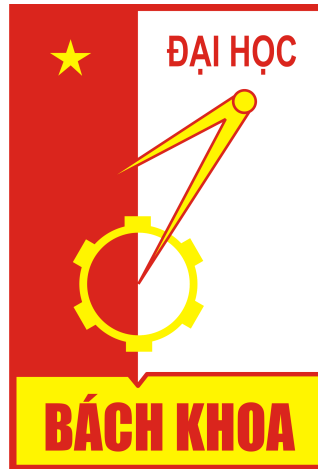


**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
**VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC**



**BÀI TẬP LỚN**  
**PHÂN LOẠI THƯ RÁC SỬ DỤNG**  
**THUẬT TOÁN NAIVE BAYES**

Học phần: Hệ hỗ trợ quyết định

Mã lớp: 142300

Giảng viên:	TS. Lê Hải Hà
Sinh viên thực hiện:	Vũ Thành Đạt
MSSV:	20206275
Lớp:	Hệ thống thông tin 01-K65

**HÀ NỘI, 07/2023**

## Lời mở đầu

Trong thời đại kỹ thuật số, ứng dụng của tin nhắn ngày càng nhiều. Kéo theo đó là tình trạng các tin nhắn rác (SMS spam) ngày càng tăng. Các tin nhắn rác không chỉ gây phiền hà cho người nhận, mà còn có thể công cụ lừa đảo. Do đó, cần có một công cụ, thuật toán để đánh dấu tin nhắn có phải là tin nhắn rác hay không.

Bài báo cáo này trình bày các bước sử dụng thuật toán Naive Bayes để phân loại thư rác (SMS spam).

Cấu trúc bài báo cáo gồm 3 phần:

- Có sở lý thuyết về thuật toán Navie Bayes.
- Đánh giá bộ dữ liệu và tiền xử lý dữ liệu.
- Ứng dụng thuật Naive Bayes phân loại thư rác

Trong quá trình làm báo cáo, em đã nghiên cứu, tìm hiểu và vận dụng những kiến thức mà thầy đã trang bị trong quá trình học tập. Tuy nhiên do kiến thức còn hạn chế nên báo cáo này không thể tránh khỏi những thiếu sót, em rất mong nhận được những ý kiến đóng góp từ thầy để có thể hoàn thiện hơn.

Qua đây, em xin gửi lời cảm ơn sâu sắc tới thầy Lê Hải Hà, giảng viên Trường Đại học Bách Khoa Hà nội, đã tận tình hướng dẫn cho em để em có thể hoàn thành báo cáo này.

Em xin chân thành cảm ơn!

*Hà Nội, 31 tháng 07 năm 2023*

Sinh viên

**Vũ Thành Đạt**

# Mục lục

<b>Chương 1</b>	<b>Thuật toán Navie Bayes</b>	<b>1</b>
<b>Chương 2</b>	<b>Đánh giá bộ dữ liệu và tiền xử lý dữ liệu</b>	<b>3</b>
2.1	Bộ dữ liệu . . . . .	3
2.2	Thống kê dữ liệu . . . . .	4
2.3	Định dạng . . . . .	4
2.4	Extract dữ liệu . . . . .	5
2.5	Trực quan hóa dữ liệu ham và spam theo số lượng ký tự, từ, câu .	6
2.6	Tiền xử lý dữ liệu . . . . .	7
2.6.1	Cài đặt các thư viện xử lý văn bản . . . . .	7
2.6.2	Xử lý dữ liệu văn bản . . . . .	7
2.6.3	Sửa nhãn từ 'spam' và 'ham' thành 1 và 0 . . . . .	10
2.6.4	Trực quan dữ liệu sau khi xử lý bằng <i>Word cloud</i> . . . . .	11
2.6.5	So sánh tổng số ký tự, từ trong tin nhắn spam và ham (hoặc không phải spam) . . . . .	12
<b>Chương 3</b>	<b>Ứng dụng thuật Naive Bayes phân loại thư rác</b>	<b>14</b>
3.1	Xây dựng mô hình . . . . .	14
3.1.1	Phương pháp Bag of Words . . . . .	14
3.1.2	Định dạng input và output . . . . .	14
3.1.3	Tách tập train và test . . . . .	14
3.1.4	Xây dựng mô hình bằng hàm MultinomialNB() của thư viện 'sklearn.naive_bayes': . . . . .	15
3.2	Đánh giá mô hình . . . . .	15
<b>Kết luận</b>		<b>17</b>
<b>Tài liệu tham khảo</b>		<b>18</b>

# Chương 1

## Thuật toán Navie Bayes

Naive Bayes là một thuật toán phân loại được mô hình hoá dựa trên định lý Bayes trong xác suất thống kê:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

trong đó:

- $P(y|X)$  gọi là posterior probability: xác suất của mục tiêu  $y$  với điều kiện có đặc trưng  $X$ .
- $P(X|y)$  gọi là likelihood: xác suất của đặc trưng  $X$  khi đã biết mục tiêu  $y$ .
- $P(y)$  gọi là prior probability của mục tiêu  $y$ .
- $P(X)$  gọi là prior probability của đặc trưng  $X$

Ở đây,  $X$  là vector các đặc trưng, có thể viết dưới dạng:

$$X = (x_1, x_2, x_3, \dots, x_n)$$

Khi đó đẳng thức trở thành:

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y) \dots P(x_n|y)P(y)}{P(x_1)P(x_2) \dots P(x_n)}$$

Trong mô hình Naive Bayes, có hai giả thiết được đặt ra:

- Các đặc trưng đưa vào mô hình là độc lập với nhau. Tức là sự thay đổi giá trị của một đặc trưng không ảnh hưởng đến các đặc trưng còn lại.
- Các đặc trưng đưa vào mô hình có ảnh hưởng ngang nhau đối với đầu ra mục tiêu.

Khi đó, kết quả mục tiêu  $y$  để  $P(y|X)$  đạt cực đại trở thành:

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

Chính vì hai giả thiết gần như không tồn tại trong thực tế trên, mô hình này mới được gọi là naive (ngây thơ). Tuy nhiên, chính sự đơn giản của nó với việc dự đoán rất nhanh kết quả đầu ra khiến nó được sử dụng rất nhiều trong thực tế trên những bộ dữ liệu lớn, đem lại kết quả khả quan. Một vài ứng dụng của Naive Bayes có thể kể đến như: lọc thư rác, phân loại văn bản, dự đoán sắc thái văn bản, ...

## Chương 2

# Đánh giá bộ dữ liệu và tiền xử lý dữ liệu

### 2.1 Bộ dữ liệu

SMS Spam Collection v.1 là một tập hợp các tin nhắn được gắn thẻ SMS đã được thu thập để nghiên cứu Spam SMS. Tập dữ liệu chứa các tin nhắn SMS bằng tiếng Anh gồm 5.572 tin nhắn, được gắn thẻ là ham (hợp pháp) hoặc spam. Bộ dữ liệu được lấy trên <https://archive.ics.uci.edu/ml/datasets/sms+spam+collection>. Được từ nhiều nguồn trên Internet, cụ thể là:

- Gồm 425 tin nhắn rác SMS được trích xuất thủ công từ trang web Grumbletext. Đây là một diễn đàn của Vương quốc Anh, trong đó người dùng điện thoại di động tuyên bố công khai về tin nhắn rác SMS, hầu hết trong số họ không báo cáo về tin nhắn rác đã nhận được. Trang web Grumbletext là: <http://www.grumbletext.co.uk/> .
- Một tập hợp con gồm 3.375 tin nhắn SMS của NUS SMS Corpus (NSC), là tập hợp của khoảng 10.000 tin nhắn hợp pháp được thu thập để nghiên cứu tại Khoa Khoa học Máy tính của Đại học Quốc gia Singapore. <http://www.comp.nus.edu.sg/~rpnlpir/downloads/corpora/smsCorpus/>
- Danh sách 450 tin nhắn ham SMS được thu thập từ Luận án Tiến sĩ của Caroline Tag có tại <http://etheses.bham.ac.uk/253/1/Tagg09PhD.pdf>.

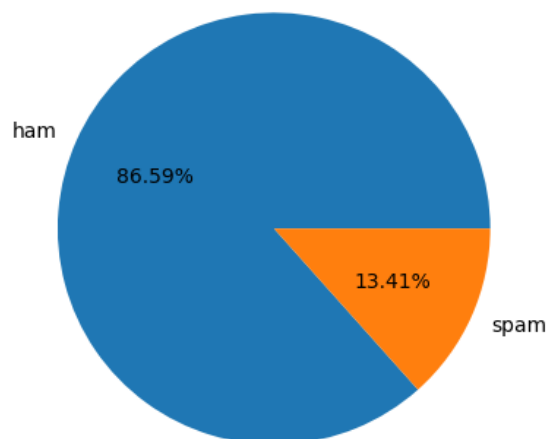
- Số lượng 1.002 tin nhắn ham SMS và 322 tin nhắn rác được trích xuất từ SMS Spam Corpus v.0.1 Big do José María Gómez Hidalgo tạo ra và được công khai tại: <http://www.esp.uem.es/jmgomez/smsspamcorp>

## 2.2 Thống kê dữ liệu

```
# Thống kê về tập dữ liệu
df.label.value_counts()

ham      4825
spam      747
Name: label, dtype: int64
```

Tập dữ liệu có tổng số 4.825 tin nhắn SMS hợp pháp (86,59%) và tổng số 747 (13,41%) tin nhắn rác.



Hình 2.1: Thống kê dữ liệu

## 2.3 Định dạng

Tập dữ liệu chứa một thông báo trên mỗi dòng. Mỗi dòng bao gồm hai cột: một cột label chứa nhãn (ham hoặc spam) và một cột message chứa văn bản nội dung tin nhắn. Trong đó tin nhắn không được sắp xếp theo thứ tự thời gian.

```

      label      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...
...      ...
5567  spam  This is the 2nd time we have tried 2 contact u...
5568  ham                        Will ü b going to esplanade fr home?
5569  ham  Pity, * was in mood for that. So...any other s...
5570  ham  The guy did some bitching but I acted like i'd...
5571  ham                        Rofl. Its true to its name

[5572 rows x 2 columns]

```

## 2.4 Extract dữ liệu

Đầu tiên, ta trích xuất các thông tin sau:

- Tổng số ký tự
- Tổng số từ
- Tổng số câu

```

df['num_char']=df['message'].apply(len)
df['num_words']=df['message'].apply(lambda x: len(str(x).split()))
df['num_sen']=df['message'].apply(lambda x: len(nltk.sent_tokenize(x)))
df

```

	label	message	num_char	num_words	num_sen
0	ham	Go until jurong point, crazy.. Available only ...	111	20	2
1	ham	Ok lar... Joking wif u oni...	29	6	2
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155	28	2
3	ham	U dun say so early hor... U c already then say...	49	11	1
4	ham	Nah I don't think he goes to usf, he lives aro...	61	13	1
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	160	30	4
5568	ham	Will ü b going to esplanade fr home?	36	8	1
5569	ham	Pity, * was in mood for that. So...any other s...	57	10	2
5570	ham	The guy did some bitching but I acted like i'd...	125	26	1
5571	ham	Rofl. Its true to its name	26	6	2

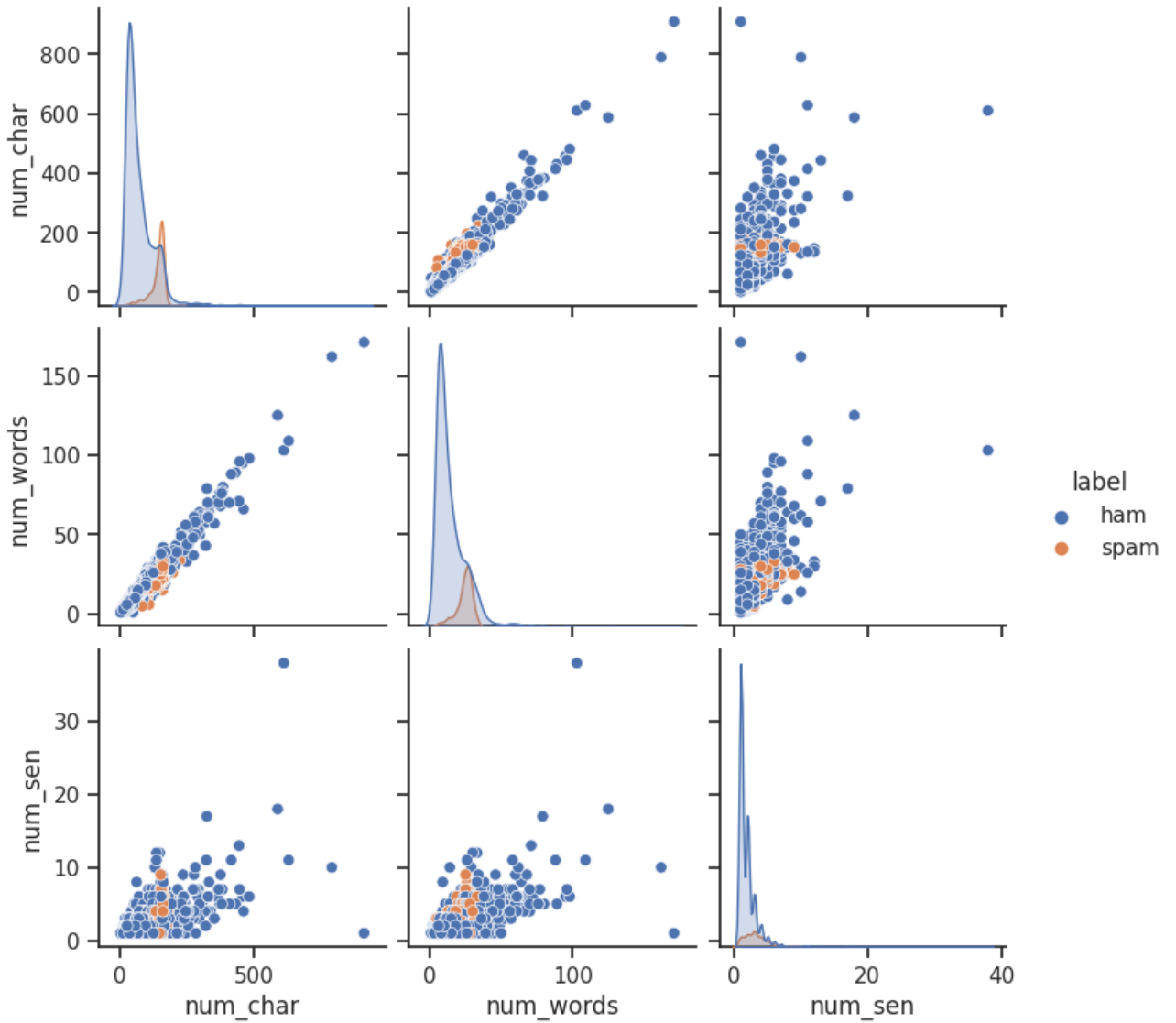
5572 rows x 5 columns



## 2.5 Trực quan hóa dữ liệu ham và spam theo số lượng ký tự, từ, câu

```
sns.pairplot(df, hue='label')  
plt.show()
```

Kết quả hiển thị ra mà hình:



Hình 2.2: Trực quan hóa dữ liệu

## 2.6 Tiền xử lý dữ liệu

### 2.6.1 Cài đặt các thư viện xử lý văn bản

```
# Import lib required for text processing
nltk . download ( ' stopwords ' )
from nltk . corpus import stopwords
stopwords . words ( ' english ' )
from nltk . stem import PorterStemmer
from wordcloud import WordCloud
import string , time
string . punctuation
stopwords . words ( ' english ' )
```

### 2.6.2 Xử lý dữ liệu văn bản

Trước khi tiến hành phân loại, việc tiền xử lý dữ liệu đầu vào là rất quan trọng để đảm bảo kết quả phân loại chính xác. Với dữ liệu dạng chuỗi, việc xử lý trước là cực kỳ cần thiết để phục vụ cho việc trích xuất các đặc trưng và phân loại.

Có nhiều phương pháp khác nhau để xử lý dữ liệu chuỗi, tùy thuộc vào từng bộ dữ liệu và mục đích phân tích. Dưới đây là một số công đoạn xử lý dữ liệu em lựa chọn trong đề tài này:

- Loại bỏ các liên kết web
- Loại bỏ email
- Loại bỏ chuỗi số

Tạo các hàm loại bỏ:

```
def remove_website_links(text):
    no_website_links = text.replace(r"http\S+", "")
    return no_website_links

def remove_numbers(text):
    removed_numbers = text.replace(r'\d+', '')
    return removed_numbers
```

```
def remove_emails(text):
    no_emails = text.replace(r"\S*@\\S*\\s?", '')
    return no_emailsmessage'].apply(lambda x: len(nltk.sent_tokenize(x)))

df

df['message'] = df['message'].apply(remove_website_links)
df['message'] = df['message'].apply(remove_numbers)
df['message'] = df['message'].apply(remove_emails)
```

- **Chuyển về chữ thường:** Bước này bao gồm chuyển toàn bộ văn bản về dạng chữ thường. Việc này là cần thiết để đảm bảo tính đồng nhất trong dữ liệu văn bản. Ví dụ, từ "Hello" và "hello" sẽ được xem như cùng một từ.
- **Phân đoạn thành từng từ:** Phân đoạn (tokenization) là quá trình chia nhỏ văn bản thành từng từ hoặc token riêng lẻ. Điều này giúp chia nhỏ văn bản thành các đơn vị có nghĩa. Ví dụ, câu "I love dogs" sẽ được phân đoạn thành ba token riêng biệt: "I", "love" và "dogs".
- **Loại bỏ các ký tự đặc biệt:** Các ký tự đặc biệt như dấu câu, ký hiệu hoặc bất kỳ ký tự không phải chữ hoặc số nào thường được loại bỏ khỏi văn bản. Bước này được thực hiện để loại bỏ nhiễu và tập trung vào nội dung văn bản chính.
- **Loại bỏ các từ dừng(stopwords):** Các từ dừng (stop words) là các từ thông thường như "the", "is" hoặc "and" không mang ý nghĩa đáng kể trong bối cảnh phân tích văn bản. Việc loại bỏ các từ dừng giúp giảm số chiều của dữ liệu và cải thiện hiệu suất tính toán. Ngoài ra, việc loại bỏ các dấu câu giúp làm sạch văn bản hơn và loại bỏ bất kỳ ký tự không cần thiết nào.
- **Rút gọn từ:** Rút gọn từ (stemming) là quá trình giảm từ về dạng gốc hoặc hình thức cơ bản nhất của nó. Nó bao gồm việc loại bỏ các hậu tố và biến đổi để thu được ý nghĩa cốt lõi của từ. Ví dụ, việc rút gọn từ sẽ chuyển đổi các từ như "running", "runs" và "ran" về dạng cơ bản chung của chúng, là "run". Bước này giúp chuẩn hóa các từ và giảm kích thước từ vựng.

Xây dựng hàm xử lý dữ liệu:

```
def transform_text(text):  
    #1.lower casing  
    text=text.lower()  
  
    #2.tokenization  
    lst=nlTK.word_tokenize(text)  
  
    #3.remove special characters stopwords and punctuation  
    l1=[]  
    useless_words=stopwords.words('english')+list(string.punctuation)  
    for word in lst:  
        if word.isalnum()==True and word not in useless_words:  
            l1.append(word)  
  
    #4.stemming  
    l2=[]  
    for word in l1:  
        ps=PorterStemmer()  
        l2.append(ps.stem(word))  
  
    return " ".join(l2).strip()  
    l1.clear()  
    l2.clear()
```

Dưới đây là giải thích chi tiết từng lệnh và cách sử dụng các hàm:

- Đầu tiên sử dụng phương thức `lower()` để chuyển toàn bộ văn bản trong biến `text` thành chữ thường để đảm bảo tính đồng nhất trong văn bản.
- Sử dụng thư viện `nlTK` để phân đoạn văn bản thành các từ riêng biệt bằng hàm `word_tokenize()`. Kết quả trả về là một danh sách các từ.
- Tiếp theo xử lý danh sách các từ đã phân đoạn bằng cách loại bỏ các ký tự đặc biệt, từ dừng và dấu câu không cần thiết. Để làm điều này, đầu tiên tạo một danh sách rỗng `l1` để lưu trữ các từ đã được lọc. Sau đó, danh sách `useless_words` được tạo ra bằng cách kết hợp danh sách các từ dừng và danh sách các ký tự đặc biệt và dấu câu. Vòng lặp `for` được sử dụng để

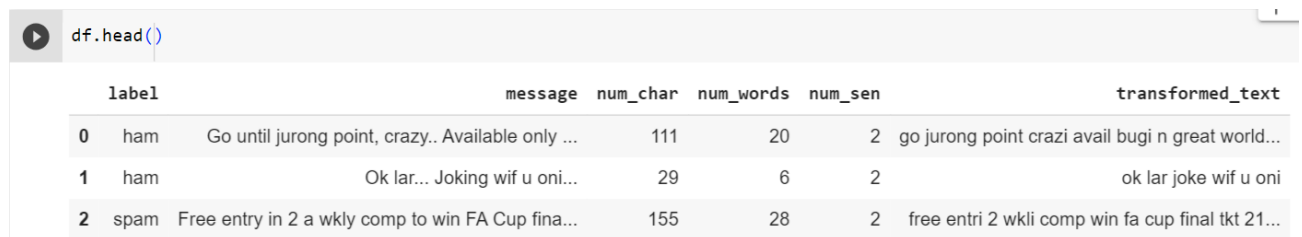
kiểm tra từng từ trong danh sách `lst` đã được phân đoạn và chỉ thêm vào danh sách `l1` nếu từ không có chứa ký tự đặc biệt, là từ dừng hoặc là dấu câu.

- Sử dụng thuật toán rút gọn từ (stemming) bằng cách sử dụng thuật toán `PorterStemmer` để đưa các từ trong danh sách `l1` về dạng gốc. Kết quả thu được sẽ được lưu trữ trong danh sách `l2`.
- Sử dụng phương thức `join()` để ghép các từ trong danh sách `l2` thành một chuỗi duy nhất, cách nhau bởi một khoảng trắng. Sau đó, hàm `strip()` được sử dụng để loại bỏ khoảng trắng thừa ở hai đầu chuỗi. Cuối cùng, danh sách `l1` và `l2` được xóa bằng cách sử dụng phương thức `clear()` để giải phóng bộ nhớ.

Gọi hàm:

```
df['transformed_text'] = df['message'].apply(transform_text)
```

Một số kết quả hiển thị sau khi xử lý:



	label	message	num_char	num_words	num_sen	transformed_text
0	ham	Go until jurong point, crazy.. Available only ...	111	20	2	go jurong point crazy avail bugi n great world...
1	ham	Ok lar... Joking wif u oni...	29	6	2	ok lar joke wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155	28	2	free entri 2 wkli comp win fa cup final tkt 21...

### 2.6.3 Sửa nhãn từ 'spam' và 'ham' thành 1 và 0

```
df['label'] = df['label'].replace({'spam': 0, 'ham': 1})
df
```

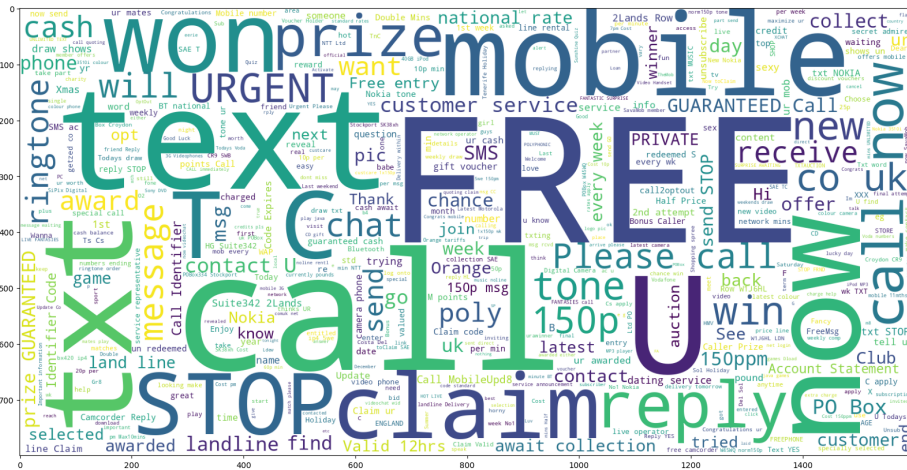
5572 rows × 6 columns

#### 2.6.4 Trực quan dữ liệu sau khi xử lý bằng *Word cloud*

- Ham:



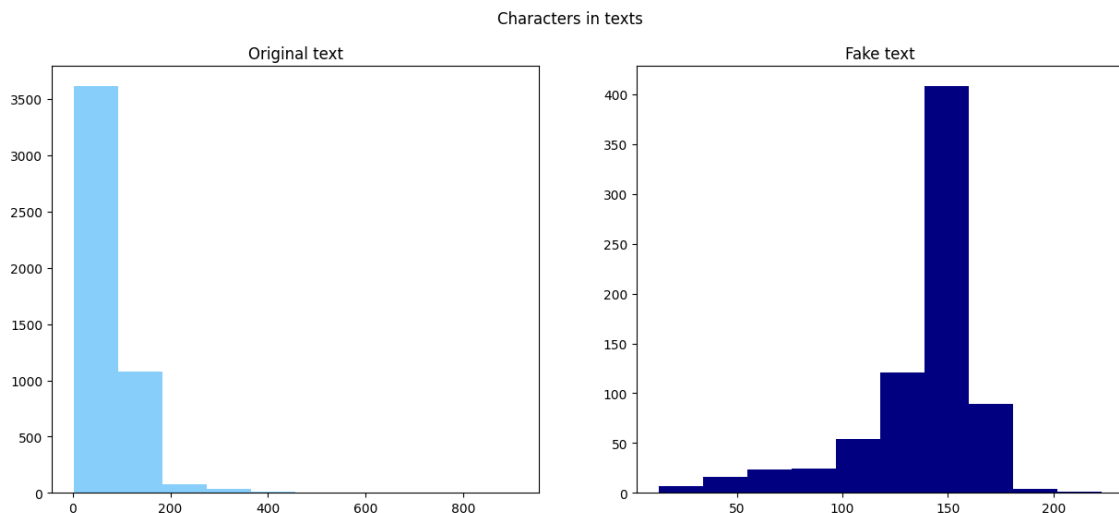
- Spam:



### 2.6.5 So sánh tổng số ký tự, từ trong tin nhắn spam và ham (hoặc không phải spam)

- So sánh ký tự

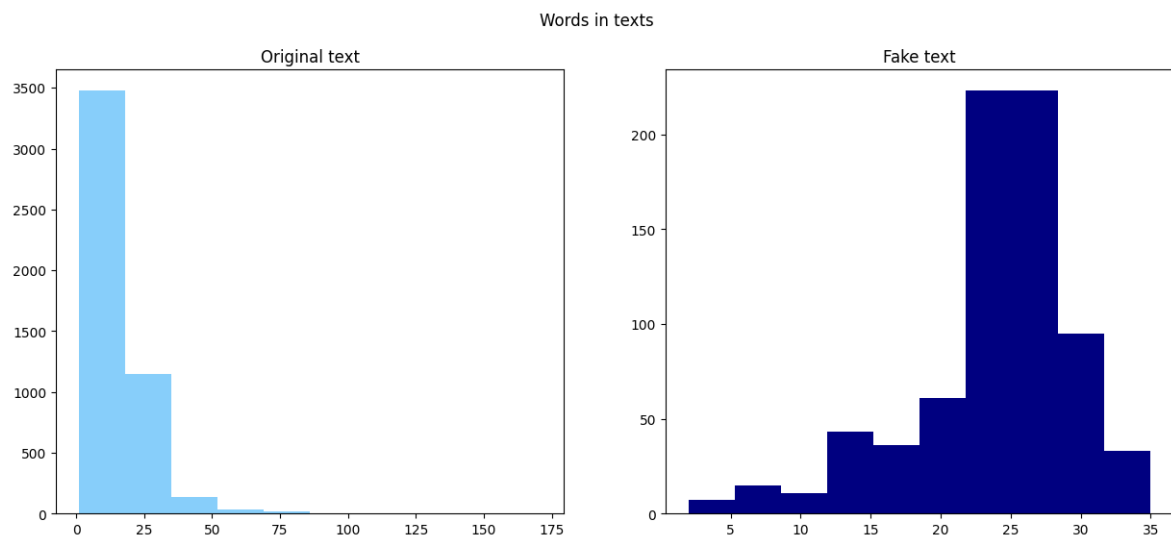
```
fig, (ax1, ax2)=plt.subplots(1,2,figsize=(15,6))
text_len=df[df['label']==1]['message'].str.len()
ax1.hist(text_len,color='lightskyblue')
ax1.set_title('Original text')
text_len=df[df['label']==0]['message'].str.len()
ax2.hist(text_len,color='navy')
ax2.set_title('Fake text')
fig.suptitle('Characters in texts')
plt.show()
```



Hình 2.4: So sánh ký tự

- So sánh từ

```
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(15,6))
text_len=df[df['label']==1]['num_words_transform']
ax1.hist(text_len,color='lightskyblue')
ax1.set_title('Original text')
text_len=df[df['label']==0]['num_words_transform']
ax2.hist(text_len,color='navy')
ax2.set_title('Fake text')
fig.suptitle('Words in texts')
plt.show()
```



Hình 2.5: So sánh từ



## Chương 3

# Ứng dụng thuật Naive Bayes phân loại thư rác

### 3.1 Xây dựng mô hình

#### 3.1.1 Phương pháp Bag of Words

Phương pháp Bag of Words (BoW) sẽ chuyển các từ, các câu, đoạn văn ở dạng text của văn bản về một vector mà mỗi phần tử là một số. BoW học được một bộ từ vựng từ tất cả các văn bản rồi mô hình các văn bản bằng cách đếm số lần xuất hiện của mỗi từ trong văn bản đó. Bag of Words không quan tâm đến thứ tự từ trong câu và cũng không quan tâm đến ngữ nghĩa của từ. Ở đây, ta sử dụng thư viện có sẵn 'sklearn' để thực hiện phương pháp này.

```
from sklearn.feature_extraction.text import CountVectorizer,
    TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

#### 3.1.2 Định dạng input và output

```
X = tfidf.fit_transform(df['message']).toarray()
y = df['label'].values
```

#### 3.1.3 Tách tập train và test

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,
        random_state=2)
```

### 3.1.4 Xây dựng mô hình bằng hàm MultinomialNB() của thư viện 'sklearn.naive\_bayes':

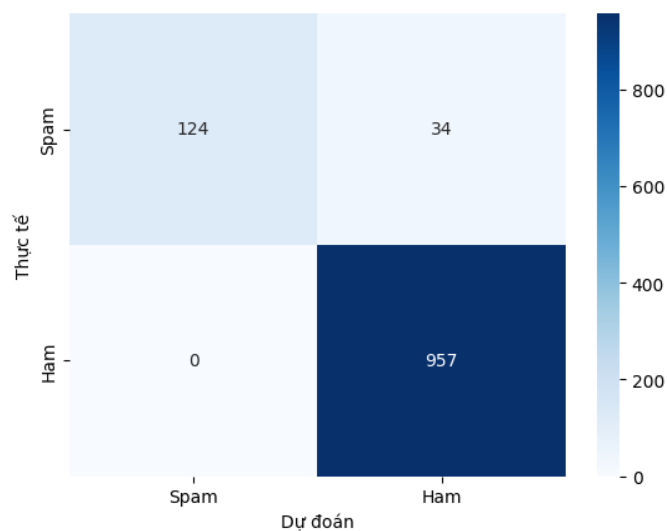
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score,
        confusion_matrix
mnb = MultinomialNB()
mnb.fit(X_train,y_train)
```

## 3.2 Đánh giá mô hình

```
y_pred2 = mnb.predict(X_test)
print("Accuracy Score -",accuracy_score(y_test,y_pred2))
print("Precision Score -",precision_score(y_test,y_pred2))
```

Accuracy Score - 0.9695067264573991  
Precision Score - 0.9656912209889001

Mô hình phân loại thư rác sử dụng thuật toán Naive Bayes trên có độ chính xác là 96.95%.



Hình 3.1: Confusion matrix

Với dữ liệu tập *test*:

- Mô hình dự đoán đúng 957/957 tin nhắn không phải Spam.
- Mô hình dự đoán đúng 124/158 tin nhắn Spam.

# Kết luận

Trong quá trình học tập trên lớp cũng như việc tự tìm hiểu, nghiên cứu kiến thức về Hệ hỗ trợ quyết định đặc biệt là về Mô hình phân lớp Bayes, em đã đạt được những kết quả sau:

- Hiểu cơ sở lý thuyết của Định lý Bayes và mô hình phân lớp Naive Bayes.
- Biết cách ứng dụng mô hình vào bài toán thực tế, cụ thể là phân loại SMS.
- Trau dồi kỹ năng sử dụng ngôn ngữ Python.
- Trau dồi thêm khả năng đọc hiểu các tài liệu chuyên ngành bằng tiếng Anh, cải thiện kỹ năng đọc tài liệu.
- Tăng cường kỹ năng viết và trình bày báo cáo.

# Tài liệu tham khảo

- [1] Tran Duc Tan, Mô hình phân lớp Naive Bayes - [https:// viblo. asia/ p/ mo-hinh-phan-lop-naive-bayes-vyDZO0A7lwj](https://viblo.asia/p/mo-hinh-phan-lop-naive-bayes-vyDZO0A7lwj) , VIBLO.
- [2] Bài 32: Naive Bayes Classifier - [https:// machinelearningcoban. com/ 2017/ 08/ 08/ nbc/](https://machinelearningcoban.com/2017/08/08/nbc/) , Machine Learning cơ bản.
- [3] Wes McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2nd Edition.