BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC KINH TẾ TP.HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH

UEH
UNIVERSITY

ĐỒ ÁN MÔN HỌC

XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Đề tài:

Dự đoán đánh giá sản phẩm điện thoại di động dựa trên bình luận của khách hàng trên các sàn thương mại điện tử

Thành viên:

Huỳnh Trịnh Tiến Khoa	31211027645
Trương Thanh Phong	31211027662
Lê Trần Khánh Phú	31211024087

Giảng viên: TS. Đặng Ngọc Hoàng Thành

Thành phố Hồ Chí Minh , ngày 21 tháng 12 năm 2023

Mục Lục

LỜI NÓI ĐẦU

Trong thời đại công nghệ hiện đại, việc mua sắm smartphone trực tuyến ngày càng trở nên phổ biến, và người dùng không chỉ là những khách hàng đơn thuần mà còn là những người chia sẻ trải nghiệm và đánh giá chi tiết về sản phẩm mà họ sở hữu. Sàn thương mại điện tử trở thành không gian mà người tiêu dùng thường xuyên thảo luận, so sánh, và đưa ra nhận xét về những chiếc điện thoại thông minh mà họ quan tâm. Đối với các nhà sản xuất và những người kinh doanh trong lĩnh vực di động, việc hiểu rõ ý kiến của khách hàng về sản phẩm của mình là quan trọng để nắm bắt xu hướng thị trường và nâng cao chất lượng sản phẩm. Đồng thời, đánh giá từ người dùng cũng đóng vai trò quyết định lớn đến quyết định mua sắm của những người tiêu dùng tiếp theo.

Đề tài nghiên cứu này nhấn mạnh vào việc dự đoán lượt đánh giá của người dùng về smartphone từ những bình luận trên sàn thương mại điện tử. Sự kết hợp giữa bộ dữ liệu UIT-ViSFD về phản hồi đối sản phẩm điện thoại thông minh của người Việt và machine learning; deep learning sẽ giúp tạo ra một mô hình dự đoán mạnh mẽ và chính xác.

Nhóm chúng em tin rằng nghiên cứu này không chỉ mang lại giá trị trong việc phân tích ý kiến về smartphone, mà còn giúp doanh nghiệp hiểu rõ hơn về nhu cầu và mong muốn của khách hàng. Sự hiểu biết sâu sắc này có thể giúp họ tối ưu hóa chiến lược marketing, phát triển sản phẩm, và tăng cường mối quan hệ với khách hàng. Với đặc trưng riêng biệt của lĩnh vực smartphone, đề tài này hy vọng mang lại những kiến thức hữu ích và ứng dụng thực tiễn, từ đó đóng góp vào sự phát triển toàn diện của ngành công nghiệp di động và cung cấp thông tin giá trị cho người tiêu dùng trong quá trình quyết định mua sắm của họ.

Do điều kiện về mặt thời gian và nhận thức còn nhiều hạn chế nên đồ án không khỏi có phần sai sót, kính mong quý giảng viên nhiệt tình góp ý để nhóm chúng em có những tiếp thu tốt hơn trong môn học này. Qua bài đồ án, nhóm chúng em xin cảm ơn giảng viên phụ trách môn Xử lý ngôn ngữ tự nhiên, TS. Đặng Ngọc Hoàng Thành đã truyền đạt những kiến thức quý báu cho học viên bằng cả tấm lòng nhiệt tình và sự tận tâm của thầy. Cảm ơn tác giả Nguyễn Văn Hiếu với danh sách teencode. Và chúng em rất cảm ơn nhóm tác giả đã cho chúng em khai thác và sử dụng bộ dữ liệu UIT-ViSD4SA cho đồ án lần này.

Chương I: Tổng Quan Đề Tài

1.1. Xác định đề tài

Đề tài của nhóm tập trung vào "Dự Đoán Đánh Giá Sản Phẩm Smartphone từ Bình Luận Người Dùng trên Sàn Thương Mại Điện Tử." Chúng em đã lựa chọn đề tài này dựa trên mong muốn nghiên cứu sâu rộng về cách người dùng tương tác và đánh giá về smartphone thông qua bình luận trên các sàn thương mại điện tử.

Thông qua bộ dữ liệu về phản hồi đối sản phẩm điện thoại thông minh của người Việt-UIT-ViSFD. Nhóm nhận thức rõ ràng về sự đa dạng trong danh sách bình luận người dùng về sản phẩm smartphone trên các sàn thương mại điện tử. Trong mỗi đoạn văn, từng từ ngữ và ý nghĩa của chúng đã trở thành những dấu hiệu quan trọng, tạo nên một kho dữ liệu đầy đủ về trải nghiệm và đánh giá từ cộng đồng người tiêu dùng. Nhóm đặt ra câu hỏi cụ thể về cách người dùng thể hiện ý kiến về smartphone, và liệu có thể dựa vào những từ ngữ này để dự đoán lượt đánh giá (số sao) một cách chính xác hay không. Điều này đồng thời phản ánh rõ ràng sự hướng dẫn của đề tài chọn lựa, hướng tới ứng dụng Học máy và Học sâu trong việc phân loại văn bản.

1.2. Mục tiêu nghiên cứu

Mục tiêu cuối cùng của nhóm là tạo ra một công cụ dự đoán đánh giá từ bộ dữ liệu UIT-ViSFD. Nhóm chúng em hy vọng rằng nghiên cứu này sẽ đóng góp vào việc hiểu sâu hơn về cách người dùng tương tác với sản phẩm trên sàn thương mại điện tử và cung cấp một cơ sở cho việc phát triển chiến lược kinh doanh dựa trên ý kiến của khách hàng.

1.3. Phương pháp nghiên cứu

- Các thuật toán: Logistics Regression cho Maxent Model, Naives Bayes cho Machine Learning và Long short term memory cho Deep Learning.
- Phương pháp biểu diễn văn bản: Term Frequency-Inverse Document Frequency.
- Các phương pháp cân bằng dữ liệu: Undersampling, RandomOverSampler.
- Các loại biểu đồ: sử dụng các loại biểu đồ phù hợp với mục đích trực quan hóa dữ liệu, đồng thời đẹp mắt, gọn gàng giúp người đọc dễ dàng quan sát, so sánh và đánh giá các số liệu được nhóm đưa ra trong bài báo cáo.

1.4. Tài nguyên sử dụng

- Ngôn ngữ lập trình: Python.
- Bộ dữ liệu về phản hồi đối với sản phẩm điện thoại thông minh của người Việt-UIT-ViSFD.
- Danh sách teencode được lấy từ github tác giả Nguyễn Văn Hiếu để hỗ trợ cho việc biên dịch các bình luận: https://gist.github.com/nguyenvanhieuvn/7d9441c10b3c2739499fc5a4d9ea06fb
- Danh sách stopword, nguồn: vnstopword.txt | Powered by Box

Chương II: Cơ sở lý thuyết

2.1. Các thuật toán được dùng

a. Maxent Model: Logistics Regression

Maxent Model, hay Maximum Entropy Model , thường được sử dụng trong bài toán phân loại, đặc biệt là trong học máy có giám sát. Maxent Model được xây dựng trên cơ sở của lý thuyết thông tin và định lý Bayes. Maxent Model: Logistic Regression là một phương pháp mạnh mẽ trong học máy, đặc biệt là khi chúng ta đối mặt với bài toán phân loại và muốn tối đa hóa sự chắc chắn thông tin trong mô hình của mình.

Thuật ngữ "Maximum Entropy" ám chỉ việc tối ưu hóa thông tin độ không chắc chắn (uncertainty) trong mô hình. Mô hình này giữ nguyên nguyên tắc ước tính log-tuyến tính, tức là cố gắng tìm ra ước tính ít sai lệch nhất có thể đối với thông tin đã cho. Trong ngữ cảnh của Maximum Entropy Classification, thông tin này thường là các đặc trưng (features) của dữ liệu.

Mô hình Logistic Regression là một dạng cụ thể của Maxent Model. Trong hồi quy logistic, chúng ta ánh xạ đầu ra của mô hình vào một phạm vi giữa 0 và 1, thường được hiểu như xác suất. Điều này làm cho nó phù hợp cho các bài toán phân loại nhị phân, nơi chúng ta quan tâm đến việc ước tính xác suất thuộc một lớp cụ thể.

Trong bài toán phân loại với Maxent Model: Logistic Regression, chúng ta ước tính xác suất của việc một quan sát thuộc vào từng lớp, sau đó chọn lớp có xác suất cao nhất làm dự đoán. Thuật ngữ "logistic" xuất phát từ hàm sigmoid logistic được sử dụng để chuyển đổi kết quả của hồi quy tuyến tính thành một phạm vi giữa 0 và 1. Một ví dụ điển hình là phân loại Email, gồm có email công việc, email gia đình, email spam, … Hay cụ thể hơn là ở bài toán này, phân loại đánh giá dựa trên bình luận của khách hàng về sản phẩm điện thoại di động trên sàn thương mại điện tử.

b. Machine Learning: Naive Bayes

Naive Bayes Classification (NBC) là một thuật toán phân loại dựa trên tính toán xác suất áp dụng định lý Bayes. Thuật toán này thuộc nhóm Supervised Learning (Học có giám sát). Có 2 mô hình thuật toán Naive Bayes thường sử dụng là: mô hình Bernoulli và mô hình Multinomial. Trong bài toán phân loại, Naive Bayes giả định tính độc lập giữa các đặc trưng, điều này có nghĩa là giá trị của mỗi đặc trưng không phụ thuộc vào giá trị của các đặc trưng khác khi đã biết nhãn lớp. Giả định này giúp đơn giản hóa tính toán và làm tăng tốc độ học của mô hình.

Trong bài toán này ta chỉ tìm hiểu mô hình Multinomial. Đặc trưng đầu vào ở đây chính là tần suất xuất hiện của từ trong văn bản đó. Tương tự như ý tưởng của nhóm là thực hiện tìm kiếm từ xuất hiện nhiều trong bình luận của khách hàng để phân loại ra bình luận đó có nội dung liên quan đến đánh giá bao nhiêu sao. Multinomial Naive Bayes được chọn vì nó phù hợp với bài toán phân loại nhiều lớp, thích hợp cho trường hợp đánh giá bình luận theo nhiều mức độ sao khác nhau. Mô hình này hoạt động dựa trên giả định rằng mỗi đặc trưng (từ trong văn bản) được mô tả bằng phân phối đa thức, và nó cực kỳ hiệu quả trong việc xử lý văn bản và dữ liệu đếm. Điều này làm cho Multinomial Naive Bayes là một lựa chọn tự nhiên cho bài toán của bạn.

c. Long short term memory

LSTM, hay Long Short-Term Memory, là một kiến trúc mạng nơ-ron hồi quy (RNN) được sử dụng trong học sâu. Nó được thiết kế để xử lý các chuỗi dữ liệu dài, nơi cần phải lưu trữ thông tin trong thời gian dài.

Trong so sánh với RNN thông thường, LSTM sử dụng cơ chế "cổng" để kiểm soát thông tin. Cụ thể, mỗi đơn vị LSTM bao gồm các thành phần quan trọng sau:

- Input Gate (Cổng Nhập): Xác định thông tin mới nào cần được thêm vào trạng thái ô nhớ.
- Forget Gate (Cổng Quên): Quyết định thông tin nào cần loại bỏ khỏi trạng thái ô nhớ.
- Output Gate (Cổng Xuất): Quyết định thông tin nào sẽ được đưa ra từ trạng thái ô nhớ ra bên ngoài.

Các cổng này giúp LSTM lựa chọn thông tin quan trọng để lưu giữ và quên đi thông tin không quan trọng, giải quyết vấn đề biến mất gradient. Các ứng dụng phổ biến của LSTM bao gồm xử lý ngôn ngữ tự nhiên, nhận dạng giọng nói, dự đoán chuỗi thời gian, và nhiều lĩnh vực khác. Nhờ vào khả năng của mình trong việc xử lý thông tin dài hạn, LSTM đã trở thành một công cụ mạnh mẽ trong nhiều ứng dụng học máy liên quan đến dữ liệu chuỗi và chuỗi thời gian.

2.2. Các phương pháp biểu diễn văn bản thành dạng vector

a. CountVectorizer

CountVectorizer là một phương pháp quan trọng trong xử lý ngôn ngữ tự nhiên (NLP) được sử dụng để biểu diễn văn bản dưới dạng dữ liệu có thể đưa vào mô hình máy học. Phương pháp này giúp chuyển đổi các đoạn văn thành vectơ số, nơi mỗi thành phần của vectơ tương ứng với số lần xuất hiện của từ trong văn bản. Điều này tạo ra một biểu diễn số học cho từng văn bản, thuận tiện cho việc sử dụng trong các mô hình máy học và các thuật toán khác. CountVectorizer hoạt động bằng cách xây dựng một "từ điển" từ toàn bộ tập văn bản và sau đó đếm số lần xuất hiện của từng từ trong từ điển trong từng văn bản cụ thể. Kết quả là một ma trận, trong đó mỗi hàng biểu diễn một văn bản và mỗi cột biểu diễn một từ trong từ điển, với giá trị ở mỗi ô là số lần xuất hiện của từ tương ứng trong văn bản đó. Và thường được sử dụng trong các tác vụ phân loại văn bản, phân tích ý kiến, hay bất kỳ nhiệm vụ nào yêu cầu biểu diễn số học của văn bản. Tuy nhiên, cần lưu ý rằng phương pháp này không giữ thông tin về thứ tự của từ trong câu và có thể không phản ánh mối quan hệ ngữ nghĩa giữa các từ.

b. TF-IDF Vectorizer

TF-IDF Vectorizer là một trong những kỹ thuật vector hóa văn bản phổ biến nhất trong lĩnh vực xử lý ngôn ngữ tự nhiên. Kỹ thuật này dựa trên hai thành phần là Term Frequency (TF) và Inverse Document Frequency (IDF) để tính toán trọng số cho mỗi từ trong văn bản.

Cụ thể, TF đại diện cho tần suất xuất hiện của một từ trong văn bản – từ càng xuất hiện nhiều thì càng quan trọng với ý nghĩa của văn bản. Trong khi đó, IDF đại diện cho

mức độ hiếm hoi của từ trong toàn bộ tập văn bản – từ càng hiếm thì IDF càng cao và từ đó càng quan trọng.

Kết hợp TF và IDF, trọng số TF-IDF thể hiện tầm quan trọng của một từ cụ thể trong một văn bản cụ thể, trong bối cảnh của toàn bộ tập văn bản, giúp tạo ra biểu diễn số học chất lượng cho mỗi văn bản. Tuy nhiên, nó không giữ thông tin về thứ tự từ và loại bỏ những từ không có trong từ điển. Thông qua đó mà TF-IDF Vectorizer trở thành một công cụ mạnh mẽ để biểu diễn và mã hóa văn bản dưới dạng vectơ số, phục vụ hiệu quả cho nhiều tác vụ xử lý ngôn ngữ tự nhiên.

2.3 Các phương pháp cân bằng dữ liệu

a) Undersampling

Undersampling là một chiến lược của phương pháp resample, nơi chúng ta giảm số lượng mẫu của lớp đa số để phù hợp với số lượng mẫu của lớp thiểu số. Mục tiêu của undersampling là giảm sự chệch giữa các lớp, giảm thiểu hiện tượng mất cân bằng trong dữ liệu. Tuy nhiên, mặc dù undersampling giúp cân bằng dữ liệu, nhược điểm của nó có thể dẫn đến mất mát thông tin quan trọng từ lớp đa số. Cụ thể, khi áp dụng undersampling, chúng ta loại bỏ ngẫu nhiên một số mẫu từ lớp đa số để giảm sự chệch giữa các lớp. Việc này có thể giúp mô hình học được mối quan hệ phức tạp và tăng khả năng tổng quát hóa trên tập dữ liệu mới. Tuy undersampling giảm mất cân bằng, nhưng cũng cần phải xem xét các tùy chọn khác như oversampling để đảm bảo rằng mô hình vẫn có đủ dữ liệu để học và không mất mát quá nhiều thông tin. Sự kết hợp linh hoạt giữa undersampling và oversampling có thể được sử dụng để tối ưu hóa cân bằng giữa chính xác và độ cân bằng trong mô hình

b) RandomOverSampler

RandomOverSampler là một phương pháp resampling trong môi trường máy học được sử dụng để giải quyết vấn đề mất cân bằng dữ liệu. Đặc biệt, nó thuộc loại "oversampling," nơi mục tiêu là tăng cường số lượng mẫu của lớp thiểu số bằng cách thêm các bản sao ngẫu nhiên từ lớp đó.

Trong quá trình oversampling, RandomOverSampler chọn mẫu ngẫu nhiên từ lớp thiểu số và thêm chúng vào tập dữ liệu để làm tăng kích thước của lớp đó. Việc lựa chọn ngẫu nhiên giúp đảm bảo sự đa dạng trong dữ liệu mới được tạo ra, tránh tình trạng quá mức lặp lại mẫu và tăng cường khả năng tổng quát hóa của mô hình. Quyết định sử dụng RandomOverSampler hay không phụ thuộc vào bối cảnh cụ thể của dữ liệu và vấn đề cụ thể mà ta đang giải quyết. Điều này giúp cải thiện hiệu suất của mô hình trên dữ liệu mất cân bằng bằng cách tạo thêm độ đa dạng và đồng đều giữa các lớp.

Chương III: Đề xuất mô hình triển khai

3.1. Các bước thực hiện

a. Đánh giá tổng quan bộ dữ liệu đầu vào.

- Loại bỏ những cột không cần thiết .
- Tính số lượng của mỗi đánh giá.

b. Khám phá dữ liệu (Data Exploration)

Hiểu trước các đặc điểm của dữ liệu sẽ cho phép ta xây dựng **mô hình phân lớp** tốt hơn (nghĩa là đạt được độ chính xác cao hơn). Qua khám phá dữ liệu sơ bộ, ta thấy dữ liệu có đặc điểm mất cân bằng về phân phối các lớp đánh giá. Bên cạnh đó là sự xuất hiện của các từ thừa hoặc không mang nhiều ý nghĩa. Do đó, cần thực hiện các bước xử lý, làm sạch và cân bằng dữ liệu trong quá trình tiền xử lý. Sau khi xử lý, dữ liệu sẽ phù hợp hơn để xây dựng các mô hình máy học.

c. Tiền xử lý dữ liệu (Data Preprocessing)

- Tạo ra các hàm chức năng để làm sạch và xử lý dữ liệu văn bản, bao gồm: chuẩn hóa văn bản, xử lý láy âm tiết, loại bỏ các common token, loại bỏ dấu câu và ký tự xuống dòng, chuẩn hóa các từ viết tắt, xử lý các từ dính nhau, xử lý các từ không có nghĩa ít hơn 3 chữ, loại bỏ stopword.
- Kiểm tra dữ liệu sau khi làm sạch.
- Tách từ.
- Rút trích đặc trưng.

d. Ứng dụng mô hình phân lớp

Xây dựng và đánh giá mô hình bằng các thuật toán Machine Learning.

Đối với đề tài "Dự Đoán Lượt Đánh Giá Sản Phẩm Smartphone từ Bình Luận Người Dùng trên Sàn Thương Mại Điện Tử," nhóm bắt đầu bằng việc chia tập dữ liệu thành tập huấn luyện và tập kiểm tra sử dụng thư viện Scikit-learn. Sau đó, chúng ta sử dụng phương pháp phổ biến trong xử lý ngôn ngữ tự nhiên là TF-IDF để vector hóa văn bản. Mô hình máy học được xây dựng là Naive Bayes. Thực hiện Grid Search để tìm ra bộ tham số tốt, sau đó đánh giá hiệu suất của mô hình bằng cách tạo bảng tóm tắt các điểm số phân loại như Precision, Recall, và F1-score cho mỗi mô hình.

Xây dựng và đánh giá mô hình bằng các thuật toán Maximum Entropy.

Tiếp theo, nhóm áp dụng thuật toán Maximum Entropy với mô hình Logistics Regression để phân loại lượt đánh giá sản phẩm. Quy trình xây dựng và đánh giá mô hình tương tự như trong phần trước, với việc sử dụng CountVectorizer và TF-IDF transformer để vector hóa văn bản và đánh giá mô hình bằng các độ đo hiệu suất trên tập kiểm tra.

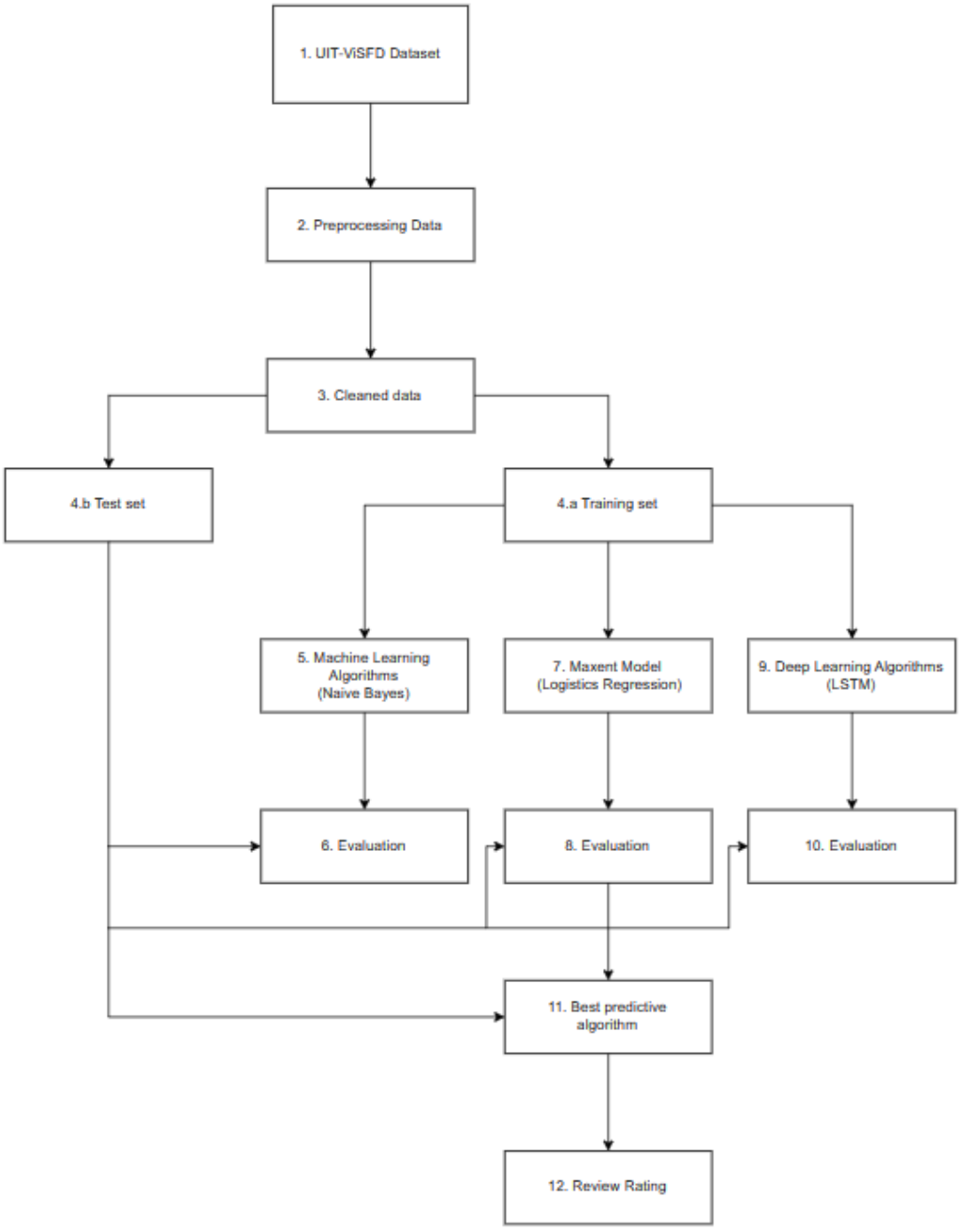
Xây dựng và đánh giá mô hình bằng các thuật toán Deep Learning

Cuối cùng, nhóm chuyển sang việc xây dựng mô hình học sâu LSTM sử dụng thư viện Keras. Mô hình này bao gồm các lớp nhúng (embedding layer), LSTM layer, và các lớp fully connected. Huấn luyện mô hình được thực hiện trên tập dữ liệu đã được xử lý, và các tham số như số lớp, số nơ-ron, hệ số học, và số vòng lặp được điều chỉnh để tối ưu hóa độ chính xác. Sau cùng, kiểm tra và đánh giá mô hình được thực hiện trên tập dữ liệu kiểm tra, và kết quả đánh giá hiệu suất được đo lường.

Đây là quy trình tổng thể để xây dựng và đánh giá mô hình trên cả ba thuật toán Machine Learning: Naive Bayes, Logistics Regression (Maximum Entropy), và LSTM (Deep Learning). Thực hiện đánh giá 3 mô hình trên bằng các độ đo như Accuracy, Precision, Recall, F1 Score và ma trận nhầm lẫn, chúng ta có thể lựa chọn mô hình tối ưu nhất giữa Logistic Regression, Naive Bayes và LSTM cho đề tài cụ thể này.

3.2. Phương hướng triển khai thực nghiệm

Nhóm thực hiện triển khai phương hướng thực hiện theo lưu đồ như sau:



3.3. Kết quả mong muốn dự báo đạt được

Kết quả mong muốn của đề tài "Dự Đoán Lượt Đánh Giá Sản Phẩm Smartphone từ Bình Luận Người Dùng trên Sàn Thương Mại Điện Tử" là tạo ra một hệ thống dự đoán độ chính xác cao về lượt đánh giá sản phẩm dựa trên bình luận người dùng. Đối với mô hình Machine Learning và mô hình Maxent, chúng em hy vọng mô hình sẽ có khả năng phân loại đúng cấp độ đánh giá dựa trên dữ liệu thực tế đã được nhãn. Trong khi đó, mô hình Deep Learning sẽ tập trung vào việc nhận biết và thu thập các từ ngữ mang ý nghĩa phân loại bình luận, làm cho mô hình tự động hơn và giảm công đoạn thủ công.

Qua quá trình so sánh hiệu suất giữa mô hình Machine Learning và Deep Learning, nếu một trong ba thuật toán có độ chính xác cao hơn, nhóm sẽ lựa chọn và triển khai mô hình đó nhằm tối ưu hóa dự đoán lượt đánh giá sản phẩm trên sàn thương mại điện tử. Kết quả cuối cùng mong đợi là có một hệ thống dự đoán hiệu quả, giúp doanh nghiệp đánh giá và quản lý đánh giá sản phẩm một cách tự động và chính xác.

Chương IV: Tổng Quan Bộ Dữ Liệu

4.1. Tổng quan bộ dữ liệu thu thập

Bộ dữ liệu về phản hồi đối với sản phẩm điện thoại thông minh của người Việt-UIT-ViSFD . Bộ dữ liệu gồm có 5 cột và 8898 dòng.

4.2. Các thuộc tính của bộ dữ liệu.

Tên thuộc tính	Kiểu dữ liệu	Mô tả
index	int64	số thứ tự các bình luận
comment	object	các bình luận của khách hàng và người tiêu dùng
n_star	int64	đánh giá của khách hàng và người tiêu dùng
date_time	object	thời gian của các bình luận và đánh giá
label	object	đánh giá tích cực/ tiêu cực/ trung tính theo các khía cạnh

Đối với n_star: 1 là cấp độ thấp nhất, thể hiện cho bình luận tiêu cực, 5 là cấp độ cao nhất, thể hiện bình luận tích cực.

4.3. Mô tả dữ liệu gốc

Để có thể biết được sơ bộ về bộ dữ liệu sử dụng, ta cần phải đưa bộ dữ liệu về dạng dataframe thông qua thư viện Pandas. Thư viện Pandas là một thư viện mã nguồn mở được xây dựng trên trên Numpy, được sử dụng thao tác và phân tích dữ liệu. Pandas được thế kế để người dùng làm việc với dữ liệu trở nên trực quan hơn.

Import thư viện pandas.

```
import pandas as pd
```

In 5 dòng đầu của dữ liệu.

	index	comment	n_star	date_time	label
0	0	Mới mua máy này Tại thegioididong thốt nốt cảm...	5	2 tuần trước	{CAMERA#Positive};{FEATURES#Positive};{BATTERY...
1	1	Pin kém còn lại miễn chê mua 8/3/2019 tính tra...	5	14/09/2019	{BATTERY#Negative};{GENERAL#Positive};{OTHERS};
2	2	Sao lúc gọi điện thoại màn hình bị chấm nhỏ nh...	3	17/08/2020	{FEATURES#Negative};
3	3	Mọi người cập nhật phần mềm lại , nó sẽ bớt tố...	3	29/02/2020	{FEATURES#Negative};{BATTERY#Neutral};{GENERAL...
4	4	Mới mua Sài được 1 tháng thấy pin rất trâu, Sà...	5	4/6/2020	{BATTERY#Positive};{PERFORMANCE#Positive};{SER...

Sau khi đã đọc được dữ liệu và chuyển thành dạng dữ liệu thành cấu trúc dataframe ta cần phải xem thử dữ liệu của ta có bao nhiêu cột và dòng.

Kiểm tra kiểu dữ liệu ở các cột

```
df.info()
```

Sau khi đã kiểm tra tổng quát bộ dữ liệu , bộ dữ liệu nhóm chúng em đưa ra có 8898 dòng dữ liệu và có 5 cột

Kiểu dữ liệu trong dataset bao gồm 3 cột kiểu Categorical và 2 cột kiểu Numerical.

4.4. Loại bỏ các cột không cần thiết

Nhóm chỉ thực hiện phân loại đánh giá dựa trên bình luận của khách hàng, vì vậy quyết định drop đi các cột còn lại.

```
df.drop(['index', 'date_time', 'label'], axis=1, inplace=True)
```

Tổng quát bộ dữ liệu sau khi đã xóa các cột, có 8898 dòng dữ liệu và còn lại 2 cột comment và n_star

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8898 entries, 0 to 8897
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   comment  8898 non-null   object
 1   n_star   8898 non-null   int64 
dtypes: int64(1), object(1)
```

4.5. Tổng quan dữ liệu sau khi loại bỏ.

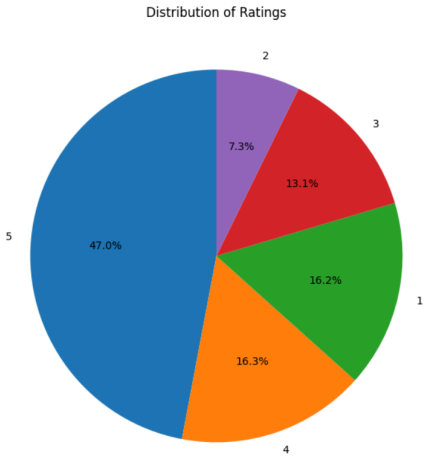
Tên thuộc tính	Kiểu dữ liệu	Mô tả	Ghi chú
comment	object	các bình luận của khách hàng và người tiêu dùng	
n_star	int64	đánh giá của khách hàng và người tiêu dùng	Đánh giá theo thang điểm 1-5

```
data.info()
```

4.6. Phân tích dữ liệu

Vẽ biểu đồ tròn để quan sát đánh giá (n_star)

Dựa vào biểu đồ, ta thấy đánh giá 5 sao chiếm gần 1 nửa (47%) trên tổng số các đánh giá, điều này cho thấy bộ dữ liệu bị mất cân bằng. Có khá ít đánh giá 2 sao, nếu đưa bộ dữ liệu vào mô hình, mô hình sẽ không đánh giá tốt các trường hợp được đánh giá 2 sao.



Vẽ biểu đồ WordCloud để quan sát các từ có trong bộ dữ liệu.

Nhìn vào biểu đồ trên, ta thấy các từ như ‘mua’, ‘thì’, ‘máy’, ‘rất’, … xuất hiện khá nhiều trong bộ dữ liệu, những từ này mang ít ý nghĩa trong xử lý văn bản nên xem xét loại bỏ những từ này ra khỏi bộ dữ liệu để làm sạch dữ liệu hơn.

Chương V: Triển khai và đánh giá

5.1. Thực hiện tiền xử lý:

Drop cột index và côt label, chỉ giữ lại cột comment và cột n_star để thực hiện phân loại đánh giá dựa trên bình luận của khách hàng.

Tạo các hàm chức năng sau:

- Chuẩn hóa văn bản tiếng Việt.
- Xử lý láy âm tiết. (Vd: ngon quá điiiii -> ngon quá đi)
- Loại bỏ các common token.
- Loại bỏ dấu câu và ký tự xuống dòng.
- Chuẩn hóa các từ viết tắt cơ bản (r -> rồi, đc -> được).
- Xử lý các từ viết dính nhau bằng cách tách ra (hơinhanh -> hơi nhanh).
- Xử lý các từ không phải tiếng Việt ngắn hơn 3 chữ.
- Loại bỏ các stopword.

a. Xác định một số chức năng để xử lý dữ liệu không chuẩn

Chuẩn hóa văn bản tiếng Việt, để chuẩn hóa kiểu gõ dấu theo kiểu cũ, chuẩn hóa bảng mã Unicode để thống nhất với nhau.

Tạo hàm chức năng có tên là ***handle_repeated_syllables*** để xóa các từ láy âm tiết, Sử dụng regex để tìm các từ có âm tiết lặp lại (ví dụ: quáááááá -> quá), chỉ giữ lại một phần và thêm vào từ gốc.

```
def handle_repeated_syllables(text):
    # Sử dụng regex để tìm các từ có âm tiết lặp lại (ví dụ: quáááááá)
    repeated_syllables_pattern = re.compile(r'(\w+?)\1+', re.UNICODE)
    # Hàm xử lý việc loại bỏ âm tiết lặp lại
    def handle_repetition(match):
        word = match.group(1)
        # Giữ lại chỉ một phần lặp lại và thêm vào từ gốc
        return word
```

Tạo hàm ***replace_common_token*** để loại bỏ các common token như PHONE, MENTION, NUMBER, DATETIME, … Đối với bộ dữ liệu này, đây là những thông tin không cần thiết nên bỏ đi bằng cách sử dụng regex.

```
def replace_common_token(txt):
    txt = re.sub(EMAIL, ' ', txt)
    txt = re.sub(URL, ' ', txt)
    txt = re.sub(MENTION, ' ', txt)
    txt = re.sub(DATETIME, ' ', txt)
    txt = re.sub(NUMBER, ' ', txt)
    txt = re.sub(PRICE, ' ', txt)
    return txt
```

Tạo hàm ***remove_unnecessary_characters*** bỏ dấu câu và các các ký tự xuống dòng bằng cách sử dụng regex và string.punctuation.

```
def remove_unnecessary_characters(text):
    RE_CLEAR = re.compile("[\n\r]+")# Thay thế các chuỗi xuống dòng (\n hoặc \r) bằng một ký tự trắng
    text = re.sub(RE_CLEAR, ' ', text)
    # Sử dụng string.punctuation để lấy tất cả các ký tự dấu câu
    translator = str.maketrans('', '', string.punctuation)
    # Loại bỏ dấu câu từ văn bản sử dụng bảng dịch (translator)
    text = text.translate(translator)
    return text
```

Đưa toàn bộ dữ liệu về ***lower*** để xử lý các bước sau.

Chuẩn hóa các từ viết tắt cơ bản bằng hàm ***normalize_acronyms***. Tải bộ từ viết tắt teencode tham khảo (bao gồm 1 cột từ viết tắt và 1 cột từ gốc) và sau đó thay những từ viết tắt của dữ liệu có trong bộ từ viết tắt bằng từ gốc trong bộ từ viết tắt.

```
def normalize_acronyms(text, teencode_file=folder+'/Data/teencode.xlsx'):
    # Đọc dữ liệu từ tệp Excel teencode.xlsx
        teencode_df = pd.read_excel(teencode_file, header=None, names=['teencode', 'replace'])
    words = []
    for word in text.strip().split():
        word = word.strip(string.punctuation)
        # Tìm kiếm trong teencode_df và thay thế
                                                    replacement = teencode_df.loc[teencode_df['teencode'].str.lower() == word, 'replace'].values
        if len(replacement) > 0:
            words.append(replacement[0])
        else:
            words.append(word)
    return ' '.join(words)
```

b. Tiền xử lý văn bản sử dụng các functions trên

Tạo hàm ***text_preprocess*** để tổng hợp lại các hàm chức năng ở trên và áp dụng cho dữ liệu input.

```
def text_preprocess(text):
    # 1. Chuẩn hóa văn bản tiếng việt
    text = text_normalize(text)
```

```
    # 2. Xử lý láy âm tiết
    text = handle_repeated_syllables(text)
    # 3. Loại bỏ các common token
    text = replace_common_token(text)
    # 4. Xóa bỏ dấu câu
    text = remove_unnecessary_characters(text)
    # 5. Đưa về lower
    text = text.lower()
    # 7. Chuẩn hóa các từ viết tắt cơ bản
    text = normalize_acronyms(text)
    return text
```

Lưu các dữ liệu đã được apply hàm Preprocess vào 1 cột mới 'comment_clean' trong dataframe.

```
df["comment_clean"] = df['comment'].apply(text_preprocess)
```

c. Tiếp tục xác định một số chức năng để xử lý dữ liệu

Tải về bộ từ điển tiếng Việt sau đó lấy tất cả từ đơn trong bộ từ điển và lấy tất cả từ đơn trong bộ dữ liệu. So khớp để xác định từ đơn trong bộ dữ liệu không nằm trong bộ từ đơn tiếng Việt.

Lấy các từ không nằm trong bộ từ đơn tiếng Việt có độ dài hơn 6 (vì từ đơn tiếng Việt dài nhất là 6 chữ). Tiến hành chia những từ này ra làm 2 phần, xét nếu 2 phần đều thuộc bộ từ đơn tiếng Việt thì tách ra bằng khoảng trắng, sau đó thu được bộ từ bao gồm 2 cột, 1 cột là các từ bị dính nhau và cột thứ 2 là các từ dính nhau được tách ra.

```
def stuck_words(text):
  for i in range(1,len(text)):
      word1 = text[:i]
      word2 = text[i:]
      if word1 in word_dict and word2 in word_dict:
          text_after = word1 + ' ' + word2
          return text_after
          break
# Gán các giá trị có thể tách ra vào cột mới stuck_word
not_vietnamese_than_6['stuck_word'] = float('nan')
for i in not_vietnamese_than_6['word']:
          not_vietnamese_than_6.loc[not_vietnamese_than_6['word']  ==  i,
'stuck_word'] = stuck_words(i)
```

Tạo hàm ***normalize_stuck*** bằng cách dựa vào bộ từ này để tách các từ bị dính nhau trong dữ liệu, hàm sử dụng tương tự với chuẩn hóa những viết tắt.

```
def normalize_stuck(text, df_stuck_words=df_stuck_words):
    words = []
    for word in text.strip().split():
        word = word.strip(string.punctuation)
        # Tìm kiếm trong teencode_df và thay thế
                                                    replacement        =
df_stuck_words.loc[df_stuck_words['word'].str.lower()       ==       word,
'stuck_word'].values
        if len(replacement) > 0:
            words.append(replacement[0])
        else:
            words.append(word)

    return ' '.join(words)
```

Cũng từ các từ đơn trong bộ dữ liệu không nằm trong bộ từ đơn tiếng Việt trên, lấy ra các từ ngắn hơn 3. Qua quan sát, những từ này không có nghĩa hoặc không có giá trị thông tin nên đưa những từ này vào stopword, tạm thời đặt tên là not_vietnamese_less_3.

Tạo hàm ***remove_stopwords*** để loại bỏ các stopword bằng cách tạo hàm get_stopwords để xác định những từ có idf thấp hơn một ngưỡng cụ thể trong bộ dữ liệu, và lấy danh sách này làm stopwords trong bộ dữ liệu. Tải về một bộ stopword tiếng Việt khác được lấy từ một số lượng lớn văn bản và so khớp, chỉ giữ lại những stopword đều thuộc 2 bộ stopword đã có sau đó bổ sung thêm tập not_vietnamese_less_3 ở trên vào. Tiến hành loại bỏ những từ có trong bộ stopword này.

```
def get_stopwords(documents, threshold=3):
    """
    :param documents: list of documents
    :param threshold:
    :return: list of words has idf <= threshold
    """
    tfidf = TfidfVectorizer(min_df=100)
    tfidf_matrix = tfidf.fit_transform(documents)
    features = tfidf.get_feature_names_out()
    stopwords = []
    print(min(tfidf.idf_), max(tfidf.idf_), len(features))
    for index, feature in enumerate(features):
        if tfidf.idf_[index] <= threshold:
            stopwords.append(feature)
    return stopwords
```

```
def remove_stopwords(line):
    words = []
    for word in line.strip().split():
        if word not in stopword:
            words.append(word)
    return ' '.join(words)
```

d. Tiền xử lý văn bản lần 2 sử dụng các functions mới

Tạo hàm ***text_preprocess2*** để tổng hợp lại các hàm chức năng ở trên và áp dụng cho dữ liệu input.

```
def text_preprocess2(text):
    # 7. Tách các từ viết dính nhau
    text = normalize_stuck(text)
    # 8. Loại bỏ các stopword tiếng Việt
    text = remove_stopwords(text)
    # 9. Loại bỏ các khoảng trắng liên tiếp
    RE_CLEAR = re.compile("\s+") # Các khoảng trắng liên tiếp
    text = re.sub(RE_CLEAR,' ', text)
    return text
```

Lưu các dữ liệu đã được apply hàm text_preprocess2 vào 1 cột mới **'comment_clean2'** trong dataframe.

```
df["comment_clean2"] = df['comment_clean'].apply(text_preprocess)
```

e. Kiểm tra độ dài của các comment đã được làm sạch

Kiểm tra độ dài của comment dài nhất và ngắn nhất xem có comment nào không hợp lệ không. Kết quả như sau:

```
Độ dài ngắn nhất: 8
Độ dài dài nhất: 796
```

Quan sát 2 dòng thì cả 2 dòng đều hợp lý nên không xử lý.

f. Áp dụng Tokenizing vào bộ dữ liệu đã tiền xử lý

Sau đó tách từ trong câu bằng lệnh **word_tokenize** từ thư viện underthesea.

```
def word_segmentation(text):
    text = underthesea.word_tokenize(text, format="text")
    return text
```

Lưu các dữ liệu đã được apply hàm ***word_segmentation*** vào 1 cột mới 'comment_token' trong dataframe.

```
df['comment_token'] = df['comment_clean2'].apply(word_segmentation)
```

5.2. Xây dựng và đánh giá mô hình phân tích dữ liệu đánh giá bình luận về sản phẩm điện thoại:

a. Xây dựng và đánh giá mô hình bằng các thuật toán máy học Naive Bayes:

- **Chia tập dữ liệu**

Sử dụng thư viện Scikit-learn (sklearn) để chia tập dữ liệu thành tập huấn luyện và tập kiểm tra. Cụ thể, đầu vào của hàm train_test_split là 2 Series X và y. Với X chứa văn bản sau khi đã được xử lý và làm sạch, được lưu trữ trong cột ***'comment_token'*** của dataframe df. Và y là dữ liệu về nhãn của các văn bản, được lưu trữ trong cột ***'n_star'*** của dataframe df.

```
X, y = df['comment_token'], df['n_star']
```

Sử dụng hàm *train_test_split()* trong sklearn để tách dữ liệu thành tập huấn luyện và tập kiểm tra. Mặc định, hàm train_test_split sẽ chia dữ liệu đầu vào thành hai phần với tỷ lệ 80% cho tập huấn luyện và 20% cho tập kiểm tra. Ta sử dụng tham số random_state thiết lập giá trị ngẫu nhiên được sử dụng để xáo trộn dữ liệu trước khi chia. Bằng cách đặt giá trị này thành một số cụ thể (trong trường hợp này là 42), chúng ta có thể đảm bảo rằng cùng một chia ngẫu nhiên được tạo ra mỗi lần chạy mã. Điều này hữu ích cho tính nhất quán trong quá trình tái tạo.

Kết quả trả về của hàm train_test_split là 4 mảng x_train, x_test, y_train, y_test chứa dữ liệu đã được chia thành tập huấn luyện và tập kiểm tra tương ứng.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

- **Vector hóa văn bản**

Ta dùng đến phương pháp TF-IDF được sử dụng trong xử lý ngôn ngữ tự nhiên để vector hóa văn bản, sử dụng 2 hàm CountVectorizer và TF-IDF transformer.

- **CountVectorizer:** sử dụng CountVectorizer để chuyển đổi văn bản thành ma trận đếm các từ, bỏ qua các từ trong danh sách stop_words (từ không có ý nghĩa như "và", "với",...), và chỉ tính toán các từ xuất hiện ít nhất min_df lần trong toàn bộ tập dữ liệu.
- **TF-IDF transformer:** được sử dụng để chuyển đổi ma trận đếm từ (CountVectorizer tạo ra) thành ma trận TF-IDF.

```
count_vectorizer = CountVectorizer(ngram_range=(1, 5),
stop_words=stopword, max_df=0.5, min_df=5)
tfidf_vectorizer = TfidfTransformer(use_idf=False, sublinear_tf = True,
norm='l2', smooth_idf=True)

X_train = count_vectorizer.fit_transform(X_train)
X_train = tfidf_vectorizer.fit_transform(X_train)

X_test = count_vectorizer.transform(X_test)
```

```
X_test = tfidf_vectorizer.transform(X_test)
```

- **Xây dựng mô hình bằng thuật toán máy học Naive Bayes**

Sử dụng MultinomialNB để xây dựng một mô hình phân loại Naive Bayes đa thức từ các vectơ đặc trưng và thực hiện Grid Search để tìm ra bộ tham số tốt nhất cho mô hình.

```
# Khởi tạo mô hình MultinomialNB
naive_bayes_classifier = MultinomialNB()

# Thiết lập các tham số cần tìm kiếm
parameters = {
    'force_alpha' : (True, False),
    'fit_prior': (True, False),
    'alpha': (1, 0.1, 0.01)
}

# Khởi tạo GridSearchCV với mô hình, tham số và số lượng fold trong cross-validation (cv)
grid_search_nb = GridSearchCV(estimator=naive_bayes_classifier, param_grid=parameters, cv=5, scoring='accuracy')

# Huấn luyện GridSearchCV trên dữ liệu
grid_search_nb.fit(X_train, y_train)

# In ra các tham số tốt nhất
print("Best Parameters:", grid_search_nb.best_params_)
```

```
Best Parameters: {'alpha': 0.1, 'fit_prior': True, 'force_alpha': True}
```

Sử dụng bộ tham số đó đưa vào mô hình:

```
MultinomialNB
MultinomialNB(alpha=0.1, force_alpha=True)
```

- **Đánh giá mô hình MultinomialNB:**

Đánh giá hiệu suất của mô hình bằng các chỉ số đánh giá, sử dụng hàm *classification_report* báo cáo được in ra để cung cấp cái nhìn tổng quan về khả năng dự đoán của mô hình Naive Bayes trên tập kiểm tra. Các độ đo trong báo cáo giúp định

lượng khả năng của mô hình trong việc phân loại các lớp khác nhau và đánh giá khả năng tổng quát của mô hình.

```
# Dự đoán trên tập test
y_pred_nb = nb_model.predict(X_test)
# Đánh giá mô hình bằng các chỉ số
report1 = metrics.classification_report(y_test, y_pred_nb, digits=3)
print(report1)
```

```
              precision    recall  f1-score   support

           1      0.557     0.731     0.633       279
           2      0.364     0.034     0.063       117
           3      0.355     0.270     0.307       222
           4      0.306     0.182     0.228       286
           5      0.746     0.906     0.819       876

    accuracy                          0.626      1780
   macro avg      0.466     0.425     0.410      1780
weighted avg      0.572     0.626     0.581      1780
```

Dựa trên báo cáo, có thể kết luận rằng mô hình có độ chính xác (accuracy) là khoảng 62.6%. Mô hình có hiệu suất khác biệt đáng kể giữa các lớp, được thể hiện qua các chỉ số precision, recall, và F1-score, kết quả cho mỗi lớp là không đồng đều. Tổng quát, mô hình có hiệu suất tốt đối với lớp 5, nhưng có hiệu suất thấp đối với các lớp khác, đặc biệt là lớp 2. Nguyên nhân là do lớp 5 chiếm nhiều phần trăm trong bộ dữ liệu, còn lớp 2 chỉ chiếm 7.3% nên hiệu suất của mô hình đối với lớp này thấp. bộ dữ liệu. Vì vậy cần xem xét xử lý mất cân bằng giữa các lớp trong dữ liệu.

Thực hiện *cross validation* để đánh giá biến động

```
#CROSS VALIDATION
cross_score = cross_val_score(nb_model, X_train,y_train, cv=10)
cross_score
```

```
array([0.5744382 , 0.57022472, 0.55758427, 0.56460674, 0.56601124,
       0.56320225, 0.55617978, 0.56320225, 0.5625879 , 0.58931083])
```

Các giá trị độ chính xác nằm trong khoảng từ khoảng 0.556 đến 0.589. Sự biến động này không quá lớn, có thể coi là đồng đều. Kiểm tra biến động giữa các fold cho thấy hiệu quả của mô hình trên nhiều tập dữ liệu khá giống nhau. Điều này có thể minh chứng mô hình không bị overfitting.

b. Xây dựng và đánh giá mô hình bằng các thuật toán Maxent:

- Chia tập dữ liệu

Sử dụng cùng tập dữ liệu X_train, y_train, X_test, y_test với mô hình máy học Naive Bayes. Bộ dữ liệu đưa vào đã được làm sạch, tách từ và trích xuất đặc trưng.

- Xây dựng mô hình bằng thuật toán máy học Logistic Regression

Logistic regression cũng là một mô hình dựa trên nguyên lý Maximum entropy. Sử dụng Logistic regression để huấn luyện một mô hình phân loại đa lớp từ các vectơ đặc trưng và thực hiện Grid Search để tìm ra bộ tham số tốt nhất cho mô hình.

```
# Tạo mô hình Logistic Regression (Softmax Regression)
model = LogisticRegression()

# Định nghĩa các tham số cần tối ưu
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100],
              'penalty': ['l2'],
               'multi_class': ['multinomial'], # xác định rằng đang thực
hiện phân loại đa lớp (softmax regression).
              'solver' : ['lbfgs','sag','saga','newton-cg']
              }

# Tạo đối tượng GridSearchCV
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5,
scoring='accuracy')

# Thực hiện grid search trên tập huấn luyện
grid_search.fit(X_train, y_train)

# Hiển thị kết quả tối ưu
print("Best parameters: ", grid_search.best_params_)
```

```
Best parameters:  {'C': 1, 'multi_class': 'multinomial', 'penalty':
'l2', 'solver': 'lbfgs'}
```

Sử dụng bộ tham số đó đưa vào mô hình:

```
LogisticRegression
LogisticRegression(C=1, multi_class='multinomial')
```

- **Đánh giá mô hình Logistic Regression:**

Đánh giá hiệu suất của mô hình bằng các chỉ số đánh giá, sử dụng hàm *classification_report* báo cáo được in ra để cung cấp cái nhìn tổng quan về khả năng dự đoán của mô hình Logistic Regression trên tập kiểm tra.

```
# Dự đoán trên tập kiểm tra
y_pred_lr = maxent_model.predict(X_test)
```

```
# Đánh giá mô hình bằng các chỉ số
report1 = metrics.classification_report(y_test, y_pred_lr, digits=3)
print(report1)
```

```
              precision    recall  f1-score   support

           1      0.572     0.699     0.629       279
           2      0.312     0.043     0.075       117
           3      0.342     0.248     0.287       222
           4      0.351     0.206     0.260       286
           5      0.734     0.917     0.815       876

    accuracy                          0.628      1780
   macro avg      0.462     0.422     0.413      1780
weighted avg      0.570     0.628     0.582      1780
```

Dựa trên báo cáo, có thể kết luận rằng mô hình có độ chính xác (accuracy) là khoảng 62.6%. Tổng thể, mô hình có vẻ hoạt động tốt đối với các lớp có đánh giá cao hơn (lớp 1 và lớp 5), nhưng hiệu suất giảm đáng kể đối với các lớp khác, đặc biệt là lớp 2 và lớp 4. Điều này có thể yêu cầu xem xét và điều chỉnh thêm mô hình để cải thiện hiệu suất trên các lớp khó dự đoán hơn.

Thực hiện *cross validation* để đánh giá biến động

```
#CROSS VALIDATION
cross_score = cross_val_score(maxent_model, X_train,y_train, cv=10)
cross_score
```

```
array([0.60252809, 0.61376404, 0.58707865, 0.59691011, 0.5997191 ,
       0.59269663, 0.58426966, 0.6011236 , 0.59634318, 0.604782  ])
```

Các giá trị độ chính xác nằm trong khoảng từ khoảng 0.584 đến 0.613. Các giá trị này có vẻ ổn định, và có vẻ mô hình đang giữ được sự ổn định trên các tập dữ liệu khác nhau. Điều này là một dấu hiệu tích cực về tính tổng quát của mô hình.

c. Xây dựng và đánh giá mô hình bằng các thuật toán học sâu LSTM:

- Chia tập dữ liệu

Sử dụng cùng tập dữ liệu X, y với mô hình máy học Naive Bayes.

Thực hiện OneHotEncoder đối với biến target. Chia tập dữ liệu thành tập train và test với tỷ lệ 80:20 sau đó thực hiện vector hóa văn bản bằng phương pháp TF-IDF đối với biến X_train, X_test.

```
encoder = OneHotEncoder()
encoded_data = encoder.fit_transform(y.values.reshape(-1, 1))
y = encoded_data
```

- Xây dựng mô hình bằng thuật toán học sâu LSTM

Mô hình học sâu được đào tạo như sau:

1. Chuẩn bị dữ liệu:

- X_train_array và X_test_array là các ma trận biểu diễn của dữ liệu đầu vào (feature matrix) cho tập huấn luyện và tập kiểm tra, tương ứng. Dữ liệu đã được chuyển đổi thành dạng ma trận để có thể sử dụng trong mô hình.
- y_train_array và y_test_array là các ma trận one-hot encoding của nhãn cho tập huấn luyện và tập kiểm tra.

2. Xây dựng mô hình LSTM:

- Mô hình bắt đầu với một lớp Embedding, trong đó mỗi từ được biểu diễn dưới dạng một vectơ có kích thước 128.
- Tiếp theo là một lớp LSTM với 128 units để học các mối quan hệ dài hạn trong dữ liệu chuỗi.
- Cuối cùng, một lớp Dense với activation function là 'sigmoid' được sử dụng để dự đoán đầu ra cho 5 nhãn.
- Mô hình được huấn luyện với hàm mất mát là 'categorical_crossentropy' và tối ưu hóa bằng thuật toán 'adam'.

3. Huấn luyện mô hình:

- Mô hình được huấn luyện trên dữ liệu tập huấn luyện với batch size là 32 và số epoch là 50. Validation split là 0.2, có nghĩa là 20% dữ liệu sẽ được sử dụng để đánh giá hiệu suất mô hình trong quá trình huấn luyện.

```
# @title LSTM
X_train_array = X_train.toarray()
X_test_array = X_test.toarray()

y_train_array = y_train.toarray()
y_test_array = y_test.toarray()

max_length = X_train_array.shape[1]
vocab_size = X_train_array.shape[1]

# Xây dựng mô hình LSTM
model = Sequential()
model.add(Embedding(X_train_array.shape[1],                    128,
input_length=max_length))
model.add(LSTM(128))
model.add(Dense(5, activation='sigmoid'))
model.compile(loss='categorical_crossentropy',          optimizer='adam',
metrics=['accuracy'])

# Huấn luyện mô hình (tăng epoch lên khoảng 500-1000)
history   =   model.fit(X_train_array,   y_train_array,   batch_size=32,
epochs=50, validation_split=0.2)
```

```
# Dự đoán nhãn cho dữ liệu mới
predicted_scores = model.predict(X_test_array)
predicted_scores = encoder.inverse_transform(predicted_scores)

# Hiển thị dự đoán
for i, score in enumerate(predicted_scores):
    print(f"Bình luận {i+1}: Điểm dự đoán - {score}")
```

- **Đánh giá mô hình LSTM:**

Đánh giá hiệu suất của mô hình bằng các chỉ số đánh giá, sử dụng hàm *classification_report* báo cáo được in ra để cung cấp cái nhìn tổng quan về khả năng dự đoán của mô hình Logistic Regression trên tập kiểm tra.

```
# Dự đoán trên tập kiểm tra
y_pred_lr = maxent_model.predict(X_test)
# Đánh giá mô hình bằng các chỉ số
report1 = metrics.classification_report(y_test, y_pred_lr, digits=3)
print(report1)
```

```
              precision    recall  f1-score   support

           1      0.000     0.000     0.000       279
           2      0.000     0.000     0.000       117
           3      0.000     0.000     0.000       222
           4      0.000     0.000     0.000       286
           5      0.492     1.000     0.660       876

    accuracy                          0.492      1780
   macro avg      0.098     0.200     0.132      1780
weighted avg      0.242     0.492     0.325      1780
```

Kết quả đánh giá cho thấy mô hình không thể dự đoán các lớp trừ lớp 5, với độ chính xác khoảng 49.2%. Đối với lớp 5, mô hình có độ chính xác đối với precision, recall và f1-score cao hơn, nhưng vẫn còn chênh lệch. Dựa vào kết quả này, mô hình có vẻ chưa hiệu quả trong việc dự đoán các lớp (1-4), có thể do sự mất cân bằng trong số lượng mẫu giữa các lớp. Vì bộ dữ liệu chứa nhiều lớp 5 sao, nhưng nguyên lý của mô hình sẽ dự đoán phần trăm xảy ra ở mỗi lớp, và lớp 5 sao luôn đạt kết quả phần trăm cao nhất, dẫn đến mô hình phân lớp cho toàn bộ tập test là lớp 5 sao. Kết quả này tệ hơn 2 mô hình Logistic Regression và Naive Bayes nên chỉ đánh giá 2 mô hình còn lại sau khi xử lý mất cân bằng dữ liệu và bỏ qua mô hình này.

d. Xử lý mất cân bằng dữ liệu:

Với bộ dữ liệu trên, các kết quả đánh giá đều không quá tốt, đặc điểm chung của các kết quả trên là luôn hoạt động tốt đối với một vài lớp và không hiệu quả đối với các lớp còn lại. Nguyên nhân là do dữ liệu bị mất cân bằng, vì thế xử lý mất cân bằng dữ liệu bằng cách giảm kích thước mẫu hoặc tăng kích thước mẫu.

- **Giảm kích thước mẫu**

Chia dữ liệu thành các lớp, sau đó xác định lớp có số lượng mẫu ít nhất và giảm mẫu các lớp khác để đảm bảo không vượt quá số lượng mẫu ít nhất. Kết hợp lại thành DataFrame mới cân bằng. Thu được bảng kết quả như sau với 3240 dòng, trong đó mỗi đánh giá là 648 dòng:

	index	Unnamed: 0	comment	n_star	comment_clean	comment_clean2	comment_length	comment_token
0	7043	7043	Có ai hay tự nhiên bị mất wf như mình ko. Phải...	1	có ai hay tự nhiên bị mất wf như mình không ph...	có ai hay tự nhiên bị mất wf như mình không ph...	148	có ai hay tự_nhiên bị mất wf như mình không ph...
1	3698	3698	Thứ nhất: độ sáng 35 % không auto, xem Faceboo...	1	thứ nhất độ sáng không auto xem facebok và nhắ...	thứ nhất độ sáng không auto xem facebok và nhắ...	796	thứ nhất độ sáng không auto xem facebok và nhắ...
2	305	305	Sài 1 tháng là thấy máy xuống cấp trầm trọng, ...	1	xài tháng là thấy máy xuống cấp trầm trọng lỗi...	xài tháng là thấy máy xuống cấp trầm trọng lỗi...	280	xài tháng là thấy máy xuống_cấp trầm_trọng lỗi...
3	2125	2125	Mình có mua dt này chưa dc một tháng mà chụp ả...	1	mình có mua điện thoại này chưa được một tháng...	mình có mua điện thoại này chưa được một tháng...	111	mình có mua điện_thoại này chưa được một tháng...
4	1253	1253	Mua 11/07/2020 đến giờ dùng nhanh hết pin và ...	1	mua đến giờ dùng nhanh hết pin và vào mục gì v...	mua đến giờ dùng nhanh hết pin và vào mục gì v...	76	mua đến giờ dùng nhanh hết pin và vào mục gì v...
...	...	...	...	...	...	...	...	...
3235	7374	7374	Quá tuyệt vời tôi mua được 3 ngày dùng ok lắm ...	5	quá tuyệt vời tôi mua được ngày dùng ok lắm pi...	quá tuyệt vời tôi mua được ngày dùng ok lắm pi...	208	quá tuyệt_vời tôi mua được ngày dùng ok lắm pi...
3236	3689	3689	Pin khá tốt, dùng cả ngày tối về vẫn còn gần 3...	5	pin khá tốt dùng cả ngày tối về vẫn còn gần ch...	pin khá tốt dùng cả ngày tối về vẫn còn gần ch...	372	pin khá tốt dùng cả ngày tối về vẫn còn gần ch...
3237	5663	5663	Máy ok về mọi thứ.\nCamera thì ko nét lắm!\nAe...	5	máy ok về mọi thứ camera thì không nét lắm anh...	máy ok về mọi thứ camera thì không nét lắm anh...	122	máy ok về mọi thứ camera thì không nét lắm anh...
3238	6884	6884	Máy rất tốt a chơi game và làm việc đều được h...	5	máy rất tốt a chơi game và làm việc đều được h...	máy rất tốt a chơi game và làm việc đều được h...	130	máy rất tốt a chơi game và làm_việc đều được h...
3239	6150	6150	hiệu năng ngon ,hơi ấm .cảm giác cầm chơi lâu ...	5	hiệu năng ngon hơi ấm cảm giác cầm chơi lâu th...	hiệu năng ngon hơi ấm cảm giác cầm chơi lâu th...	158	hiệu_năng ngon_hơi ấm cảm_giác cầm chơi lâu th...

3240 rows × 8 columns

Đánh giá hiệu suất của mô hình bằng các chỉ số đánh giá, sử dụng hàm *classification_report* báo cáo được in ra để cung cấp cái nhìn tổng quan về khả năng dự đoán của 2 mô hình Logistic Regression và Naive Bayes trên tập kiểm tra, thu được kết quả như sau:

```
Classification Report for Logistic Regression model:
              precision    recall  f1-score   support

           1      0.584     0.510     0.544       157
           2      0.311     0.288     0.299       132
           3      0.264     0.320     0.289       122
           4      0.277     0.267     0.272       116
           5      0.550     0.587     0.568       121

    accuracy                          0.400       648
   macro avg      0.397     0.394     0.394       648
weighted avg      0.407     0.400     0.402       648

------------------------------
Classification Report for Naive Bayes model:
              precision    recall  f1-score   support

           1      0.540     0.478     0.507       157
           2      0.311     0.318     0.315       132
           3      0.291     0.320     0.305       122
```

```
              4       0.349     0.319     0.333        116
              5       0.604     0.669     0.635        121

    accuracy                            0.423        648
   macro avg        0.419     0.421     0.419        648
weighted avg        0.424     0.423     0.423        648
```

Cả hai mô hình có khả năng dự đoán khác nhau đối với các lớp. Điều này có thể phản ánh sự hiệu quả của mỗi mô hình đối với từng lớp cụ thể. Naive Bayes có vẻ cho kết quả tốt hơn với một số chỉ số đánh giá (như precision và F1-score) so với Logistic Regression trên tập kiểm thử. Cả hai mô hình đều đạt hiệu quả cao hơn ở lớp 1 và 5, đạt hiệu quả thấp hơn ở 3 lớp còn lại. Kết quả đạt được của 2 mô hình trên bộ dữ liệu giảm mẫu tệ hơn trước khi giảm mẫu, nên đây không phải một phương pháp hiệu quả để cân bằng dữ liệu.

- **Tăng kích thước mẫu**

Sử dụng RandomOverSampler để tăng kích thước mẫu. Thu được bảng kết quả như sau với 20925 dòng, trong đó mỗi đánh giá là 4185 dòng:

	index	Unnamed: 0	comment	n_star	comment_clean	comment_clean2	comment_length	comment_token
0	7043	7043	Có ai hay tự nhiên bị mất wf như mình ko. Phải...	1	có ai hay tự nhiên bị mất wf như mình không ph...	có ai hay tự nhiên bị mất wf như mình không ph...	148	có ai hay tự_nhiên bị mất wf như mình không ph...
1	3698	3698	Thứ nhất: độ sáng 35 % không auto, xem Faceboo...	1	thứ nhất độ sáng không auto xem facebok và nhắ...	thứ nhất độ sáng không auto xem facebok và nhắ...	796	thứ nhất độ sáng không auto xem facebok và nhắ...
2	305	305	Sài 1 tháng là thấy máy xuống cấp trầm trọng, ...	1	xài tháng là thấy máy xuống cấp trầm trọng lỗi...	xài tháng là thấy máy xuống cấp trầm trọng lỗi...	280	xài tháng là thấy máy xuống_cấp trầm_trọng lỗi...
3	2125	2125	Mình có mua dt này chưa dc một tháng mà chụp ả...	1	mình có mua điện thoại này chưa được một tháng...	mình có mua điện thoại này chưa được một tháng...	111	mình có mua điện_thoại này chưa được một tháng...
4	1253	1253	Mua 11/07/2020 đến giờ dùng nhanh hết pin và ...	1	mua đến giờ dùng nhanh hết pin và vào mục gì v...	mua đến giờ dùng nhanh hết pin và vào mục gì v...	76	mua đến giờ dùng nhanh hết pin và vào mục gì v...
...	...	...	...	...	...	...	...	...
3235	7374	7374	Quá tuyệt vời tôi mua được 3 ngày dùng ok lắm ...	5	quá tuyệt vời tôi mua được ngày dùng ok lắm pi...	quá tuyệt vời tôi mua được ngày dùng ok lắm pi...	208	quá tuyệt_vời tôi mua được ngày dùng ok lắm pi...
3236	3689	3689	Pin khá tốt, dùng cả ngày tối về vẫn còn gần 3...	5	pin khá tốt dùng cả ngày tối về vẫn còn gần ch...	pin khá tốt dùng cả ngày tối về vẫn còn gần ch...	372	pin khá tốt dùng cả ngày tối về vẫn còn gần ch...
3237	5663	5663	Máy ok về mọi thứ.\nCamera thì ko nét lắm!\nAe...	5	máy ok về mọi thứ camera thì không nét lắm anh...	máy ok về mọi thứ camera thì không nét lắm anh...	122	máy ok về mọi thứ camera thì không nét lắm anh...
3238	6884	6884	Máy rất tốt ạ chơi game và làm việc đều được h...	5	máy rất tốt ạ chơi game và làm việc đều được h...	máy rất tốt ạ chơi game và làm việc đều được h...	130	máy rất tốt ạ chơi game và làm_việc đều được h...
3239	6150	6150	hiệu năng ngon ,hơi ấm .cảm giác cầm chơi lâu ...	5	hiệu năng ngon hơi ấm cảm giác cầm chơi lâu th...	hiệu năng ngon hơi ấm cảm giác cầm chơi lâu th...	158	hiệu_năng ngon_hơi ấm cảm_giác cầm chơi lâu th...

3240 rows × 8 columns

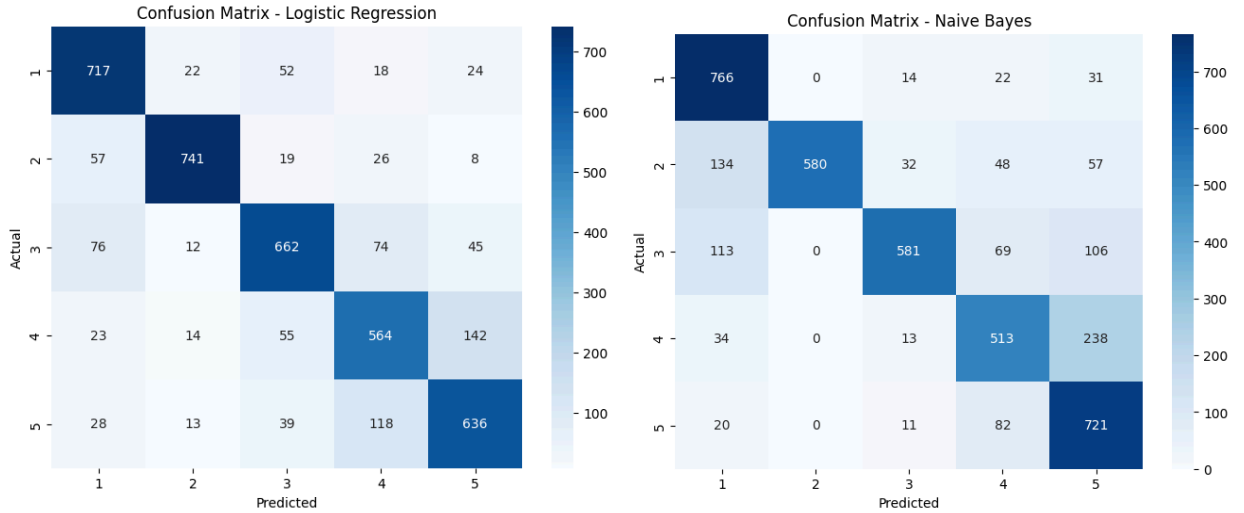
Đánh giá hiệu suất của mô hình bằng các chỉ số đánh giá, sử dụng hàm *classification_report* báo cáo được in ra để cung cấp cái nhìn tổng quan về khả năng dự đoán của 2 mô hình Logistic Regression và Naive Bayes trên tập kiểm tra, thu được kết quả như sau:

```
Classification Report for Logistic Regression model:
              precision    recall  f1-score   support

           1       0.768     0.852     0.808       833
           2       0.943     0.839     0.888       851
           3       0.785     0.757     0.771       869
           4       0.719     0.707     0.713       798
           5       0.731     0.772     0.751       834

    accuracy                          0.786      4185
   macro avg       0.789     0.785     0.786      4185
```

```
weighted avg       0.790     0.786     0.787       4185

------------------------------
Classification Report for Naive Bayes model:
              precision    recall  f1-score   support

           1       0.749     0.906     0.820       833
           2       1.000     0.686     0.814       851
           3       0.875     0.695     0.775       869
           4       0.707     0.630     0.667       798
           5       0.617     0.881     0.726       834

    accuracy                          0.760      4185
   macro avg       0.790     0.760     0.760      4185
weighted avg       0.792     0.760     0.761      4185
```

Cả hai mô hình đều có hiệu suất tốt với một số lớp nhất định. Ví dụ, Logistic Regression có precision cao hơn đối với lớp 2, trong khi Naive Bayes có recall cao hơn đối với lớp 1 và lớp 5. Kết quả phụ thuộc vào mục tiêu cụ thể của bạn. Đối với một số lớp, Logistic Regression có hiệu suất tốt hơn, trong khi Naive Bayes làm tốt hơn ở các lớp khác. Nhìn vào các kết quả trên thì kết quả của 2 mô hình Naive Bayes và Logistic Regression đối với bộ dữ liệu tăng kích thước mẫu khá tốt. Nên tiếp tục phân tích 2 mô hình của bộ dữ liệu này để chọn mô hình tốt nhất.

Sử dụng *ma trận nhầm lẫn* đánh giá cho 2 mô hình:



Nhìn vào cả hai mô hình, ta thấy biểu đồ của ma trận nhầm lẫn của hai mô hình đều thể hiện được mô hình khá tốt. Tuy nhiên mô hình Naive Bayes đạt hiệu quả hơn trên lớp 1 và 5 so với 3 lớp còn lại, mô hình Logistic Regression có sự chênh lệch giữa các lớp không quá lớn như mô hình Naive Bayes. Và từ kết quả trên, ta có mô hình Logistic Regression có độ chính xác cao hơn, vì thế chọn mô hình này để áp dụng sử dụng và thiết kế giao diện.

e. Áp dụng mô hình

Lưu mô hình Logistic Regression và lưu bộ dữ liệu cân bằng bằng cách tăng mẫu để sử dụng về sau. Lập hàm ***predict_rate*** dự đoán kết quả đánh giá là bao nhiêu sao, với kết quả tương ứng từ 1 đến 4. Trong hàm này sẽ thực hiện các bước làm sạch dữ liệu, tách dữ liệu và trích xuất đặc trưng.

```
   def predict_rate(text):
       text = text_preprocess(text)
       text = text_preprocess2(text)
       text = underthesea.word_tokenize(text, format="text")

       balanced_df = pd.read_csv(folder +'/Data/balanced_df.csv')
       X, y = balanced_df['comment_token'], balanced_df['n_star']
          X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
       # Chia dữ liệu thành tập huấn luyện và tập kiểm tra
             count_vectorizer = CountVectorizer(ngram_range=(1, 5),
stop_words=stopword, max_df=0.5, min_df=5)
       tfidf_vectorizer = TfidfTransformer(use_idf=False, sublinear_tf =
True, norm='l2', smooth_idf=True)

       X_train = count_vectorizer.fit_transform(X_train)
       X_train = tfidf_vectorizer.fit_transform(X_train)

       X_test = count_vectorizer.transform([text])
       X_test = tfidf_vectorizer.transform(X_test)
       # Dự đoán dữ liệu mới
       maxent_model = joblib.load(folder + '/Output/maxent_model.pkl')
       y_pre = maxent_model.predict(X_test)
       return str(y_pre[0])
```

Nhóm tiến hành lấy 2 câu ví dụ để đánh giá mô hình Logistic Regression. Và thu được kết quả như sau:

```
text = input("Bạn thấy sản phẩm điện thoại này như thế nào: ")
print(f'Dự đoán bình luận của bạn là {predict_rate(text)} sao.')
```

```
Bạn thấy sản phẩm điện thoại này như thế nào: Chán phèo
Dự đoán bình luận của bạn là 1 sao
```

```
text = input("Bạn thấy sản phẩm điện thoại này như thế nào: ")
print(f'Dự đoán bình luận của bạn là {predict_rate(text)} sao.')
```

```
Bạn thấy sản phẩm điện thoại này như thế nào: Máy có thiết kế đẹp, giao hàng nhanh
Dự đoán bình luận của bạn là 5 sao.
```

Kết quả cho thấy, phân loại khá ổn.

Chương VI: Kết Luận

6.1. Thiết kế giao diện

a. Tài nguyên sử dụng:

Thư viện Gradio: được sử dụng bằng ngôn ngữ Python. Thư viện này cho phép xây dựng và triển khai các ứng dụng web. Điểm nổi bật nhất của thư viện này là thực hiện trực quan dữ liệu chỉ với rất ít dòng mã. Một lợi ích khác của thư viện này là làm quá trình thực hiện trở nên nhanh chóng và dễ dàng hơn. Với Gradio, có thể thử nghiệm các đầu vào khác nhau. Việc xác thực mô hình dễ dàng hơn bao giờ hết vì nó có sẵn các liên kết công khai. Do vậy việc triển khai và phân phối các ứng dụng web trở nên rất dễ dàng.

b. Code giao diện:

```
def combine(a):
    return predict_rate(a) + '⭐' # Chỗ này điền dự đoán của mô hình
def mirror(x):
    return x


    import gradio as gr

with gr.Blocks() as demo:

     txt = gr.Textbox(label="Bạn thấy sản phẩm điện thoại này như thế
nào", lines=2)
    txt_2 = gr.Textbox(value="", label="Kết quả đánh giá")
    btn = gr.Button(value="Submit")
    btn.click(combine, inputs=[txt], outputs=[txt_2])
    gr.Markdown("## Đánh giá ví dụ")
    gr.Examples(
         ["Tôi thấy sản phẩm này đẹp", "Cấu hình máy mạnh", "Máy chạy quá
chậm"],
        [txt],
        txt_2,
        combine,
        cache_examples=True)
if __name__ == "__main__":
    demo.launch()
```

c. Chức năng của giao diện:

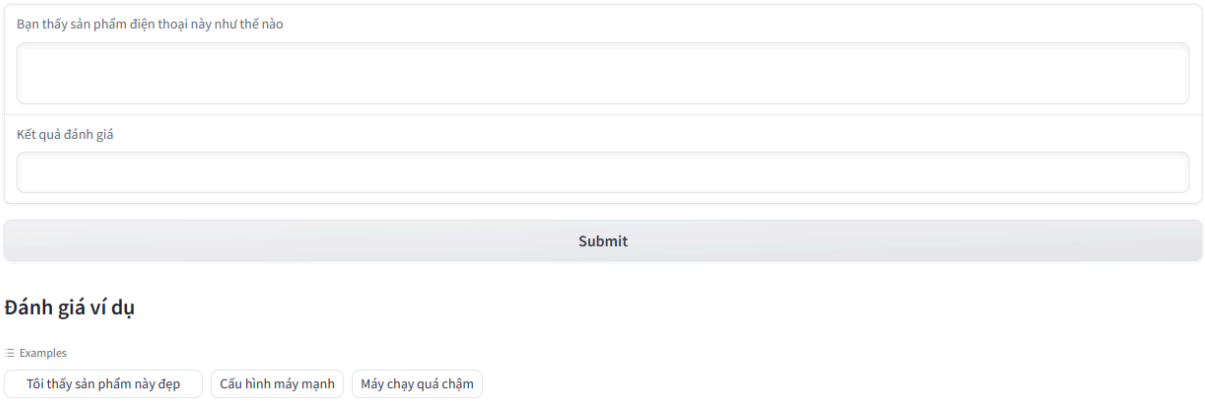
Giao diện có chức năng cho phép người dùng nhập vào bình luận về sản phẩm và phân loại bình luận đó được bao nhiêu sao. Nếu người dùng không biết nhập gì có thể nhấn ví dụ để xem kết quả phân loại.

- Bước 1: Nhập vào nội dung bình luận.
- Bước 2: Nhấn Submit.

- Bước 3: Chọn ví dụ nếu không biết nhập gì.

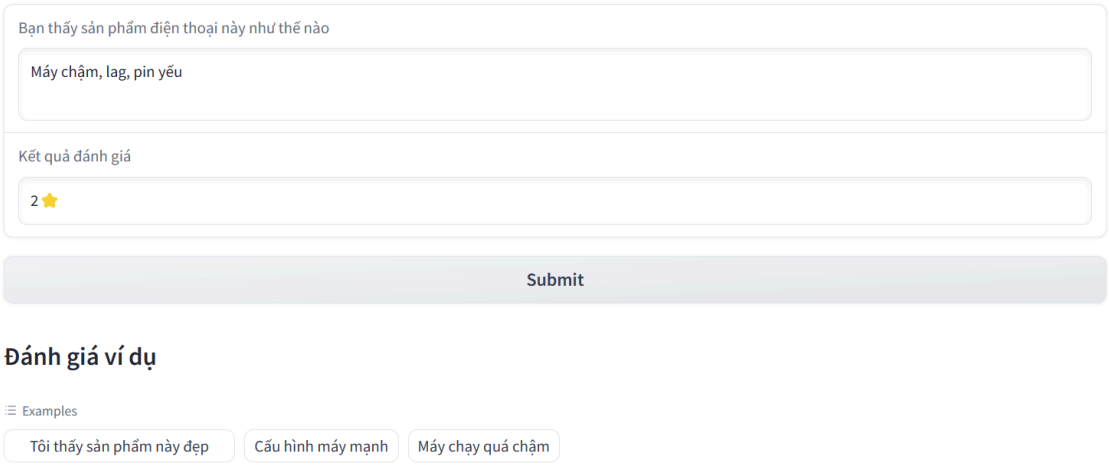
d. Thiết kế:

Giao diện được thiết kế như sau:



e. Chạy thử:

Chạy thử đánh giá "Máy chậm, lag, pin yếu" và thu được kết quả như sau:



6.2. Đánh giá bài nghiên cứu

a. Điểm mạnh

Bài nghiên cứu của nhóm đã thành công xây dựng được mô hình dự đoán lượt đánh giá sản phẩm điện thoại thông minh dựa trên bình luận của người dùng. Bắt đầu từ giai đoạn thu thập, khám phá, làm sạch và chuẩn bị dữ liệu, sau đó mới đến các bước xây dựng mô hình và triển khai. Điều này giúp quá trình phân tích được thực hiện một cách có hệ thống, tránh bỏ sót các bước quan trọng.

Trong khâu làm sạch và tiền xử lý dữ liệu, nhóm đã áp dụng nhiều kỹ thuật để xử lý nhiễu, giá trị bất thường, cũng như chuẩn hóa văn bản tiếng Việt. Điều này cho phép nâng cao chất lượng dữ liệu đầu vào và do đó cải thiện hiệu quả huấn luyện mô hình.

Một điểm mạnh nữa là bài nghiên cứu đã trình bày và so sánh 3 mô hình máy học phổ biến là Naive Bayes, Logistic Regression và LSTM dựa trên các tiêu chí khách quan như độ chính xác, precision, recall và f1-score. Việc này giúp lựa chọn được mô hình phù hợp nhất cho bài toán phân loại đánh giá sản phẩm.

b. Hạn chế

Mặc dù bài nghiên cứu đã đạt được một số kết quả tích cực ban đầu, nhưng vẫn tồn tại một số hạn chế cần khắc phục. Một trong số đó là kích thước mẫu của tập dữ liệu không đồng đều, việc tăng mẫu hoặc giảm mẫu đều có thể gây ảnh hưởng cho bộ dữ liệu. Vì thế thu thập thêm dữ liệu đa dạng sẽ giúp mô hình không bị overfitting và tăng độ chính xác.

Bước tiền xử lý dữ liệu chưa thật sự tốt đối với mô hình học sâu LSTM. Vì thực hiện còn đơn giản nên mô hình đưa ra kết quả hoàn toàn là lớp 5. Đề xuất xây dựng phương pháp tiền xử lý khác như gán nhãn đối với các từ chắc chắn hoặc nhúng qua mạng neural để vector hóa nhằm thu được bộ dữ liệu phù hợp hơn đối với mô hình học sâu hoặc tinh chỉnh các tham số để mô hình chạy tốt hơn.

Đồ án đánh giá phân loại dựa theo sao, nhưng không có các tiêu chí nào để xếp hạng từng cấp độ sao. Vì thế kết quả thu được khá mơ hồ và không được rõ ràng. Đánh giá sao thu được chỉ nhìn được kết quả tổng thể của sản phẩm chứ không nắm được chi tiết từng bộ phận của sản phẩm. Nhất là đối với các sản phẩm có khía cạnh tốt, khía cạnh chưa tốt, hoặc người dùng có yêu cầu đặc biệt đối với một chức năng nào đó của sản phẩm thì đánh giá bằng cấp độ sao không thể đáp ứng được nhu cầu của người tiêu dùng hoặc doanh nghiệp.

c. Định hướng phát triển

Đưa ra các tiêu chí là từng cấp độ sao hoặc phân tích cảm xúc trong văn bản và quan tâm phân loại đến từng khía cạnh như pin, tổng thể, màn hình, camera, … Để có thể có cái nhìn chi tiết với từng khía cạnh của một sản phẩm chứ không chỉ tổng thể sản phẩm được đánh giá bao nhiêu sao để các doanh nghiệp và người tiêu dùng hiểu rõ được ưu và khuyết điểm của sản phẩm.

Sử dụng các mô hình khác, hoặc thay đổi bước tiền xử lý hoặc tinh chỉnh tham số để có kết quả phân lớp tốt hơn. Việc tối ưu hóa thêm siêu tham số và kiến trúc mạng neural cho mô hình LSTM cũng sẽ giúp nâng cao hiệu quả của thuật toán học sâu.

Có thể mở rộng phạm vi sang các ngành hàng tiêu dùng khác điện thoại thông minh, đồng thời xây dựng REST API để dễ dàng tích hợp vào các ứng dụng thực tế của doanh nghiệp.

Phát triển giao dịch người dùng, thêm các tính năng khác như đưa ra các biểu đồ thể hiện số lượng từng cấp độ sao. Hoặc nhập vào URL của một trang bán hàng điện tử và xuất ra bảng báo cáo tổng hợp về thông tin sản phẩm, xếp hạng của từng cấp độ sao, thông tin của từng khía cạnh là tốt hay xấu hay trung tính, …

Tài Liệu Tham Khảo

Hiếu N. V. (2023, September 7). Phân loại văn bản tiếng Việt sử dụng machine learning. Lập Trình Không Khó. https://nguyenvanhieu.vn/phan-loai-van-ban-tieng-viet/

Luc Phan, L., Huynh Pham, P., Thi-Thanh Nguyen, K., Khai Huynh, S., Thi Nguyen, T., Thanh Nguyen, L., Van Huynh, T., & Van Nguyen, K. (2021). SA2SL: From aspect-based sentiment analysis to social listening system for business intelligence. In Knowledge Science, Engineering and Management (pp. 647–658). Springer International Publishing.

Nguyen, K. T.-T., Huynh, S. K., Phan, L. L., Pham, P. H., Nguyen, D.-V., & Van Nguyen, K. (2021). Span detection for aspect-based sentiment analysis in Vietnamese. In arXiv [cs.CL]. http://arxiv.org/abs/2110.07833

Phạm Hữu, Q. (2018, October 14). Phân tích phản hồi khách hàng hiệu quả với Machine learning(Vietnamese Sentiment Analysis). Viblo; Sun* AI Research Team. https://viblo.asia/p/phan-tich-phan-hoi-khach-hang-hieu-qua-voi-machine-learningvietnamese-sentiment-analysis-Eb85opXOK2G

Pham, T. (n.d.). Machinelearningcoban.com. Retrieved December 21, 2023, from https://forum.machinelearningcoban.com/t/chia-se-model-sentiment-analysis-aivivn-com-top-5/4537

Van, C. P. (2020, September 18). Logistic Regression - Bài toán cơ bản trong Machine Learning. Viblo. https://viblo.asia/p/logistic-regression-bai-toan-co-ban-trong-machine-learning-924lJ4rzKPM

Van, H. N. (n.d.). Danh sách một số teencode phổ biến thống kê từ dữ liệu.

Vnstopword.Txt. (n.d.). Box.com. Retrieved December 21, 2023, from https://app.box.com/s/7wlco0lc8z3hi141f0zq

Link Github

[RainbowBunnyyy/Natural-Language-Programming (github.com)](https://github.com/RainbowBunnyyy/Natural-Language-Programming)

Phụ Lục

Tên	MSSV	Đánh giá
Huỳnh Trịnh Tiến Khoa	31211027645	100%
Trương Thanh Phong	31211027662	100%
Lê Trần Khánh Phú	31211024087	100%