# Hướng Dẫn

## **Đồ Án: Lọc thư rác bằng NaiveBayes**

### **Chương 1: Dữ liệu và source code.**

#### 1.1. Dữ liệu.

Dữ liệu để train và test được lấy trong ebook, có 6 cơ sở dữ liệu:

Link: [http://www.aueb.gr/users/ion/data/enron-spam/](http://www.aueb.gr/users/ion/data/enron-spam/" \o "http://www.aueb.gr/users/ion/data/enron-spam/)

Link tải trực tiếp 1 trong các cơ sở dữ liệu( link dự phòng)

Link: [http://www.aueb.gr/users/ion/data/enron-spam/preprocessed/enron1.tar.gz](http://www.aueb.gr/users/ion/data/enron-spam/pre processed/enron1.tar.gz" \o "http://www.aueb.gr/users/ion/data/enron-spam/pre processed/enron1.tar.gz)

#### 1.2. Source code:

Yêu cầu:

+ Python (khiến nghị phiên bản mới nhất)

+ Thư viện nltk

+ Thư viện sklearn (khi tải 2 thư viện trên python sẽ tự động tích hợp các thư viện yêu cầu)

+ Môi trường python(khiến nghị, tránh xung đột)

Code bao gồm:   
 + Filteremail.py

+ NaiveBayes\_emails.py

+ Loaddata.py

Link source: [https://github.com/tamton23/spam\_email\_Naive\_Bayes\_implementations](https://github.com/tamton23/spam_email_Naive_Bayes_implementations" \o "https://github.com/tamton23/spam_email_Naive_Bayes_implementations)

### **Chương 2: Thực thi tập lệnh**

Trước khi thực thi trong tập lệnh Loaddata.py cập nhật đường link dữ liệu.

#### 2.1. Thực thi trực tiếp ra kết quả: 2 cách chạy

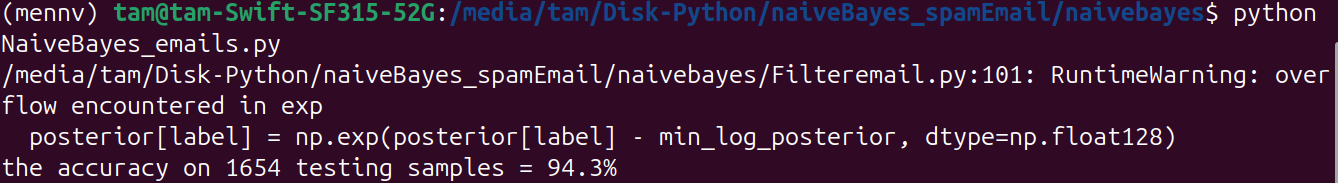
Cách 1:

* python NaiveBayes\_emails.py

Cách 2: Trong giao diện lập trình python sử dụng lệnh

>> exec(open(“NaiveBayes\_emails.py”).read())

Kết quả của dòng lệnh:



#### 2.2. Thực thi chi tiết:

B1. Load dữ liệu và gán nhãn

B2. Định dạng lại nội dung email - ( \_clean\_data())

B3. Features - ( \_get\_features())

B4. Tính prior( xác suất tiên nghiệm) - (\_get\_prior())

B5. Likelihood: P(features|labels) - (\_get\_likelihood())

B6. Posterior(xác suất hậu nghiệm) - ( \_get\_posterior())

B7. Train - test

#### 2.2.1. Load dữ liệu và gán nhãn:

Trong môi trường Python chạy lệnh python.

*>>> e\_mails = [ ]*

*>>> labels = [ ]*

*>>> import Loaddata as l*

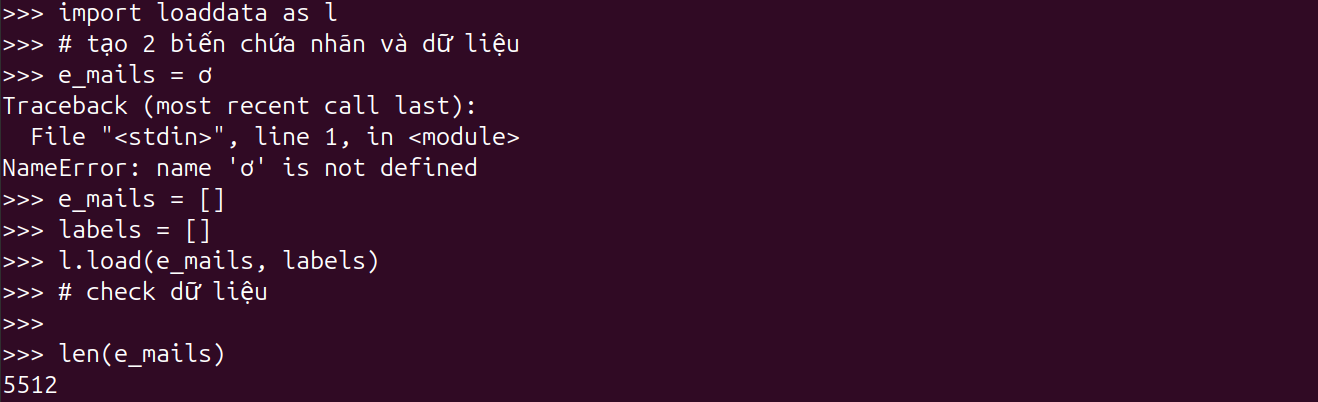
*>>> l.load(e\_mails, labels)*

kiểm tra danh sách xem đã có dữ liệu chưa.

*>>> len(e\_mails)*

5512

Hiển thị:



#### 2.2.2. Định dạng nội dung email: sử dụng func \_clean\_text(tham số) trong Filteremail.py

*>>> import Filteremail as fl*

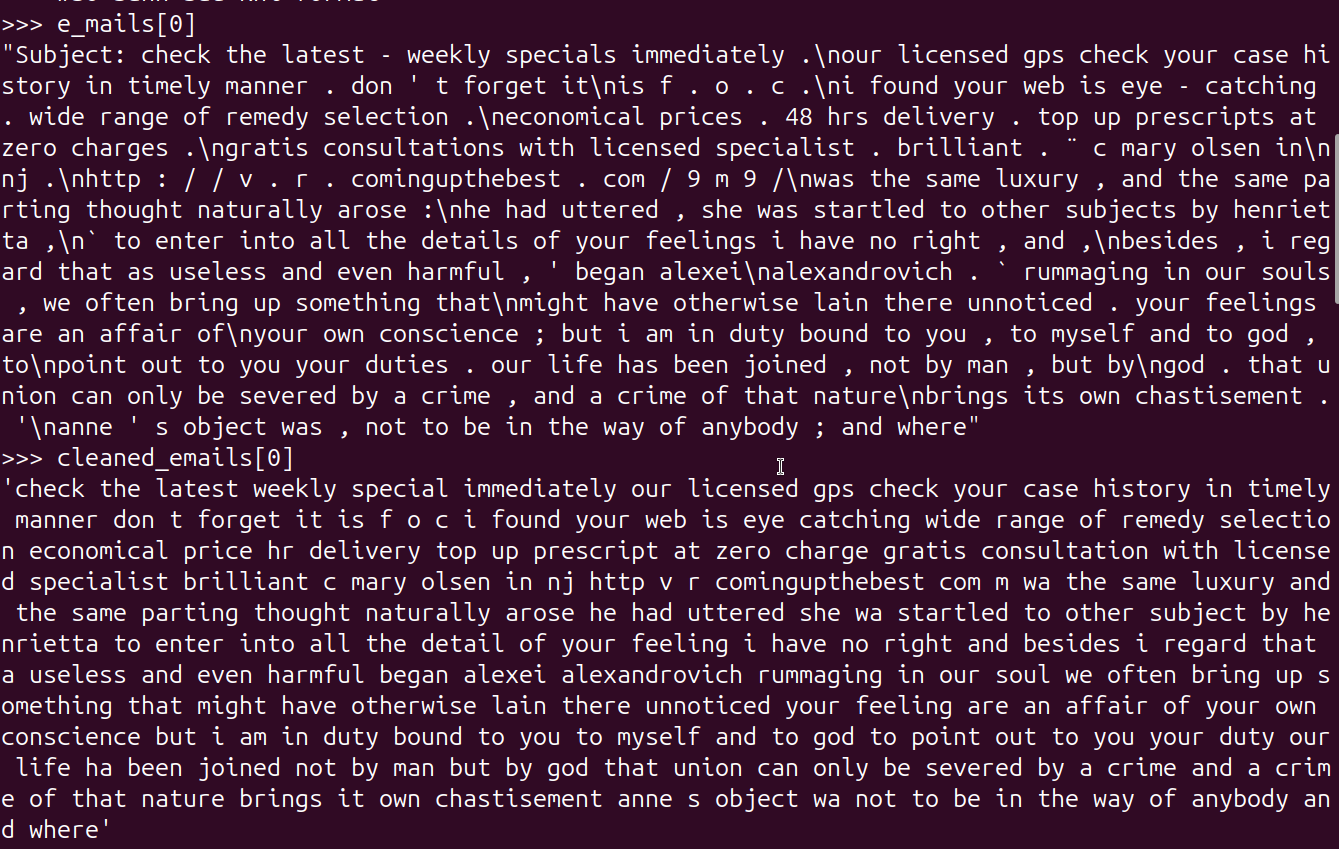
*>>> cleaned\_emails = fl.\_clean\_text(e\_mails)*

So sánh email chưa định dạng và email đã định dạng

*>>> e\_mail[0]*

*>>> cleaned\_emails[0]*

Hiển thị:



#### 2.2.3. Features: \_get\_features. của Filteremail.py

Nhập tiếp tục:

*>>> term\_docs = fl.\_get\_feature(cleaned\_emails)*

Xem có bao nhiêu thuật ngữ xuất hiện trong email.

*>>> print(term\_docs[0])*

#### 2.2.4. Prior: \_get\_label\_index(), \_get\_prior()

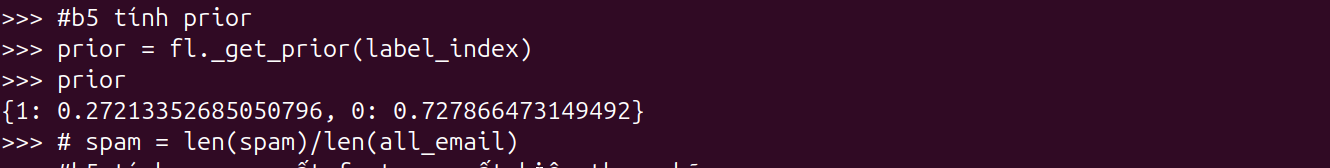
Ta cần group nhãn trước khi tính xác suất tiên nghiệm.

*>>> label\_index = fl.\_get\_label\_index(labels)*

Tính prior.

*>>> prior = fl.\_get\_prior(label\_index)*

Xem kết quả:



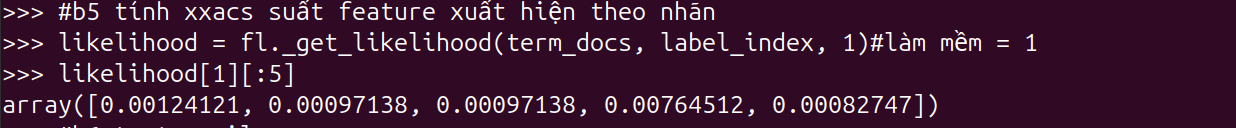
#### 2.2.5. Likelihood: \_get\_likelihood(tham số 1,ts 2,ts 3)

Đưa vào 3 đối số: feature, label\_index và số làm mềm

*>>> likelihood = fl.\_get\_likelihood(term\_docs, label\_index, 1)*

Xem xác suất của 5 email thuộc nhãn 1

*>>> likelihood[1][:5]*



#### 2.2.6. Posterior(xác suất hậu nghiệm):\_get\_posterior

Khi đã có term\_docs, prior, likelihood ta sẽ tính posterior.

*>>> posterior = fl.\_get\_posterior(term\_docs, prior, likelihood)*

Xem xác suất của 5 email đầu.

*>> posterior[:5]*

#### 2.2.7. Train - Test:

*>>> exec(open(“NaiveBayes\_emails.py").read())*

Tính được xác suất sấp xỉ 94% với 70% email train và 30% email test