các từ khóa "gọi", "đặt", "ship", "nhận" cho thấy quy trình mua hàng diễn ra suôn sẻ và nhanh chóng.

Để duy trì và phát triển những điểm mạnh này, bạn có thể thực hiện các biện pháp sau:

- Tăng cường chất lượng nội dung:
 - Đa dạng hóa thể loại: Cung cấp nhiều thể loại sách, truyện khác nhau để đáp ứng nhu cầu của nhiều đối tượng khách hàng.
 - Cập nhật xu hướng: Luôn cập nhật những xu hướng mới nhất về sách, truyện để thu hút độc giả.
 - Tổ chức các sự kiện văn hóa: Tổ chức các buổi ra mắt sách, tọa đàm với tác giả để tạo sự tương tác giữa tác giả và độc giả.
 - Nâng cao chất lượng dịch vụ khách hàng:

- Đào tạo nhân viên: Tổ chức các khóa đào tạo để nâng cao kỹ năng giao tiếp, giải quyết vấn đề của nhân viên.
- Xây dựng hệ thống chăm sóc khách hàng chuyên nghiệp: Đáp ứng nhanh chóng các thắc mắc, khiếu nại của khách hàng.
- Tích hợp các công cụ hỗ trợ khách hàng: Chatbot, FAQ, form liên hệ... để khách hàng dễ dàng tìm kiếm thông tin.
- Tăng cường tương tác với cộng đồng:
 - Tổ chức các cuộc thi, sự kiện: Tạo ra các hoạt động hấp dẫn để thu hút sự tham gia của khách hàng.
 - Xây dựng các nhóm cộng đồng: Tạo điều kiện cho khách hàng tương tác, chia sẻ với nhau.

10. Tài liệu tham khảo

1. API là gì?
<https://support.apple.com/vi-vn/guide/shortcuts-mac/apd2e30c9d45/mac>
2. API (Giao diện lập trình ứng dụng) là gì? <https://aws.amazon.com/vi/what-is/api/>
3. Giao thức HTTP và HTTPS là gì? Tại sao nên sử dụng HTTPS?
<https://cystack.net/vi/blog/http-vs-https-la-gi#giao-thuc-http-la-gi>
4. Giới thiệu về k-fold cross validation
<https://trituenhantao.io/kien-thuc/gioi-thieu-ve-k-fold-cross-validation/>
5. Hyperparameter Tuning with GridSearchCV
<https://www.mygreatlearning.com/blog/gridsearchcv/>
6. Tự học ML | Điều chỉnh siêu tham số SVM bằng GridSearchCV | ML
<https://nanado.edu.vn/tu-hoc-ml-dieu-chinh-sieu-tham-so-svm-bang-gridsearchcv-ml-a5345.html>
7. SMOTE for Imbalanced Classification with Python
<https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/>

8. Understanding TF-IDF for Machine Learning

<https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/>

9. Bài 6: Logistic Regresion (Hỏi quy Logistic)

<https://trituenhantao.io/machine-learning-co-ban/bai-6-logistic-regression-hoi-quy-logistic/>

10. SVM quá khó hiểu! Hãy đọc bài này

<https://trituenhantao.io/kien-thuc/svm-qua-kho-hieu-hay-doc-bai-nay/>

11. Random Forest là gì? Ứng dụng của thuật toán trong Machine Learning

<https://bizflycloud.vn/tin-tuc/random-forest-la-gi-20240814100430828.htm>

12. Unveiling the Power: TF-IDF vs Bag of Words

<https://myscale.com/blog/text-analysis-tf-idf-vs-bag-of-words/>

13. In Depth: Parameter tuning for SVC

<https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-75821539476>