BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC KINH TẾ TP HỒ CHÍ MINH

TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ

✍

ĐỒ ÁN MÔN HỌC

ĐỀ TÀI:

PHÂN TÍCH ĐÁNH GIÁ CỦA KHÁCH HÀNG TRÊN CÁC TRANG THƯƠNG MẠI ĐIỆN TỬ ĐỂ THU THẬP PHẢN HỒI CỦA KHÁCH HÀNG VỀ SẢN PHẨM (TÍCH CỰC VÀ TIÊU CỰC)

Học phần: Xử Lý Ngôn Ngữ Tự Nhiên

Nhóm Sinh Viên:

1. ĐỒNG ĐAN HOÀI
2. NGUYỄN QUỲNH KHÁNH HÀ
3. HUỲNH TRẦN ANH THY

Chuyên Ngành: KHOA HỌC DỮ LIỆU

Khóa: K46

Giảng Viên: TS. Đặng Ngọc Hoàng Thành

TP. Hồ Chí Minh, Ngày 15 tháng 12 năm 2022

MỤC LỤC

MỤC LỤC **1**

CHƯƠNG 1. TỔNG QUAN **3**

1.1. Giới Thiệu Bài Toán 3

1.2. Lý Do Chọn Lựa Đề Tài 3

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT **5**

2.1. Các Phương Pháp Tiền Xử Lý Dữ Liệu 5

2.1.1. Tiền Xử Lý Dữ Liệu *5*

2.1.2. Text Mining *5*

2.2. Các Mô Hình Phân Tích Dữ Liệu 8

2.2.1. Mô Hình Logistic Regression *8*

2.2.2. Mô Hình Naive Bayes *11*

CHƯƠNG 3. CÁC KẾT QUẢ THỰC NGHIỆM **13**

3.1. Bộ Dữ Liệu 13

3.1.1. Tổng quan về bộ dữ liệu *13*

3.1.2. Tiền xử lý dữ liệu *16*

3.1.3. Text Mining *21*

3.1.4. WordCloud *23*

3.2. Phân Chia Dữ Liệu 26

3.3. Huấn Luyện Dữ Liệu với Naive Bayes và Logistic Regression 28

3.3.1. Naive Bayes *28*

3.3.2. Logistic Regression *30*

3.4. Các Kết Quả Thực Nghiệm 30

3.4.1. Naive Bayes *30*

3.4.2 Logistic Regression *31*

3.5. Phân Tích và Đánh Giá 31

CHƯƠNG 4. KẾT LUẬN **32**

4.1. Các Kết Quả Đạt Được 32

4.2. Những Hạn Chế và Hướng Phát Triển 32
4.2.1. Hạn chế *32*
4.2.2. Hướng phát triển *32*
PHỤ LỤC I (Link Github) **34**
PHỤ LỤC II **34**
TÀI LIỆU THAM KHẢO **35**

CHƯƠNG 1. TỔNG QUAN

1.1. Giới Thiệu Bài Toán

Ngày nay, những tiến bộ của công nghệ thông tin đã làm thay đổi cách thức trao đổi buôn bán truyền thông giúp cho khách hàng dễ dàng truy cập thông tin và trao đổi ý kiến về sản phẩm và dịch vụ trên một quy mô lớn trong thời gian thực. Sự ra đời của mạng xã hội và các website đánh giá trực tuyến (như: Agoda, TripAdvisor, Yelp, Amazon...) cho phép khách hàng có cơ hội đưa ra ý kiến của mình thông qua các bài bình luận về sản phẩm, dịch vụ. Với sự bùng nổ của dữ liệu lớn (Big Data), các ý kiến bình luận của cộng đồng trực tuyến cần được thu thập và khai thác một cách tự động, cho phép các nhà kinh doanh theo dõi hành vi mua sắm, phát hiện sở thích và đánh giá sự hài lòng của khách hàng về chất lượng sản phẩm, dịch vụ. Vì thế, "Sentiment Analysis and Classification" đã trở thành tiêu điểm của rất nhiều nhà đầu tư. Hiện nay, có rất nhiều mô hình giúp người nghiên cứu có thể có những cái nhìn tổng quan nhất về vấn đề, thu thập được các nhận định, đánh giá từ khách hàng để cải thiện sản phẩm theo nhu cầu của người tiêu dùng nâng cao doanh số và thu lợi nhuận cao. Để làm rõ các vấn đề trên, nhóm chúng em cùng nhau tìm hiểu bài toán ***Phân tích đánh giá của khách hàng trên các trang thương mại điện tử để thu thập phản hồi của khách hàng về sản phẩm (tích cực và tiêu cực).***

Để giải quyết bài toán này, nhóm chúng em đã sử dụng công cụ hỗ trợ *Python, thư viện NLTK và hai thuật toán Logistic Regression, Naive Bayes*. Sơ lược về quá trình thực hiện như sau: đầu tiên nhóm chúng em lựa chọn bộ dữ liệu *"Womens Clothing E-Commerce Reviews"*. Tiếp đến tiến hành xử lý bước tiền xử lý dữ liệu trước khi sử dụng vào các thuật toán. Tiếp theo là vẽ biểu đồ WordCloud để nhìn thấy các *"Sentiment"* của khách hàng đối với sản phẩm. Cuối cùng là xây dựng mô hình Logistic Regression và Naive Bayes để phân tích đánh giá của khách hàng về các sản phẩm nhằm đưa ra phương án cải thiện tốt nhất.

1.2. Lý Do Chọn Lựa Đề Tài

Bài toán này tập trung vào việc sử dụng các kỹ thuật Ngôn ngữ tự nhiên để tìm ra các xu hướng chung trong suy nghĩ bằng văn bản của khách hàng. Mục tiêu là dự đoán liệu khách hàng có hài lòng với sản phẩm họ đã mua hay không bằng cách sử dụng thông tin trong văn bản đánh giá của họ. Một trong những thách thức trong nghiên cứu này là trích xuất thông tin hữu ích từ biến Văn bản Đánh giá bằng cách sử dụng các kỹ thuật khai thác văn bản. Nhóm chúng em sẽ xây dựng các mô hình phân loại "Sentiment" bằng cách sử dụng các thuật toán (Logistic Regression và Naive Bayes). Từ những kết quả của việc nghiên cứu này, công ty có thể đưa ra các chiến lược kinh doanh phù hợp cho từng sản phẩm 1 cách đúng đắn như tăng số lượng sản xuất khi sản phẩm đó nhận được nhiều

đánh giá hài lòng hay bãi bỏ sản phẩm khi chúng nhận quá nhiều lời đánh giá không hài lòng của khách hàng,...

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Các Phương Pháp Tiền Xử Lý Dữ Liệu

2.1.1. Tiền Xử Lý Dữ Liệu

a. Khái niệm

Tiền xử lý đề cập đến các phép biến đổi được áp dụng cho dữ liệu của chúng ta trước khi đưa nó vào thuật toán.

Tiền xử lý dữ liệu là một kỹ thuật được sử dụng để chuyển đổi dữ liệu thô thành một tập dữ liệu sạch. Nói cách khác, bất cứ khi nào dữ liệu được thu thập từ các nguồn khác nhau, nó được thu thập ở định dạng thô không khả thi cho việc phân tích.

b. Tại sao cần tiền xử lý dữ liệu

Để đạt được kết quả tốt hơn từ mô hình được áp dụng trong các dự án ML , định dạng của dữ liệu phải theo cách phù hợp. Một số mô hình ML được chỉ định cần thông tin ở định dạng được chỉ định, ví dụ: thuật toán Rừng ngẫu nhiên không hỗ trợ giá trị null, do đó, để thực thi thuật toán rừng ngẫu nhiên, giá trị rỗng phải được quản lý từ tập dữ liệu thô ban đầu.

Một khía cạnh khác là tập dữ liệu nên được định dạng theo cách mà nhiều hơn một thuật toán ML và Học sâu được thực thi trong một tập dữ liệu và tốt nhất trong số chúng được chọn.

2.1.2. Text Mining

a. Khái niệm

Khai thác văn bản, còn được gọi là khai thác dữ liệu văn bản, là quá trình chuyển đổi văn bản phi cấu trúc thành định dạng có cấu trúc để xác định các mẫu có ý nghĩa và hiểu biết mới. Bằng cách áp dụng các kỹ thuật phân tích nâng cao, chẳng hạn như Naïve Bayes, Support Vector Machines (SVM) và các thuật toán học sâu khác, các công ty có thể khám phá và khám phá các mối quan hệ ẩn trong dữ liệu phi cấu trúc của họ.

Văn bản là một trong những kiểu dữ liệu phổ biến nhất trong cơ sở dữ liệu. Tùy thuộc vào cơ sở dữ liệu, dữ liệu này có thể được tổ chức như 3 kiểu dữ liệu sau:

- **Structured data**: Dữ liệu này được chuẩn hóa thành định dạng bảng với nhiều hàng và cột, giúp lưu trữ và xử lý dễ dàng hơn cho các thuật toán phân tích và máy học. Dữ liệu có cấu trúc có thể bao gồm các đầu vào như tên, địa chỉ và số điện thoại.
- **Unstructured data**: Dữ liệu này không có định dạng dữ liệu được xác định trước. Nó có thể bao gồm văn bản từ các nguồn, như mạng xã hội hoặc đánh giá sản phẩm hoặc các định dạng đa phương tiện như tệp video và âm thanh.

- **Semi-structured data**: Như tên gợi ý, dữ liệu này là sự pha trộn giữa các định dạng dữ liệu có cấu trúc và phi cấu trúc. Mặc dù nó có một số tổ chức, nhưng nó không có đủ cấu trúc để đáp ứng các yêu cầu của cơ sở dữ liệu quan hệ. Ví dụ về dữ liệu bán cấu trúc bao gồm các tệp XML, JSON và HTML.

Vì 80% dữ liệu trên thế giới nằm ở định dạng phi cấu trúc nên khai thác văn bản là một phương pháp cực kỳ có giá trị trong các tổ chức. Các công cụ khai thác văn bản và kỹ thuật xử lý ngôn ngữ tự nhiên (NLP) cho phép chuyển đổi tài liệu phi cấu trúc thành định dạng có cấu trúc để cho phép phân tích và tạo ra thông tin chuyên sâu chất lượng cao . Đổi lại, điều này sẽ cải thiện việc ra quyết định của các tổ chức, dẫn đến kết quả kinh doanh tốt hơn.

b. Kỹ thuật

Quá trình khai thác văn bản bao gồm một số hoạt động cho phép người dùng suy luận thông tin từ dữ liệu văn bản phi cấu trúc. Trước khi người dùng có thể áp dụng các kỹ thuật khai thác văn bản khác nhau, người dùng phải bắt đầu với quá trình tiền xử lý văn bản, đó là thực hành làm sạch và chuyển đổi dữ liệu văn bản thành định dạng có thể sử dụng được. Thực tiễn này là một khía cạnh cốt lõi của xử lý ngôn ngữ tự nhiên (NLP) và nó thường liên quan đến việc sử dụng các kỹ thuật như nhận dạng ngôn ngữ, mã thông báo, gắn thẻ một phần của bài phát biểu, phân đoạn và phân tích cú pháp để định dạng dữ liệu phù hợp để phân tích. Khi quá trình tiền xử lý văn bản hoàn tất, người có thể áp dụng các thuật toán khai thác văn bản để thu thập thông tin chi tiết từ dữ liệu. Một số kỹ thuật khai thác văn bản phổ biến bao gồm:

- **Information retrieval**: Truy xuất thông tin (IR) trả về thông tin hoặc tài liệu có liên quan dựa trên một tập hợp truy vấn hoặc cụm từ được xác định trước. Các hệ thống IR sử dụng các thuật toán để theo dõi hành vi của người dùng và xác định dữ liệu liên quan. Truy xuất thông tin thường được sử dụng trong các hệ thống danh mục thư viện và các công cụ tìm kiếm phổ biến, như Google. Một số nhiệm vụ phụ IR phổ biến bao gồm:
- **Tokenization**: Đây là quá trình chia văn bản dạng dài thành các câu và từ được gọi là "tokens". Sau đó, chúng được sử dụng trong các mô hình, giống như mô hình túi từ (bag-of-words), cho các tác vụ phân cụm văn bản và so khớp tài liệu.
- **Stemming**: Điều này đề cập đến quá trình tách các tiền tố và hậu tố khỏi các từ để lấy được hình thức và ý nghĩa của từ gốc. Kỹ thuật này cải thiện việc truy xuất thông tin bằng cách giảm kích thước của tệp lập chỉ mục.
- **Xử lý ngôn ngữ tự nhiên (NLP)**: Xử lý ngôn ngữ tự nhiên, phát triển từ ngôn ngữ học tính toán, sử dụng các phương pháp từ nhiều ngành khác nhau, chẳng hạn như khoa học máy tính, trí tuệ nhân tạo, ngôn ngữ học và khoa học dữ liệu, để cho phép máy tính hiểu ngôn ngữ của con người ở cả dạng viết và lời nói. Bằng cách phân tích cấu trúc câu và ngữ pháp, các nhiệm vụ phụ của NLP cho phép máy tính "đọc". Các nhiệm vụ phụ phổ biến bao gồm:

- **Summarization**: Kỹ thuật này cung cấp một bản tóm tắt các đoạn văn bản dài để tạo ra một bản tóm tắt ngắn gọn, mạch lạc về các điểm chính của tài liệu.
- **Part-of-Speech (PoS) tagging**: Kỹ thuật này gán một thẻ cho mọi mã thông báo (token) trong tài liệu dựa trên phần lời nói của nó—tức là biểu thị danh từ, động từ, tính từ, v.v. Bước này cho phép phân tích ngữ nghĩa trên văn bản phi cấu trúc.
- **Text categorization**: Nhiệm vụ này, còn được gọi là phân loại văn bản, chịu trách nhiệm phân tích các tài liệu văn bản và phân loại chúng dựa trên các chủ đề hoặc danh mục được xác định trước. Nhiệm vụ phụ này đặc biệt hữu ích khi phân loại các từ đồng nghĩa và từ viết tắt.
- **Sentiment analysis**: Tác vụ này phát hiện tâm lý tích cực hoặc tiêu cực từ các nguồn dữ liệu bên trong hoặc bên ngoài, cho phép bạn theo dõi các thay đổi về thái độ của khách hàng theo thời gian. Nó thường được sử dụng để cung cấp thông tin về nhận thức về thương hiệu, sản phẩm và dịch vụ. Những hiểu biết sâu sắc này có thể thúc đẩy các doanh nghiệp kết nối với khách hàng và cải thiện quy trình cũng như trải nghiệm người dùng.

- **Information extraction**: Trích xuất thông tin (IE) hiển thị các phần dữ liệu có liên quan khi tìm kiếm các tài liệu khác nhau. Nó cũng tập trung vào việc trích xuất thông tin có cấu trúc từ văn bản tự do và lưu trữ các thực thể, thuộc tính và thông tin quan hệ này trong cơ sở dữ liệu. Các nhiệm vụ con khai thác thông tin phổ biến bao gồm:
- **Feature selection**: Lựa chọn tính năng hoặc lựa chọn thuộc tính là quá trình chọn các tính năng (thứ nguyên) quan trọng để đóng góp nhiều nhất cho đầu ra của mô hình phân tích dự đoán.
- **Feature extraction**: Trích xuất tính năng là quá trình chọn một tập hợp con các tính năng để cải thiện độ chính xác của nhiệm vụ phân loại. Điều này đặc biệt quan trọng đối với việc giảm kích thước.
- **Named-entity recognition (NER)**: Nhận dạng thực thể được đặt tên còn được gọi là nhận dạng thực thể hoặc trích xuất thực thể, nhằm mục đích tìm và phân loại các thực thể cụ thể trong văn bản, chẳng hạn như tên hoặc vị trí. Ví dụ: NER xác định "California" là một địa điểm và "Mary" là tên của một người phụ nữ.

c. Ứng dụng

Phần mềm phân tích văn bản đã tác động đến cách thức hoạt động của nhiều ngành công nghiệp, cho phép họ cải thiện trải nghiệm người dùng sản phẩm cũng như đưa ra các quyết định kinh doanh nhanh hơn và tốt hơn. Một số trường hợp sử dụng bao gồm:

- **Dịch vụ khách hàng**: Có nhiều cách khác nhau để các công ty thu thập phản hồi của khách hàng từ người dùng của mình. Khi được kết hợp với các công cụ phân tích văn bản, hệ thống phản hồi, chẳng hạn như chatbot, khảo sát khách hàng, NPS (điểm số của người quảng bá mạng), đánh giá trực tuyến, vé hỗ trợ và hồ sơ truyền thông xã hội, cho phép các công ty cải thiện trải nghiệm khách hàng của họ một cách nhanh chóng. Khai thác văn bản và phân tích tình cảm có thể cung cấp một cơ chế để các công ty ưu tiên các điểm yếu chính cho khách hàng của họ, cho phép

các doanh nghiệp phản hồi các vấn đề cấp bách trong thời gian thực và tăng sự hài lòng của khách hàng.

- **Quản lý rủi ro**: Khai thác văn bản cũng có các ứng dụng trong quản lý rủi ro, nơi nó có thể cung cấp thông tin chi tiết về xu hướng của ngành và thị trường tài chính bằng cách theo dõi sự thay đổi trong cảm xúc và bằng cách trích xuất thông tin từ báo cáo phân tích và báo cáo nghiên cứu chuyên sâu. Điều này đặc biệt có giá trị đối với các tổ chức ngân hàng vì dữ liệu này mang lại sự tự tin hơn khi xem xét các khoản đầu tư kinh doanh trên nhiều lĩnh vực khác nhau.
- **Bảo trì**: Khai thác văn bản cung cấp một bức tranh phong phú và đầy đủ về hoạt động và chức năng của sản phẩm và máy móc. Theo thời gian, khai thác văn bản tự động hóa việc ra quyết định bằng cách tiết lộ các mẫu tương quan với các vấn đề và quy trình bảo trì phòng ngừa và phản ứng. Phân tích văn bản giúp các chuyên gia bảo trì tìm ra nguyên nhân gốc rễ của các thách thức và lỗi nhanh hơn.
- **Chăm sóc sức khỏe**: Các kỹ thuật khai thác văn bản ngày càng có giá trị đối với các nhà nghiên cứu trong lĩnh vực y sinh, đặc biệt là để phân cụm thông tin. Điều tra thủ công các nghiên cứu y học có thể tốn kém và mất thời gian; khai thác văn bản cung cấp một phương pháp tự động hóa để trích xuất thông tin có giá trị từ tài liệu y khoa.
- **Lọc thư rác**: Thư rác thường đóng vai trò là điểm vào để tin tặc lây nhiễm phần mềm độc hại vào hệ thống máy tính. Khai thác văn bản có thể cung cấp một phương pháp để lọc và loại trừ những e-mail này khỏi hộp thư đến, cải thiện trải nghiệm người dùng tổng thể và giảm thiểu rủi ro tấn công mạng cho người dùng cuối.

2.2. Các Mô Hình Phân Tích Dữ Liệu

2.2.1. Mô Hình Logistic Regression

a. Khái niệm

Loại mô hình thống kê này (còn được gọi là mô hình logit) thường được sử dụng để phân loại và phân tích dự đoán. Hồi quy logistic ước tính xác suất xảy ra một sự kiện, chẳng hạn như đã bỏ phiếu hoặc không bỏ phiếu, dựa trên tập dữ liệu nhất định gồm các biến độc lập. Vì kết quả là một xác suất nên biến phụ thuộc có giới hạn từ 0 đến 1. Trong hồi quy logistic, phép biến đổi logit được áp dụng trên tỷ lệ cược—nghĩa là xác suất thành công chia cho xác suất thất bại. Điều này còn thường được gọi là tỷ lệ cược log hoặc logarit tự nhiên của tỷ lệ cược và hàm logistic này được biểu thị bằng các công thức sau:

$$logit(pi) = \frac{1}{1+e^{-pi}}$$

$$ln(\frac{pi}{1-pi}) = \beta_0 + \beta_1 \times x_1 + \dots + \beta_k \times x_k$$

Trong phương trình hồi quy logistic này, logit(pi) là biến phụ thuộc hoặc biến phản hồi và x là biến độc lập. Tham số beta hoặc hệ số trong mô hình này thường được ước tính

thông qua ước tính khả năng tối đa (MLE). Phương pháp này kiểm tra các giá trị khác nhau của beta thông qua nhiều lần lặp lại để tối ưu hóa cho phù hợp nhất với tỷ lệ chênh lệch nhật ký. Tất cả các lần lặp này tạo ra hàm khả năng ghi nhật ký và hồi quy logistic tìm cách tối đa hóa hàm này để tìm ước tính tham số tốt nhất. Sau khi tìm thấy hệ số tối ưu (hoặc các hệ số nếu có nhiều hơn một biến độc lập), xác suất có điều kiện cho từng quan sát có thể được tính toán, ghi lại và cộng lại với nhau để tạo ra xác suất dự đoán. Đối với phân loại nhị phân, xác suất nhỏ hơn 0,5 sẽ dự đoán 0 trong khi xác suất lớn hơn 0 sẽ dự đoán 1. Sau khi mô hình được tính toán, cách tốt nhất là đánh giá mức độ mô hình dự đoán biến phụ thuộc, được gọi là mức độ tốt của Phù hợp. Phép thử Hosmer–Lemeshow là một phương pháp phổ biến để đánh giá mức độ phù hợp của mô hình.

b. Giải thích hồi quy logistic

Tỷ lệ chênh lệch nhật ký có thể khó hiểu trong phân tích dữ liệu hồi quy logistic. Do đó, việc lũy thừa các ước tính beta là phổ biến để chuyển đổi kết quả thành tỷ lệ chênh lệch (Odds Ratio-OR), giúp dễ dàng giải thích kết quả. OR đại diện cho tỷ lệ cược mà một kết quả sẽ xảy ra với một sự kiện cụ thể, so với tỷ lệ cược của kết quả xảy ra khi không có sự kiện đó. Nếu OR lớn hơn 1, thì sự kiện đó có khả năng tạo ra một kết quả cụ thể cao hơn. Ngược lại, nếu OR nhỏ hơn 1, thì sự kiện đó có tỷ lệ xảy ra kết quả đó thấp hơn. Dựa trên phương trình ở trên, việc giải thích tỷ lệ chênh lệch có thể được biểu thị như sau: tỷ lệ cược thành công thay đổi theo $e^{c\beta_1}$ lần cho mỗi lần tăng đơn vị c trong x. Để sử dụng một ví dụ, giả sử chúng ta ước tính tỷ lệ sống sót trên tàu Titanic với điều kiện người đó là nam giới và tỷ lệ chênh lệch đối với nam giới là 0,0810. Có thể giải thích tỷ lệ chênh lệch là tỷ lệ sống sót của nam giới giảm theo hệ số 0,0810 so với nữ giới, giữ nguyên tất cả các biến số khác.

c. Các loại hồi quy logistic

Có ba loại mô hình hồi quy logistic, được xác định dựa trên phản ứng phân loại.

- Hồi quy logistic nhị phân (**Binary logistic regression**): Theo cách tiếp cận này, biến phản hồi hoặc biến phụ thuộc có bản chất phân đôi—tức là nó chỉ có hai kết quả có thể xảy ra (ví dụ: 0 hoặc 1). Một số ví dụ phổ biến về việc sử dụng nó bao gồm dự đoán xem một e-mail có phải là thư rác hay không phải thư rác hoặc khối u ác tính hay không ác tính. Trong hồi quy logistic, đây là cách tiếp cận được sử dụng phổ biến nhất và nói chung, nó là một trong những cách phân loại phổ biến nhất để phân loại nhị phân.
- Hồi quy logistic đa thức (**Multinomial logistic regression**): Trong loại mô hình hồi quy logistic này, biến phụ thuộc có ba hoặc nhiều kết quả có thể xảy ra; tuy nhiên, các giá trị này không có thứ tự cụ thể. Ví dụ: các hãng phim muốn dự đoán thể loại phim mà khán giả có khả năng xem để tiếp thị phim hiệu quả hơn. Mô

hình hồi quy logistic đa thức có thể giúp hãng phim xác định mức độ ảnh hưởng của tuổi tác, giới tính và tình trạng hẹn hò của một người đối với loại phim họ thích. Sau đó, hãng phim có thể định hướng chiến dịch quảng cáo của một bộ phim cụ thể tới một nhóm người có khả năng sẽ đi xem bộ phim đó.

- Hồi quy logistic thông thường (**Ordinal logistic regression**): Loại mô hình hồi quy logistic này được tận dụng khi biến phản hồi có ba hoặc nhiều hơn kết quả có thể xảy ra, nhưng trong trường hợp này, các giá trị này có một thứ tự xác định. Ví dụ về các câu trả lời theo thứ tự bao gồm thang điểm từ A đến F hoặc thang đánh giá từ 1 đến 5.

d. Logistic regression và machine learning

Trong học máy, hồi quy logistic thuộc họ các mô hình học máy có giám sát. Nó cũng được coi là một mô hình phân biệt đối xử, có nghĩa là nó cố gắng phân biệt giữa các lớp (hoặc danh mục). Không giống như thuật toán tổng quát, chẳng hạn như naïve bayes, nó không thể tạo ra thông tin của lớp mà nó đang cố gắng dự đoán (ví dụ: hình ảnh con mèo).

Hồi quy logistic cũng có thể dễ bị overfitting, đặc biệt khi có nhiều biến dự đoán trong mô hình. Chính quy hóa thường được sử dụng để xử phạt các tham số có hệ số lớn khi mô hình có số chiều cao.

e. Các trường hợp sử dụng hồi quy logistic

Hồi quy logistic thường được sử dụng cho các vấn đề dự đoán và phân loại. Một số trường hợp sử dụng này bao gồm:

- Phát hiện gian lận (**Fraud detection**): Các mô hình hồi quy logistic có thể giúp các nhóm xác định các điểm bất thường về dữ liệu, giúp dự đoán gian lận. Một số hành vi hoặc đặc điểm nhất định có thể có mối liên hệ cao hơn với các hoạt động gian lận, điều này đặc biệt hữu ích cho ngân hàng và các tổ chức tài chính khác trong việc bảo vệ khách hàng của họ. Các công ty dựa trên SaaS (Software as a Service) cũng đã bắt đầu áp dụng các phương pháp này để loại bỏ tài khoản người dùng giả mạo khỏi bộ dữ liệu của họ khi tiến hành phân tích dữ liệu về hiệu quả kinh doanh.
- Dự đoán bệnh (**Disease prediction**): Trong y học, phương pháp phân tích này có thể được sử dụng để dự đoán khả năng mắc bệnh hoặc bệnh tật cho một nhóm dân số nhất định. Các tổ chức chăm sóc sức khỏe có thể thiết lập dịch vụ chăm sóc phòng ngừa cho những cá nhân có xu hướng mắc các bệnh cụ thể cao hơn.
- Dự đoán về sự rời bỏ (**Churn prediction**): Các hành vi cụ thể có thể là biểu hiện của sự rời bỏ trong các chức năng khác nhau của một tổ chức. Ví dụ: các nhóm quản lý và nhân sự có thể muốn biết liệu có những người có thành tích cao trong công ty có nguy cơ rời khỏi tổ chức hay không; loại hiểu biết sâu sắc này có thể

thúc đẩy các cuộc trò chuyện để hiểu các vấn đề trong công ty, chẳng hạn như văn hóa hoặc lương thưởng. Ngoài ra, tổ chức bán hàng có thể muốn tìm hiểu xem khách hàng nào của họ có nguy cơ chuyển công việc kinh doanh của họ sang nơi khác. Điều này có thể nhắc các nhóm thiết lập chiến lược duy trì để tránh bị mất doanh thu.

2.2.2. Mô Hình Naive Bayes

a. Khái niệm

Các phương thức Naive Bayes là một tập hợp các thuật toán học có giám sát dựa trên việc áp dụng định lý Bayes với giả định "naive" về tính độc lập có điều kiện giữa mọi cặp tính năng được cung cấp giá trị của biến lớp. Định lý Bayes phát biểu mối quan hệ sau, biến lớp y đã cho và vectơ đặc trưng phụ thuộc x_1 thông qua x_n:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Sử dụng giả định độc lập có điều kiện naive rằng

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y),$$

đối với tất cả i , mối quan hệ này được đơn giản hóa thành

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Vì $P(x_1, \ldots, x_n)$ là hằng số với đầu vào, nên chúng ta có thể sử dụng quy tắc phân loại sau:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

và chúng ta có thể sử dụng ước tính Maximum A Posteriori (MAP) để ước tính P(y) và $P(x_i|y)$; cái trước là tần suất tương đối của lớp y trong training set.

Các bộ phân loại naive Bayes khác nhau chủ yếu khác nhau bởi các giả định mà chúng đưa ra liên quan đến việc phân phối $P(x_i|y)$.

Mặc dù các giả định dường như quá đơn giản của chúng, các bộ phân loại naive Bayes đã hoạt động khá tốt trong nhiều tình huống trong thế giới thực, nổi tiếng là phân loại tài liệu và lọc thư rác. Chúng yêu cầu một lượng nhỏ dữ liệu huấn luyện để ước tính các tham số cần thiết.

Người học và người phân loại Naive Bayes có thể cực kỳ nhanh so với các phương pháp phức tạp hơn. Việc tách rời các phân phối tính năng có điều kiện của lớp có nghĩa là mỗi phân phối có thể được ước tính độc lập dưới dạng phân phối một chiều. Điều này đến lượt nó giúp giảm bớt các vấn đề bắt nguồn từ lời nguyền về chiều.

Mặt khác, mặc dù naive Bayes được biết đến như một công cụ phân loại tốt, nhưng nó được biết đến là một công cụ ước tính tồi, vì vậy xác suất đầu ra từ predict_proba không được coi trọng quá.

b. Các loại Naive Bayes

- ***Gaussian naive Bayes***
- ***Complement naive Bayes***
- ***Bernoulli naive Bayes***
- ***Categorical naive Bayes***

CHƯƠNG 3. CÁC KẾT QUẢ THỰC NGHIỆM

3.1. Bộ Dữ Liệu

3.1.1. Tổng quan về bộ dữ liệu

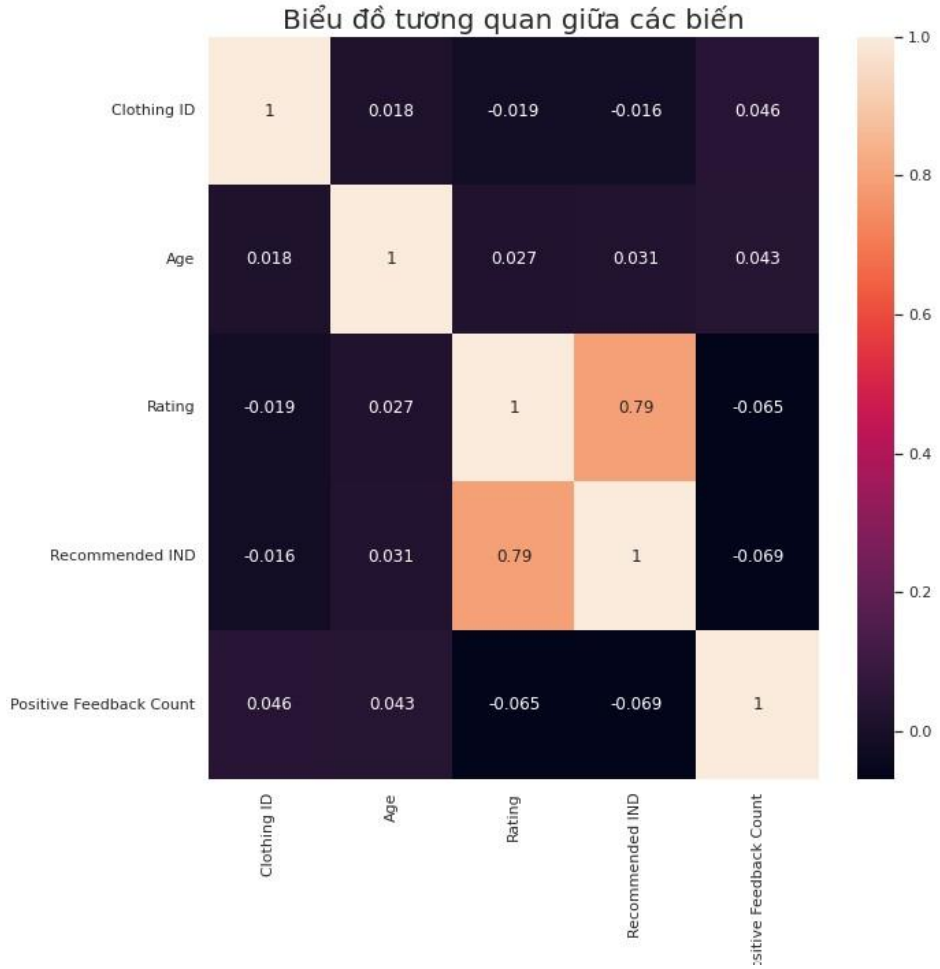
Dữ liệu gồm 10 biến và 23.486 dòng.

Giải thích các biến:

- Clothing ID: Mã sản phẩm
- Age: Tuổi của khách hàng
- Title: Tiêu đề bài review
- Review Text: Nội dung bài review
- Rating: Đánh giá (từ 1-5 sao)
- Recommended IND: Biến nhị phân cho biết sản phẩm có được recommend hay không (1 = recommended, 0 = unrecommended)
- Positive Feedback Count: Số lượng khách hàng nhận thấy đánh giá này là tích cực
- Division Name: Tên phân loại sản phẩm
- Department Name: Tên nhóm sản phẩm
- Class Name: Tên lớp sản phẩm

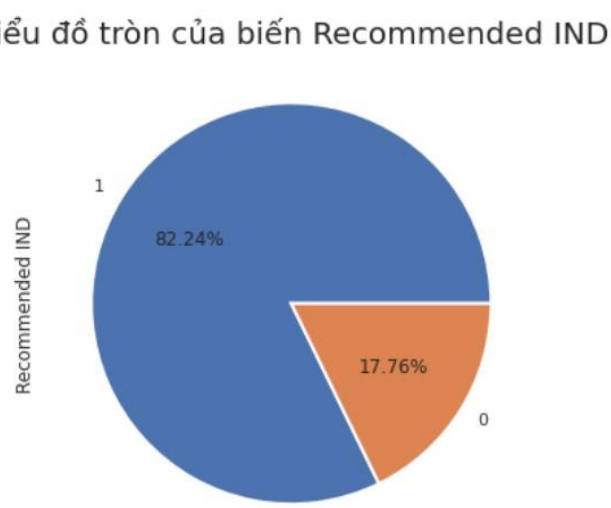
```
[42] print('Các cột hiện có của bộ dữ liệu:')
     for x in df.columns:
         print('>',x)
     print('\nSố dòng của bộ dữ liệu:',len(df))

     Các cột hiện có của bộ dữ liệu:
     > Clothing ID
     > Age
     > Title
     > Review Text
     > Rating
     > Recommended IND
     > Positive Feedback Count
     > Division Name
     > Department Name
     > Class Name

     Số dòng của bộ dữ liệu: 23486
```

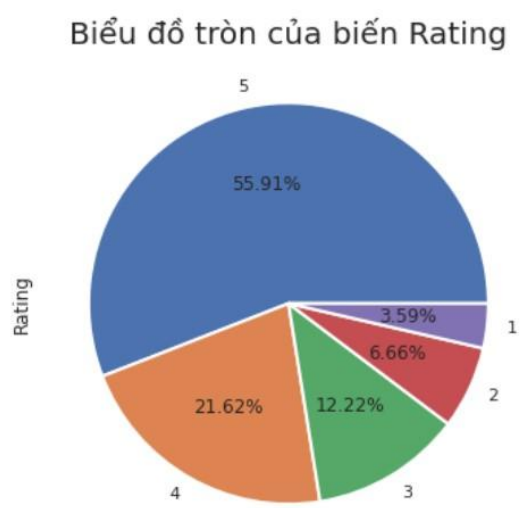
Biểu đồ tương quan giữa các biến định lượng cho thấy biến Recommend IND và Rating có độ tương quan cao với nhau (0.79).



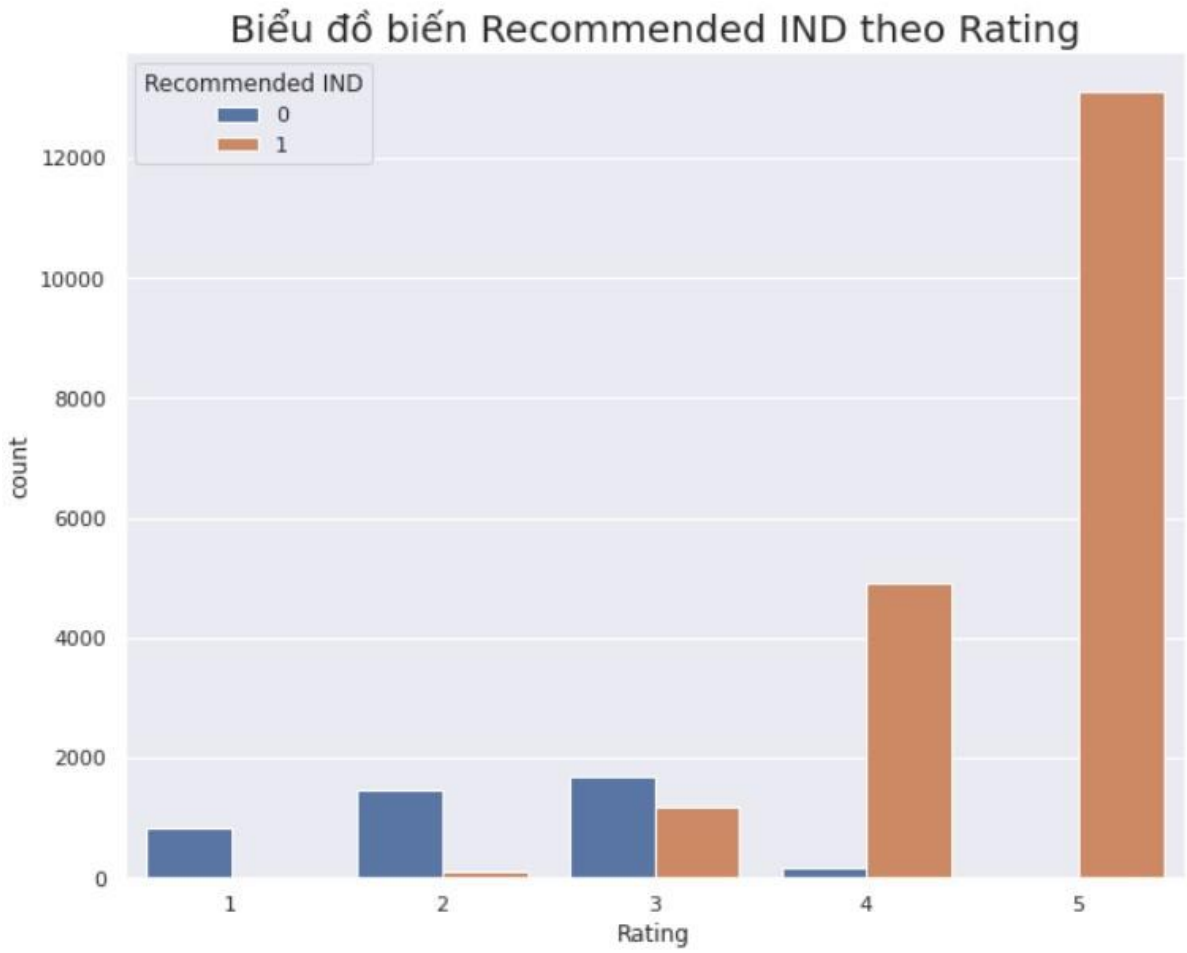
Bộ dữ liệu có 82.24% đánh giá tích cực (Recommended = 1), và 17.76% đánh giá tiêu cực (Recommended IND = 0)

Đồng thời có 55.91% đánh giá 5 sao, 21.62% đánh giá 4 sao, và 22.47% còn lại là đánh giá từ 1-3 sao.





Biểu đồ cột giữa biến Rating và Recommended IND cho thấy sự tương quan giữa 2 biến. Với Rating càng cao thì số lượng Recommended IND = 1 càng nhiều và ngược lại.

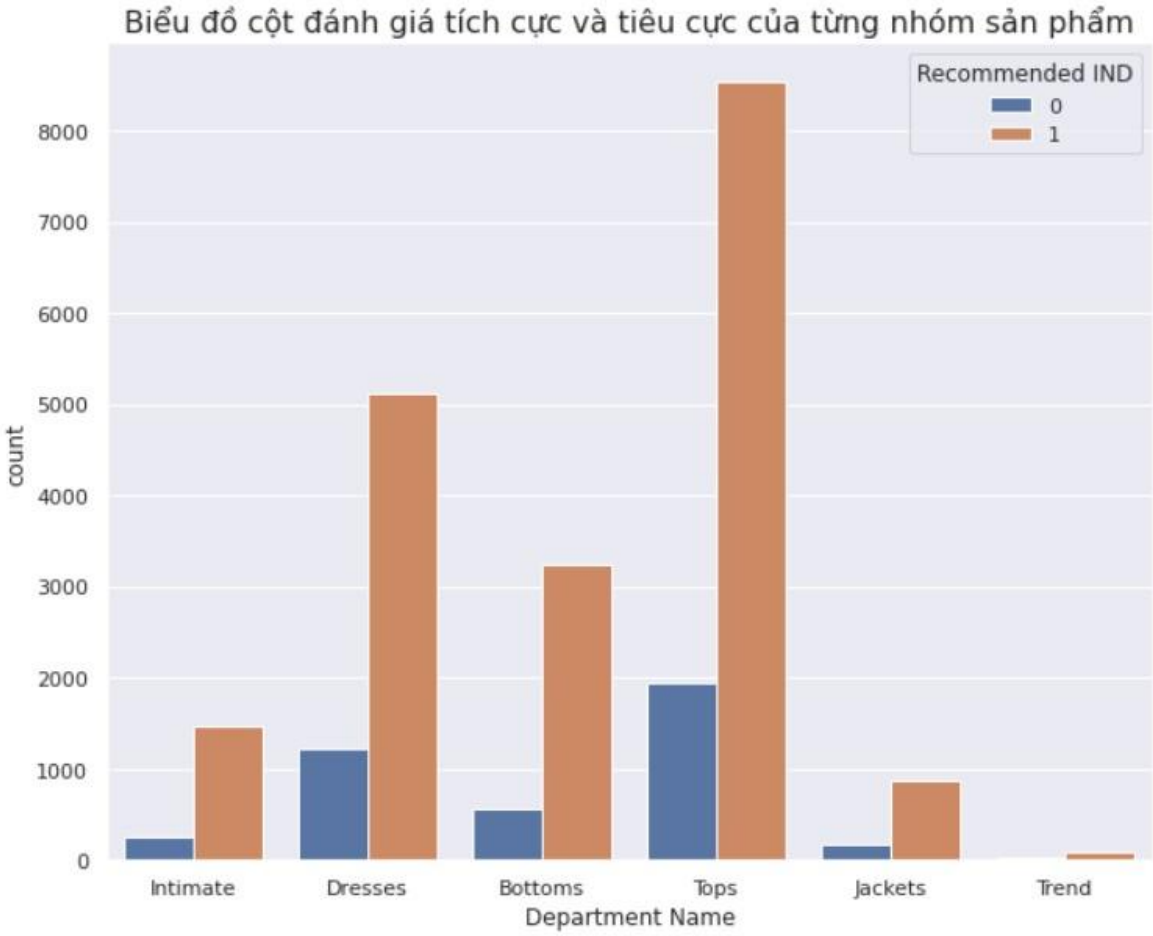


Sản phẩm được chia thành 5 nhóm chính theo biến Department Name. Trong đó áo (tops) có số lượng review nhiều nhất, tiếp đến là đầm (dresses) và quần (bottoms)

```
[125] df['Department Name'].value_counts()

Tops        10468
Dresses      6319
Bottoms      3799
Intimate     1735
Jackets      1032
Trend         119
Name: Department Name, dtype: int64
```

Trong đó nhóm sản phẩm áo nhận được nhiều đánh giá tích cực nhất và cũng đồng thời nhận nhiều đánh giá tiêu cực nhất.

Nhóm sản phẩm trend nhận được ít đánh giá nhất.



3.1.2. Tiền xử lý dữ liệu

Bộ dữ liệu tồn tại nhiều giá trị bị thiếu của biến Title, Review Text, Division Name, Department Name, Class Name. Vì missing values chiếm dưới 20% bộ dữ liệu nên ta sẽ drop các cột chứa giá trị bị thiếu.

```
[9] print('Số giá trị bị thiếu của các cột :\n', len(df) - df.count())

Số giá trị bị thiếu của các cột :
 Unnamed: 0                   0
Clothing ID                   0
Age                           0
Title                      3810
Review Text                 845
Rating                        0
Recommended IND               0
Positive Feedback Count       0
Division Name                14
Department Name              14
Class Name                   14
dtype: int64
```

```
[ ] # Xóa dữ liệu bị thiếu
    df.dropna(inplace= True)
    df.isna().sum()
```

```
Clothing ID                0
Age                        0
Title                      0
Review Text                0
Rating                     0
Recommended IND            0
Positive Feedback Count    0
Division Name              0
Department Name            0
Class Name                 0
dtype: int64
```

Số lượng review nhất nhất trên 1 sản phẩm là 871, và ít nhất là 1 review.

```
[11] # Đếm số lượng review trên từng sản phẩm (Clothing ID)
     review_count = df['Clothing ID'].value_counts()

     print('Số lượng review nhiều nhất trên 1 sản phẩm:', review_count.max())
     print('Số lượng review ít nhất trên 1 sản phẩm:', review_count.min())
```

```
Số lượng review nhiều nhất trên 1 sản phẩm: 871
Số lượng review ít nhất trên 1 sản phẩm: 1
```

Vì vậy ta sẽ lọc những sản phẩm có số lượng review quá ít (< 5)

```
✓ [12] # Chỉ giữ lại những sản phẩm có số lượng review > 5
       df = df.groupby('Clothing ID').filter(lambda x : x['Clothing ID'].shape[0]>5)
```

Yêu cầu đồ án là Phân tích đánh giá của khách hàng về sản phẩm (tích cực và tiêu cực). Vì vậy ta sẽ tạo dataframe mới 'text_df' để xử lý trên các biến: Recommended IND, Review Text, Title. Với biến Recommended IND là biến target phân loại.

```
[14] # Tạo dataframe 'text_df' gồm 3 biến Title, Review Text và Recommended IND
     text_df = df[['Title', 'Review Text', 'Recommended IND']]
     text_df.head()
```

	Title	Review Text	Recommended IND
2	Some major design flaws	I had such high hopes for this dress and reall...	0
3	My favorite buy!	I love, love, love this jumpsuit. it's fun, fl...	1
5	Not for the very petite	I love tracy reese dresses, but this one is no...	0
6	Cagrcoal shimmer fun	I aded this in my basket at hte last mintue to...	1
7	Shimmer, surprisingly goes with lots	I ordered this in carbon for store pick up, an...	1

Sau đó gộp biến Title (tiêu đề bài đánh giá) và Review Text (nội dung bài đánh giá) thành 1 biến mới là Review.

```
[15] # Tạo biến mới 'Review' được gộp từ Title và Review Text
     text_df['Review'] = text_df['Title'] + ' ' + text_df['Review Text']

     # Drop 2 biến Title và Review Text cũ
     text_df = text_df.drop(labels=['Title','Review Text'] , axis=1)
     text_df.head()
```

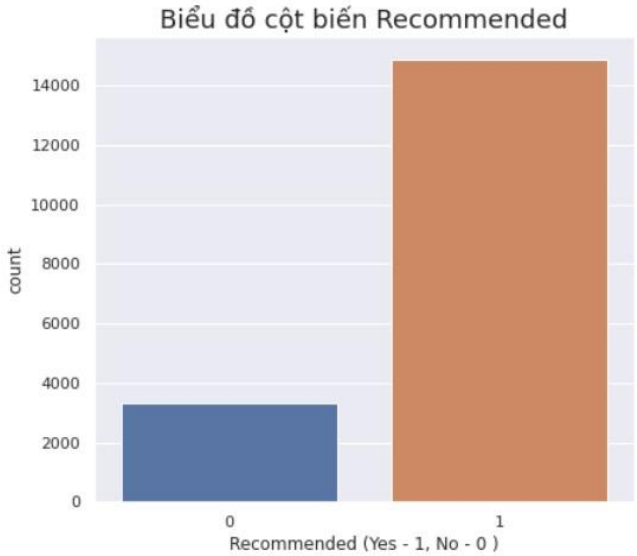
	Recommended IND	Review
2	0	Some major design flaws I had such high hopes ...
3	1	My favorite buy! I love, love, love this jumps...
5	0	Not for the very petite I love tracy reese dre...
6	1	Cagrcoal shimmer fun I aded this in my basket ...
7	1	Shimmer, surprisingly goes with lots I ordered...

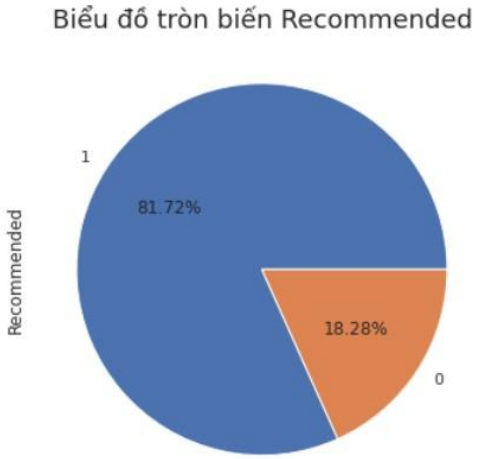
Đổi tên biến “Recommended IND” thành “Recommended” để tránh xảy ra nhầm lẫn với tên biến chứa khoảng trống.

```
[16] # Đổi tên biến 'Recommended IND' thành Recommended
     text_df = text_df.rename(columns={"Recommended IND": "Recommended"})
     text_df.head()
```

	Recommended	Review
2	0	Some major design flaws I had such high hopes ...
3	1	My favorite buy! I love, love, love this jumps...
5	0	Not for the very petite I love tracy reese dre...
6	1	Cagrcoal shimmer fun I aded this in my basket ...
7	1	Shimmer, surprisingly goes with lots I ordered...

Vì biến phân loại nhị phân Recommended có phân phối chênh lệch nhau khá lớn (với giá trị 1 chiếm 81.72% và 0 chiếm chỉ có 18.28%), dẫn đến hiện tượng mất cân bằng dữ liệu nghiêm trọng khi thực hiện các bài toán mô hình phân lớp, có thể gây ảnh hưởng đáng kể tới khả năng dự báo của mô hình. Khi đó độ chính xác có thể đạt được rất cao và có thể gây ngộ nhận về chất lượng của mô hình phân lớp (overfitting)





Vì vậy ta sẽ tiến hành boostrap resampling (lấy mẫu có hoàn lại) đối với mẫu thiểu số (ở đây là Recommended = 0) để giúp cho mô hình phân lớp tránh bị thiên lệch.

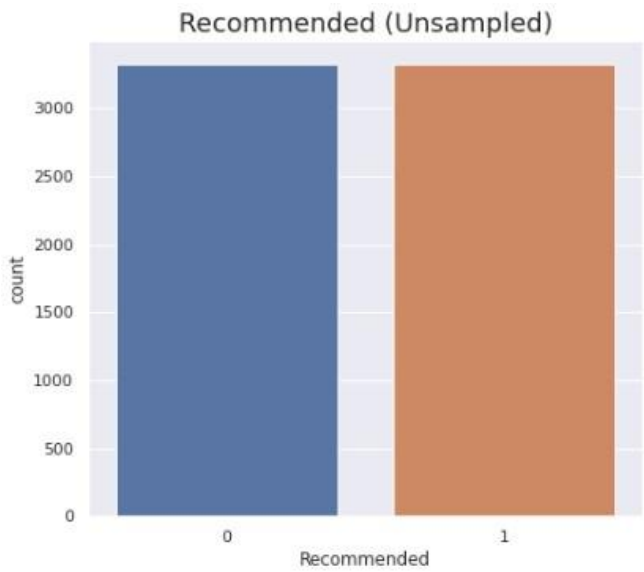
Đồng thời trong quá trình thử nghiệm, nhóm nhận thấy mô hình học máy sử dụng thư viện NLTK có tốc độ xử lý khá chậm (bộ dữ liệu lớn hơn 20.000 xử lý khoảng 10 phút). Nguyên nhân ở đây có thể vì NLTK không phải là một thư viện mạnh trong việc xử lý mô hình ML, hoặc có thể do cách phân chia train, test data theo dạng (feature, labels) của

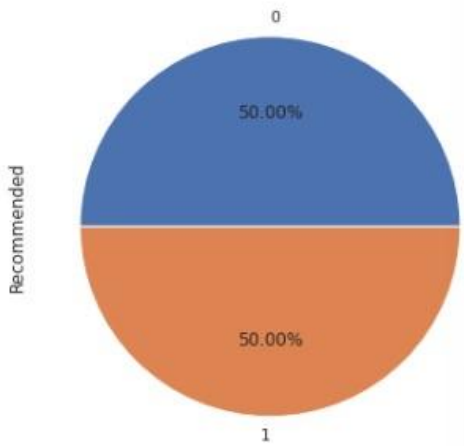
NLTK thay vì vector hóa văn bản bằng TF-IDF sklearn dẫn đến tốc độ xử lý không tối ưu.

Vì vậy để cải thiện tốc độ trong quá trình xử lý, nhóm sẽ tiến hành downsampling bộ dữ liệu xuống còn khoảng hơn 6600 dòng.

```
[20] # Tạo 2 dataframe ứng với Recommend = 1 và = 0
     df_majority = text_df[(text_df['Recommended']==1)]
     df_minority = text_df[(text_df['Recommended']==0)]

     # resample df_majority
     df_majority_resampled = resample(df_majority,
                                      replace=True,
                                      n_samples=len(df_minority), #downsampling
                                      random_state=40)

     # Kết hợp df minority và df majority đã resampled
     df_resampled = pd.concat([df_minority, df_majority_resampled])
```

Phân phối của bộ dữ liệu theo biến phân loại sau khi đã downsampling.





Hàm xử lý biến text đầu vào xóa các ký tự đặc biệt. số và dấu câu

```
[22] # Hàm xử lý biến text đầu vào
     def remove_pattern(input_txt, pattern):
         r = re.findall(pattern, input_txt)
         for word in r:
             input_txt = re.sub(word, "", input_txt)
         return input_txt
```

```
[23] df_resampled["Cleaned_Review"] = np.vectorize(remove_pattern)(df_resampled["Review"], "@[\w]*")
```

```
[24] # Xóa các ký tự đặc biệt, số và dấu chấm câu
     df_resampled["Cleaned_Review"] = df_resampled["Cleaned_Review"].str.replace("[^a-zA-Z#]", " ")
     df_resampled.head()
```

	Recommended	Review	Cleaned_Review
2	0	Some major design flaws I had such high hopes ...	Some major design flaws I had such high hopes ...
5	0	Not for the very petite I love tracy reese dre...	Not for the very petite I love tracy reese dre...
10	0	Dress looks like it's made of cheap material D...	Dress looks like it s made of cheap material D...
22	0	Not what it looks like First of all, this is n...	Not what it looks like First of all this is n...
26	0	Huge disappointment I have been waiting for th...	Huge disappointment I have been waiting for th...

Xóa các từ quá ngắn (từ có độ dài nhỏ hơn 3)

```
[25] # Xóa từ quá ngắn (từ có độ dài nhỏ hơn 3)
     df_resampled["Cleaned_Review"] = df_resampled["Cleaned_Review"].apply(lambda x: " ".join([w for w in x.split() if len(w)>3]))
     df_resampled.head()
```

	Recommended	Review	Cleaned_Review
2	0	Some major design flaws I had such high hopes ...	Some major design flaws such high hopes this d...
5	0	Not for the very petite I love tracy reese dre...	very petite love tracy reese dresses this very...
10	0	Dress looks like it's made of cheap material D...	Dress looks like made cheap material Dress run...
22	0	Not what it looks like First of all, this is n...	what looks like First this pullover styling th...
26	0	Huge disappointment I have been waiting for th...	Huge disappointment have been waiting this swe...

3.1.3. Text Mining

Sau khi xử lý biến text đầu vào, nhóm tiến hành khai thác văn bản (Text Mining). Đầu tiên tạo hàm "cleaning" với bộ dữ liệu data chứa các câu lệnh: tokenize (ngăn cách các từ trong văn bản bởi dấu phẩy), Remove Puncs (loại bỏ đi các dấu chấm câu), Removing Stopwords (loại bỏ các từ như "the", "a", "an", "in",...), lemma (tạo lại thành mệnh đề), cuối cùng joining (return lại dữ liệu biến text sau khi đã xử lý).

Tạo hàm thành công tiếp tục apply dữ liệu vào biến "Cleaned_Review" của data "df_resampled" thông qua hàm "cleaning".

```python
def cleaning(data):

    #1. Tokenize
    text_tokens = word_tokenize(data.replace("'", "").lower())

    #2. Remove Puncs
    tokens_without_punc = [w for w in text_tokens if w.isalpha()]

    #3. Removing Stopwords
    tokens_without_sw = [t for t in tokens_without_punc if t not in stop_words]

    #4. lemma
    text_cleaned = [WordNetLemmatizer().lemmatize(t) for t in tokens_without_sw]

    #joining
    return " ".join(text_cleaned)
```

```python
[29] df_resampled["Cleaned_Review"] = df_resampled["Cleaned_Review"].apply(cleaning)
     df_resampled["Cleaned_Review"].head()
```

```
2     major design flaw high hope dress really wante...
5     petite love tracy reese dress petite foot tall...
10    dress look like made cheap material dress run ...
22    look like first pullover styling side zipper p...
26    huge disappointment waiting sweater coat ship ...
Name: Cleaned_Review, dtype: object
```

Dùng lệnh *split().value_counts()*: Liệt kê số lượng các từ trong "Cleaned_Review" sau khi được cắt ra

```python
[30] rare_words = pd.Series(" ".join(df_resampled["Cleaned_Review"]).split()).value_counts()
     rare_words
```

```
dress      4117
love       2745
size       2708
like       2622
look       2457
           ...
plainer       1
groom         1
poetic        1
row           1
sotra         1
Length: 7065, dtype: int64
```

* Kết quả nhận được ta thấy có:

4117 từ dress xuất hiện, 2745 từ love xuất hiện, 2708 từ size xuất hiện, 2622 từ like xuất hiện, …

Hiển thị các từ có độ dài <= 2

```
[31] rare_words = rare_words[rare_words <= 2]
```

```
[32] rare_words.index

Index(['coal', 'solves', 'prefered', 'simultaneously', 'widen', 'shee',
       'caved', 'rustic', 'timely', 'community',
       ...
       'housewife', 'xxxxxs', 'exists', 'leap', 'streamline', 'plainer',
       'groom', 'poetic', 'row', 'sotra'],
      dtype='object', length=3798)
```

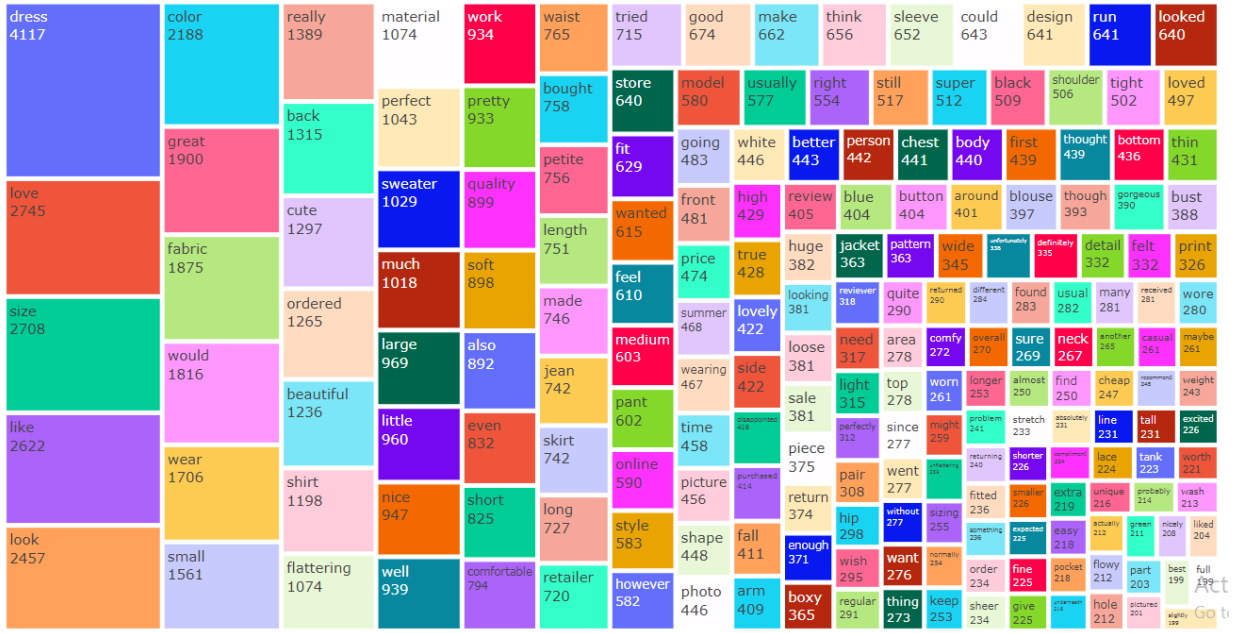
3.1.4. WordCloud

Đối với phần WordCloud, nhóm tiến hành vẽ các biểu đồ thể hiện tần số xuất hiện của các từ ngữ trong “Cleaned_Review”.

Vẽ biểu đồ thể hiện top 200 từ có tần số xuất hiện nhiều nhất sau khi Cleaning

```
import plotly.express as px
FreqOfWords = df_resampled["Cleaned_Review"].str.split(expand=True).stack().value_counts()
FreqOfWords_top200 = FreqOfWords[:200]

fig = px.treemap(FreqOfWords_top200, path=[FreqOfWords_top200.index], values=0, width=1200, height=700)
fig.update_layout(title_text='Top 200 từ xuất hiện nhiều trong Dataset (After Cleaning)',
                  title_x=0.5, title_font=dict(size=20)
                  )
fig.update_traces(textinfo="label+value")
fig.show()
```

Top 200 từ xuất hiện nhiều trong Dataset (After Cleaning)



Kết quả nhận được: Từ “dress” có tần số xuất hiện nhiều nhất là 4117 và có 3 từ có tần số thấp nhất trong top 200 này là: “best”, “full”, “slightly”.

Để vẽ được WordCloud ta cần import thư viện WordCloud trong NLTK.

WordCloud của tất cả đánh giá của khách hàng

Đầu tiên tạo 1 biến “all_word” chứa list các từ được cắt nhỏ ra từ “df_resampled[“Cleaned_Review”]. Sau đó tiến hành vẽ biểu đồ theo thông số có sẵn.

```
[35] all_words = " ".join([sentence for sentence in df_resampled["Cleaned_Review"]])

     from wordcloud import WordCloud
     wordcloud = WordCloud(background_color='white', width = 800, height = 500, random_state = 42, max_font_size = 100).generate(all_words)

     plt.figure(figsize = (15, 8))
     plt.imshow(wordcloud, interpolation = "bilinear")
     plt.axis("off")
     plt.show()
```

WordCloud về những đánh giá tích cực của khách hàng

Tạo list “all_words” với các từ có “Recommended” = 1. Sau đó tiến hành vẽ biểu đồ như trên WordCloud của tất cả đánh giá của khách hàng.

```
[36] all_words = " ".join([sentence for sentence in df_resampled["Cleaned_Review"][df_resampled["Recommended"]==1]])

     wordcloud = WordCloud(background_color='white', width = 800, height = 500, random_state = 42, max_font_size = 100).generate(all_words)

     plt.figure(figsize = (15, 8))
     plt.imshow(wordcloud, interpolation = "bilinear")
     plt.axis("off")
     plt.show()
```

WordCloud về những đánh giá tiêu cực của khách hàng

Tạo list "all_words" với các từ có "Recommended" = 0. Sau đó tiến hành vẽ biểu đồ như trên WordCloud của tất cả đánh giá của khách hàng.

```
[37] all_words = " ".join([sentence for sentence in df_resampled["Cleaned_Review"][df_resampled["Recommended"]==0]])

     wordcloud = WordCloud(background_color='white', width = 800, height = 500, random_state = 42, max_font_size = 100).generate(all_words)

     plt.figure(figsize = (15, 8))
     plt.imshow(wordcloud, interpolation = "bilinear")
     plt.axis("off")
     plt.show()
```

3.2. Phân Chia Dữ Liệu

Trước khi chạy 2 mô hình phân lớp, nhóm sẽ tiến hành phân bộ dữ liệu thành train và test set độc lập với nhau

Training set sẽ dùng để huấn luyện mô hình, và testing set sẽ dùng để kiểm thử, đánh giá độ chính xác của mô hình phân lớp đã được huấn luyện.

Đầu tiên ta sẽ drop biến Review và giữ lại 2 biến Recommended và Cleaned_Review.

```
[100] df_cleaned_resampled = df_resampled.drop(['Review'], axis = 1)
```

```
df_cleaned_resampled
```

	Recommended	Cleaned_Review
2	0	major design flaw high hope dress really wante...
5	0	petite love tracy reese dress petite foot tall...
10	0	dress look like made cheap material dress run ...
22	0	look like first pullover styling side zipper p...
26	0	huge disappointment waiting sweater coat ship ...
...	...	...
5499	1	dress dress closet ordered little short taste ...
9986	1	simple cute dress still look good wear legging...
9652	1	amazing work pant bought pant green black comf...
4856	1	close like another reviewer really wanted love...
14895	1	tank nice quality beautiful color nice bought ...

6656 rows × 2 columns

Train và test set ở đây được lấy với tỷ lệ 90/10 (test_size = 0.1)

Từ train và test set, ta sẽ tiến hành phân tách mỗi set thành 2 mảng theo positive và negative.

Với phần tử có Recommended = 1 ta sẽ append vào mảng positive (train_pos, test_pos)

Với phần tử có Recommended = 0 ta sẽ append vào mảng negative (train_neg, test_neg)

Đồng thời ta sẽ tạo một mảng reviews có cấu trúc (words, recommended).

Lý do nhóm thực hiện phân chia dữ liệu như trên vì mô hình phân lớp của thư viện NLTK xử lý và nhận giá trị theo cấu trúc list (featureset, labels).

Ngoài ra nếu chạy mô hình bằng thư viện Sklearn ta có thể vectơ hóa dữ liệu bằng IF-IDF, hoặc Bag of Words để thực hiện phân tách dữ liệu, hoặc tokenize texts_to_sequences bằng Keras.

```
[103] train, test = train_test_split(df_cleaned_resampled,test_size = 0.1)
```

```
[104] # Tạo 2 mảng tách train sets theo positive, negative
      train_pos = []
      train_neg = []

      for index, row in train.iterrows():
          if row['Recommended'] == 1:
              train_pos.append(row)
          elif row['Recommended'] == 0:
              train_neg.append(row)

      # Tạo mảng review
      reviews = []
      for index, row in train.iterrows():
          words_filtered = [e.lower() for e in row.Cleaned_Review.split() if len(e) >= 3]
          reviews.append((words_filtered,row.Recommended))

      # Tạo 2 mảng tách test sets theo positive, negative
      test_pos = []
      test_neg = []

      for index, row in test.iterrows():
          if row['Recommended'] == 1:
              test_pos.append(row)
          elif row['Recommended'] == 0:
              test_neg.append(row)
```

Sau đó nhóm sẽ định nghĩa hàm truy xuất features gồm hàm lấy words trong mảng reviews (get_words_in_review), hàm lấy feature của words (get_words_features), và hàm truy xuất features (extract_features) để chuẩn bị cho quá trình huấn luyện mô hình.

```
[77] def get_words_in_reviews(reviews):
         all = []
         for (words, recommended) in reviews:
             all.extend(words)
         return all

     def get_words_features(wordlist):
         wordlist = nltk.FreqDist(wordlist)
         features = wordlist.keys()
         return features

     w_features = get_words_features(get_words_in_reviews(reviews))

     def extract_features(document):
         document_words = set(document)
         features = {}
         for word in w_features:
             features['containts(%s)' % word] = (word in document_words)
         return features
```

3.3. Huấn Luyện Dữ Liệu với Naive Bayes và Logistic Regression

3.3.1. Naive Bayes

Tạo bộ dữ liệu training mới từ mảng reviews và hàm extract_features ở trên để phù hợp với mô hình huấn luyện.

Tiếp theo fit vào mô hình naive Bayes để tiến hành huấn luyện mô hình.

Input:

```
[ ] # Huấn luyện mô hình Naive Bayes
    training_set = nltk.classify.apply_features(extract_features,reviews)
    naivebayesclassifier = nltk.NaiveBayesClassifier.train(training_set)
```

Bắt đầu lấy mỗi review ở bộ mỗi dữ liệu test, tích cực và tiêu cực, đưa vào mô hình để tiến hành dự đoán liệu review đó là tích cực hay là tiêu cực và có đúng với thực tế hay không.

Input:

```
[ ] # Đánh giá độ chính xác của mô hình training so với test sets
    neg_cnt = 0
    pos_cnt = 0
    for obj in test_neg:
        res =  naivebayesclassifier.classify(extract_features(obj.Cleaned_Review.split()))
        if(res == 0):
            neg_cnt = neg_cnt + 1
    for obj in test_pos:
        res =  naivebayesclassifier.classify(extract_features(obj.Cleaned_Review.split()))
        if(res == 1):
            pos_cnt = pos_cnt + 1

    print('[Negative]: %s/%s '  % (len(test_neg),neg_cnt))
    print('[Positive]: %s/%s '  % (len(test_pos),pos_cnt))
```

Mô hình phân lớp naive Bayes có độ chính xác khoảng 91% cho các đánh giá negative, và khoản 85% cho các đánh giá positive.

Output:

```
[Negative]: 338/309
[Positive]: 328/279
```

Kiểm tra xem năm từ có tỷ lệ chênh lệch giữa tích cực và tiêu cực hoặc giữa tiêu cực và tích cực nào là cao nhất

Input:

```
# Hàm trả về most_informative_features
naivebayesclassifier.show_most_informative_features(5)
```

Kết quả cho thấy với từ “shame” và “horrible” nó có tỷ lệ xuất hiện trong review tiêu cực (0) gấp 26.4 lần so với trong review tích cực (1)

Từ “dressed” có tỷ lệ xuất hiện trong review tích cực (1) gấp 24 lần so với trong review tiêu cực (0)

Output:

```
Most Informative Features
     containts(horrible) = True              0 : 1      =     26.4 : 1.0
        containts(shame) = True              0 : 1      =     26.4 : 1.0
      containts(dressed) = True              1 : 0      =     24.0 : 1.0
containts(disappointment) = True              0 : 1      =     19.5 : 1.0
         containts(poor) = True              0 : 1      =     16.6 : 1.0
```

Tạo hàm show_most_informative_features_in_list() để đánh giá một từ là tích cực hoặc là tiêu cực có tỷ lệ cao nhất.

Input:

```
# Chỉnh sửa lại hàm của nltk để hiển thị kết quả trực quan hơn
def show_most_informative_features_in_list(classifier, n=10):
    cpdist = classifier._feature_probdist       # probability distribution for feature values given labels
    feature_list = []
    for (fname, fval) in classifier.most_informative_features(n):
        def labelprob(l):
            return cpdist[l, fname].prob(fval)
        labels = sorted([l for l in classifier._labels if fval in cpdist[l, fname].samples()],
                        key=labelprob)
        feature_list.append([fname, labels[-1]])
    return feature_list
```

Kiểm tra 10 từ có tỷ lệ cao nhất

Input:

```
show_most_informative_features_in_list(naivebayesclassifier, 10)
```

Output:

```
[['containts(horrible)', 0],
 ['containts(shame)', 0],
 ['containts(dressed)', 1],
 ['containts(disappointment)', 0],
 ['containts(poor)', 0],
 ['containts(cooler)', 1],
 ['containts(ripped)', 0],
 ['containts(flirty)', 1],
 ['containts(everyday)', 1],
 ['containts(unflattering)', 0]]
```

3.3.2. Logistic Regression

Tạo bộ dữ liệu training mới từ mảng reviews và hàm extract_features ở trên để phù hợp với mô hình huấn luyện.

Tiếp theo fit vào mô hình Logistic Regression để tiến hành huấn luyện mô hình.

Input:

```
# Huấn luyện mô hình Logistic Regression
training_set = nltk.classify.apply_features(extract_features,reviews)
logregressionclassifier = SklearnClassifier(LogisticRegression()).train(training_set)
```

Bắt đầu lấy mỗi review ở bộ mỗi dữ liệu test, tích cực và tiêu cực, đưa vào mô hình để tiến hành dự đoán liệu review đó là tích cực hay là tiêu cực và có đúng với thực tế hay không.

Input:

```
[ ] # Đánh giá độ chính xác của mô hình training so với test sets
    neg_cnt = 0
    pos_cnt = 0
    for obj in test_neg:
        res =  logregressionclassifier.classify(extract_features(obj.Cleaned_Review.split()))
        if(res == 0):
            neg_cnt = neg_cnt + 1
    for obj in test_pos:
        res =  logregressionclassifier.classify(extract_features(obj.Cleaned_Review.split()))
        if(res == 1):
            pos_cnt = pos_cnt + 1

    print('[Negative]: %s/%s '  % (len(test_neg),neg_cnt))
    print('[Positive]: %s/%s '  % (len(test_pos),pos_cnt))
```

Mô hình phân lớp Logistic Regression có độ chính xác khoảng 88% cho các đánh giá negative, và khoảng 87% cho các đánh giá positive.

Output:

```
[Negative]: 338/296
[Positive]: 328/286
```

3.4. Các Kết Quả Thực Nghiệm

3.4.1. Naive Bayes

Đánh giá độ chính xác

- Accuracy: ~88%
- Precision: ~85%
- Recall: ~91%

- F-score: ~88%

Input:

```
Accuracy  = (neg_cnt + pos_cnt)/(len(test_neg)+ len(test_pos)) * 100
Precision = pos_cnt/len(test_pos) * 100
Recall = pos_cnt/(len(test_neg) - neg_cnt + pos_cnt) * 100
F_score = (2 * Precision  * Re Loading... cision + Recall)
print('[Accuracy]: %s '  % Accuracy)
print('[Precision]: %s '  % Precision)
print('[Recall]: %s '  % Recall)
print('[F-score]: %s '  % F_score)
```

Output:

```
[Accuracy]: 88.28828828828829
[Precision]: 85.0609756097561
[Recall]: 90.5844155844156
[F-score]: 87.7358490566038
```

3.4.2 Logistic Regression

Đánh giá độ chính xác

- Accuracy: ~87%
- Precision: ~87%
- Recall: ~87%
- F-score: ~87%

Input:

```
Accuracy  = (neg_cnt + pos_cnt)/(len(test_neg)+ len(test_pos)) * 100
Precision = pos_cnt/len(test_pos) * 100
Recall = pos_cnt/(len(test_neg) - neg_cnt + pos_cnt) * 100
F_score = (2 * Precision  * Recall)/(Precision + Recall)
print('[Accuracy]: %s '  % Accuracy)
print('[Precision]: %s '  % Precision)
print('[Recall]: %s '  % Recall)
print('[F-score]: %s '  % F_score)
```

Output:

```
[Accuracy]: 87.38738738738738
[Precision]: 87.1951219512195
[Recall]: 87.1951219512195
[F-score]: 87.1951219512195
```

3.5. Phân Tích và Đánh Giá

Nhận xét:

- Có 3 giá trị độ chính xác của mô hình naive Bayes đều cao hơn của mô hình Logistic Regression, ngoại trừ Precision.
- Mô hình naive Bayes có sự biến động của 4 giá trị độ chính xác cao hơn so với của mô hình Logistic Regression

- Cả 4 giá trị độ chính của mô hình Logistic Regression đều là khoảng 87%
- Trong 4 giá trị độ chính xác của mô hình naive Bayes thì Recall có giá trị cao nhất (khoảng 91%)

Đánh giá:

Mặc dù mô hình Logistic Regression có độ tương đồng cao về độ chính xác, nhưng đối với bộ dữ liệu và bối cảnh này thì nên sử dụng mô hình naive Bayes (có giá trị độ tin cậy cao hơn) để dự đoán quan điểm, tích cực và tiêu cực, của khách hàng về chất lượng sản phẩm và dịch vụ.

CHƯƠNG 4. KẾT LUẬN

4.1. Các Kết Quả Đạt Được

Thông qua quá trình tìm hiểu và nghiên cứu, nhóm đã xây dựng được mô hình phân lớp phân tích phản hồi của khách hàng về sản phẩm (tích cực, tiêu cực) bằng mô hình Naive Bayes và Logistic Regression sử dụng thư viện NLTK, và kết quả 2 mô hình đạt độ chính xác tương đối cao.

Đồng thời nhóm đã ứng dụng những kiến thức đã học về Xử lý ngôn ngữ tự nhiên để thực hiện Text Mining (khai thác và xử lý dữ liệu văn bản)

4.2. Những Hạn Chế và Hướng Phát Triển

4.2.1. Hạn chế

Nhóm chưa khai thác và sử dụng được hết các biến trong bộ dữ liệu như các thông tin về sản phẩm, tên sản phẩm, thông tin khách hàng,... trong việc phân loại phản hồi của khách hàng theo từng nhóm sản phẩm, từng nhóm khách hàng khác nhau để có thể phân tích nguyên nhân của các đánh giá tiêu cực đồng thời đưa ra hướng cải thiện sản phẩm.

Vì không có quá nhiều thời gian nghiên cứu nên bộ dữ liệu đầu vào mà nhóm sử dụng là bộ dữ liệu có sẵn trên Kaggle mà chưa thể khai thác và thu thập dữ liệu đầu vào theo thời gian thực từ các trang web, các sàn thương mại điện tử.

4.2.2. Hướng phát triển

Từ việc phân tích phản hồi của khách hàng theo biến phân loại nhị phân (tích cực và tiêu cực), nhóm có thể phát triển xây dựng mô hình học máy phân loại đa lớp (với biến phân loại Rating 1-5 sao hoặc theo nhiều loại cảm xúc khác nhau: giận dữ, hạnh phúc, buồn bã, bất ngờ,...)

Ngoài ra có thể bổ sung thêm thông tin về khách hàng, sản phẩm để xây dựng mô hình phân lớp đánh giá phản hồi của khách hàng theo từng nhóm khách hàng, hoặc theo từng loại sản phẩm khác nhau.

Việc xử lý ngôn ngữ tự nhiên hiện tại nhóm chỉ phân tích trên ngôn ngữ là tiếng Anh. Trong tương lai có thể phân tích nhận diện cảm xúc của khách hàng bằng tiếng Việt (tham khảo và áp dụng các mô hình trong các dự án của nhóm Vietnamese NLP Research Group - Under The Sea)

Trong quá trình thu thập phản hồi khách hàng có thể lọc ra những phản hồi không phù hợp (toxic comment classification) để dự đoán liệu phản hồi của khách hàng có thật sự phù hợp hay không.

PHỤ LỤC I (Link Github)

Link github mã nguồn:
https://github.com/KhanhHa1109/Project_CustomeReview_SentimentAnalysis

PHỤ LỤC II

Bảng phân công công việc	
Nguyễn Quỳnh Khánh Hà	- EDA, tiền xử lý dữ liệu, train-test split - Báo cáo chương 3, chương 4
Huỳnh Trần Anh Thy	- Text Mining, WordCloud - Báo cáo chương 1, chương 3
Đồng Đan Hoài	- Huấn luyện, đánh giá mô hình phân lớp - Báo cáo chương 2, chương 3

TÀI LIỆU THAM KHẢO

1. Slide bài giảng môn Xử lý ngôn ngữ tự nhiên
2. https://www.nltk.org/book/ch06.html
3. https://github.com/nltk/nltk
4. https://www.nltk.org/api/nltk.classify.html
5. https://www.nltk.org/api/nltk.classify.naivebayes.html
6. https://www.nltk.org/api/nltk.classify.scikitlearn.html
7. https://www.kaggle.com/code/kadirduran/nlp-sentiment-classification-with-ml-and-dl-models
8. https://www.kaggle.com/code/kartikeyat/preprocessing-naive-bayes-evaluation#Train-2-Models
9. ibm.com/cloud/learn/text-mining
10. https://www.ibm.com/topics/logistic-regression