AI VIET NAM – AI COURSE 2024

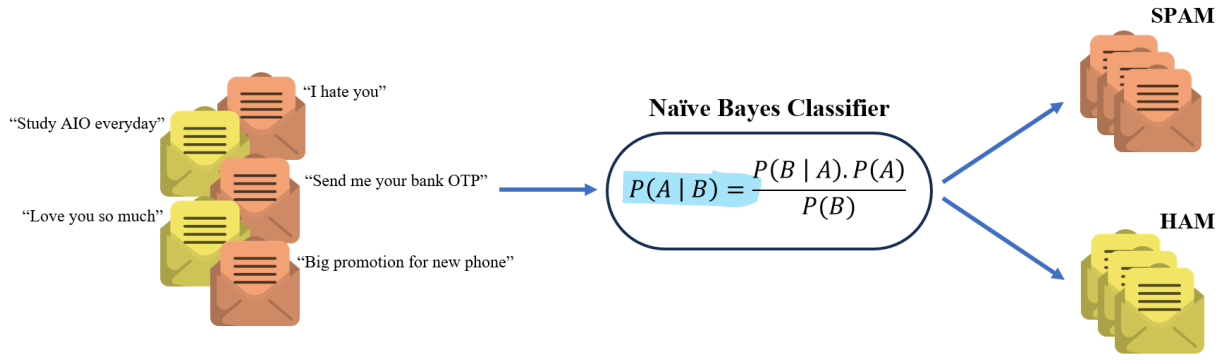
Phân loại tin nhắn spam với Naive Bayes

Dinh-Thang Duong, Anh-Khoi Nguyen, Quang-Vinh Dinh

Ngày 2 tháng 8 năm 2024

I. Giới thiệu

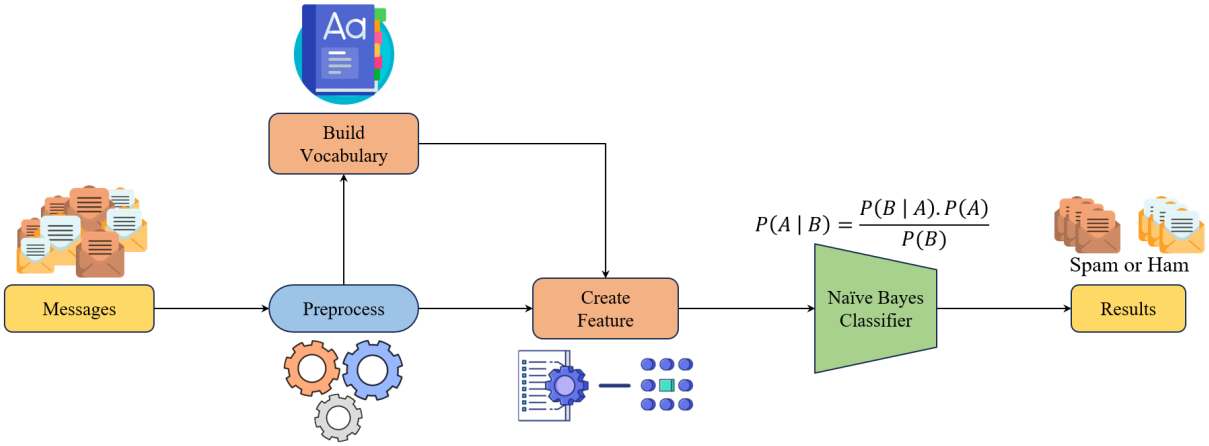
Text Classification **(Tạm dịch: Phân loại văn bản)** là một trong những bài toán phổ biến trong lĩnh vực Machine Learning và Natural Language Processing. Trong đó, nhiệm vụ của chúng ta là xây dựng một chương trình có khả năng phân loại văn bản vào các phân lớp do chúng ta quy định. Các ứng dụng phổ biến liên quan đến loại chương trình này có thể kể đến phát hiện các bình luận tiêu cực trên không gian mạng, các đánh giá tích cực của sản phẩm...



Trong project này, chúng ta sẽ xây dựng một chương trình Text Classification liên quan đến việc phân loại một đoạn tin nhắn là tin nhắn spam hay không, sử dụng thuật toán Naive Bayes. Theo đó, Input/Output của chương trình bao gồm:

- **Input:** Một đoạn tin nhắn (text).
- **Output:** Có là tin nhắn spam hay không (bool).

Dựa vào các thông tin trên, ta sẽ xây dựng được một luồng xử lý (pipeline) cho bài toán này như sau:



Theo đó, với bộ dữ liệu có nhãn về tin nhắn spam hoặc không spam, chúng ta sẽ đưa qua một số bước tiền xử lý dữ liệu để tách ra các đặc trưng và nhãn tương ứng. Khi đã chuẩn bị bộ dữ liệu cho việc huấn luyện, ta thực hiện xây dựng mô hình Naive Bayes Classifier. Cuối cùng, sử dụng mô hình Naive Bayes đã huấn luyện được, ta có thể dự đoán một tin nhắn bất kì có là spam hay không. Như vậy, chương trình trong project của chúng ta đã hoàn tất.

II. Cài đặt chương trình

Trong phần này, ta sẽ tiến hành cài đặt chương trình phân loại tin nhắn rác hay không với Naive Bayes. Chương trình được cài đặt trên Google Colab. Các bước thực hiện như sau:

1. **Tải bộ dữ liệu:** Đầu tiên, chúng ta cùng nhìn qua bộ dataset sẽ được sử dụng trong bài này thông qua bảng sau:

Category	Message
ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got a...
ham	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entr...
spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it s...
...	...

   Quan sát thấy bộ dữ liệu sẽ gồm có 2 cột:

   1. **Category:** gồm 2 nhãn là *Ham* và *Spam*, với ý nghĩa như sau:
      - *Ham:* Là những tin nhắn bình thường, không có mục đích quảng cáo hoặc lừa đảo hoặc nói cách khác là người nhận mong muốn nhận được.
      - *Spam:* Là những tin nhắn không mong muốn, thường có mục đích quảng cáo sản phẩm, dịch vụ, hoặc lừa đảo.
   2. **Message:** là những nội dung bên trong một Messages.

   Nhiệm vụ của chúng ta là dựa vào nội dung Message để phân loại nhị phân với Naive Bayes, để xem xét rằng, liệu với nội dung như thế này thì Message đó là *Spam* hay *Ham*. Để huấn luyện mô hình Naive Bayes giải quyết bài toán này, ta cần tải bộ dữ liệu này về máy. Chúng ta có thể tải tại đây hoặc trực tiếp tại trang Kaggle của dataset này tại đây. Trong python, với đường dẫn google drive của bộ dữ liệu, ta có thể dùng lệnh sau đây để tải về một cách tự động về máy:

```
# https://drive.google.com/file/d/1N7rk-kfnDFIGMeX0ROVTjKh71gcgx-7R/view?usp=sharing
!gdown --id 1N7rk-kfnDFIGMeX0ROVTjKh71gcgx-7R
```

2. **Import các thư viện cần thiết:** Trước tiên, chúng ta cùng điểm qua một số như viện chính được sử dụng trong bài:

   - **string:** Cung cấp các hàm cơ bản để thao tác với chuỗi ký tự.
   - **nltk (Natural Language Toolkit):** Một trong những thư viện xử lý ngôn ngữ tự nhiên phổ biến nhất trong Python.
   - **pandas:** Cung cấp các cấu trúc dữ liệu hiệu quả và các công cụ để làm việc với dữ liệu.

- **numpy:** Cung cấp các đối tượng mảng đa chiều và các hàm toán học để làm việc với các mảng này.
- **scikit-learn:** Thư viện học máy phổ biến, giúp xây dựng và triển khai các mô hình học máy phức tạp một cách nhanh chóng.

Hình 1: Logo của một số bộ dữ liệu kể trên.

Trong môi trường code, các bạn thực thi đoạn code sau:

```
import string
import nltk
nltk.download('stopwords')
nltk.download('punkt')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
```

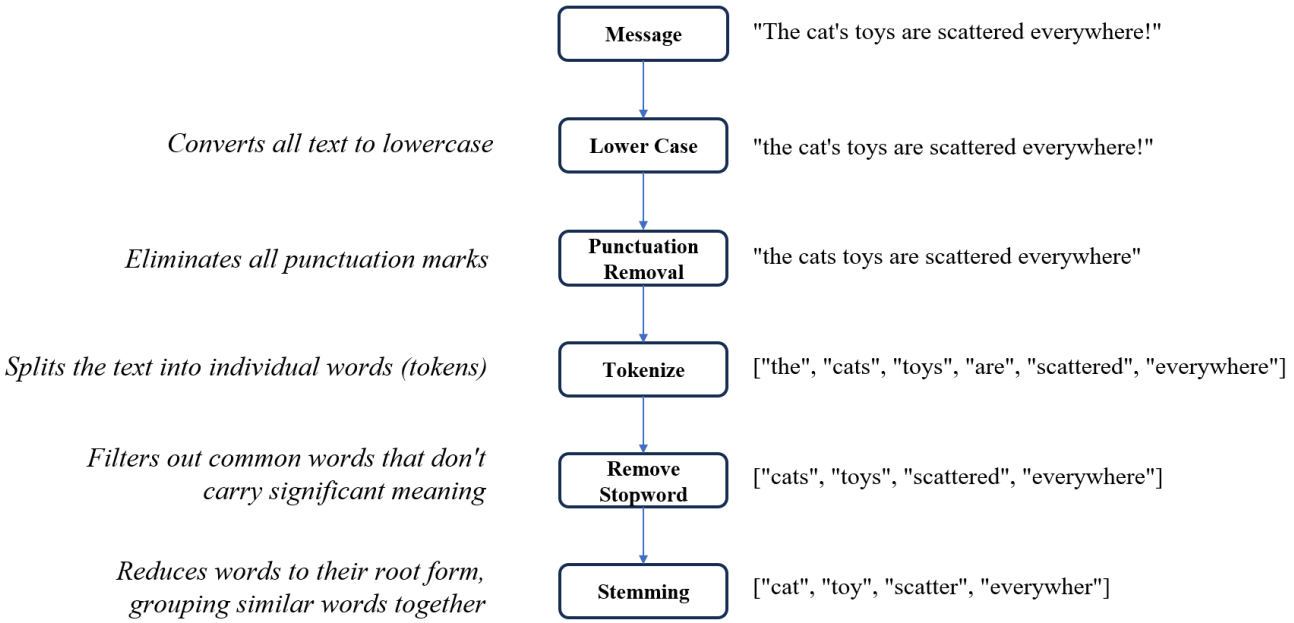
3. **Đọc dữ liệu:** Để đọc bộ dữ liệu có dạng là một file .csv, chúng ta sẽ dùng thư viện **pandas**. Để tách riêng biệt phần đặc trưng và nhãn, sau khi có dataframe, chúng ta đọc và lưu trữ dữ liệu của từng cột vào 2 biến tương ứng *messages* và *labels*:

```
DATASET_PATH = '/content/2cls_spam_text_cls.csv'
df = pd.read_csv(DATASET_PATH)

messages = df['Message'].values.tolist()
labels = df['Category'].values.tolist()

```

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...

4. **Tiền xử lý dữ liệu:**

- **Tiền xử lý dữ liệu đặc trưng:** Như chúng ta đã biết, nội dung của các Messages (như thư/SMS/email...) vô cùng nhiều và đa dạng các tổ hợp từ hoặc ký tự với nhau. Trong các trường hợp đặc biệt, chúng có thể là những từ viết tắt *(như "'m, 's, 're, ...)*, từ không mang nhiều ý nghĩa *(như các dấu câu (.,'/)*, các biến thể của từ *(ví dụ: change, changing, changes, changed, changer, ...)* và rất nhiều trường hợp đặc biệt khác. Vì vậy, bước Tiền xử lý dữ liệu là vô cùng quan trọng với các tác vụ liên quan đến ngôn ngữ. Với bài này, chúng ta sẽ đối phó với chúng với một số hàm đơn giản như hình sau:

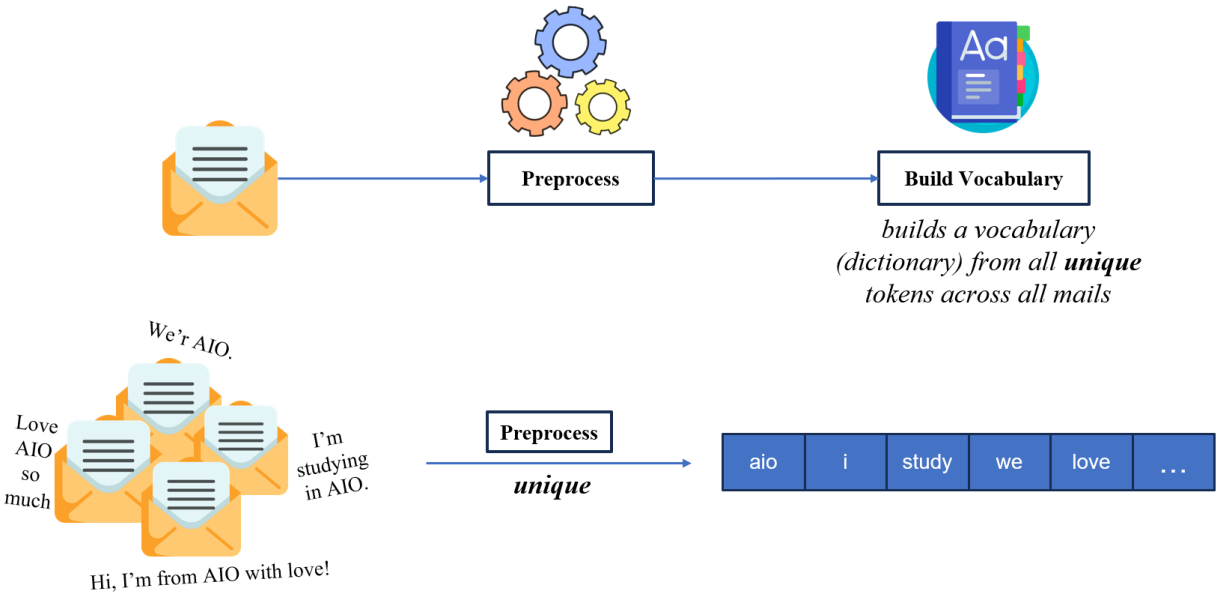


Tương ứng với đoạn code sau:

```python
def lowercase(text):
    return text.lower()

def punctuation_removal(text):
    translator = str.maketrans('', '', string.punctuation)

    return text.translate(translator)

def tokenize(text):
    return nltk.word_tokenize(text)

def remove_stopwords(tokens):
    stop_words = nltk.corpus.stopwords.words('english')

    return [token for token in tokens if token not in stop_words]

def stemming(tokens):
    stemmer = nltk.PorterStemmer()

    return [stemmer.stem(token) for token in tokens]
```

```python

def preprocess_text(text):
    text = lowercase(text)
    text = punctuation_removal(text)
    tokens = tokenize(text)
    tokens = remove_stopwords(tokens)
    tokens = stemming(tokens)

    return tokens

def preprocess_text(text):
    text = lowercase(text)
    text = punctuation_removal(text)
    tokens = tokenize(text)
    tokens = remove_stopwords(tokens)
    tokens = stemming(tokens)

    return tokens

messages = [preprocess_text(message) for message in messages]

```

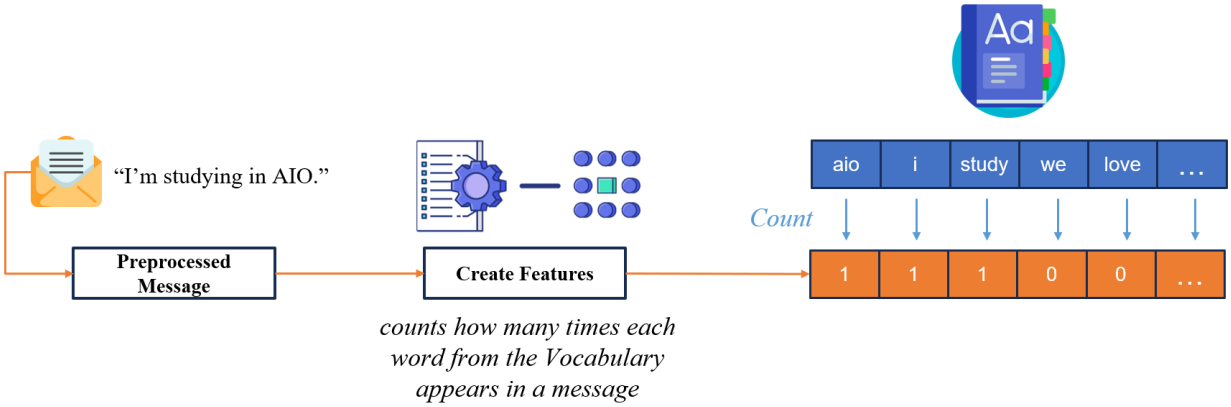
Tiếp theo, chúng ta cần tạo một bộ từ điển (Dictionary), chứa tất các từ hoặc ký tự có xuất hiện trong toàn bộ Messages sau khi được tiền xử lý và không tính trùng lặp.




```python
def create_dictionary(messages):
    dictionary = []

    for tokens in messages:
        for token in tokens:
            if token not in dictionary:
                dictionary.append(token)

    return dictionary
```

```
dictionary = create_dictionary(messages)
```

Kế đến, chúng ta cần tạo ra những đặc trưng đại diện cho thông tin (là các từ) của các Message. Một trong những các đơn giản chính là dựa vào tần suất xuất hiện của từ. Với mỗi Message, vector đại diện sẽ có kích thước bằng với số lượng từ có trong Dictionary.




```
def create_features(tokens, dictionary):
    features = np.zeros(len(dictionary))

    for token in tokens:
        if token in dictionary:
            features[dictionary.index(token)] += 1

    return features

X = np.array([create_features(tokens, dictionary) for tokens in
    messages])
```

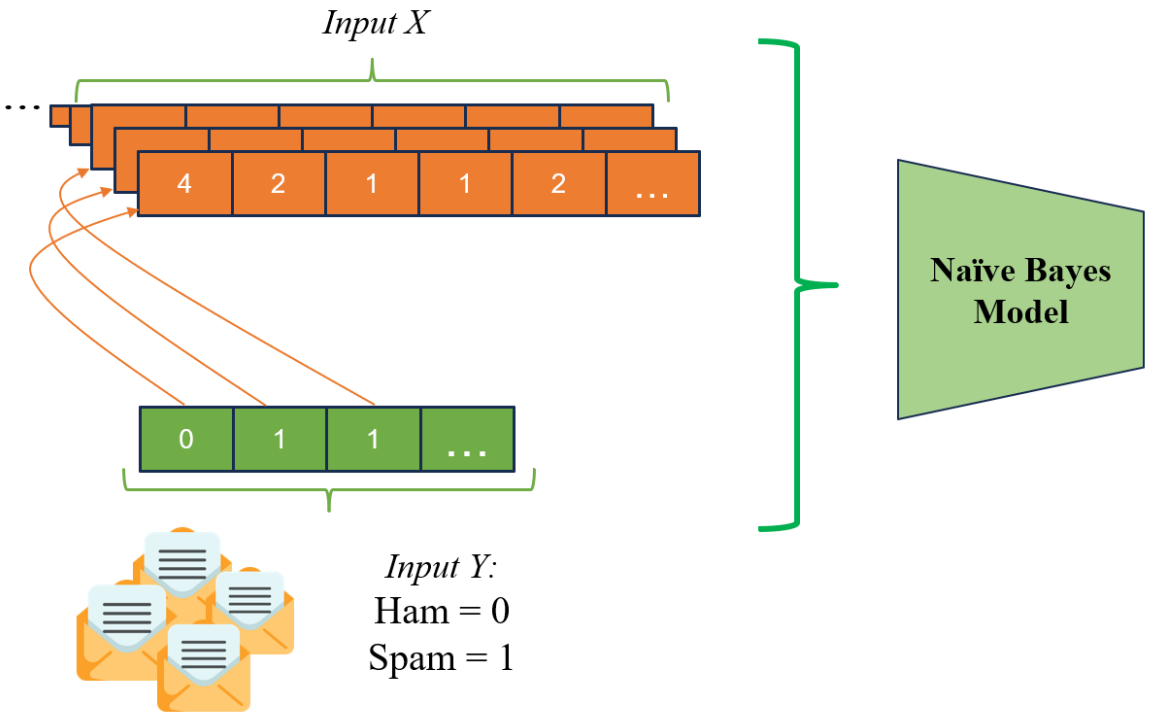
- **Tiền xử lý dữ liệu nhãn:** Đối với nhãn của bài toán này, chúng ta sẽ xử lý đơn giản bằng cách chuyển 2 nhãn *ham* và *spam* thành các con số *0* và *1* để máy tính có thể hiểu, cụ thể như sau:

```
le = LabelEncoder()
y = le.fit_transform(labels)
print(f'Classes: {le.classes_}')
print(f'Encoded labels: {y}')

# >> Classes: ['ham' 'spam']
# >> Encoded labels: [0 0 1 ... 0 0 0]

```

5. **Chia bộ dữ liệu train/val/test:** Khi tiến hành huấn luyệnh một mô hình machine learning, chúng ta sẽ không lấy toàn bộ bộ dữ liệu hiện có để huấn luyện mà sẽ tách ra làm 3 bộ khác nhau. Tại bước này, chúng ta chia bộ dữ liệu thành 3 phần: Train, Validation và Test theo tỉ lệ lần lượt là 7/2/1 (trên tỉ lệ 100% của bộ dữ liệu gốc). Ngoài ra, chúng ta thêm tham số *SEED* để duy trì kết quả giống nhau sau mỗi lần chạy lại.

```
VAL_SIZE = 0.2
TEST_SIZE = 0.125
SEED = 0

X_train, X_val, y_train, y_val = train_test_split(X, y,
                                                  test_size=VAL_SIZE,
                                                  shuffle=True,
                                                  random_state=SEED)

X_train, X_test, y_train, y_test = train_test_split(X_train, y_train,
                                                    test_size=TEST_SIZE,
                                                    shuffle=True,
                                                    random_state=SEED)

```

6. **Huấn luyện mô hình:** Sau các bước trên, chúng ta đã tạo ra 2 Input cần thiết, công việc bây giờ chỉ cần truyền chúng vào mô hình Gaussian Naive Bayes và tiến hành huấn luyện bằng các hàm trong thư viện *sklearn*.



Theo đó, chúng ta thực thi đoạn code sau để khởi tạo và thực hiện việc huấn luyện mô hình Naive Bayes:

```
model = GaussianNB()
```

```
print('Start training...')
model = model.fit(X_train, y_train)
print('Training completed!')

# >> Start training...
# >> Training completed!
# >> CPU times: user 397 ms, sys: 162 ms, total: 559 ms
# >> Wall time: 633 ms

```

7. **Đánh giá mô hình:** Sau khi huấn luyện, chúng ta đến phần đánh giá hiệu suất của mô hình. Bắt đầu với việc cho mô hình đã huấn luyện dự đoán trên tập Validation và Test. Sau đó, sử dụng độ đo *Accuracy Score* để đánh giá mô hình.

```
y_val_pred = model.predict(X_val)
y_test_pred = model.predict(X_test)

val_accuracy = accuracy_score(y_val, y_val_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)

print(f'Val accuracy: {val_accuracy}')
print(f'Test accuracy: {test_accuracy}')

# >> Val accuracy: 0.8816143497757848
# >> Test accuracy: 0.8602150537634409

```

8. **Thực hiện dự đoán:** Cuối cùng, để sử dụng mô hình cho các Message mới, chúng ta sẽ phải thực lại các công đoạn *Tiền xử lý, tạo đặc trưng* cho Message mới này và truyền vào mô hình Naive Bayes. Lúc này, mô hình sẽ trả về giá trị 0 hoặc 1, do đó, cần gọi hàm `inverse_transform()` để chuyển đổi lại về nhãn ban đầu là Ham hoặc Spam.

```
def predict(text, model, dictionary):
    processed_text = preprocess_text(text)
    features = create_features(text, dictionary)
    features = np.array(features).reshape(1, -1)
    prediction = model.predict(features)
    prediction_cls = le.inverse_transform(prediction)[0]

    return prediction_cls

test_input = 'I am actually thinking a way of doing something useful'
prediction_cls = predict(test_input, model, dictionary)
print(f'Prediction: {prediction_cls}')

# >> Prediction: ham

```

III. Câu hỏi trắc nghiệm

1. Naive Bayes là một thuật toán học máy thuộc loại nào?
   (*a*) Học có giám sát.
   (*b*) Học không giám sát.
   (*c*) Học bán giám sát.
   (*d*) Học tăng cường.

2. Trong Naive Bayes, "Naive"(ngây thơ) ám chỉ điều gì?
   (*a*) Thuật toán đơn giản và dễ hiểu.
   (*b*) Giả định về tính độc lập của các đặc trưng.
   (*c*) Thuật toán chỉ hoạt động với dữ liệu đơn giản.
   (*d*) Thuật toán không hiệu quả bằng các phương pháp khác.

3. Naive Bayes thường được sử dụng trong các bài toán nào?
   (*a*) Hồi quy tuyến tính.
   (*b*) Phân cụm.
   (*c*) Phân loại văn bản.
   (*d*) Tối ưu hóa.

4. Trong Naive Bayes, P(X|Y) được gọi là gì?
   (*a*) Xác suất tiên nghiệm.
   (*b*) Xác suất hậu nghiệm.
   (*c*) Hàm hợp lý.
   (*d*) Xác suất biên.

5. Khi nào Naive Bayes có thể hoạt động kém hiệu quả?
   (*a*) Khi có nhiều đặc trưng độc lập.
   (*b*) Khi có quá nhiều dữ liệu huấn luyện.
   (*c*) Khi các đặc trưng có mối tương quan mạnh.
   (*d*) Khi dữ liệu có phân phối chuẩn.

6. Công thức nào sau đây thể hiện đúng nguyên lý của Naive Bayes?
   (*a*) P(Y|X) = P(X|Y) * P(Y) / P(X)
   (*b*) P(X|Y) = P(Y|X) * P(X) / P(Y)
   (*c*) P(X,Y) = P(X|Y) * P(Y)
   (*d*) P(Y) = P(X|Y) * P(Y|X) / P(X)

7. Tại sao cần tiền xử lý dữ liệu trong bài toán phân loại email?

(a) Để tăng kích thước dữ liệu.

(b) Để giảm nhiễu và chuẩn hóa dữ liệu.

(c) Để thay đổi nội dung email.

(d) Để mã hóa email.

8. Chọn đáp án đúng cho đoạn code sau:

```
def remove_stopwords(tokens):
stop_words = nltk.corpus.stopwords.words('english')
return [token for token in tokens if token not in stop_words]

input_text = ["Pho", "is", "a", "popular", "Vietnamese", "noodle", "soup"
    ]
result = remove_stopwords(input_text)
print(result)
```

(a) `["Pho", "is", "a", "popular", "Vietnamese", "noodle", "soup"]`

(b) `["Pho", "popular", "Vietnamese", "noodle", "soup"]`

(c) `["is", "a"]`

(d) `["Pho", "Vietnamese", "noodle", "soup"]`

9. Trong tiền xử lý văn bản, việc loại bỏ stopwords có tác dụng gì?

(a) Tăng độ chính xác của mô hình.

(b) Giảm kích thước từ điển và loại bỏ các từ ít quan trọng.

(c) Thêm ngữ cảnh cho văn bản.

(d) Tăng số lượng đặc trưng.

10. Phương pháp nào được sử dụng để tạo vector đặc trưng cho mỗi email trong đoạn code?

(a) TF-IDF.

(b) Word2Vec.

(c) Đếm tần suất xuất hiện của từ.

(d) One-hot encoding.

11. Đáp án nào miêu tả đầy đủ và chính xác nhất đoạn code sau ?

```
def create_features(tokens, dictionary):
    features = np.zeros(len(dictionary))
    for token in tokens:
        if token in dictionary:
            features[dictionary.index(token)] += 1

    return features
```

(a) Tạo vector cho từng message, mỗi phần tử tương ứng với thứ tự các từ trong dictionary.

(b) Tạo vector có kích thước bằng với dictionary cho từng message, mỗi phần tử tương ứng với thứ tự các từ trong dictionary

(*c*) Tạo vector có kích thước bằng với dictionary, mỗi phần tử tương ứng với thứ tự các từ trong dictionary và được gán giá trị bằng 1 nếu xuất hiện trong message.

(*d*) Tạo vector có kích thước bằng với dictionary, mỗi phần tử tương ứng với thứ tự các từ trong dictionary và được gán giá trị bằng tần suất của từ đó trong message.