



TÍNH TOÁN HIỆU SUẤT CAO

TRIỂN KHAI PYTHON TRÊN DOCKER IMAGE OFFICIAL

NHÓM 5
GV HƯỚNG DẪN: TS. NGUYỄN QUỐC HÙNG



BỐ CỤC BÀI TRÌNH BÀY

Đề xuất mô hình và
phương hướng triển
khai

Cài đặt và triển khai
ứng dụng

Chương 1

Giới thiệu tính toán
hiệu suất cao (HPC) và
dự án thực hiện

Chương 2

Xây dựng ứng dụng
Python

Chương 3

Chương 4

Chương 5

Kết luận và hướng
phát triển

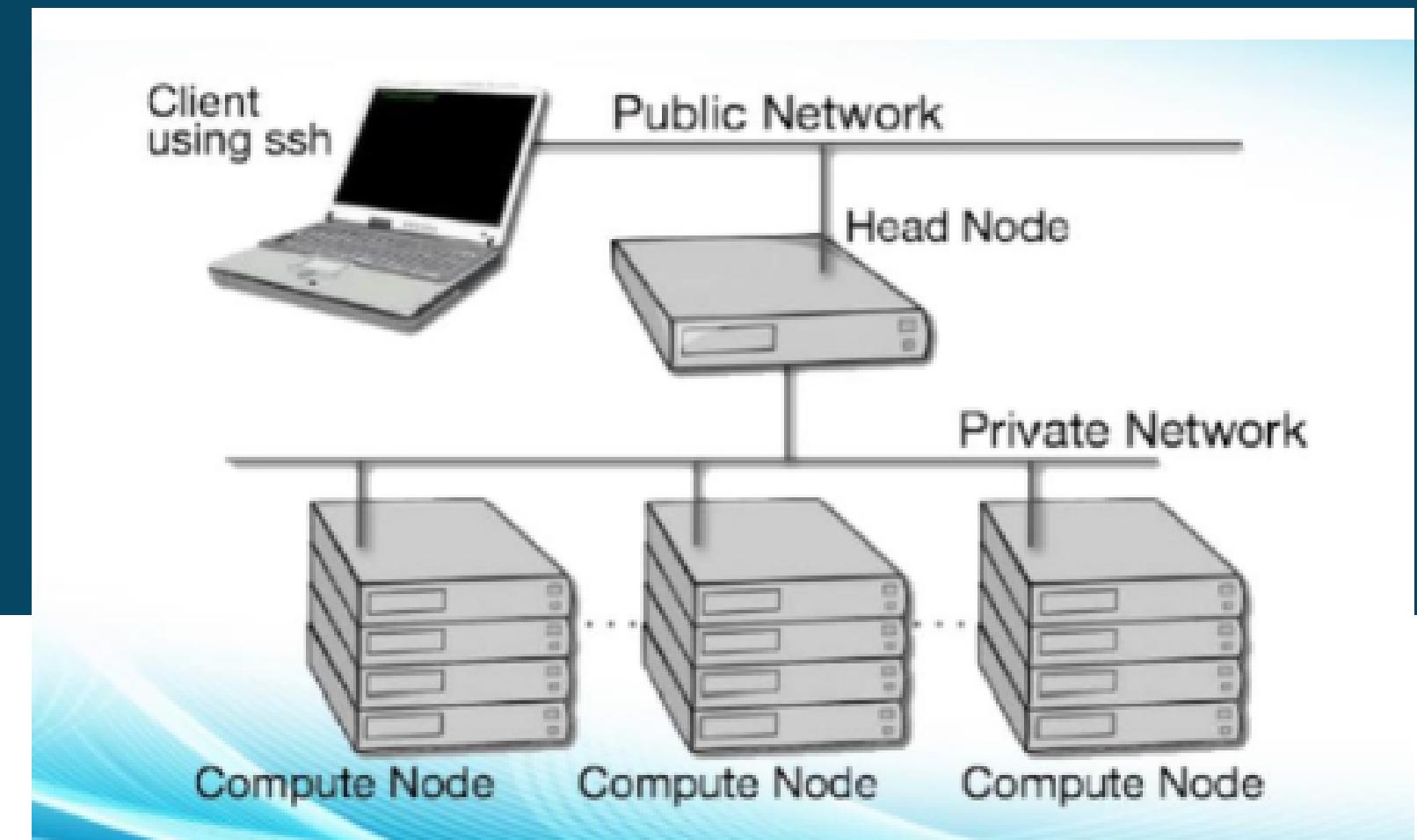


HPC là gì?

Các hệ thống máy tính có khả năng xử lý dữ liệu và thực hiện các phép tính phức tạp với tốc độ cao hơn rất nhiều so với máy tính thông thường.

Mô hình triển khai hệ thống HPC

- Máy tính kết nối nội bộ (Private Network)
- Kết nối với máy chủ điều khiển (Head Node) qua SSH (Public Network)
- Mỗi máy tính là một nút tính toán
- Có thể mở rộng thêm nút tính toán để nâng cao hiệu năng



KIẾN TRÚC CỦA HỆ THỐNG

MASSIVELY PARALLEL PROCESSING

- Hệ thống Cray XC50 là một hệ thống MPP được sử dụng cho nghiên cứu khoa học và kỹ thuật.
- Hệ thống IBM Power System 9 là một hệ thống MPP được sử dụng cho các ứng dụng kinh doanh.

CLUSTER COMPUTING

- Hệ thống Beowulf là một hệ thống CC miễn phí và mã nguồn mở.
- Hệ thống HTCondor là một hệ thống CC được sử dụng để phân phối các tác vụ tính toán trên nhiều máy tính.



ỨNG DỤNG & XU HƯỚNG

ỨNG DỤNG

- Khoa học
- Kỹ thuật
- Tài chính
- Quân sự

XU HƯỚNG

- HPC đám mây
- HPC trí tuệ
nhân tạo
- HPC lượng tử

ỨNG DỤNG MÔ HÌNH PHÂN LỚP

- Xây dựng và đánh giá mô hình bằng các thuật toán Machine Learning.
- Xây dựng và đánh giá mô hình bằng các thuật toán Maximum Entropy.

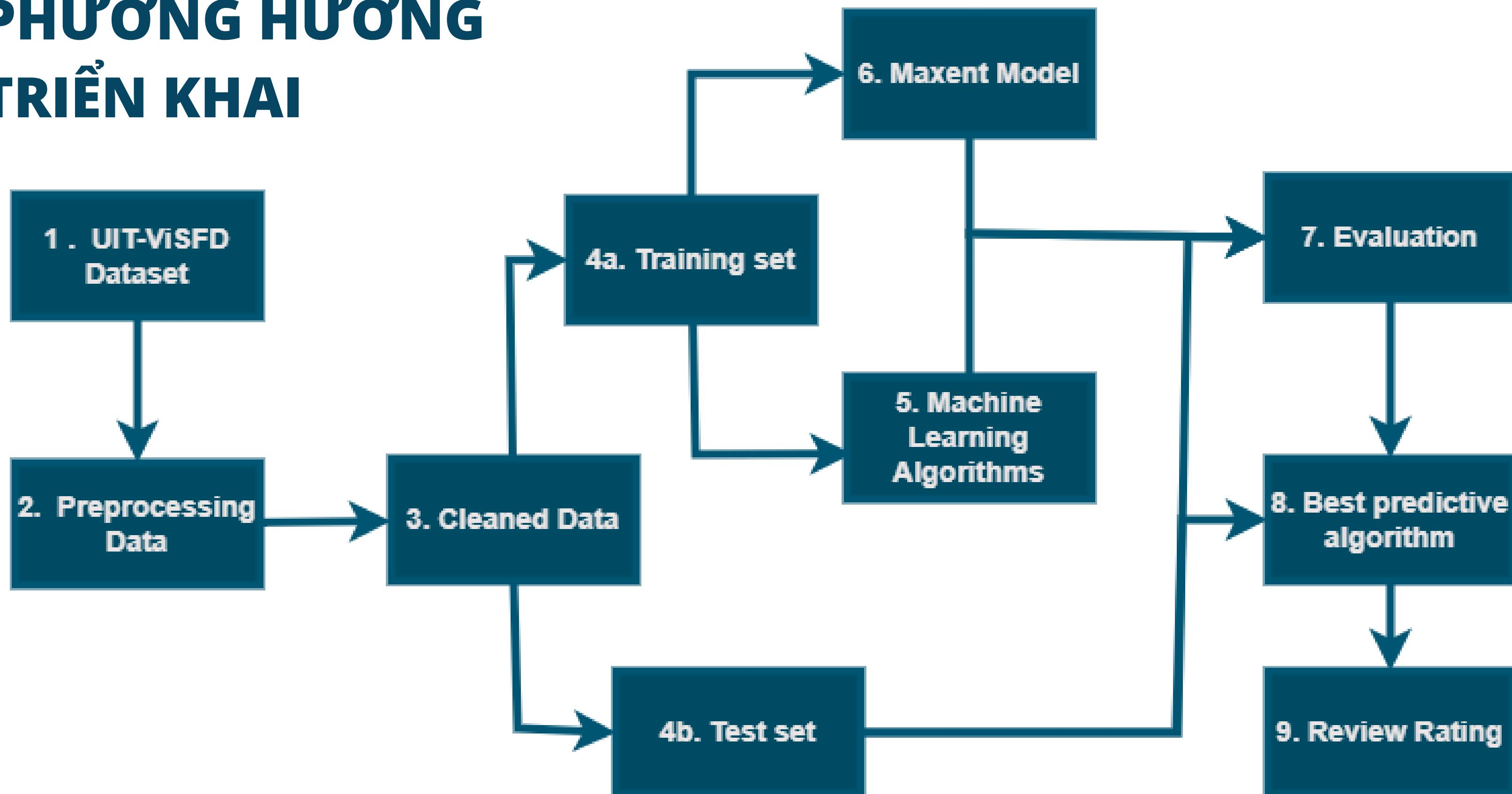
01

Naive Bayes

02

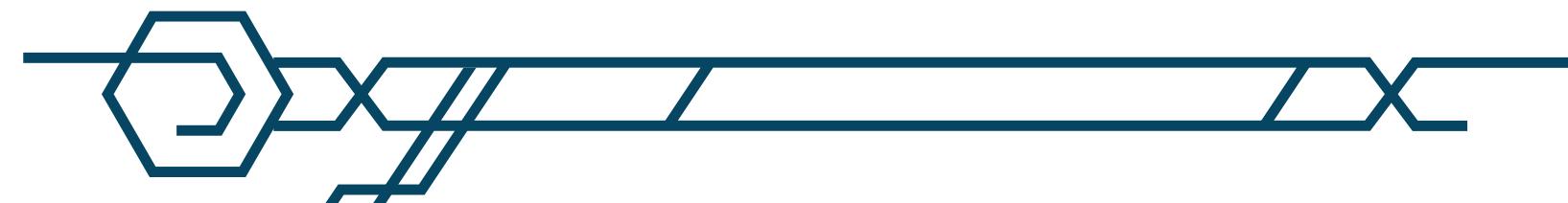
Logistic Regression

PHƯƠNG HƯỚNG TRIỂN KHAI



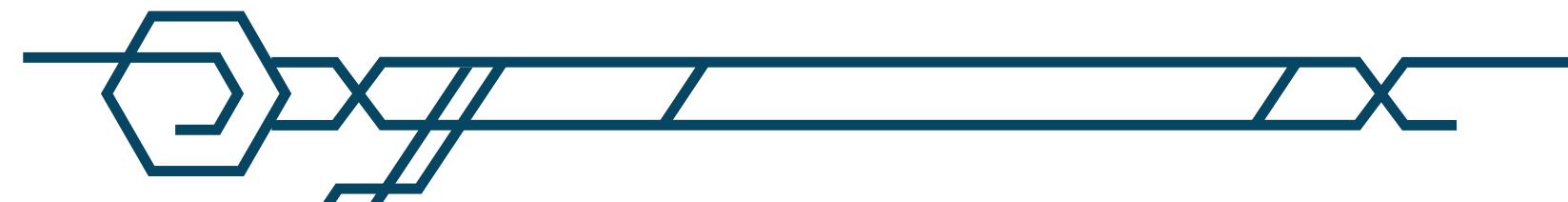
TỔNG QUAN BỘ DỮ LIỆU

Tên thuộc tính	Kiểu dữ liệu	Mô tả
index	int64	số thứ tự các bình luận
comment	object	các bình luận của khách hàng và người tiêu dùng
n_star	int64	đánh giá của khách hàng và người tiêu dùng
date_time	object	thời gian của các bình luận và đánh giá
label	object	đánh giá tích cực/ tiêu cực/ trung tính theo các khía cạnh



TỔNG QUAN BỘ DỮ LIỆU

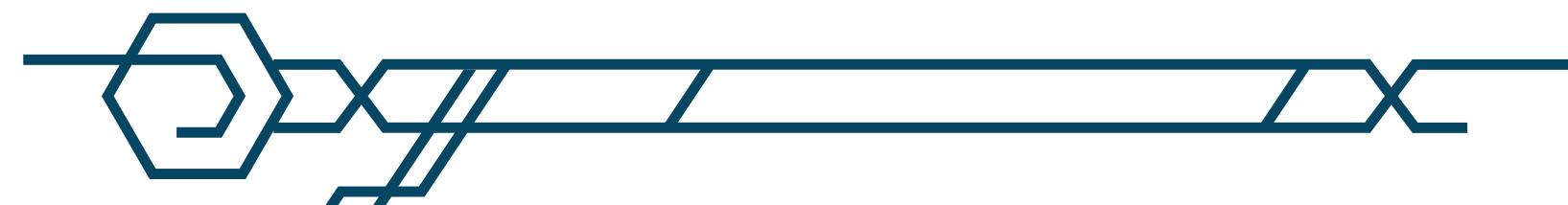
Tên thuộc tính	Kiểu dữ liệu	Mô tả
index	int64	số thứ tự các bình luận
comment	object	các bình luận của khách hàng và người tiêu dùng
n_star	int64	đánh giá của khách hàng và người tiêu dùng
date_time	object	thời gian của các bình luận và đánh giá
label	object	đánh giá tích cực/ tiêu cực/ trung tính theo các khía cạnh



TỔNG QUAN BỘ DỮ LIỆU

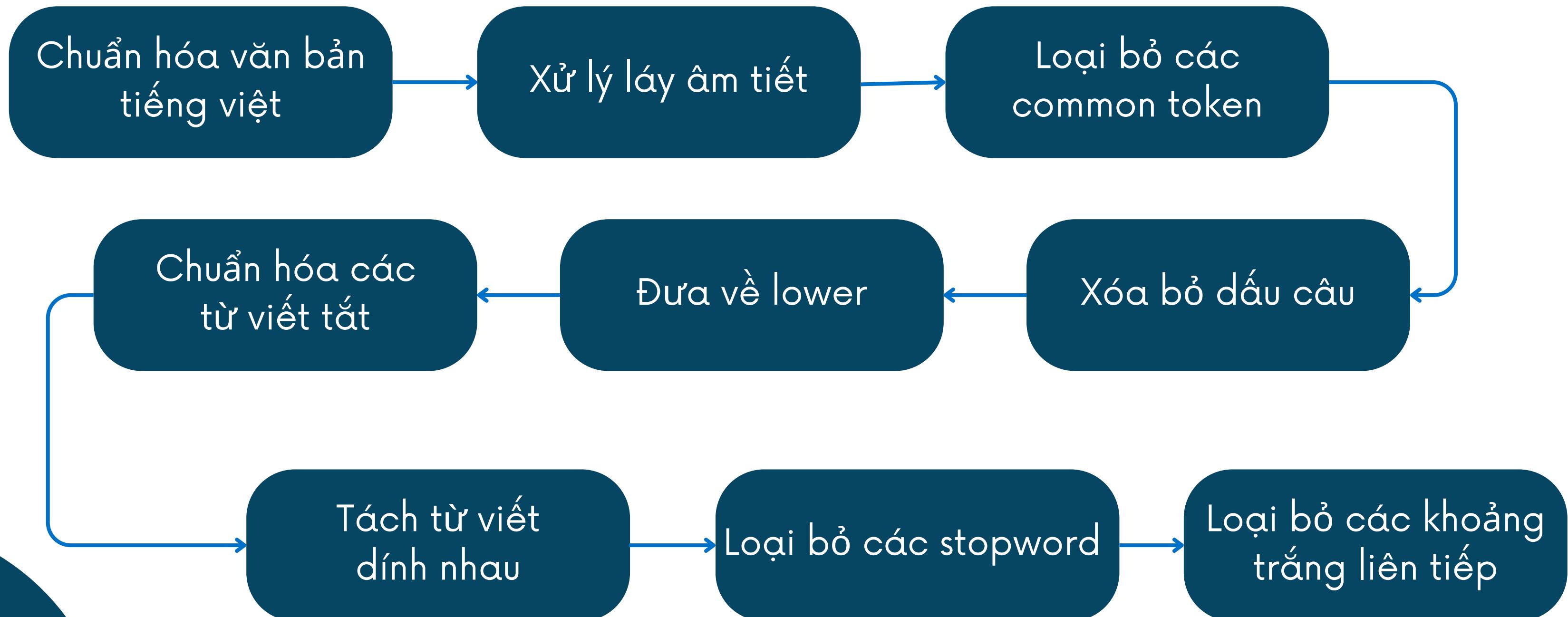
index	comment	n_star	date_time	label
0	Mới mua máy này Tại thegioididong thót nốt cầm...	5	2 tuần trước	{CAMERA#Positive};{FEATURES#Positive};{BATTERY...
1	Pin kém còn lại miễn chê mua 8/3/2019 tình trạ...	5	14/09/2019	{BATTERY#Negative};{GENERAL#Positive};{OTHERS};
2	Sao lúc gọi điện thoại màn hình bị chấm nhỏ nh...	3	17/08/2020	{FEATURES#Negative};
3	Moi người cập nhật phần mềm lại , nó sẽ bớt tố...	3	29/02/2020	{FEATURES#Negative};{BATTERY#Neutral};{GENERAL...
4	Mới mua Sài được 1 tháng thấy pin rất trâu, Sà...	5	4/6/2020	{BATTERY#Positive};{PERFORMANCE#Positive};{SER...

Đối với n_star: 1 là cấp độ thấp nhất, thể hiện cho bình luận tiêu cực, 5 là cấp độ cao nhất, thể hiện bình luận tích cực.



TIỀN XỬ LÝ DỮ LIỆU

Tạo các hàm chức năng xử lý cơ bản:



TIỀN XỬ LÝ DỮ LIỆU

Tách từ

- word_tokenize

Phân chia dữ liệu

- Train: 80%
- Test: 20%

Vector hóa dữ liệu

- TF-IDF Vectorizer
- CountVectorizer

TIỀN XỬ LÝ DỮ LIỆU

1. Loại bỏ các cột không cần thiết

```
df.drop(['index', 'date_time', 'label'], axis=1, inplace=True)
```

Tên thuộc tính	Kiểu dữ liệu	Mô tả	Ghi chú
comment	object	các bình luận của khách hàng và người tiêu dùng	
n_star	int64	đánh giá của khách hàng và người tiêu dùng	Đánh giá theo thang điểm 1-5

TIỀN XỬ LÝ DỮ LIỆU

2. Chuẩn hóa văn bản tiếng Việt

Chuẩn hóa văn bản tiếng Việt, để chuẩn hóa kiểu gõ dấu theo kiểu cũ, chuẩn hóa bảng mã Unicode để thống nhất với nhau.

3. Xử lý lấy âm tiết

```
def handle_repeated_syllables(text):
    # Sử dụng regex để tìm các từ có âm tiết lặp lại (ví dụ: quááááá)
    repeated_syllables_pattern = re.compile(r'(\w+?)\1+', re.UNICODE)
    # Hàm xử lý việc loại bỏ âm tiết lặp lại
    def handle_repetition(match):
        word = match.group(1)
        # Giữ lại chỉ một phần lặp lại và thêm vào từ gốc
        return word
```

TIỀN XỬ LÝ DỮ LIỆU

4. Loại bỏ các common token

Loại bỏ các common token như PHONE, MENTION, NUMBER, DATETIME, ... Đối với bộ dữ liệu này, đây là những thông tin không cần thiết nên bỏ đi bằng cách sử dụng regex.

```
def replace_common_token(txt):
    txt = re.sub(EMAIL, ' ', txt)
    txt = re.sub(URL, ' ', txt)
    txt = re.sub(MENTION, ' ', txt)
    txt = re.sub(DATETIME, ' ', txt)
    txt = re.sub(NUMBER, ' ', txt)
    txt = re.sub(PRICE, ' ', txt)
    return txt
```

TIỀN XỬ LÝ DỮ LIỆU

5. Loại bỏ dấu câu và ký tự xuống dòng

Bỏ dấu câu và các ký tự xuống dòng bằng cách sử dụng regex và string.punctuation.

```
def remove_unnecessary_characters(text):
    RE_CLEAR = re.compile("[\n\r]+")# Thay thế các chuỗi xuống dòng (\n
hoặc \r) bằng một ký tự trắng
    text = re.sub(RE_CLEAR, ' ', text)
    # Sử dụng string.punctuation để lấy tất cả các ký tự dấu câu
    translator = str.maketrans('', '', string.punctuation)
    # Loại bỏ dấu câu từ văn bản sử dụng bảng dịch (translator)
    text = text.translate(translator)
return text
```

TIỀN XỬ LÝ DỮ LIỆU

6. Chuẩn hóa các từ viết tắt cơ bản

Tải bộ từ viết tắt teencode tham khảo (bao gồm 1 cột từ viết tắt và 1 cột từ gốc) và sau đó thay những từ viết tắt của dữ liệu có trong bộ từ viết tắt bằng từ gốc trong bộ từ viết tắt.

```
def normalize_acronyms(text, teencode_file=folder+'/Data/teencode.xlsx'):
    # Đọc dữ liệu từ tệp Excel teencode.xlsx
    teencode_df = pd.read_excel(teencode_file, header=None,
names=['teencode', 'replace'])
    words = []
    for word in text.strip().split():
        word = word.strip(string.punctuation)
        # Tìm kiếm trong teencode_df và thay thế
        replacement = teencode_df.loc[teencode_df['teencode'].str.lower() == word, 'replace'].values
        if len(replacement) > 0:
            words.append(replacement[0])
        else:
            words.append(word)
    return ' '.join(words)
```

TIỀN XỬ LÝ DỮ LIỆU

7. Xử lý các từ viết dính nhau

```
def stuck_words(text):
    for i in range(1,len(text)):
        word1 = text[:i]
        word2 = text[i:]
        if word1 in word_dict and word2 in word_dict:
            text_after = word1 + ' ' + word2
            return text_after
    break
# Gán các giá trị có thể tách ra vào cột mới stuck_word
not_vietnamese_than_6['stuck_word'] = float('nan')
for i in not_vietnamese_than_6['word']:
    not_vietnamese_than_6.loc[not_vietnamese_than_6['word'] == i,
    'stuck_word'] = stuck_words(i)
```

```
def normalize_stuck(text, df_stuck_words=df_stuck_words):
    words = []
    for word in text.strip().split():
        word = word.strip(string.punctuation)
        # Tìm kiếm trong teencode_df và thay thế
        replacement =
        df_stuck_words.loc[df_stuck_words['word'].str.lower() == word,
        'stuck_word'].values
        if len(replacement) > 0:
            words.append(replacement[0])
        else:
            words.append(word)

    return ' '.join(words)
```

TIỀN XỬ LÝ DỮ LIỆU

8. Loại bỏ các stopword

```
def get_stopwords(documents, threshold=3):
    """
    :param documents: list of documents
    :param threshold:
    :return: list of words has idf <= threshold
    """

    tfidf = TfidfVectorizer(min_df=100)
    tfidf_matrix = tfidf.fit_transform(documents)
    features = tfidf.get_feature_names_out()
    stopwords = []
    print(min(tfidf.idf_), max(tfidf.idf_), len(features))
    for index, feature in enumerate(features):
        if tfidf.idf_[index] <= threshold:
            stopwords.append(feature)
    return stopwords
```

```
def remove_stopwords(line):
    words = []
    for word in line.strip().split():
        if word not in stopwords:
            words.append(word)
    return ' '.join(words)
```

TIỀN XỬ LÝ DỮ LIỆU

9. Text_preprocess tổng hợp các hàm chức năng

```
def text_preprocess(text):
    # 1. Chuẩn hóa văn bản tiếng việt
    text = text_normalize(text)
    # 2. Xử lý lấy âm tiết
    text = handle_repeated_syllables(text)
    # 3. Loại bỏ các common token
    text = replace_common_token(text)
    # 4. Xóa bỏ dấu câu
    text = remove_unnecessary_characters(text)
    # 5. Đưa về lower
    text = text.lower()
    # 7. Chuẩn hóa các từ viết tắt cơ bản
    text = normalize_acronyms(text)
    # 8. Tách các từ viết dính nhau
    text = normalize_stuck(text)
    # 9. Loại bỏ các stopword tiếng Việt
    text = remove_stopwords(text)
    # 10. Loại bỏ các khoảng trắng liên tiếp
    RE_CLEAR = re.compile("\s+")
    # Các khoảng trắng liên tiếp
    text = re.sub(RE_CLEAR, ' ', text)
return text
```

TIỀN XỬ LÝ DỮ LIỆU

10. Tách từ trong câu bằng lệnh `word_tokenize`

```
def word_segmentation(text):
    text = underthesea.word_tokenize(text, format="text")
    return text
```

XÂY DỰNG MÔ HÌNH

Xây dựng mô hình bằng thuật toán máy học Naive Bayes

Sử dụng MultinomialNB

```
# Khởi tạo mô hình MultinomialNB
naive_bayes_classifier = MultinomialNB()

# Thiết lập các tham số cần tìm kiếm
parameters = {
    'force_alpha' : (True, False),
    'fit_prior': (True, False),
    'alpha': (1, 0.1, 0.01)
}

# Khởi tạo GridSearchCV với mô hình, tham số và số lượng fold trong
# cross-validation (cv)
grid_search_nb = GridSearchCV(estimator=naive_bayes_classifier,
param_grid=parameters, cv=5, scoring='accuracy')

# Huấn luyện GridSearchCV trên dữ liệu
grid_search_nb.fit(x_train, y_train)

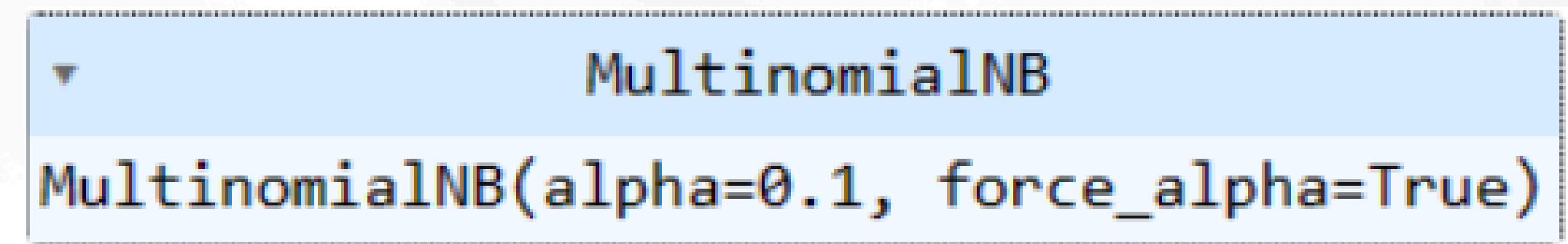
# In ra các tham số tốt nhất
print("Best Parameters:", grid_search_nb.best_params_)

Best Parameters: {'alpha': 0.1, 'fit_prior': True, 'force_alpha': True}
```

XÂY DỰNG MÔ HÌNH

Xây dựng mô hình bằng thuật toán máy học Naive Bayes

Sử dụng bộ tham số đó đưa vào mô hình



XÂY DỰNG MÔ HÌNH

Đánh giá hiệu suất của mô hình

Sử dụng hàm *classification_report*

```
# Dự đoán trên tập test  
y_pred_nb = nb_model.predict(x_test)  
# Đánh giá mô hình bằng các chỉ số  
report1 = metrics.classification_report(y_test, y_pred_nb, digits=3)  
print(report1)
```

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

1	0.557	0.731	0.633	279
2	0.364	0.034	0.063	117
3	0.355	0.270	0.307	222
4	0.306	0.182	0.228	286
5	0.746	0.906	0.819	876

accuracy			0.626	1780
macro avg	0.466	0.425	0.410	1780
weighted avg	0.572	0.626	0.581	1780

XÂY DỰNG MÔ HÌNH

Đánh giá hiệu suất của mô hình

Thực hiện *cross validation* để đánh giá biến động

```
#CROSS VALIDATION
cross_score = cross_val_score(nb_model, X_train,y_train, cv=10)
cross_score

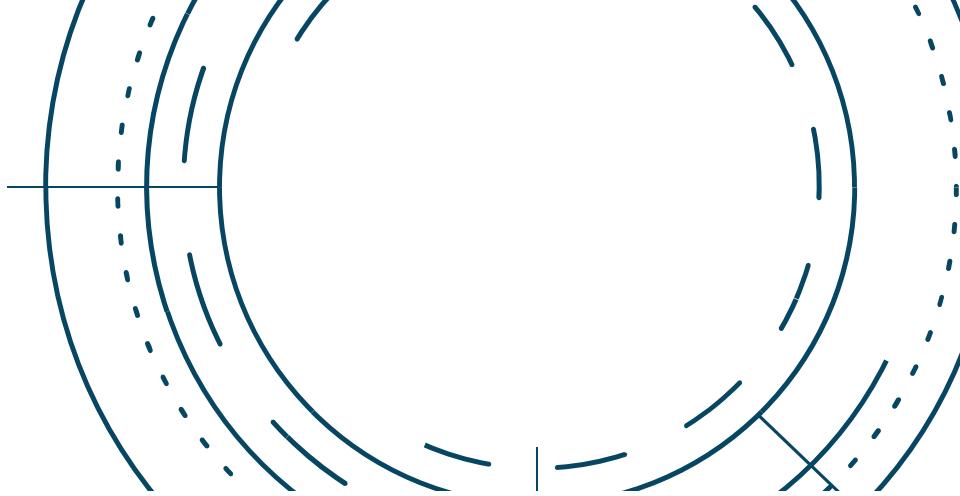
array([0.5744382 , 0.57022472, 0.55758427, 0.56460674, 0.56601124,
       0.56320225, 0.55617978, 0.56320225, 0.5625879 , 0.58931083])
```

CHIA TẬP DỮ LIỆU

Sử dụng cùng tập dữ liệu X_train, y_train, X_test, y_test với mô hình máy học Naive Bayes.

LOGISTIC REGRESSION

Huấn luyện một mô hình phân loại đa lớp từ các vector đặc trưng và thực hiện Grid Search để tìm ra bộ tham số tốt nhất cho mô hình.

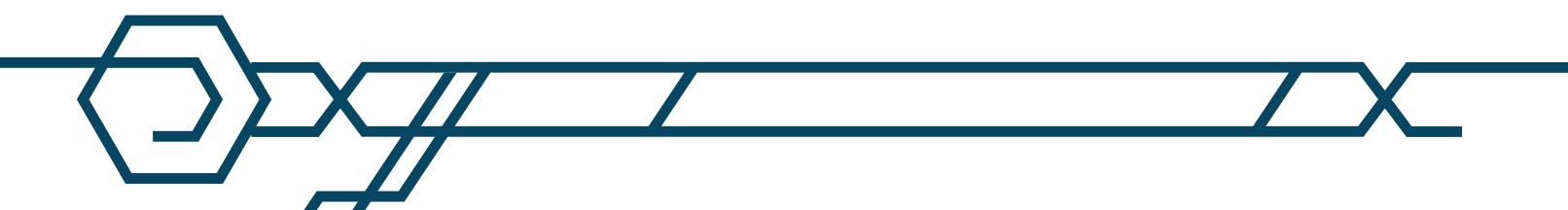


```
# Tạo mô hình Logistic Regression (Softmax Regression)
model = LogisticRegression()

# Định nghĩa các tham số cần tối ưu
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100],
              'penalty': ['l2'],
              'multi_class': ['multinomial'], # xác định rằng đang thực hiện phân loại đa lớp (softmax regression).
              'solver' : ['lbfgs','sag','saga','newton-cg']}
}

# Tạo đối tượng GridSearchCV
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5,
                           scoring='accuracy')

# Thực hiện grid search trên tập huấn luyện
```





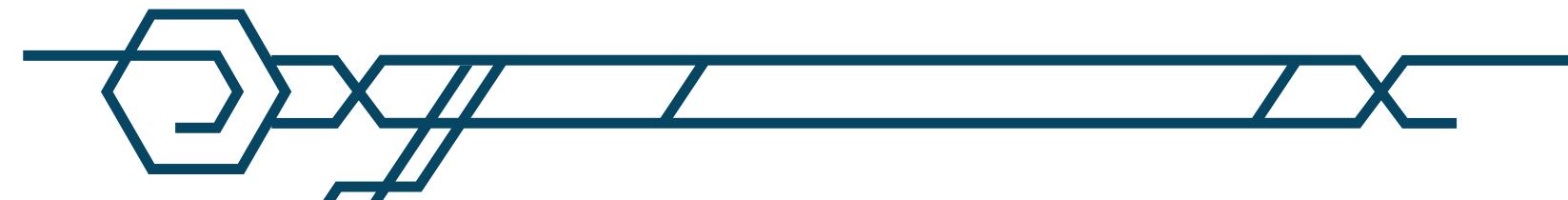
```
grid_search.fit(X_train, y_train)

# Hiển thị kết quả tối ưu
print("Best parameters: ", grid_search.best_params_)

Best parameters: {'C': 1, 'multi_class': 'multinomial', 'penalty': 'l2',
'solver': 'lbfgs'}
```

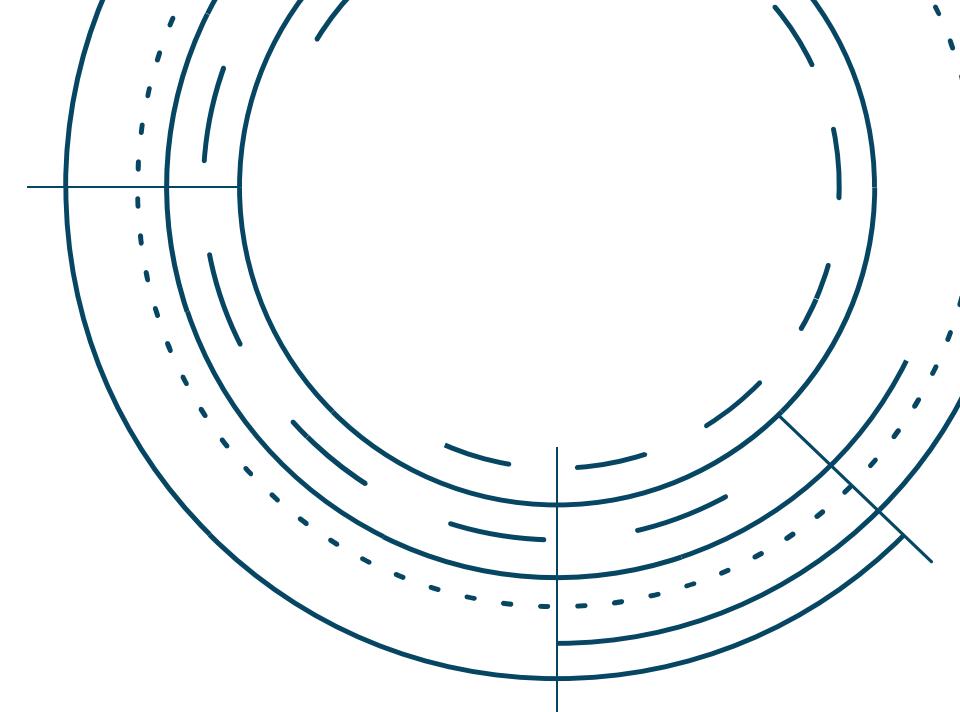
Sử dụng bộ tham số đó đưa vào mô hình:

```
LogisticRegression
LogisticRegression(C=1, multi_class='multinomial')
```



```
# Dự đoán trên tập kiểm tra  
y_pred_lr = maxent_model.predict(x_test)  
# Đánh giá mô hình bằng các chỉ số  
report1 = metrics.classification_report(y_test, y_pred_lr, digits=3)  
print(report1)
```

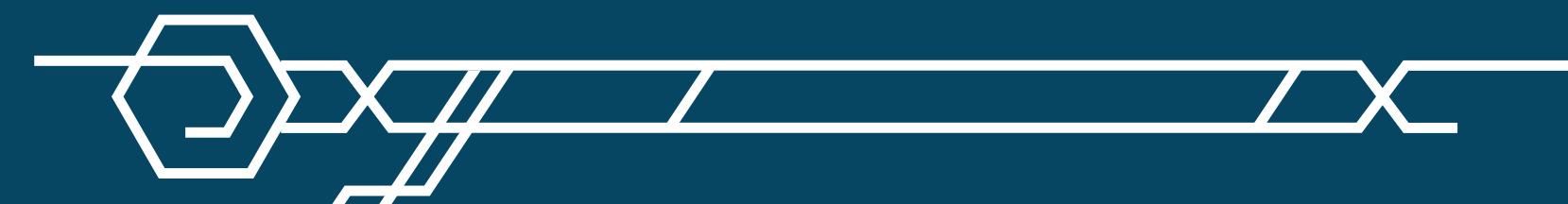
	precision	recall	f1-score	support
1	0.572	0.699	0.629	279
2	0.312	0.043	0.075	117
3	0.342	0.248	0.287	222
4	0.351	0.206	0.260	286
5	0.734	0.917	0.815	876
accuracy			0.628	1780
macro avg	0.462	0.422	0.413	1780
weighted avg	0.570	0.628	0.582	1780



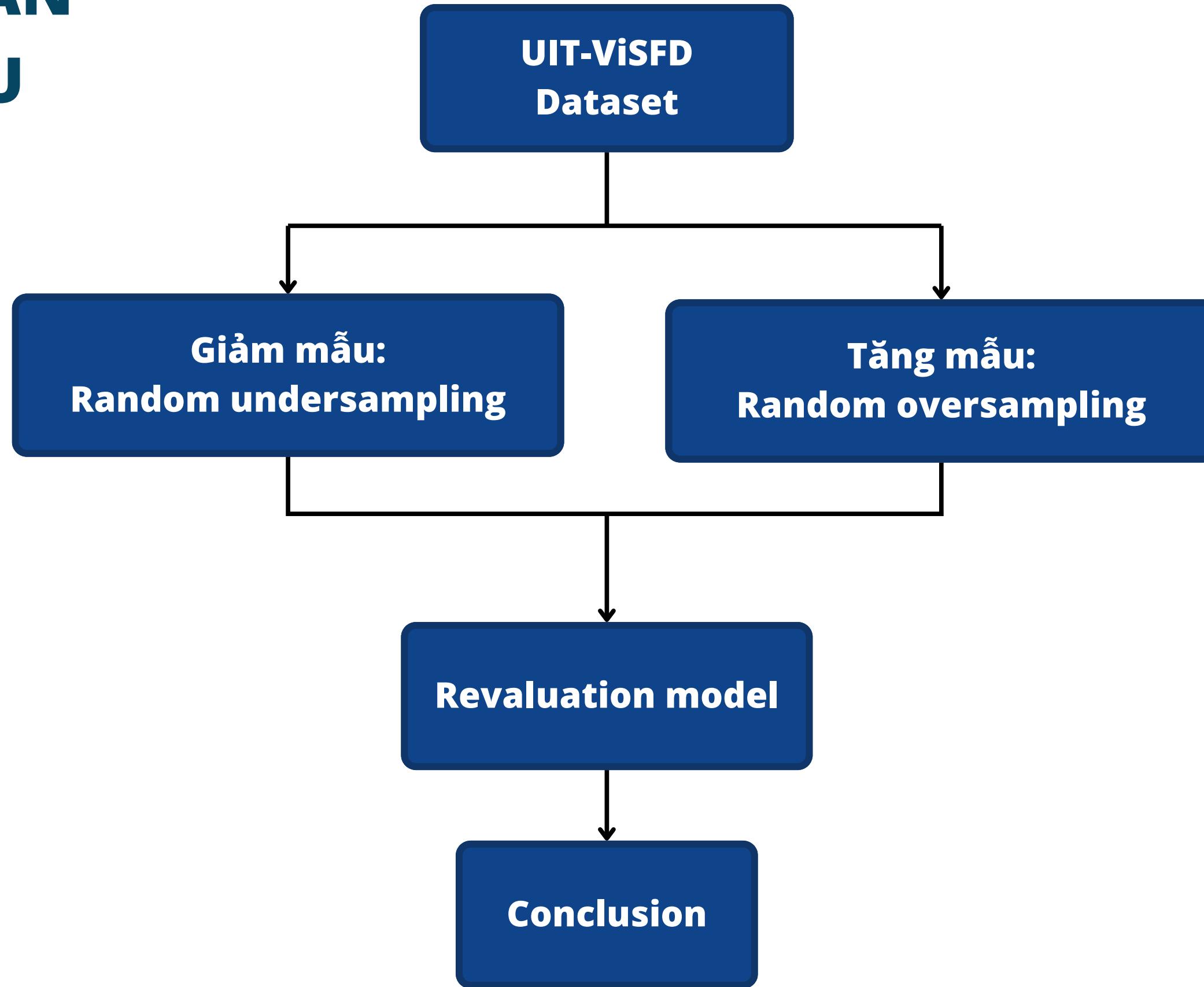
Thực hiện *cross validation* để đánh giá biến động

```
#CROSS VALIDATION
cross_score = cross_val_score(maxent_model, x_train,y_train, cv=10)
cross_score

array([0.60252809, 0.61376404, 0.58707865, 0.59691011, 0.5997191 ,
       0.59269663, 0.58426966, 0.6011236 , 0.59634318, 0.604782 ])
```



XỬ LÝ MẤT CÂN BẰNG DỮ LIỆU



GIẢM MẪU

Sử dụng **RandomUnderSampling** để giảm kích thước mẫu.
Thu được kết quả với **3240** dòng, trong đó mỗi đánh giá là **648** dòng.

Revaluation model result:

Classification Report for Naive Bayes model:				
	precision	recall	f1-score	support
1	0.540	0.478	0.507	157
2	0.311	0.318	0.315	132
3	0.291	0.320	0.305	122
4	0.349	0.319	0.333	116
5	0.604	0.669	0.635	121
accuracy			0.423	648
macro avg	0.419	0.421	0.419	648
weighted avg	0.424	0.423	0.423	648

Classification Report for Logistic Regression model:				
	precision	recall	f1-score	support
1	0.584	0.510	0.544	157
2	0.311	0.288	0.299	132
3	0.264	0.320	0.289	122
4	0.277	0.267	0.272	116
5	0.550	0.587	0.568	121
accuracy			0.400	648
macro avg	0.397	0.394	0.394	648
weighted avg	0.407	0.400	0.402	648

TĂNG MẪU

Sử dụng **RandomOverSampling** để tăng kích thước mẫu.

Thu được kết quả với **20925** dòng, trong đó mỗi đánh giá là **4185** dòng

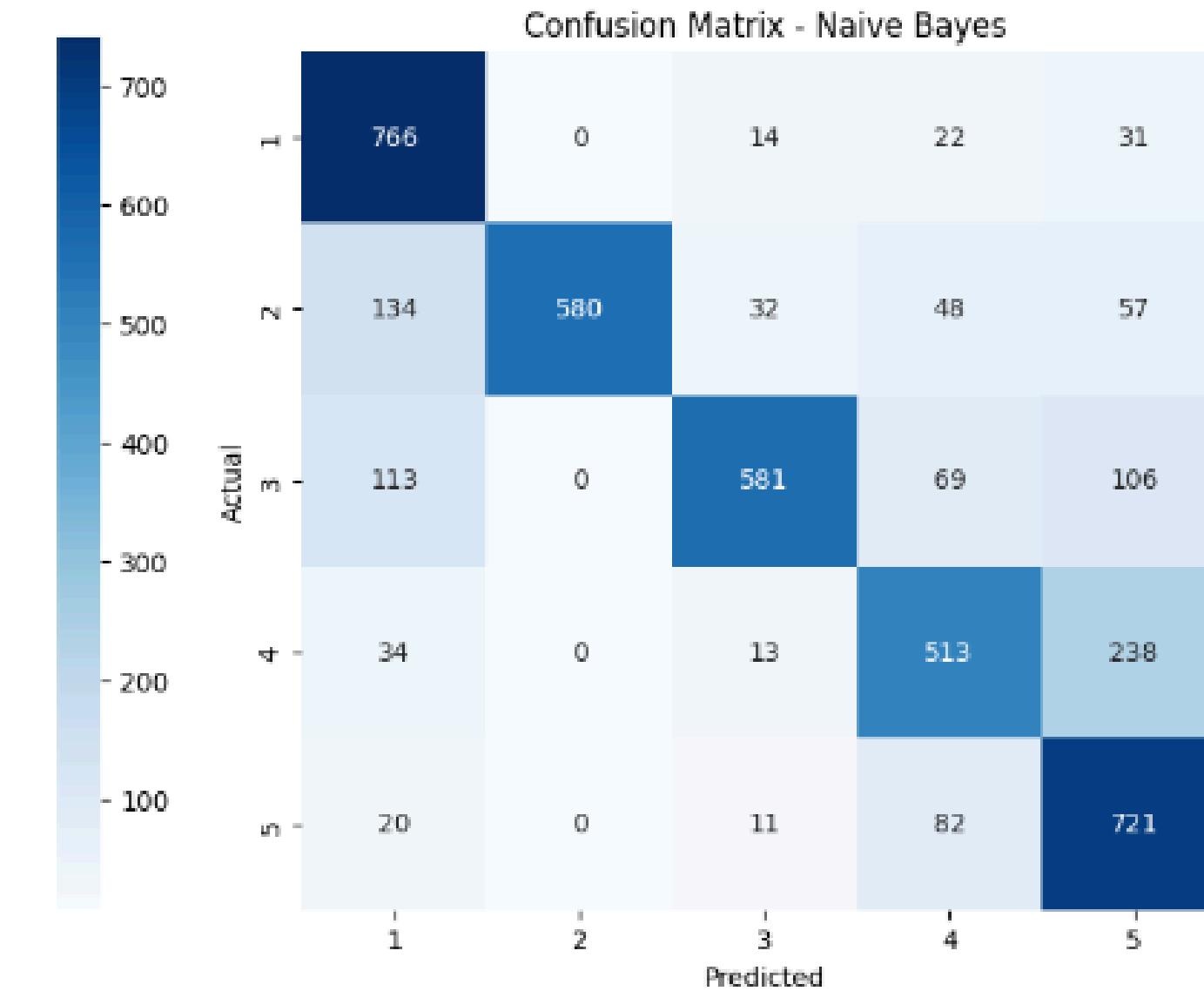
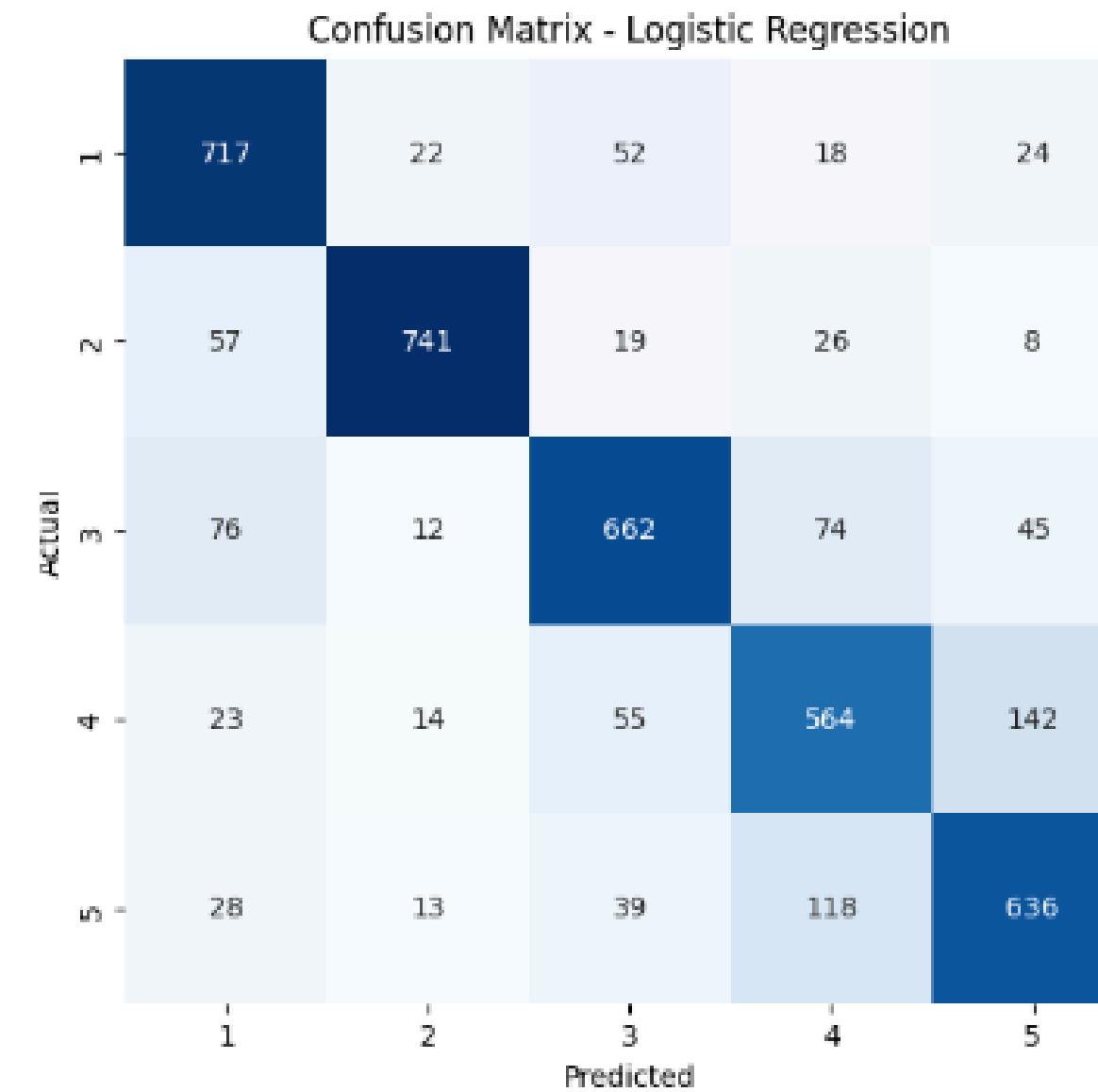
Revaluation model result:

Classification Report for Naive Bayes model:				
	precision	recall	f1-score	support
1	0.749	0.906	0.820	833
2	1.000	0.686	0.814	851
3	0.875	0.695	0.775	869
4	0.707	0.630	0.667	798
5	0.617	0.881	0.726	834
accuracy			0.760	4185
macro avg	0.790	0.760	0.760	4185
weighted avg	0.792	0.760	0.761	4185

Classification Report for Logistic Regression model:				
	precision	recall	f1-score	support
1	0.768	0.852	0.808	833
2	0.943	0.839	0.888	851
3	0.785	0.757	0.771	869
4	0.719	0.707	0.713	798
5	0.731	0.772	0.751	834
accuracy			0.786	4185
macro avg	0.789	0.785	0.786	4185
weighted avg	0.790	0.786	0.787	4185

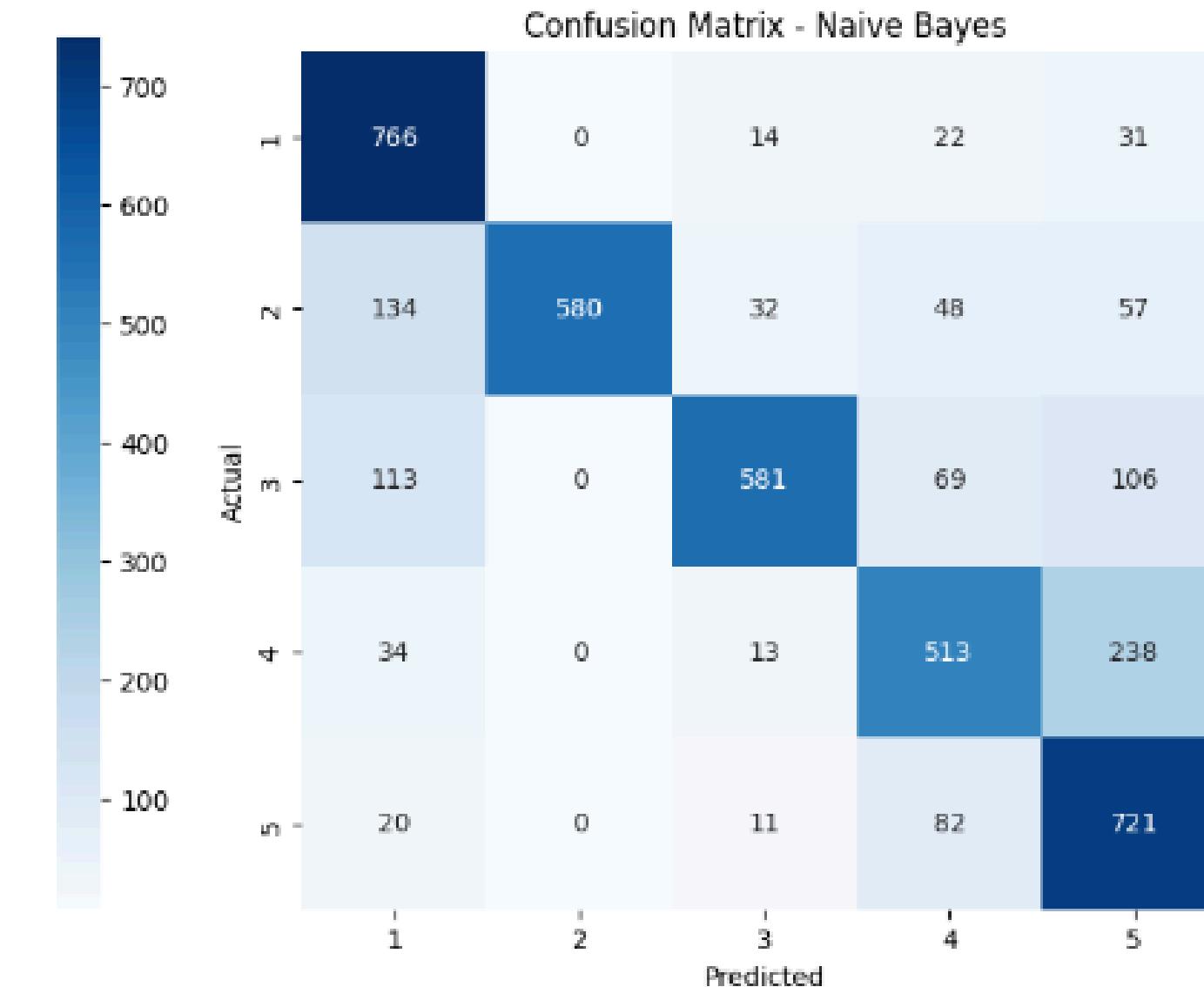
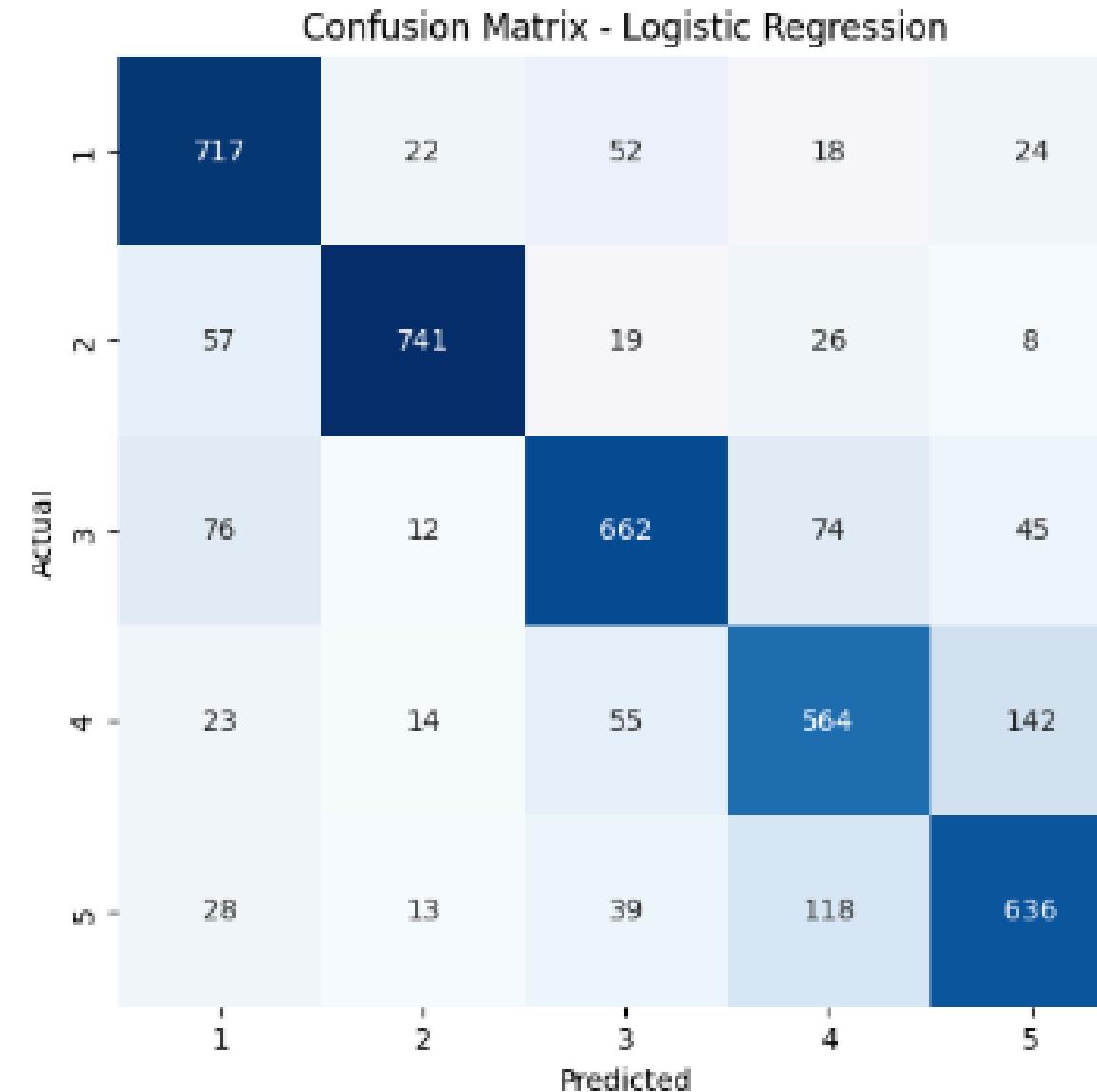
ĐÁNH GIÁ MÔ HÌNH

Sử dụng ma trận nhầm lẫn đánh giá cho 2 mô hình



ĐÁNH GIÁ MÔ HÌNH

Sử dụng ma trận nhầm lẫn đánh giá cho 2 mô hình



→ Chọn Logistic Regression

XÂY DỰNG GIAO DIỆN

Giao diện có chức năng cho phép người dùng

- Nhập vào bình luận về sản phẩm và phân loại bình luận đó được bao nhiêu sao.
- Nhấn ví dụ để xem kết quả phân loại.

Bạn thấy sản phẩm điện thoại này như thế nào

Kết quả đánh giá

Submit

Đánh giá ví dụ

≡ Examples

Tôi thấy sản phẩm này đẹp

Cấu hình máy mạnh

Máy chạy quá chậm

CÀI ĐẶT HỆ THỐNG

Bước 1. Tạo thư mục có chứa đồ án:

Có 2 cách để tạo thư mục.

- **Cách 1:** Tạo thư mục mới, đặt tên ***python_app*** trên máy chủ.
- **Cách 2:** Nhập câu lệnh ***mkdir*** trên Windows Powershell.

Kiểm tra thư mục ***python_app*** đã có trong máy chủ hay chưa.

Sau đó di chuyển vào trong thư mục đồ án bằng lệnh ***cd python_app*** trên Windows Powershell.

CÀI ĐẶT HỆ THỐNG

Bước 2. Thêm các tệp cần thiết đưa vào thư mục:

Thực hiện thêm các tệp sau :

stopword_train.txt - danh sách các stopword được tổng hợp

balanced_df.csv - bộ dữ liệu sau khi đã tiền xử lý và xử lý mất cân bằng.

maxent_model.pkl - mô hình máy học Logistic Regression đã được xuất ra

teencode.xlsx - bộ từ viết tắt teencode được tổng hợp từ tài liệu tham khảo.

CÀI ĐẶT HỆ THỐNG

Bước 3. Tạo tệp requirements.txt:

Tệp ***requirements.txt*** chứa danh sách tất cả các thư viện và phiên bản cụ thể mà dự án cần.

Nội dung của tệp ***requirements.txt***

- underthesea
- openpyxl
- gradio
- joblib

CÀI ĐẶT HỆ THỐNG

Bước 4. Tạo file *python_app.py*:

Tạo file *python_app.py*, có 2 cách để tạo:

- Trong VS Code, chọn New File ... rồi đặt tên “*python_app.py*”.
- Nhập câu lệnh ***touch python_app.py*** trên Windows Powershell.

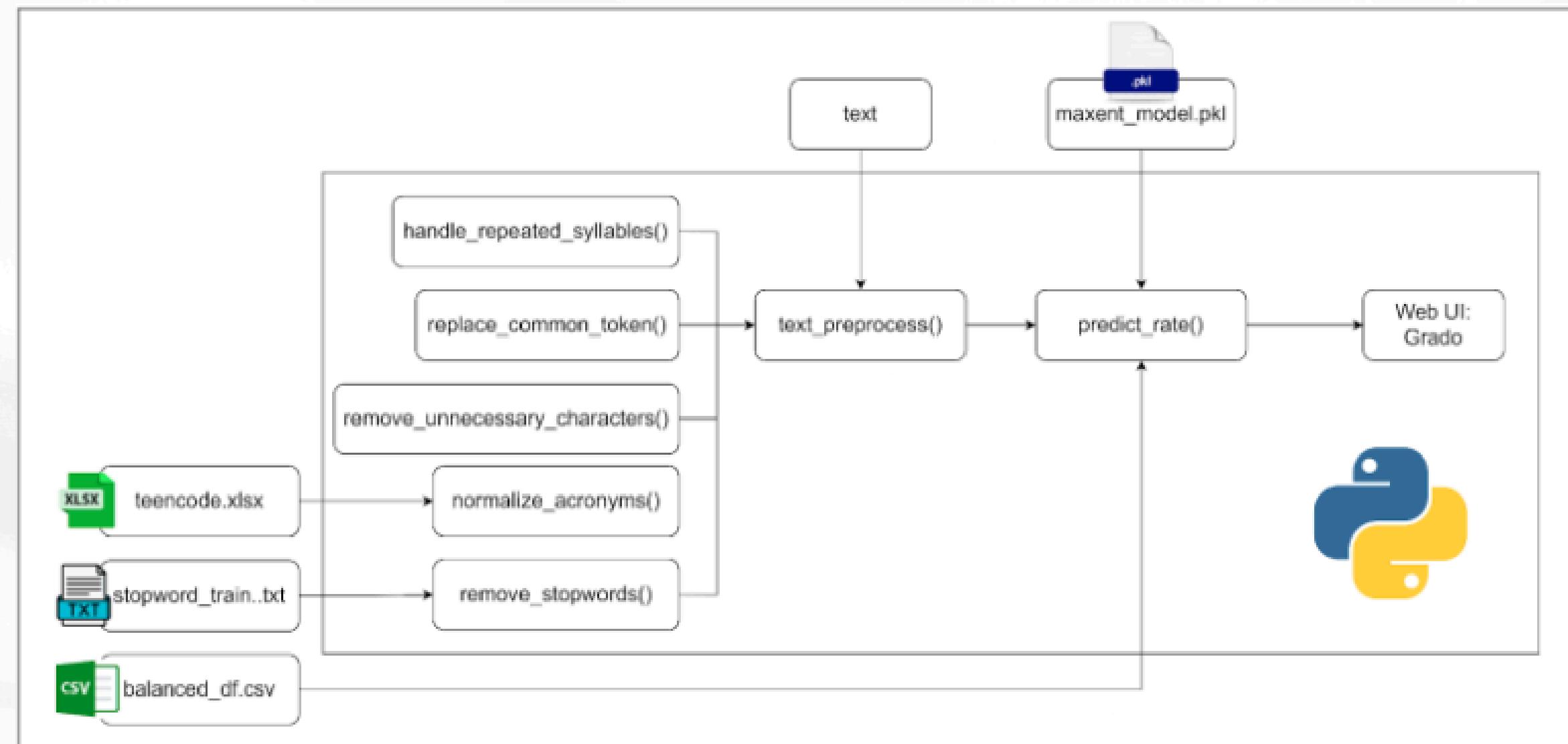
Thực hiện nhập mã trong file *python_app.py* và đoạn mã gồm các thành phần chính sau:

1. Import các thư viện cần thiết
2. Định nghĩa các hàm xử lý văn bản
3. Tiền xử lý văn bản
4. Dự đoán đánh giá sản phẩm:
5. Giao diện người dùng

CÀI ĐẶT HỆ THỐNG

Bước 4. Tạo file *python_app.py*:

Cấu trúc file *python_app.py* như sau:



CÀI ĐẶT HỆ THỐNG

Bước 5. Tạo Dockerfile:

Tạo Dockerfile, có 2 cách để tạo Dockerfile

- Trong VS Code, chọn **New File** ... rồi đặt tên "**Dockerfile**".
- Nhập câu lệnh **touch Dockerfile** trên Windows Powershell

Khai báo thông số cho Dockerfile

- Mở **Dockerfile** vừa tạo lên: có thể mở bằng **note** hoặc **VS Code**
- Nhập nội dung cho Dockerfile

CÀI ĐẶT HỆ THỐNG

Bước 5. Tạo Dockerfile:

Nội dung Dockerfile:

```
FROM python:3.10
WORKDIR /code
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
CMD [ "python", "python_app.py" ]
```

CÀI ĐẶT HỆ THỐNG

Bước 5. Tạo Dockerfile:

Cấu trúc thư mục *python_app*:

```
└── python_app/
    ├── balanced_df.csv
    ├── Dockerfile
    ├── maxent_model.pkl
    ├── python_app.py
    ├── requirements.txt
    ├── stopword_train.txt
    └── teencode.xlsx
```

CÀI ĐẶT HỆ THỐNG

Bước 6. Khởi tạo một image Docker từ Dockerfile trong thư mục hiện tại:

Gõ lệnh ***docker build -t pythonapp*** . trên Windows Powershell.

Như vậy, sau khi lệnh này được thực thi, Docker sẽ đọc Dockerfile từ thư mục hiện tại và sử dụng nó để xây dựng một Docker Image có tên là ***pythonapp***.

Thực hiện kiểm tra đã có ***Image pythonapp*** chưa:

- Gõ lệnh ***docker images*** trên Windows Powershell.
- Hoặc kiểm tra trong Docker

CÀI ĐẶT HỆ THỐNG

Bước 6. Khởi tạo một image Docker từ Dockerfile trong thư mục hiện tại:

```
PS D:\Docker\python_app> docker images
REPOSITORY      TAG          IMAGE ID      CREATED       SIZE
pythonapp        latest        0aa0c979463b    36 hours ago  1.65GB
ubuntu           latest        bf3dc08bfed0    4 weeks ago   76.2MB
busybox          latest        65ad0d468eb1    12 months ago  4.26MB
hello-world      latest        d2c94e258dcb    13 months ago  13.3kB
```

CÀI ĐẶT HỆ THỐNG

Bước 7. Khởi tạo một container Docker từ image **pythonapp**:

Gõ lệnh ***docker run -it -p 7860:7860 pythonapp*** trên Windows Powershell.

Như vậy, sau khi lệnh này được thực thi, Docker sẽ đọc Dockerfile từ thư mục hiện tại và sử dụng nó để xây dựng một Docker Image có tên là ***pythonapp***

Thực hiện kiểm tra đã có container chưa:

- Gõ lệnh ***docker ps*** trên Windows Powershell.
- Hoặc kiểm tra trong Docker

CÀI ĐẶT HỆ THỐNG

Bước 7. Khởi tạo một container Docker từ image pythonapp:

```
PS D:\Docker\python_app> docker run -it -p 7860:7860 pythonapp
Caching examples at: '/code/gradio_cached_examples/5'
Caching example 1/3
/usr/local/lib/python3.10/site-packages/sklearn/base.py:376: InconsistentVersionWarning: Trying to unpickle estimator LogisticRegression from version 1.2.2 when using version 1.5.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model\_persistence.html#security-maintainability-limitations
    warnings.warn(
Caching example 2/3
/usr/local/lib/python3.10/site-packages/sklearn/base.py:376: InconsistentVersionWarning: Trying to unpickle estimator LogisticRegression from version 1.2.2 when using version 1.5.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model\_persistence.html#security-maintainability-limitations
    warnings.warn(
Caching example 3/3
/usr/local/lib/python3.10/site-packages/sklearn/base.py:376: InconsistentVersionWarning: Trying to unpickle estimator LogisticRegression from version 1.2.2 when using version 1.5.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model\_persistence.html#security-maintainability-limitations
    warnings.warn(
Running on local URL: http://0.0.0.0:7860
Running on public URL: https://9950d45d07602283f0.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run 'gradio deploy' from Terminal to deploy to Spaces (https://huggingface.co/spaces)
```

CÀI ĐẶT HỆ THỐNG

Bước 7. Khởi tạo một container Docker từ image pythonapp:

```
Running on local URL: http://0.0.0.0:7860
```

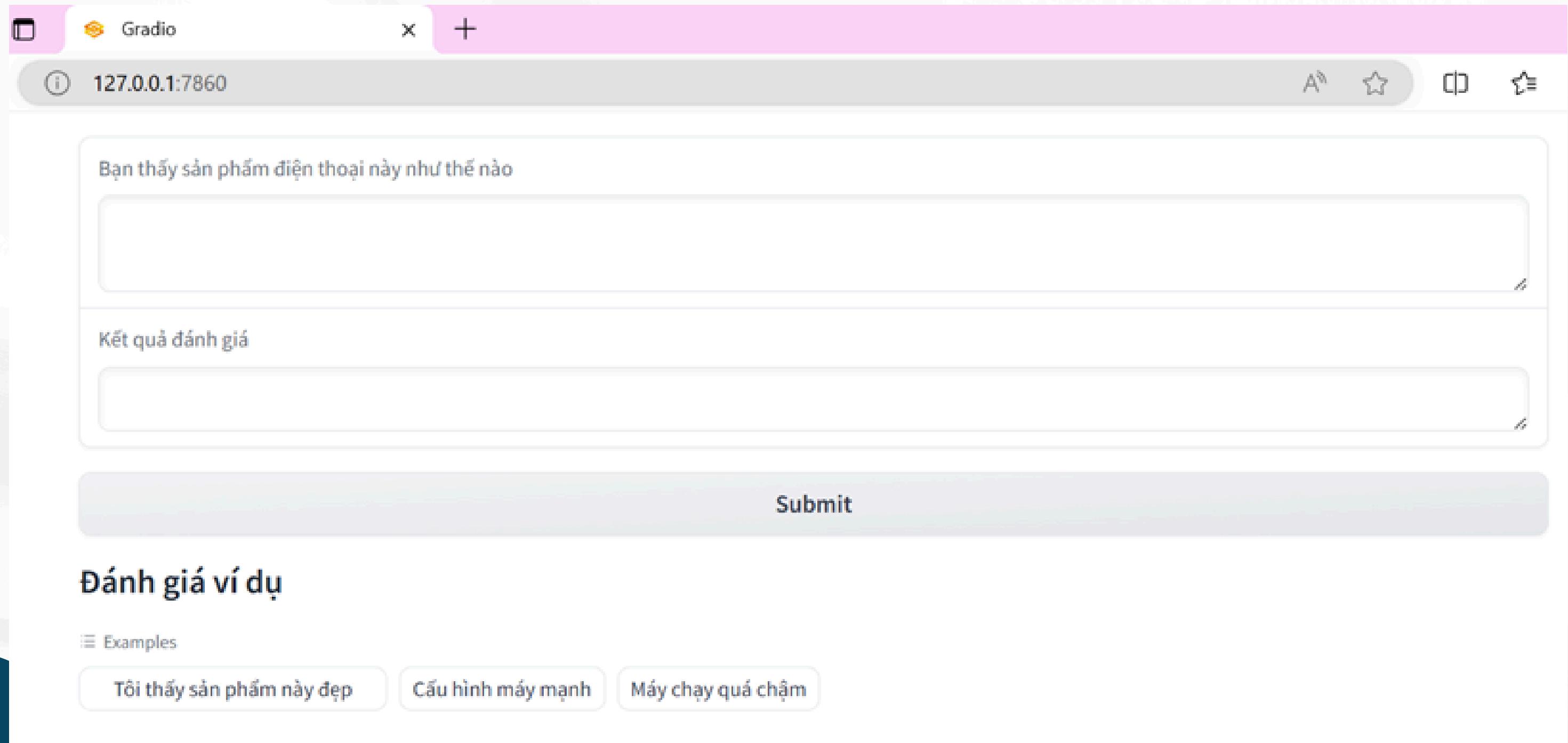
```
Running on public URL: https://9950d45d07602283f0.gradio.live
```

Ứng dụng trên 2 URL:

- Local URL
- Public URL

TRUY CẬP VÀ THỬ NGHIỆM

Mở trình duyệt lên và gõ link miền trang chủ localhost: **http://localhost:7860** hoặc **http://127.0.0.1:7860**.



TRUY CẬP VÀ THỬ NGHIỆM

- Hoàn toàn khen ứng dụng mất 3.2s để chạy.
- Đánh giá 5 sao

The screenshot shows a web application interface for product review. At the top, there's a header bar with a refresh icon, the URL '127.0.0.1:7860', and several small icons for settings and navigation. Below the header, a large text area contains a review text: 'Bạn thấy sản phẩm điện thoại này như thế nào' (How do you feel about this phone product?) followed by a larger block of text: 'Chiếc điện thoại này thật sự tuyệt vời! Màn hình sắc nét, hiệu suất mượt mà và thời lượng pin ấn tượng. Camera chụp ảnh đẹp, đặc biệt là chụp đêm. Thiết kế sang trọng, cầm rất chắc tay. Rất hài lòng với sản phẩm này, đáng đồng tiền bát gạo!' (This phone is truly wonderful! The screen is sharp, the performance is smooth, and the battery life is impressive. The camera takes nice pictures, especially at night. The design is elegant, it holds very well in your hand. I am very satisfied with this product, it is worth the price!). Below this is a section titled 'Kết quả đánh giá' (Review results) which displays a '5 ★' rating. At the bottom right of this section is a large, light-gray 'Submit' button. At the very bottom of the page, there's a heading 'Đánh giá ví dụ' (Example review), a 'Examples' link, and three buttons labeled 'Tôi thấy sản phẩm này đẹp' (I think this product is beautiful), 'Cấu hình máy mạnh' (Strong machine configuration), and 'Máy chạy quá chậm' (The machine runs too slowly).

TRUY CẬP VÀ THỬ NGHIỆM

- **Vừa khen vừa chê ứng dụng mất 3s để chạy.**
- **Đánh giá 4 sao.**

Bạn thấy sản phẩm điện thoại này như thế nào

Điện thoại này có camera chụp ảnh đẹp, thiết kế sang trọng và hiệu suất ổn định. Tuy nhiên, pin hao nhanh và giao diện hơi khó dùng lúc đầu. Tổng thể là sản phẩm tốt nhưng có vài điểm cần cải thiện.

Kết quả đánh giá

4 ★

Submit

Đánh giá ví dụ

≡ Examples

Tôi thấy sản phẩm này đẹp

Cấu hình máy mạnh

Máy chạy quá chậm

TRUY CẬP VÀ THỬ NGHIỆM

- Hoàn toàn chê ứng dụng mất 3s để chạy.
- Đánh giá 1 sao.

Bạn thấy sản phẩm điện thoại này như thế nào

Thất vọng với điện thoại này. Pin kém, màn hình mờ, hay giật lag và camera chụp tệ trong điều kiện thiếu sáng. Thiết kế không chắc tay và dễ bám vân tay. Không đáng giá tiền.

Kết quả đánh giá

1 ★

Submit

Đánh giá ví dụ

≡ Examples

Tôi thấy sản phẩm này đẹp

Cấu hình máy mạnh

Máy chạy quá chậm

TRUY CẬP VÀ THỬ NGHIỆM

comment	Local URL	Public URL
Bình luận toàn khen	3.2 s	4.8 s
Vừa khen vừa chê	3 s	3.8 s
Bình luận toàn chê	3 s	3.8 s

THỬ NGHIỆM KHI TẮT CONTAINER

Tiến hành thử nghiệm khi stop container thì trang web có còn hoạt động hay không.

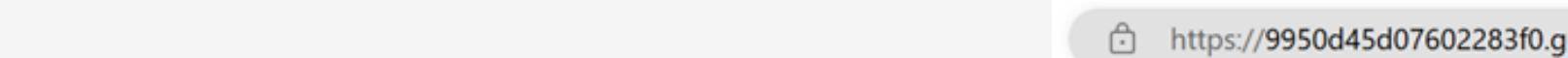
```
PS D:\Docker\python_app> docker stop 135
```

```
PS D:\Docker\python_app> docker ps
```

CONTAINER	ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
-----------	----	-------	---------	---------	--------	-------	-------



localhost:7860



https://9950d45d07602283f0.gradio.live



No interface is running right now

Hmmm... can't reach this page

localhost refused to connect.

Try:

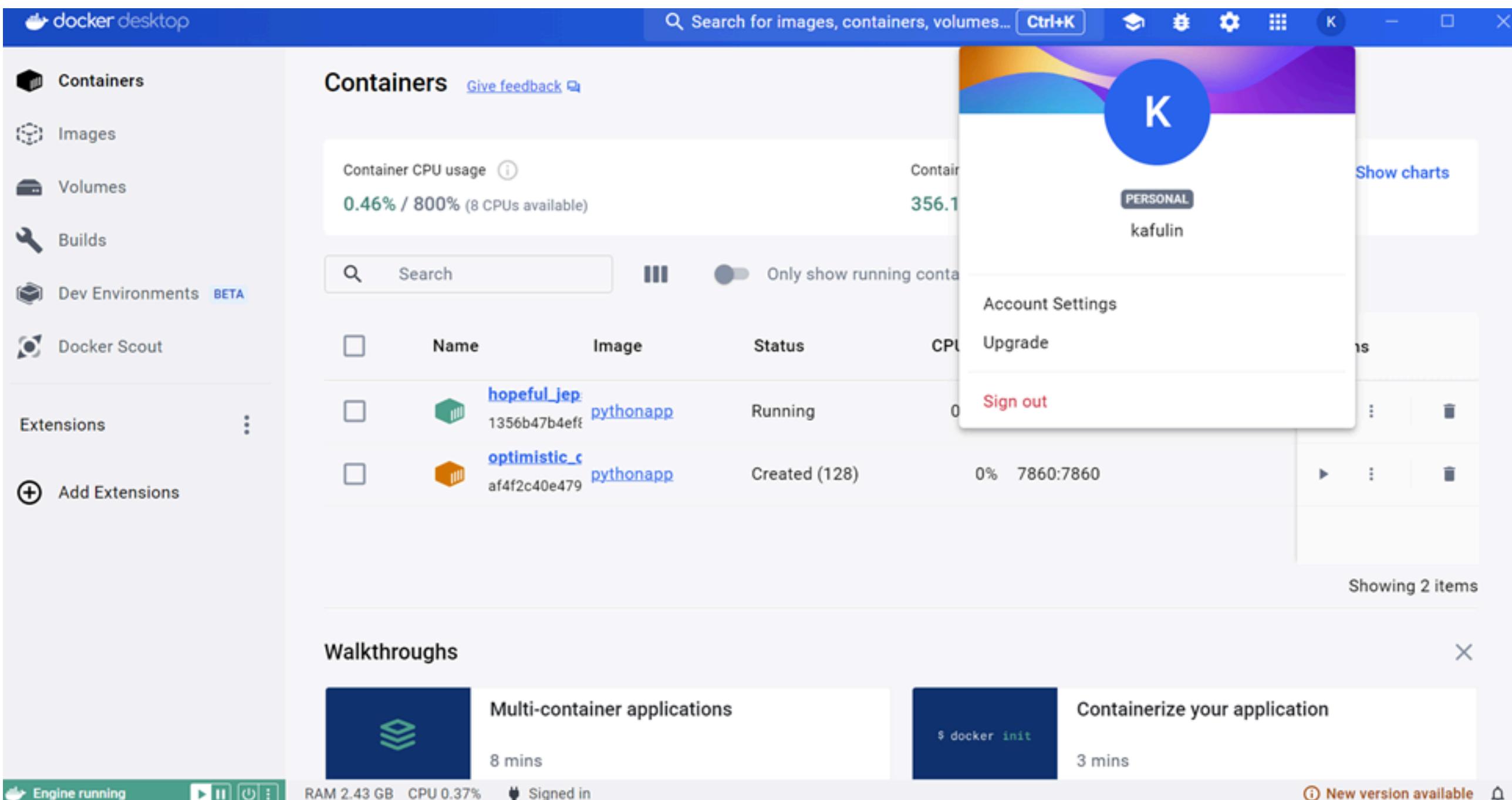
- Search the web for localhost
- Checking the connection
- [Checking the proxy and the firewall](#)

ERR_CONNECTION_REFUSED

Refresh

ĐẨY DOCKER IMAGE LÊN DOCKER HUB

B1: Kiểm tra username trong Docker



ĐẨY DOCKER IMAGE LÊN DOCKER HUB

B2: Đăng nhập vào Docker Hub

```
PS D:\Docker\python_app> docker login  
Authenticating with existing credentials...  
Login Succeeded
```

B3: Kiểm tra ID của image cần sử dụng

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
pythonapp	latest	0aa0c979463b	38 hours ago	1.65GB
ubuntu	latest	bf3dc08bfed0	4 weeks ago	76.2MB
busybox	latest	65ad0d468eb1	12 months ago	4.26MB
hello-world	latest	d2c94e258dcb	13 months ago	13.3kB

Gắn tag cho image

```
PS D:\Docker> docker tag 0aa0c979463b kafulin/pythonapp:latest
```

ĐẨY DOCKER IMAGE LÊN DOCKER HUB

B4: Đẩy image lên Docker Hub

```
PS D:\Docker\python_app> docker push kafulin/pythonapp:latest
The push refers to repository [docker.io/kafulin/pythonapp]
1c49d8afe0fd: Pushed
3c9943024e23: Pushed
1df735381fa7: Pushed
f12a8d06148a: Pushed
b43c1d914be6: Mounted from library/python
0a1f19c0d90d: Mounted from library/python
a8273e3b1197: Mounted from library/python
cbe4fb5e267b: Mounted from library/python
734c0f0b65c2: Mounted from library/node
8845ab872c1c: Mounted from library/node
d7d4c2f9d26b: Mounted from library/node
bbe1a212f7e9: Mounted from library/node
latest: digest: sha256:e7b475580f30ee2f81ea2014720f91625f76c605f3e7992dbae66a19faa09b8a size: 2844
```

Kiểm tra image đã được đẩy lên trên Docker Hub chưa

B1: Truy cập Docker Hub

B2: Kiểm tra Hồ Sơ Người Dùng (Profile)



Lê Trần Khánh Phú [Edit profile](#)

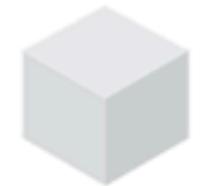
Community User Joined April 8, 2024

[Repositories](#)

Starred

Contributed

Displaying 1 to 1 repositories



[kafulin/pythonapp](#)

By [kafulin](#) • Updated 10 minutes ago

↓1 · ☆0

Kiểm tra image đã được đẩy lên trên Docker Hub chưa

B3: Thử tải Image từ Docker Hub

```
PS D:\Docker\python_app> docker pull kafulin/pythonapp
Using default tag: latest
latest: Pulling from kafulin/pythonapp
c6cf28de8a06: Pull complete
891494355808: Pull complete
6582c62583ef: Pull complete
bf2c3e352f3d: Pull complete
a99509a32390: Pull complete
946285778af4: Pull complete
b2ab5d29389b: Pull complete
d76e704b1be9: Pull complete
c757e6dc09b0: Pull complete
70e301755c28: Pull complete
a9b269a8ea17: Pull complete
be968585335b: Pull complete
Digest: sha256:e7b475580f30ee2f81ea2014720f91625f76c605f3e7992dbae66a19faa09b8a
Status: Downloaded newer image for kafulin/pythonapp:latest
docker.io/kafulin/pythonapp:latest
```

What's Next?

View a summary of image vulnerabilities and recommendations → [docker scout quickview kafulin/pythonapp](#)

B4: Kiểm tra image có trong Docker chưa

```
PS D:\Docker\python_app> docker images
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
kafulin/pythonapp  latest   0aa0c979463b  39 hours ago  1.65GB
```

BÀN LUẬN VÀ ĐÁNH GIÁ TÍNH NĂNG TRUY CẬP VÀO LOCAL URL VÀ PUBLIC URL

1. Hiệu năng truy cập từ local link và public link

Tốc độ phản hồi

- Local link: Phản hồi nhanh hơn (3s) vì không phải đi qua nhiều mạng trung gian.
- Public link: Phản hồi chậm hơn (3.8s) vì yêu cầu phải đi qua mạng internet công cộng

Nguyên nhân

- Độ trễ mạng
- Tải mạng
- Cấu trúc hạ tầng

BÀN LUẬN VÀ ĐÁNH GIÁ TÍNH NĂNG TRUY CẬP VÀO LOCAL URL VÀ PUBLIC URL

2. Vấn đề với public URL thay đổi sau mỗi lần khởi động lại container

Sự thay đổi URL

- Mỗi khi container được khởi động lại, Gradio cung cấp một URL public mới.
- URL thay đổi có thể gây gián đoạn dịch vụ và không thuận tiện cho người dùng hoặc các hệ thống cần truy cập thường xuyên.

Nguyên nhân

- Gradio hoặc dịch vụ tương tự cấp phát URL public mới mỗi lần ứng dụng khởi động lại như một biện pháp bảo mật và quản lý tài nguyên.
- Cơ chế cấp phát động của các dịch vụ này khiến cho URL không cố định.

BÀN LUẬN VÀ ĐÁNH GIÁ TÍNH NĂNG TRUY CẬP VÀO LOCAL URL VÀ PUBLIC URL

3. Đánh giá tổng thể

Ưu điểm

- Local access: Truy cập qua local link có hiệu năng tốt, phản hồi nhanh, phù hợp cho môi trường phát triển và thử nghiệm cục bộ
- Ease of use: Giúp dễ dàng triển khai và quản lý các dịch vụ liên quan đến ứng dụng của bạn

Nhược điểm

- Public access: Phản hồi từ public link chậm hơn do các yếu tố mạng.
- URL dynamics: Việc URL public thay đổi mỗi lần container khởi động lại gây ra sự bất tiện và gián đoạn dịch vụ

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

KẾT LUẬN

- Việc triển khai ứng dụng Python sử dụng Docker Official Image cho dự án `python_app` mang lại nhiều lợi ích đáng kể, bao gồm tính nhất quán của môi trường, dễ quản lý và triển khai, cùng với sự hỗ trợ mạnh mẽ từ cộng đồng. Cấu trúc thư mục rõ ràng và hợp lý giúp dễ dàng quản lý các thành phần của dự án và đảm bảo môi trường phát triển và sản xuất giống nhau. Tuy nhiên, vẫn còn một số vấn đề cần được tối ưu hóa như hiệu quả quản lý dữ liệu lớn, bảo mật mô hình và tối ưu hóa Dockerfile để cải thiện hiệu suất và bảo mật.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

HƯỚNG PHÁT TRIỂN

- Quản lý dữ liệu lớn
- Bảo mật mô hình
- Tối ưu hóa Dockerfile
- Phân chia mã nguồn
- Cải thiện tính liên tục của URL public
- Automate URL update
- Tích hợp CI/CD
- Khả năng mở rộng



THANKS FOR LISTENING

ANY QUESTION?

