

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



Thành viên:

21120369 - Nguyễn Minh Vũ

21120558 - Phùng Hoài Thi

Báo cáo thực hành:

PHÂN LOẠI THƯ ĐIỆN TỬ

**CSC18101 – Trí tuệ nhân tạo cho an ninh
thông tin**

Tp. Hồ Chí Minh, tháng 10/2024

Mục lục

1	Nội dung thực hành	4
2	Chuẩn bị dữ liệu	5
3	Xử lý dữ liệu	6
3.1	Xử lý các đặc trưng	6
3.2	Xử lý mẫu	6
4	Huấn luyện mô hình	8
4.1	CountVectorizer	8
4.2	Naive Bayes Classifier	9
4.2.1	Khái niệm về Naive Bayes Classifier	9
4.2.2	Multinomial Naive Bayes	11
4.3	Tạo pipeline	13
4.3.1	Hyperparameters	13
4.3.2	Pipeline	13
4.3.3	Dự đoán trên tập val	14
5	Thử nghiệm thực tế	15
5.1	Dữ liệu	15
5.2	Ứng dụng	15
5.2.1	Chức năng 1	15
5.2.2	Chức năng 2	16

Danh sách hình

5.1	Chức năng 1	15
5.2	Ví dụ chức năng 1 (Spam)	16
5.3	Ví dụ chức năng 1 (Non-Spam)	16
5.4	Chức năng 2	16
5.5	Ví dụ chức năng 2	17
5.6	Kết quả file output.csv	17

Danh sách bảng

1.1	Bảng đánh giá công việc	4
2.1	Dữ liệu Enron-Spam	5
3.1	Dữ liệu sau khi xử lý	7
4.1	Ví dụ về CountVectorizer	8
4.2	Ví dụ về phân loại văn bản trong bài toán Bếp và Nhà tắm	12
4.3	Minh hoạ <i>Multinomial Naive Bayes</i> cho tập huấn luyện .	12
4.4	Classification Report	14

Nội dung thực hành

Yêu cầu	Phụ trách	Mức độ hoàn thành
Đọc dữ liệu	Vũ	100%
Tiền xử lý dữ liệu	Vũ	100%
Mô hình	Vũ	100%
Cài đặt chức năng 1	Thi	100%
Cài đặt chức năng 2	Thi	100%
Viết báo cáo	Vũ, Thi	100%
Làm slide	Vũ, Thi	100%

Bảng 1.1: Bảng đánh giá công việc

Chuẩn bị dữ liệu

Trong bài thực hành, Enron-Spam [2] được lưu dưới dạng file .csv (**train.csv** để huấn luyện và **val.csv** để so sánh chất lượng các mô hình). Ta sẽ sử dụng *pandas* để đọc dữ liệu từ file như sau:

```
train_csv = pandas.read_csv("train.csv")
val_csv = pandas.read_csv("val.csv")
```

Kiểm tra số lượng mẫu của tập train và tập val:

```
train_data.shape() #(27284, 6)
val_data.shape() #(3084, 6)
```

Dữ liệu sau khi đọc có dạng bảng gồm các đặc trưng như sau:

Unnamed: 0	Message ID	Subject	Message	Spam/Ham	split
0	0	christmas ...	NaN	ham	0.038415
1	1	vastar res...	gary , pro...	ham	0.696509
2	2	calpine da...	- calpine ...	ham	0.587792
3	3	re : issue...	fyi - see ...	ham	-0.055438
5	5	mcmullen g...	jackie , s...	ham	-0.419658

Bảng 2.1: Dữ liệu Enron-Spam

Trong đó:

- Subject: Tiêu đề của emails.
- Message: Nội dung của emails.
- Spam/Ham: Đánh dấu một mail là Spam hoặc Ham.
- Unnamed: 0/ Message ID/ split: các đặc trưng không liên quan (không ảnh hưởng đến việc huấn luyện mô hình) ta sẽ tiến hành loại bỏ ra khỏi dữ liệu.

Xử lý dữ liệu

Bước tiếp theo, ta cần xử lý các dữ liệu bị thiếu, mất, lặp hoặc không ảnh hưởng đến mô hình.

3.1 Xử lý các đặc trưng

Ta chỉ giữ lại những cột đặc trưng quan trọng hỗ trợ cho mô hình như nội dung Subject, Message và nhãn Spam/Ham từ bảng 2.1. Việc chọn giữ các đặc trưng cần thiết thay vì loại bỏ những cột không quan trọng thuận tiện cho phần 5 vì dữ liệu đầu vào có thể khác nhau.

```
data = data[['Subject', 'Message', 'Spam/Ham']]
```

3.2 Xử lý mẫu

Công việc ta cần xử lý tiếp theo đó là với các mẫu dữ liệu mà không chứa giá trị hoặc giá trị bị trùng lặp để làm giảm số lượng dữ liệu mà vẫn giữ lại thông tin.

Đầu tiên, ta sẽ loại bỏ các mẫu bị trùng lặp và chỉ giữ lại một mẫu duy nhất.

```
data.drop_duplicates(subset='Message')  
data.drop_duplicates(subset='Subject')
```

Tiếp theo đó, các dữ liệu bị mất hoặc không chứa giá trị ở 2 cột nội dung **Subject** và **Message**, ta sẽ không loại bỏ ngay vì điều này sẽ làm mất thông tin dữ liệu. Cụ thể hơn, nếu 1 trong 2 cột bị mất dữ liệu và ta loại bỏ nó ra, cả mẫu cũng sẽ bị loại bỏ dù cho nó vẫn chứa dữ liệu. Thay vào đó, ta sẽ đưa dữ liệu không có giá trị về dạng chuỗi kí tự rỗng.

```
data.fill_na("")
```

Bên cạnh đó, để dữ liệu có sự đồng nhất với nhau cho việc huấn luyện mô hình, ta gộp 2 đặc trưng chứa nội dung **Subject** và **Message** thành một đặc trưng duy nhất, gọi là **Text**.

```
data['Text'] = data['Subject'] + ' ' + data['Message']
```

Cuối cùng, ta thay các *flag* (cờ) ở cột đặc trưng **Spam/Ham** thành giá trị nhị phân (đánh dấu 1 nếu là Spam và ngược lại).

```
data['spam'] = data['Spam/Ham'].apply(lambda x:1 if x ==  
'spam' else 0)
```

Kết hợp tất cả lại với nhau, ta sẽ tạo thành một hàm xử lý dữ liệu:

```
def preprocess(data):  
    data = data[['Subject', 'Message', 'Spam/Ham']]  
    data = data.drop_duplicates(subset='Message')  
    data = data.drop_duplicates(subset='Subject')  
    data = data[~(data['Subject'].isnull() &  
data['Message'].isnull())]  
    data = data.fillna('')  
    data['Text'] = data['Subject'] + ' ' + data['Message']  
    data.drop(['Subject', 'Message'], axis=1, inplace=True)  
    data['spam'] = data['Spam/Ham'].apply(lambda x: 1 if x==  
'spam' else 0)  
    return data
```

Dữ liệu sau khi xử lý sẽ có dạng như sau:

Spam/Ham	Text	spam
ham	christmas tree farm ...	0
ham	vastar resources , i...	0
ham	calpine daily gas no...	0

Bảng 3.1: Dữ liệu sau khi xử lý

Huấn luyện mô hình

Sau khi đã xử lý dữ liệu, ta phải số hoá dữ liệu và đưa cho mô hình học. Nhóm quyết định sử dụng phương pháp **CountVectorizer** kèm theo đó là mô hình **MultinomialNB** từ thư viện **sk-learn**. Bên cạnh đó, để mô hình được gọn gàng phục vụ cho phần 5, nhóm sẽ kết hợp với **Pipeline**.

4.1 CountVectorizer

Đầu tiên là **CountVectorizer**. **CountVectorizer** sử dụng phương pháp đếm tần suất xuất hiện của một từ trong một văn bản trên toàn bộ bộ ngữ liệu (corpus) từ dữ liệu hay còn gọi là mô hình túi từ (**Bag-of-Words**). Ta xem xét ví dụ sau:

```
docs= ["Trời hôm nay mưa", "Hôm nay tôi đi học",  
       "Trời mưa nhưng tôi vẫn mặc áo mưa đi học"]
```

Khi triển khai **CountVectorizer**, ta sẽ có bảng dữ liệu như sau:

	áo	đi	hôm	học	mưa	mặc	nay	nhưng	tôi	trời	vẫn
docs[0]	0	0	1	0	1	0	1	0	0	1	0
docs[1]	0	1	1	1	0	0	1	0	1	0	0
docs[2]	1	1	0	1	2	1	0	1	1	1	1

Bảng 4.1: Ví dụ về CountVectorizer

Một số quan sát có được:

- Có tất cả 11 từ khác biệt trong bộ ngữ liệu docs, được thể hiện qua các cột.
- Có tất cả 3 văn bản (mẫu/sample), được thể hiện qua các dòng.

- Các từ đều được viết thường và sắp xếp theo thứ tự bảng chữ cái.

Sau khi nắm được nguyên lý hoạt động cơ bản của mô hình túi từ, ta sẽ áp dụng nó cho bộ dữ liệu **Enron-Spam** mà ta đã xử lý trước đó.

```
#Chia tập huấn luyện
x_train, x_val, y_train, y_val = train_data.Text,
val_data.Text, train_data.spam, val_data.spam
#Mô hình CountVectorizer
cv = CountVectorizer(ngram_range=(1,2))
#Biến đổi dữ liệu
x_train_cv = cv.fit_transform(x_train)
x_val_cv = cv.transform(x_val)
```

Nhóm đã thử điều chỉnh các tham số của mô hình và đúc kết được kết quả tốt nhất là tăng chuỗi từ lên chuỗi 2 từ liên tiếp nhau (mặc định là 1 từ đơn). Trong khi đó, các tham số khác như *max_features*, *max_df*, *min_df* đều làm giảm hiệu suất của mô hình.

Kết quả ta đạt được có 1233930 từ và cụm từ khác biệt nhau:

```
print(x_train_cv.shape) #(19721, 1233930)
print(x_val_cv.shape) #(2751, 1233930)
```

4.2 Naive Bayes Classifier

4.2.1 Khái niệm về Naive Bayes Classifier

Xét bài toán phân lớp với C lớp khác nhau. Thay vì yêu cầu tìm ra lớp của mỗi dữ liệu $\mathbf{x} \in \mathbb{R}$ chính xác thì ta có thể tìm xác suất cho đầu ra đó rơi vào mỗi lớp với xác suất $p(y = c|x)$ hay $p(c|x)$. Từ ta có thể xác định lớp của mỗi điểm dữ liệu dựa vào lớp có xác suất cao nhất:

$$c = \arg \max_{c \in \{1, \dots, C\}} p(c|\mathbf{x}) \quad (4.1)$$

Tuy nhiên, cách tính trực tiếp xác suất p của lớp c khi biết \mathbf{x} khá phức tạp. Thay vào đó, áp dụng quy tắc Bayes sẽ đơn giản hơn:

$$c = \arg \max_c p(c|\mathbf{x}) = \arg \max_c \frac{p(\mathbf{x}|c)p(c)}{p(\mathbf{x})} = \arg \max_c p(\mathbf{x}|c)p(c) \quad (4.2)$$

Trong biểu thức trên:

- Dấu " = " thứ hai có được từ quy tắc Bayes.
- Dấu " = " thứ ba xảy ra vì $p(\mathbf{x})$ không phụ thuộc vào c .
- Thành phần $p(c)$ là xác suất để 1 điểm bất kỳ thuộc vào lớp c .
- Thành phần $p(\mathbf{x}|c)$ là phân phối của các điểm dữ liệu trong c .

Vì \mathbf{x} là một biến dữ liệu nhiều chiều nên để có thể ước lượng được phân phối đó thì ta sẽ phải cần một tập huấn luyện cực kì lớn. Để tối giản việc tính toán, ta có thể giả sử rằng các thành phần của biến ngẫu nhiên \mathbf{x} là độc lập với nhau khi biết c :

$$p(\mathbf{x}|c) = p(x_1, x_2, \dots, x_d|c) = \prod_{i=1}^d p(x_i|c) \quad (4.3)$$

Giả thiết *ngây thơ* (*naive*) này thực sự mang lại độ hiệu quả bất ngờ mặc dù trên thực tế, hiếm khi có thể tìm được dữ liệu mà các thành phần của nó độc lập hoàn toàn với nhau. Cái tên *Naive Bayes Classifier* [1] cũng ra đời từ các giả thiết này.

Nhờ vào tính đơn giản của NBC mà nó có tốc độ huấn luyện và kiểm thử rất nhanh. Ở bước huấn luyện, phân phối $p(c)$ và $p(x_i|c)$ với $i = 1, \dots, d$ sẽ được xác định dựa vào tập huấn luyện. Đối với bước kiểm thử, nhãn của một điểm dữ liệu mới \mathbf{x} sẽ được xác định bằng:

$$c = \arg \max_{c \in \{1, \dots, C\}} p(c) \prod_{i=1}^d p(x_i|c) \quad (4.4)$$

hoặc:

$$c = \arg \max_{c \in \{1, \dots, C\}} (\log(p(c)) + \sum_{i=1}^d \log(p(x_i|c))) \quad (4.5)$$

Biểu thức 4.5 được dùng để thay thế khi d lớn và các xác suất nhỏ dẫn đến biểu thức 4.4 rất nhỏ có thể gây ra sai số. Cuối cùng việc tính toán xác suất $p(\mathbf{x}_i|c)$ phụ thuộc vào loại dữ liệu và có ba loại phân phối thường được dùng phổ biến là *Gaussian*, *Multinomial* và *Bernoulli Naive Bayes*.

4.2.2 Multinomial Naive Bayes

Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà vector đặc trưng được xây dựng dựa trên ý tưởng Bag-of-Words (BoW). Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển (trình bày ở phần 4.1). Giá trị của thành phần thứ i trong mỗi vector chính là số lần từ thứ i xuất hiện trong văn bản đó. Khi đó, $p(x_i|c)$ tỉ lệ với tần suất từ thứ i (hay đặc trưng thứ i cho trường hợp tổng quát) xuất hiện trong các văn bản của class c . Giá trị này có thể được tính bằng:

$$\lambda_{ci} = p(x_i|c) = \frac{N_{ci}}{N_c} \quad (4.6)$$

Trong đó:

- N_{ci} là tổng số lần từ thứ i xuất hiện trong các văn bản của lớp c .
- N_c là tổng số từ (kể cả lặp) xuất hiện trong lớp c . Nói cách khác, nó là tổng độ dài của tất cả văn bản thuộc lớp c . Ta có thể suy ra rằng $N_c = \sum_{i=1}^d N_{ci}$ và $\sum_{i=1}^d \lambda_{ci} = 1$.

Tuy nhiên, điều này có một hạn chế là biểu thức 4.6 sẽ có giá trị bằng 0 nếu như có một từ mới chưa xuất hiện trong bộ ngữ liệu dẫn đến biểu thức 4.4 cũng bằng 0 bất kể giá trị còn lại có như thế nào. Để khắc phục điều này, *Multinomial Naive Bayes* sử dụng kỹ thuật *Laplace Smoothing*:

$$\hat{\lambda}_{ci} = \frac{N_{ci} + \alpha}{N_c + d\alpha} \quad (4.7)$$

với α là một số dương để tránh trường hợp tử số bằng 0 và mẫu số được cộng thêm $d\alpha$ để đảm bảo tổng xác suất bằng 1.

Hãy đến với một ví dụ cơ bản sau đây để hoàn tất phần lý thuyết trên. Giả sử ta có tập dữ liệu như sau:

	Docs	Nội dung	Nhãn
Tập huấn luyện	d1	dao chén đĩa muống	Bếp
	d2	khăn gương vôi	Nhà tắm
	d3	nồi chén chén đĩa	Bếp
	d4	muống chén đĩa	Bếp
Tập kiểm thử	d5	khăn gương chén đĩa dao	???

Bảng 4.2: Ví dụ về phân loại văn bản trong bài toán Bếp và Nhà tắm

Đối với tập huấn luyện trên, ta thấy rằng:

$$p(\text{Bếp}) = \frac{3}{4}, \quad p(\text{Nhà tắm}) = \frac{1}{4} \quad (4.8)$$

$$|V| = \{\text{dao, chén, đĩa, muống, khăn, gương, vôi, nồi, đĩa}\} = 9$$

	Docs	dao	chén	đĩa	muống	khăn	gương	vôi	nồi	đĩa
Bếp	d1	1	1	1	1	0	0	0	0	0
	d3	0	2	0	0	0	0	0	1	1
	d4	0	1	1	1	0	0	0	0	0
	Tổng	1	4	2	2	0	0	0	1	1
	$\hat{\lambda}_{\text{Bếp}}$	$\frac{2}{20}$	$\frac{5}{20}$	$\frac{3}{20}$	$\frac{3}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{2}{20}$	$\frac{2}{20}$
Nhà tắm	d2	0	0	0	0	1	1	1	0	0
	Tổng	0	0	0	0	1	1	1	0	0
	$\hat{\lambda}_{\text{Nhà tắm}}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{2}{12}$	$\frac{2}{12}$	$\frac{2}{12}$	$\frac{1}{12}$	$\frac{1}{12}$

Bảng 4.3: Minh hoạ *Multinomial Naive Bayes* cho tập huấn luyện

Bây giờ ta sẽ gán nhãn cho dữ liệu d5: $[1, 1, 1, 0, 1, 1, 0, 0, 0]$:

$$\begin{aligned}
p(\text{Bếp}|d5) &\propto p(\text{Bếp}) \prod_{i=1}^9 p(x_i|\text{Bếp}) \\
&\propto \left(\frac{3}{4}\right) \cdot \frac{2}{20} \cdot \frac{5}{20} \cdot \frac{3}{20} \cdot \frac{1}{20} \cdot \frac{1}{20} \approx 0.7 \times 10^{-5} \\
p(\text{Nhà tắm}|d5) &\propto p(\text{Nhà tắm}) \prod_{i=1}^9 p(x_i|\text{Nhà tắm}) \\
&\propto \left(\frac{1}{4}\right) \cdot \frac{1}{12} \cdot \frac{1}{12} \cdot \frac{1}{12} \cdot \frac{2}{12} \cdot \frac{2}{12} \approx 0.4 \times 10^{-5} \\
p(x_5|\text{Bếp}) &> p(x_5|\text{Nhà tắm}) \Rightarrow d5 \in \text{class}(\text{Bếp})
\end{aligned}$$

4.3 Tạo pipeline

Giờ ta đã hiểu cơ chế hoạt động của mô hình *Multinomial Naive Bayes*, tiếp theo ta sẽ tiến hành cài đặt và huấn luyện cho mô hình.

4.3.1 Hyperparameters

Nhóm sử dụng 2 siêu tham số đã giúp cho mô hình đạt độ hiệu quả cao:

```
parameters = [{
    'vect__ngram_range': [(1, 2)],
    'mnb__alpha': [(0.1)]
}]
```

Lý do chọn $\alpha = 0.1$ thay vì $\alpha = 1$ như trong ví dụ trước đó là vì $\alpha = 0.1$ làm cho mô hình thực hiện việc smoothing ít hơn, giữ cho các xác suất các từ phổ biến có ưu thế hơn trong khi vẫn tránh sự ảnh hưởng bởi những từ hiếm mà không làm mất thông tin. Bên cạnh đó, tăng số từ liên tiếp lên 2 giúp cho vec-tơ từ được đa dạng hơn và thành quả là mô hình đạt độ chính xác cao hơn.

4.3.2 Pipeline

Thay vì phải lưu 2 mô hình *CountVectorizer* và *MultinomialNB* tách biệt nhau thì ta có thể áp dụng Pipeline để gộp cả hai lại, giúp tiết kiệm tài nguyên hơn.

```
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('mnb', MultinomialNB())
])
grid_search = GridSearchCV(pipeline, parameters)
complete_pipeline = grid_search.fit(x_train, y_train)
```

4.3.3 Dự đoán trên tập val

```
y_val_pred = complete_pipeline.predict(x_val)
print("Accuracy = %.2f%%" % (accuracy_score(y_val,
y_val_pred) * 100))
print(classification_report(y_val, y_val_pred, digits=4))
```

Kết quả dự đoán cho tập val cho thấy mô hình đạt độ chính xác xấp xỉ 99.45%. Nhóm đã cố gắng tối ưu hiệu suất mô hình bằng việc điều chỉnh các siêu tham số của *CountVectorizer* cũng như *MultinomialNB* và đây là kết quả cao nhất mà nhóm đạt được.

Class	Precision	Recall	F1-score	Support
0	0.9951	0.9944	0.9948	1433
1	0.9939	0.9947	0.9943	1318
Accuracy			0.9945	2751
Macro avg	0.9945	0.9946	0.9945	2751
Weighted avg	0.9945	0.9945	0.9945	2751

Bảng 4.4: Classification Report

Thử nghiệm thực tế

5.1 Dữ liệu

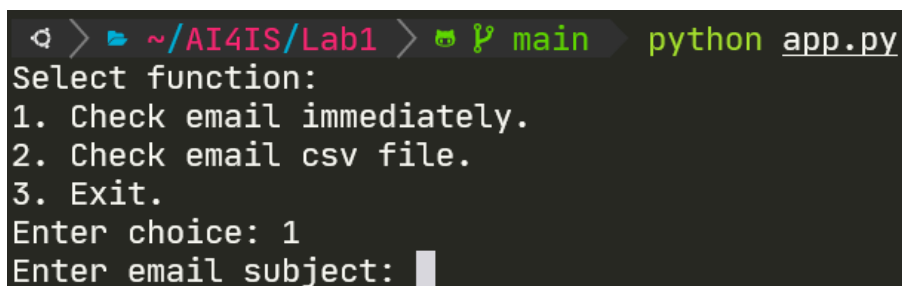
Nhóm thu thập dữ liệu test các function từ dữ liệu Enron Spam mà chưa xuất hiện tại Train và Val data.

5.2 Ứng dụng

Nhóm đóng gói model bằng thư viện 'joblib' và chạy độc lập trên một file python script. File bao gồm 2 chức năng theo yêu cầu và một số cài đặt cơ bản.

5.2.1 Chức năng 1

- Người dùng chạy ứng dụng và chọn chức năng 1.



```
> ~ / AI4IS / Lab1 > main > python app.py
Select function:
1. Check email immediately.
2. Check email csv file.
3. Exit.
Enter choice: 1
Enter email subject: █
```

Hình 5.1: Chức năng 1

- Nhập Subject và Message, ứng dụng sẽ đưa ra kết luận (Spam/Non-spam).


```

❏ > ~/AI4IS/Lab1 > 🐍 ? main ?1 python app.py
Select function:
1. Check email immediately.
2. Check email csv file.
3. Exit.
Enter choice: 1
Enter email subject: despot r j 9
Enter email message: why pay full retail for the name - brand
product when a generic version is now available to you at a 60
% discount ? generic drugs are just as safe as their brand -
name counterparts . http : / / goget . com / ? blingsoft

This is Spam email.

```

Hình 5.2: Ví dụ chức năng 1 (Spam)

```

Select function:
1. Check email immediately.
2. Check email csv file.
3. Exit.
Enter choice: 1
Enter email subject: Cake - Take home assignment - Data Scientist Intern
Enter email message: Dear Nguyen Minh Vu, Thank you for your interest in Cake's
careers. As the first step in our selection process, we would like to invite yo
u to complete the assignment for the Data Scientist Intern position. Please fi
nd the description and guidelines below, and submit your results within 7 days.
If you have any questions or concerns, feel free to reply to this email. Don't
forget to confirm upon receiving this email to let us know you have received i
t. Good luck with the challenge! We look forward to reviewing your submission.

This is not a Spam email.

```

Hình 5.3: Ví dụ chức năng 1 (Non-Spam)

5.2.2 Chức năng 2

- Người dùng chạy ứng dụng và chọn chức năng 2.

```

❏ > ~/AI4IS/Lab1 > 🐍 ? main ?1 python app.py
Select function:
1. Check email immediately.
2. Check email csv file.
3. Exit.
Enter choice: 2
Enter file path: 

```

Hình 5.4: Chức năng 2

- Người dùng nhập được dẫn file csv và ứng dụng trả kết quả dự đoán dưới dạng classification report, sau đó xuất kết quả tại file output.csv.

```

> ~/AI4IS/Lab1 > main ?1 python app.py
Select function:
1. Check email immediately.
2. Check email csv file.
3. Exit.
Enter choice: 2
Enter file path: test.csv
Accuracy: 1.0

```

	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	10
1	1.0000	1.0000	1.0000	11
accuracy			1.0000	21
macro avg	1.0000	1.0000	1.0000	21
weighted avg	1.0000	1.0000	1.0000	21

```

Output saved to output.csv

```

Hình 5.5: Ví dụ chức năng 2

- Nội dung file output.csv.

```

output.csv
1 Message ID,Spam/Ham,Spam/Ham Predict
2 9,ham,ham
3 19,ham,ham
4 30,ham,ham
5 31,ham,ham
6 49,ham,ham
7 53,ham,ham
8 64,ham,ham
9 74,ham,ham
10 88,ham,ham
11 92,ham,ham
12 19401,spam,spam
13 19044,spam,spam
14 19045,spam,spam
15 19061,spam,spam
16 19078,spam,spam
17 19094,spam,spam
18 19098,spam,spam
19 19099,spam,spam
20 19100,spam,spam
21 19101,spam,spam
22 19102,spam,spam

```

Hình 5.6: Kết quả file output.csv

Tài liệu tham khảo

- [1] Vũ Hữu Tiệp. “Machine learning cơ bản”. In: *Nhà xuất bản Khoa học và Kỹ thuật* (2018).
- [2] Metsis Vangelis, Androutsopoulos Ion, and P Geogios. “Spam filtering with naive bayes-which naive bayes?” In: *Third conference on email and anti-spam (CEAS)*. 2006.