# **I. Giới thiệu bài toán.**

Môi trường mạng xã hội mà một môi trường mở, tự do, nơi cảm xúc cá nhân được đề cao, là nơi có thể dễ dàng thu thập những ý kiến của người dùng về một sản phẩm, dịch vụ, chủ đề hay tin tức bất kỳ. Tuy nhiên, một vấn đề tồn tại đối với bất kỳ trang web lớn nào hiện nay là làm thế nào để xử lý nội dung độc hại. Đặc biệt là trong thời kỳ công nghệ đi quá sâu vào đời sống, dữ liệu sản sinh ra quá nhiều.

Quora muốn giải quyết vấn đề này trực tiếp để giữ cho nền tảng của họ trở thành một nơi mà người dùng có thể cảm thấy an toàn khi chia sẻ kiến ​​thức của họ với thế giới. Quora là một nền tảng cho phép mọi người học hỏi lẫn nhau. Trên Quora, mọi người có thể đặt câu hỏi và kết nối với những người khác, những người đóng góp thông tin chi tiết độc đáo và câu trả lời chất lượng. Một thách thức quan trọng là loại bỏ những câu hỏi thiếu chân thành - những câu hỏi được đặt ra dựa trên những tiền đề sai lầm hoặc có ý định đưa ra một tuyên bố hơn là tìm kiếm những câu trả lời hữu ích.

Bài toán đặt ra là làm thế nào để phát hiện câu hỏi là chân thành hay không. Kaggle đã tổ chức cuộc thi này nhằm xây dựng các mô hình học máy giải quyết vấn đề trên.

Phát biểu bài toán: cho một câu hỏi/phát biểu bất kỳ (dạng text, tiếng Anh) xác định nhãn của câu đó là insincere (nhãn 1) hay các trường hợp còn lại (nhãn 0):

* Input: string
* Output: 0 or 1

Với đặc trưng bài toán như trên cùng với việc sử dụng các mô hình học máy để giải quyết, bài toán được phân vào nhóm các bài toán phân loại (classification) trong học máy.

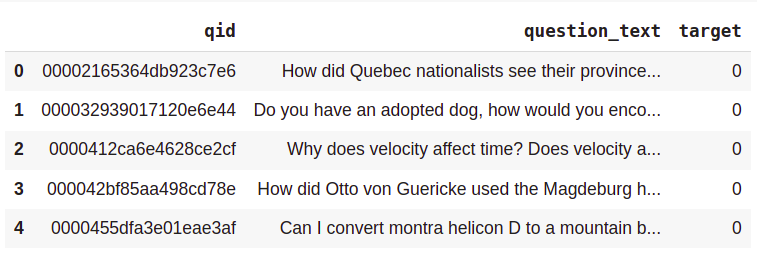
# **II. Về bộ dữ liệu**

Bộ dữ liệu được sử dụng là các dữ liệu được sưu tập từ mạng xã hội Quora và đã được gán nhãn sẵn (0 hoặc 1).

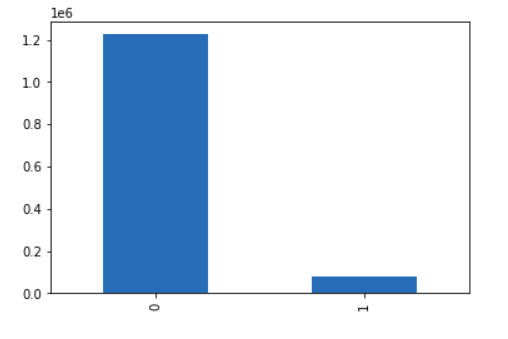
Một số phân tích về bộ dữ liệu:

* Bao gồm 3 trường: qid, question\_text (input), target (nhãn)
* Không bản ghi nào bị khuyết thiếu
* Có 1306122 bản ghi trong tập train.csv

Mẫu dữ liệu:



Phân phối dữ liệu theo nhãn:



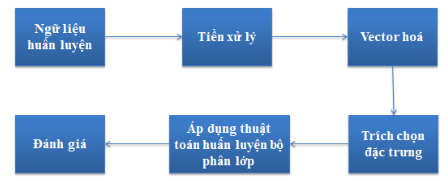
Biểu đồ trên cho thấy dữ liệu mất cân bằng nghiêm trọng, nhãn 0 tỉ lệ quá lớn.

# **III. Phân loại văn bản với các thuật toán machine learning.**

## 1. Bài toán phân loại và pipeline

Với đề tài này chúng ta có thể quy về lớp các bài toán phân loại văn bản. Có rất nhiều cách tiếp cận khác nhau, ở đây em sử dụng các phương pháp học có giám sát. Nhìn chung mô hình sẽ gồm 2 giai đoạn: giai đoạn huấn luyện và giai đoạn phân lớp.

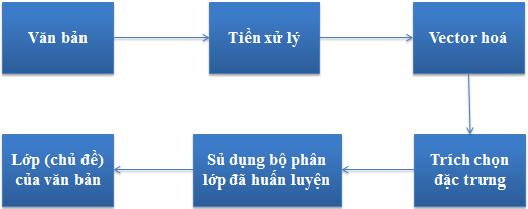
Giai đoạn huấn luyện:



Trong đó:

* Ngữ liệu huấn luyện: dữ liệu đầu vào, chính là các câu của người dùng
* Tiền xử lý: chuyển đổi dữ liệu sao cho phù hợp
* Vector hóa: mã hóa văn bản, chuyển đổi văn bản từ dạng text thành các mô hình trọng số ví dụ như vector.
* Trích chọn đặc trưng: loại bỏ những đặc trưng dư thừa
* Thuật toán huấn luyện: mô hình machine learning mà ta sử dụng
* Đánh giá: đánh giá hiệu suất của bộ phân lớp

Giai đoạn phân lớp:



## 2. Đôi nét về Tf-idf

Tf, viết tắt của thuật ngữ term frequency, một đại lượng chỉ tần số xuất hiện của một từ trong văn bản. Nhưng do văn bản dài ngắn khác nhau nên để chuẩn hóa người ta sử dụng công thức:

Tf(t, d) = (số lần xuất hiện của từ t trong văn bản d)/(tổng số từ trong văn bản d)

Idf, viết tắt của thuật ngữ inverse document frequency, đại lượng dùng để ước lượng mức quan trọng của một từ. Khi ta sử dụng trọng số tf thì mức độ quan trọng của một từ là như nhau. Tuy nhiên có vài từ thường xuyên xuất hiện ở mọi văn bản hay mọi comment thì nó không thể hiện nhiều ý nghĩa của văn bản. Công thức tính:

IDF(t, D) = log(Tổng số văn bản trong tập D / số văn bản có từ t)

Người ta thường sử dụng trọng số tf\*idf là một cách để đánh trọng số từ trong văn bản.

## 3. Tiền xử lý

Dữ liệu trước khi được vector hóa cần được tiền xử lý để loại bỏ bớt nhiễu. Quá trình này giúp nâng cao hiệu suất và giảm độ phức tạp cho thuật toán huấn luyện.

Tùy vào định dạng dữ liệu, mục đích của mô hình phân loại mà chúng ta sẽ có các phương pháp tiền xử lý khác nhau. Ở bài tập này em có các bước tiền xử lý như sau:

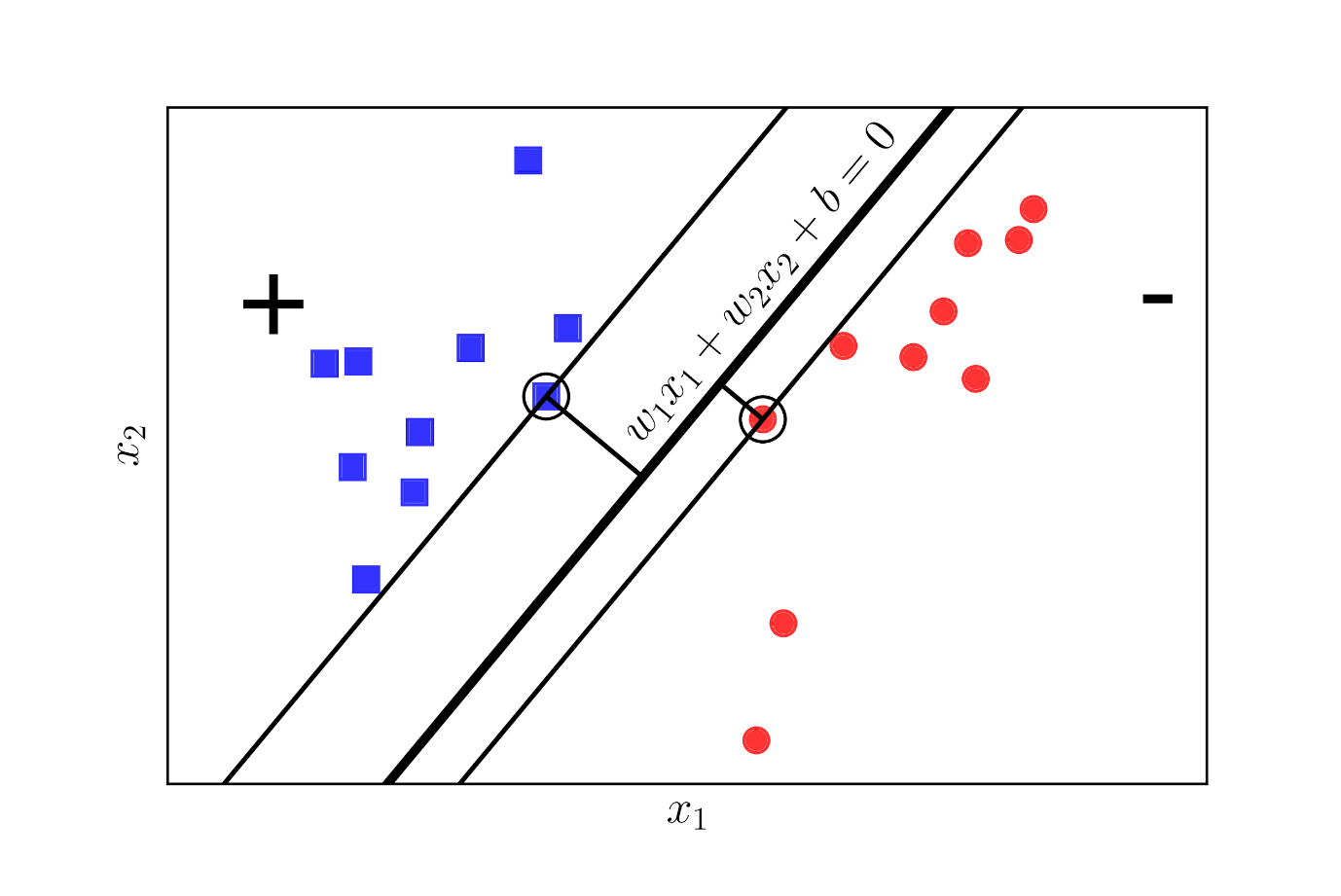
* Đưa các comment về dạng chữ thường, chuẩn utf8
* Loại bỏ hoàn toàn các dấu câu
* Tách từ: đây là một bước thường có trong xử lý text. Tuy nhiên ở bài này, sau khi thử nghiệm nhóm em phát hiện kết quả tách từ thực sự tồi.
* Xóa các thẻ html, các kí tự đặc biệt

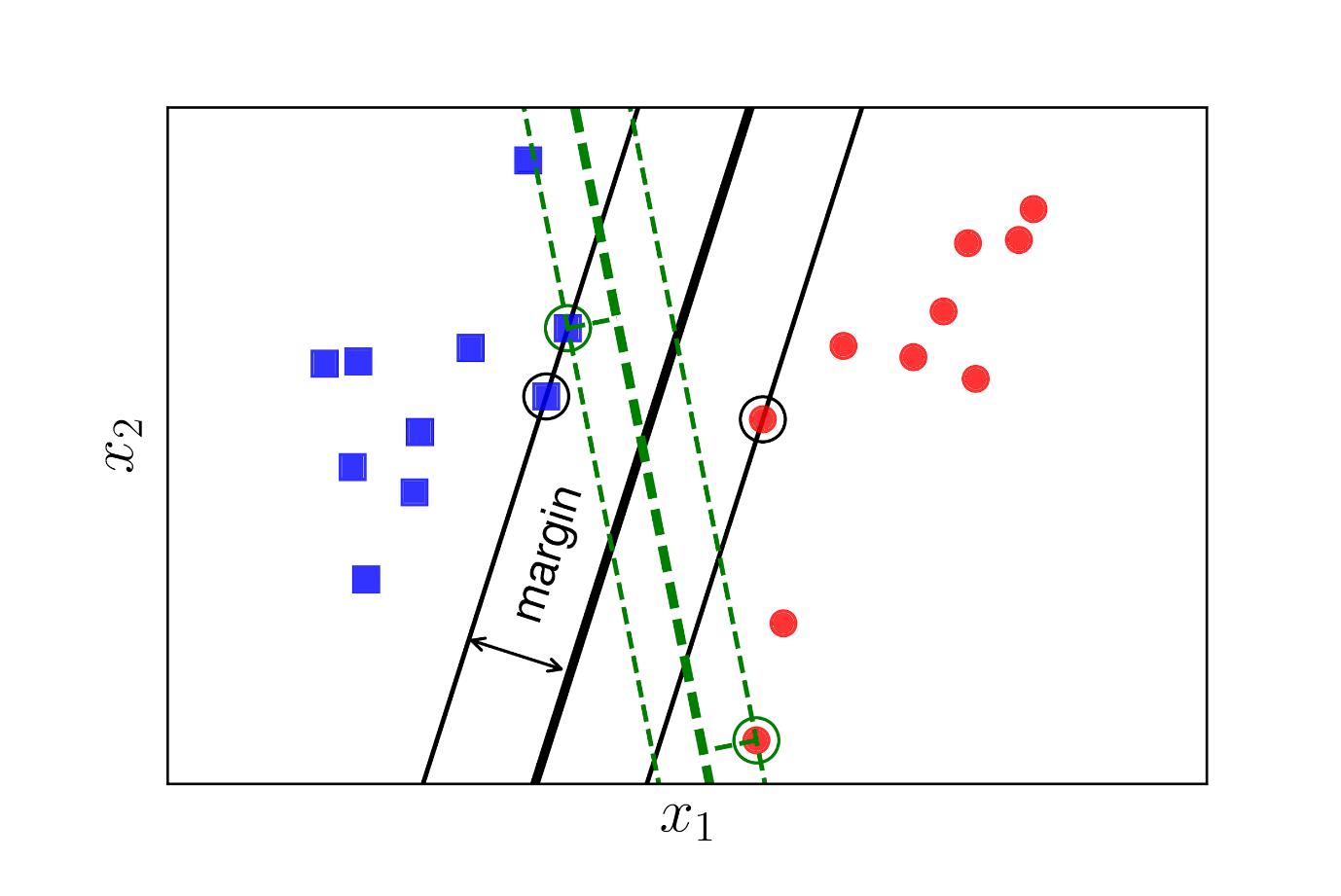
## 4. Các mô hình được sử dụng

### 4.1 Support Vector Machine

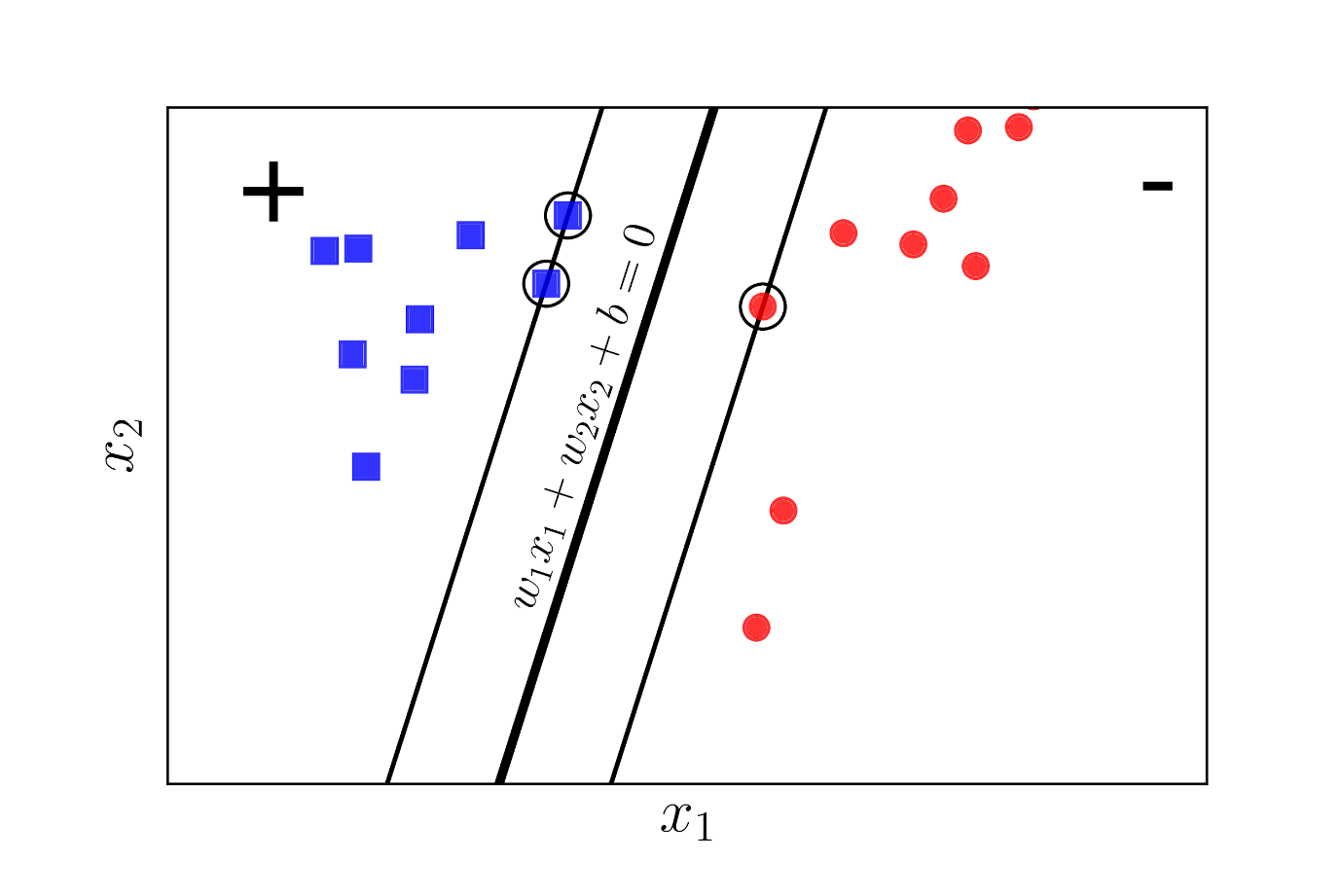
Support Vector Machine (SVM - hay máy vector hỗ trợ) là một mô hình phân lớp dữ liệu được đề xuất bởi Cortes and Vapnik vào năm 1995.

Ý tưởng chính của nó là cho trước một tập huấn luyện, được biểu diễn trong không gian vector hữu hạn chiều, mô hình sẽ cố gắng tìm ra một siêu phẳng (hyperplane) tốt nhất chia các điểm dữ liệu thành 2 lớp ( có thể nhiều lớp hơn, nhưng để đơn giản chúng ra sẽ cố định là lớp), giả sử là pos và neg. Chất lượng của siêu phẳng này được quyết định bởi khoảng cách (gọi là biên - margin) của điểm dữ liệu gần nhất của mỗi lớp tới siêu phẳng. Khi đó, khoảng cách biên càng lớn thì mô hình chúng ta có được càng tốt. Bên dưới là hình minh họa.





Sau khi mô hình tìm được siêu phẳng tối ưu:



Khi đó, giả sử w là bộ trọng số hay siêu phẳng ta tìm được. Để xác định lớp của dữ liệu chúng ta sử dụng hàm sau:

Trong đó sgn là hàm dấu, nhận giá trị 1 nếu không âm (pos) và là -1 nếu âm (neg), còn b là bias.

Bài toán tối ưu cho SVM là một bài toán đối ngẫu.

### 4.2. Naive Bayes

Naive Bayes là một thuật toán phân lớp bằng xác suất dựa trên công thức xác suất naive bayes.

Giả sử với các lớp ta có là C như: 1, 2, ..., C. Có một điểm dữ liệu . Ta tính xác suất để điểm dữ liệu này rơi vào class c theo công thức sau:

Từ công thức này ta tính được xác suất đầu ra là c với đầu vào là x. Để có thể xác định class cho điểm dữ liệu x ta sẽ đi tìm c sao cho cách chọn ra class c có xác suất là cao nhất.

với

Biểu thức trên thường khó tính trực tiếp. Vì vậy người ta đã áp dụng công thức Bayes được:

với

Ta rút gọn được p(x) vì p(x) không phụ thuộc vào c. P(c) có thể coi là xác suất một điểm rơi vào c, điều này ta có thể tính toán bằng công thức:

p(c) = (số dữ liệu có class là c / tổng số dữ liệu)

Thành phần cần tính toán còn lại là p(x | c), thường thì rất khó tính vì x là biến ngẫu nhiên nhiều chiều, rất khó để tìm được phân phối phù hợp. Để cho đơn giản thì người ta coi các thành phần của biến ngẫu nhiên x là độc lập với nhau. Khi đó ta có:

Các công thức trên là toàn bộ tư tưởng chung của thuật toán Naive Bayes, khác nhau ở chỉ ở chỗ tìm các phân phối để tính toán xác suất trên. Thực tế và đặc biệt là đối với dữ liệu dạng text thì các thành phần của x luôn luôn liên quan đến nhau để tạo ra ngữ cảnh. Nhưng kết quả mà thuật toán naive bayes mang lại rất khả quan (điều này cũng giải thích cho chữ naive - ngây thơ).

Phân phối xác suất thường dùng cho các bài toán đa lớp là multinomial naive bayes. Thuật toán này chính là mô hình ngôn ngữ bậc 1. Rất thú vị !

# **IV. Triển khai và đánh giá mô hình**

Em sử dụng thư viện sklearn (scikit-learn) cho việc xây dựng pipeline và Google Colab cho việc training model.

Do đây là một bài toán phân lớp dữ liệu không cân bằng nên độ đo chúng em sử dụng để tối ưu hyperparameter f1 score. Đồng thời cũng chỉ sử dụng tập Train.csv có sẵn mà không submit lên kaggle nên để kiểm tra mô hình em chia tập train thành 3 thập dữ liệu theo tỉ lệ sau:

* Train: 70% dùng để huấn luyện mô hình
* Dev: 10% dùng để tìm siêu tham số (hyperparameter)
* Test: 20% dùng để đánh giá mô hình

Với Multinomial Naive Bayes: accuracy = 95%

|  | precision | recall | f-score | support |
| --- | --- | --- | --- | --- |
| 0 | 0.97 | 0.97 | 0.97 | 245202 |
| 1 | 0.58 | 0.54 | 0.56 | 16023 |
| macro avg | 0.77 | 0.76 | 0.77 | 261225 |
| weighted avg | 0.9 | 0.9 | 0.88 | 261225 |

Tham số tối ưu:

alpha = 0.01

max\_df (cho Tfidf transform): kết quả gần như không phụ thuộc vào max\_df, là giá trị dùng để loại bỏ các từ xuất hiện quá nhiều trong dữ liệu (từ dừng)

ngram\_range = (1, 4)

Với SVM: accuracy = 95%

|  | precision | recall | f-score | support |
| --- | --- | --- | --- | --- |
| 0 | 0.98 | 0.97 | 0.97 | 245218 |
| 1 | 0.59 | 0.70 | 0.63 | 16007 |
| macro avg | 0.78 | 0.83 | 0.8 | 261225 |
| weighted avg | 0.96 | 0.95 | 0.95 | 6011 |

Sau khi thử nghiệm heuristic với các bộ tham số khác nhau, em rút ra nhận xét:

* Tham số C của mô hình SVM không ảnh hưởng nhiều đến kết quả
* Tham số max\_df cũng không ảnh hưởng quá nhiều
* ngram\_range của mô hình Tfidf transform giúp cải thiện kết quả

Bộ tham số tối ưu:

* max\_df: 0.7
* ngram\_range: 1-4