THỰC HÀNH 1: PHÂN LỚP VĂN BẢN CƠ BẢN

1. Naive Bayes cơ bản.

Bộ dữ liệu 1:

```
train = [
    ["just plain boring", "-"],
    ["entirely predictable and lacks energy", "-"],
    ["no surprises and very few laughs", "-"],
    ["very powerful", "+"],
    ["the most fun film of the summer", "+"]
]

test = "predictable with no fun"
```

Mỗi câu được tổ chức thành 1 list gồm 2 phần tử: s = train[0]

Phần tử thứ 1: là câu văn bản. s[0]

Phần tử thứ 2: là nhãn "+" hoặc là "-". s[1]

Hãy hiện thực Naive bayes cơ bản theo từng bước như trong bài giảng đã học

Các bước thực hiện:

+ **Bước 1**: Xây dựng **tập** từ vựng V

+ **Bước 2**: Xây dựng **danh sách** từ vựng cho các lớp + và -.

+ **Bước 3**: Xây dựng tập nhãn (y_train).

+ **Bước 4**: Tính xác suất cho 2 lớp: + và -.

Gợi ý: Sử dụng hàm count() trong python.

+ **Bước 5**: Tách các từ trong văn bản test ra thành danh sách các từ theo khoảng trắng.

Gợi ý: Sử dụng hàm split().

+ **Bước 6**: Tính xác suất theo các lớp "+" và lớp "-" lần lượt cho các từ trong test.

Lưu ý: bỏ đi từ "with".

+ **Bước 7**: So sánh kết quả xác suất của 2 lớp và kết luận nhãn.

Hiện thực với thư viện sklearn.

Các bạn sử dụng MultinomialNB và CountVectorizer để biến đổi dữ liệu.

Các bước thực hiện:

+ **Bước 1**: Xây dựng X_train, y_train và X_test từ bộ dữ liệu ban đầu.

Gợi ý: X_train là danh sách các văn bản, y_train là danh sách các nhãn.

+ **Bước 2**: Mã hoá dữ liệu:

2.1: Mã hoá văn bản: Dùng CountVectorizer để đếm tần suất xuất hiện các từ.

2.2: Mã hoá nhãn: Dùng LabelEncoder để mã hoá nhãn thành số.

VD: + mã hoá thành 1, - mã hoá thành 0

+ **Bước 3**: Áp dụng bộ phân lớp MultinomialNB để huấn luyện mô hình.

+ **Bước 4**: Dự đoán cho câu dữ liệu test.

Gợi ý: dùng hàm predict.

+ **Bước 5**: Nhãn dự đoán bởi mô hình sẽ là 0 hoặc 1 (theo mã hoá ở bước 2). Do đó, để chuyển lại thành nhãn ban đầu, ta dùng *inverse_transform* trong hàm *LabelEncoder*.

2. N-gram.

Mặc định, CountVectorizer sẽ sử dụng 1-gram. Khi sử dụng 1 gram, CountVectorizer sẽ đếm tần suất cho các trường hợp như sau:

```
['and',
 'boring',
 'energy',
 'entirely',
 'few',
 'film',
 'fun',
 'just',
 'lacks',
 'laughs',
 'most',
 'no',
 'of',
 'plain',
 'powerful',
 'predictable',
 'summer',
 'surprises',
 'the',
 'very']
```

Khi sử dụng 2-gram, ta cần khai báo thêm 2 tham số:

+ **analyzer**='word'

+ **ngram_range**=(2, 2).

Khi sử dụng 2-gram, CountVectorizer sẽ đếm tần suất cho các trường hợp như sau:

```
['and lacks',
 'and very',
 'entirely predictable',
 'few laughs',
 'film of',
 'fun film',
 'just plain',
 'lacks energy',
 'most fun',
 'no surprises',
 'of the',
 'plain boring',
 'predictable and',
 'surprises and',
 'the most',
 'the summer',
 'very few',
 'very powerful']
```

Hãy thử nghiệm lại bài toán phân tích cảm xúc trên với CountVectorizer là 2-gram. Kết quả dự đoán ra có giống như với 1-gram hay không?

Bên cạnh *CountVectorizer* dùng để đếm tần suất xuất hiện của từng phần từ, còn có kỹ thuật *TfIdfVectorizer* dùng để tính trọng số của từng phần tử trong câu theo TF-IDF (xem thêm về phương pháp tính trọng số TF-IDF).

Bài tập: Hãy hiện thực bộ phân lớp để dự đoán kết quả cho bộ dữ liệu 2 như sau:

```
train = [
    ["sản_phẩm A rất tốt.", "+"],
    ["tôi không thích sản_phẩm A.", "-"],
    ["màu_sắc của A thật tốt.", "+"],
    ["sản_phẩm A thật kinh_khủng.", "-"],
]

test = [
    "sản_phẩm A khá tốt."
    "màu_sắc quá kinh_khủng."
]
```

BÀI TẬP VẬN DỤNG:

Bài 1: Hiện thực lại mô hình Naive Bayes bằng tay cho Bộ dữ liệu 1 và Bộ dữ liệu 2.

Bài 2: Hiện thực lại mô hình Naive Bayes bằng thư viện sklearn cho Bộ dữ liệu 1 và Bộ dữ liệu 2. (Sử dụng *CountVectorizer* 1-gram)

BÀI TẬP NGHIÊN CỨU:

Bộ dữ liệu: **UIT-VSFC**

Công bố khoa học: K. V. Nguyen, V. D. Nguyen, P. X. V. Nguyen, T. T. H. Truong and N. L. Nguyen, "***UIT-VSFC: Vietnamese Students' Feedback Corpus for Sentiment Analysis***", KSE 2018.

Số lượng dữ liệu: khoảng hơn 16,000 câu.

Số lượng nhãn: 3 nhãn gồm tích cực (*positive*), tiêu cực (*negative*) và trung tính (*neural*).

Sử dụng cho tác vụ **sentiment-based** trong bộ dữ liệu.

Link:

https://drive.google.com/drive/folders/1xclbjHHK58zk2X6iqbvMPS2rcy9y9E0X

Yêu cầu:

1) Hiện thực 2 mô hình phân lớp văn bản: **Naive Bayes** và **Logistic Regression** với 2 kỹ thuật *CountVectorizer* và *TfIdfVectorizer*.

2) So sánh hiệu năng của 2 mô hình trên các độ đo: *accuracy* và *macro-f1 score*. Sử dụng *ma trận nhầm lẫn (confuse matrix)* để biện luận khả năng dự đoán trên từng nhãn.

Gợi ý:

Đọc dữ liệu UIT-VSFC:

```
import pandas as pd

X_train = pd.read_csv(UIT-VSFC/train/sents.txt',  sep='\n',
header=None, index_col=None)
y_train = pd.read_csv('UIT-VSFC/train/sentiments.txt',
sep='\n', header=None, index_col=None)
```

```
X_dev = pd.read_csv('UIT-VSFC/dev/sents.txt', sep='\n',
header=None, index_col=None)
y_dev = pd.read_csv('UIT-VSFC/dev/sentiments.txt', sep='\n',
header=None, index_col=None)

X_test = pd.read_csv('UIT-VSFC/test/sents.txt',  sep='\n',
header=None, index_col=None)
y_test = pd.read_csv('UIT-VSFC/test/sentiments.txt',
sep='\n', header=None, index_col=None)

y_train = y_train.values.flatten()
y_dev = y_dev.values.flatten()
y_test = y_test.values.flatten()
```

Hiển thị Confusion metrics.

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sn


cf = confusion_matrix(y_true, y_pred)
sn.heatmap(cf, annot=True, cmap="Greys",fmt='g', cbar=True,
annot_kws={"size": 30})
```