

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN MÔN MÁY HỌC**

**PHÂN TÍCH CẢM XÚC CỦA MỘT BÌNH LUẬN  
(FEEDBACK) CỦA KHÁCH HÀNG TRÊN CÁC TRANG  
THƯƠNG MẠI ĐIỆN TỬ**

**GV: PGS.TS LÊ ĐÌNH DUY – THS. PHẠM NGUYỄN TRƯỜNG AN  
LỚP: CS114.K21**

**DANH SÁCH THÀNH VIÊN NHÓM:**

- 1. NGUYỄN ĐỨC THỊNH – 18521442**
- 2. HUỖNH MINH TUẤN – 18521596**
- 3. NGUYỄN MINH THÔNG – 18521459**

# MỤC LỤC

1. MÔ TẢ BÀI TOÁN
2. MÔ TẢ BỘ DỮ LIỆU
3. THỰC HIỆN BÀI TOÁN
4. DỰ ĐOÁN CHO MỘT CÂU MỜI ĐƯỢC NHẬP VÀO

# 1. MÔ TẢ BÀI TOÁN

## ❖ Nội dung:

Nhu cầu mua hàng qua mạng của người dùng ngày càng phổ biến do sự tiện lợi, chi phí rẻ, có nhiều chương trình khuyến mãi hấp dẫn, có thể ngồi ở nhà để xem sản phẩm mà không cần phải đến tận nơi. Tuy nhiên, nhược điểm là người dùng không thể tận mắt đánh giá sản phẩm như mua trực tiếp tại các cửa hàng được. Vì vậy, các feedback của những người dùng đã sử dụng qua sản phẩm đóng vai trò quan trọng trong việc đánh giá chất lượng các sản phẩm tương ứng, các bình luận chủ yếu gồm 2 loại: tích cực hoặc tiêu cực. Tuy nhiên, số lượng các bình luận trên các trang thương mại điện tử rất nhiều, gây khó khăn khi đánh giá từng bình luận bằng tay, vì vậy, các thuật toán máy học sẽ hỗ trợ đắc lực cho việc phân loại này.

# 1. MÔ TẢ BÀI TOÁN

## ❖ Mục đích bài toán:

- Đóng góp cho việc phân tích và khai thác ý kiến, nhất là ý kiến của khách hàng dành cho các doanh nghiệp, cửa hàng, dịch vụ,...nhằm đưa ra dữ liệu cho các hệ thống phản hồi tự động, điều này là rất thiết yếu bởi lẽ đa phần cơ sở kinh doanh nào cũng đều có trang web riêng để chạy theo kịp với sự phát triển của nền công nghệ 4.0 như hiện nay.

- Dựa vào những ý kiến phản hồi đó mà các doanh nghiệp có thể đưa ra các chiến lược, chính sách cải thiện chất lượng phục vụ và sản phẩm, cũng như cải thiện uy tín, chất lượng của chính doanh nghiệp đó.

# 1. MÔ TẢ BÀI TOÁN

## ❖ Input/Output:

- Input: Một câu, từ đánh giá (feedback) bất kỳ bằng tiếng Việt.
- Output: Model dự đoán nhãn cho câu, từ đánh giá (feedback) đó mang ý nghĩa tích cực hay tiêu cực.

## 2. MÔ TẢ BỘ DỮ LIỆU

### 2.1. Cách thức xây dựng

- Thu thập các đánh giá, nhận xét của khách hàng từ các website thương mại điện tử lớn như: [lazada](#), [shopee](#), [Thế giới di động](#),...

- Thu thập dữ liệu dựa trên đặc tính chung của 1 website, các class chứa các element. Vì vậy quá trình crawl diễn ra như sau:

  - + Xác định class chứa comment

  - + Dùng đoạn code Javascript sau (sử dụng cửa sổ Inspect Element) để crawl

```
# let listItem = '';  
for(item of document.getElementsByClassName("<class name>")){  
  listItem = listItem + item.innerText + '\n';  
}
```

- Ngoài ra dữ liệu còn được lấy tại VLSP (Vietnamese Language and Speech Processing).

## 2. MÔ TẢ BỘ DỮ LIỆU

### 2.2. Số lượng

Gồm 8000 dòng dữ liệu (hơn 50% tự crawl còn lại là của VLSP). Dữ liệu crawl được được gán nhãn bằng tay và merge chung thành bộ dataset hoàn chỉnh.

## 3. THỰC HIỆN BÀI TOÁN

### 3.1. Install package và import các thư viện cần thiết:

- Install **joblib** (hỗ trợ lưu lại file trọng số của model) và **pyvi** (hỗ trợ tách từ tiếng Việt)

- Import các thư viện:

```
import pandas as pd
import numpy as np
import re
from gensim.models import KeyedVectors
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
import os
from sklearn.model_selection import GridSearchCV
from sklearn.externals import joblib
import seaborn as sn
import operator
import matplotlib.pyplot as plt
```



# 3. THỰC HIỆN BÀI TOÁN

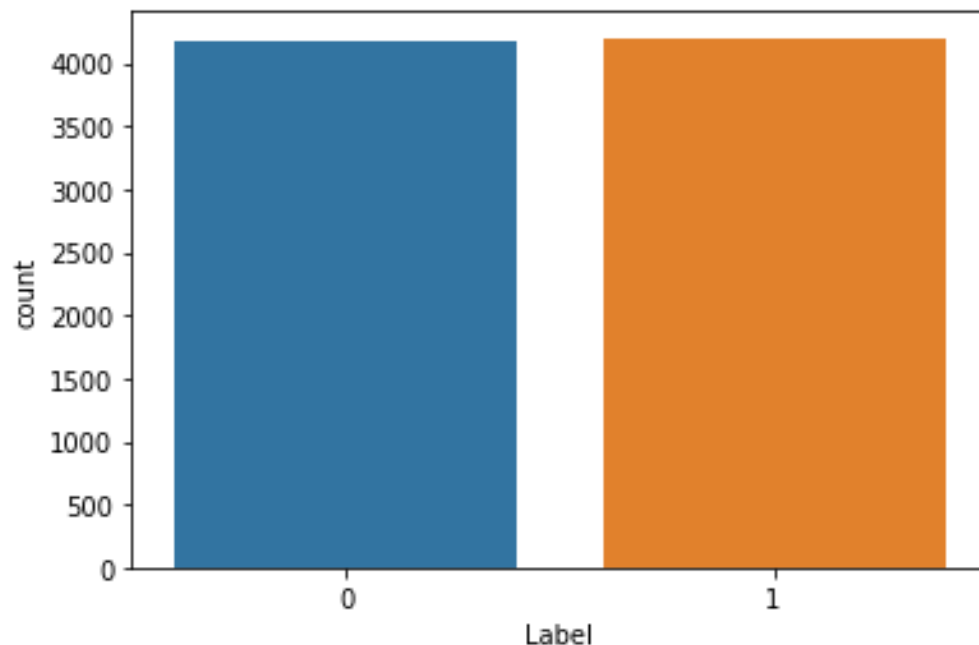
## 3.2. Load dataset:

Load dữ liệu để tiến hành giải quyết bài toán

## 3.3. Trực quan hóa dữ liệu và tiền xử lý

```
sn.countplot(df['Label'])
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f8429c780f0>



*Data thu thập được đã gần như cân bằng*

# 3. THỰC HIỆN BÀI TOÁN

## 3.3. Trục quan hóa dữ liệu và tiền xử lý

- ❖ *Tiền xử lý*: Văn bản đầu vào của chúng ta có thể chứa nhiều ký tự, dấu câu hoặc khoảng trắng thừa, các từ viết tắt, viết hoa, viết sai chính tả... điều này có thể làm ảnh hưởng tới các bước ở sau này nên chúng ta cần phải xử lý nó trước, dữ liệu được tiền xử lý bằng cách loại bỏ các dấu câu, fix lại các từ ngữ người dùng như ok, ko, thnks,... loại bỏ các khoảng trống dư thừa, đưa toàn bộ về chữ thường (lower case), quy đổi một số quy định về dạng ngữ nghĩa.

```

def standardize_data(row):
    row = re.sub(r'([A-Z])\1+', lambda m: m.group(1).upper(), row, flags=re.IGNORECASE)
    row = row.lower()
    replace_list = {
        'òa': 'oà', 'óa': 'oá', 'òà': 'oà', 'õa': 'oã', 'ọa': 'oạ', 'òe': 'oè', 'óe': 'oé', 'òe': 'oè',
        'õe': 'oẽ', 'ọe': 'oẹ', 'ùy': 'uỳ', 'úy': 'uý', 'ũy': 'uỷ', 'ũy': 'uỹ', 'uy': 'uy', 'uà': 'ùa',
        'à': 'à', 'õ': 'õ', 'u': 'u', 'õ': 'õ', 'ò': 'ò', 'ó': 'ó', 'ã': 'ã', 'ă': 'ă', 'á': 'á',
        'ă': 'ă', 'ò': 'ò', 'è': 'è', 'ě': 'ě', 'ă': 'ă', 'ù': 'ù', 'ě': 'ě', 'ò': 'ò', 'i': 'i',
        'è': 'è', 'àk': 'u' à ', 'a': 'à', 'i': 'i', 'ă': 'ă', 'ừ': 'ừ', 'e': 'è', 'y': 'ỹ', 'a': 'á',
        'ô kê': ' ok ', 'okie': ' ok ', ' o kê ': ' ok ',
        'okey': ' ok ', 'ôkê': ' ok ', 'oki': ' ok ', ' oke ': ' ok ', 'okay': ' ok ', 'okê': ' ok ',
        ' tks ': 'u' cảm ơn ', 'thks': 'u' cảm ơn ', 'thanks': 'u' cảm ơn ', 'ths': 'u' cảm ơn ', 'thank': 'u' cảm ơn ',
        '★': 'star ', '*': 'star ', '🌟': 'star ', '👍': 'u' positive ',
        'kg ': 'u' không ', 'not': 'u' không ', 'u' kg ': 'u' không ', '"k ': 'u' không ', 'kh ': 'u' không ', 'kô': 'u' không ', 'hok': 'u' không ', ' kp ': 'u' không phải ',
        'he he': ' positive ', 'hehe': ' positive ', 'hihi': ' positive ', 'haha': ' positive ', 'hjhj': ' positive ',
        ' lol ': ' negative ', ' cc ': ' negative ', 'cute': 'u' dễ thương ', 'huhu': ' negative ', ' vs ': 'u' với ', 'wa': ' quá ', 'wá': 'u' quá ', 'j': 'u' gì ',
        ' sz ': 'u' cỡ ', 'size': 'u' cỡ ', 'u' đx ': 'u' được ', 'dk': 'u' được ', 'dc': 'u' được ', 'đk': 'u' được ',
        'đc': 'u' được ', 'authentic': 'u' chuẩn chính hãng ', 'u' aut ': 'u' chuẩn chính hãng ', 'u' auth ': 'u' chuẩn chính hãng ', 'thick': 'u' positive ', 'store':
        'shop': 'u' cửa hàng ', 'sp': 'u' sản phẩm ', 'gud': 'u' tốt ', 'god': 'u' tốt ', 'wel done': ' tốt ', 'good': 'u' tốt ', 'gút': 'u' tốt ',
        'sầu': 'u' xấu ', 'gut': 'u' tốt ', 'u' tot ': 'u' tốt ', 'u' nice ': 'u' tốt ', 'perfect': 'rất tốt', 'bt': 'u' bình thường ',
        'time': 'u' thời gian ', 'qá': 'u' quá ', 'u' ship ': 'u' giao hàng ', 'u' m ': 'u' mình ', 'u' mik ': 'u' mình ',
        'ế': 'ế', 'product': 'sản phẩm', 'quality': 'chất lượng', 'chat': 'chất ', 'excelent': 'hoàn hảo', 'bad': 'tệ', 'fresh': ' tươi ', 'sad': ' tệ ',
        'date': 'u' hạn sử dụng ', 'hsd': 'u' hạn sử dụng ', 'quickly': 'u' nhanh ', 'quick': 'u' nhanh ', 'fast': 'u' nhanh ', 'delivery': 'u' giao hàng ', 'u' síp ': 'u'
        'beautiful': 'u' đẹp tuyệt vời ', 'u' tl ': 'u' trả lời ', 'u' r ': 'u' rồi ', 'u' shopE ': 'u' cửa hàng ', 'u' order ': 'u' đặt hàng ',
        'chất lg': 'u' chất lượng ', 'u' sd ': 'u' sử dụng ', 'u' dt ': 'u' điện thoại ', 'u' nt ': 'u' nhắn tin ', 'u' tl ': 'u' trả lời ', 'u' sài ': 'u' xài ', 'u' bjo': 'u' b
        'thik': 'u' thích ', 'u' sop ': 'u' cửa hàng ', ' fb ': ' facebook ', ' face ': ' facebook ', ' very ': 'u' rất ', 'u'quả ng ': 'u' quảng ',
        'dep': 'u' đẹp ', 'u' xau ': 'u' xấu ', 'delicious': 'u' ngon ', 'u'hàg': 'u' hàng ', 'u'quà': 'u' quà ',
        'iu': 'u' yêu ', 'fake': 'u' giả mạo ', 'trl': 'trả lời', '><': 'u' positive ',
        ' por ': 'u' tệ ', ' poor ': 'u' tệ ', 'ib': 'u' nhắn tin ', 'rep': 'u' trả lời ', 'u'fback': ' feedback ', 'feedback': ' feedback ',
        #dưới 3* quy về 1*, trên 3* quy về 5*
        '6 sao': ' 5star ', '6 star': ' 5star ', '5star': ' 5star ', '5 sao': ' 5star ', '5sao': ' 5star ',
        'starstarstarstarstar': ' 5star ', '1 sao': ' 1star ', '1sao': ' 1star ', '2 sao': ' 1star ', '2sao': ' 1star ',
        '2 starstar': ' 1star ', '1star': ' 1star ', '0 sao': ' 1star ', '0star': ' 1star ',
    }
    for k,v in replace_list.items():
        row = row.replace(k,v)
    row = row.replace(","," ").replace(".", " ") \
        .replace(";"," ").replace(""," ") \
        .replace(":"," ").replace(""," ") \
        .replace("'", " ").replace("'", " ") \
        .replace("!", " ").replace("?", " ") \
        .replace("-", " ").replace("?", " ")
    row = row.strip()
    return row

```

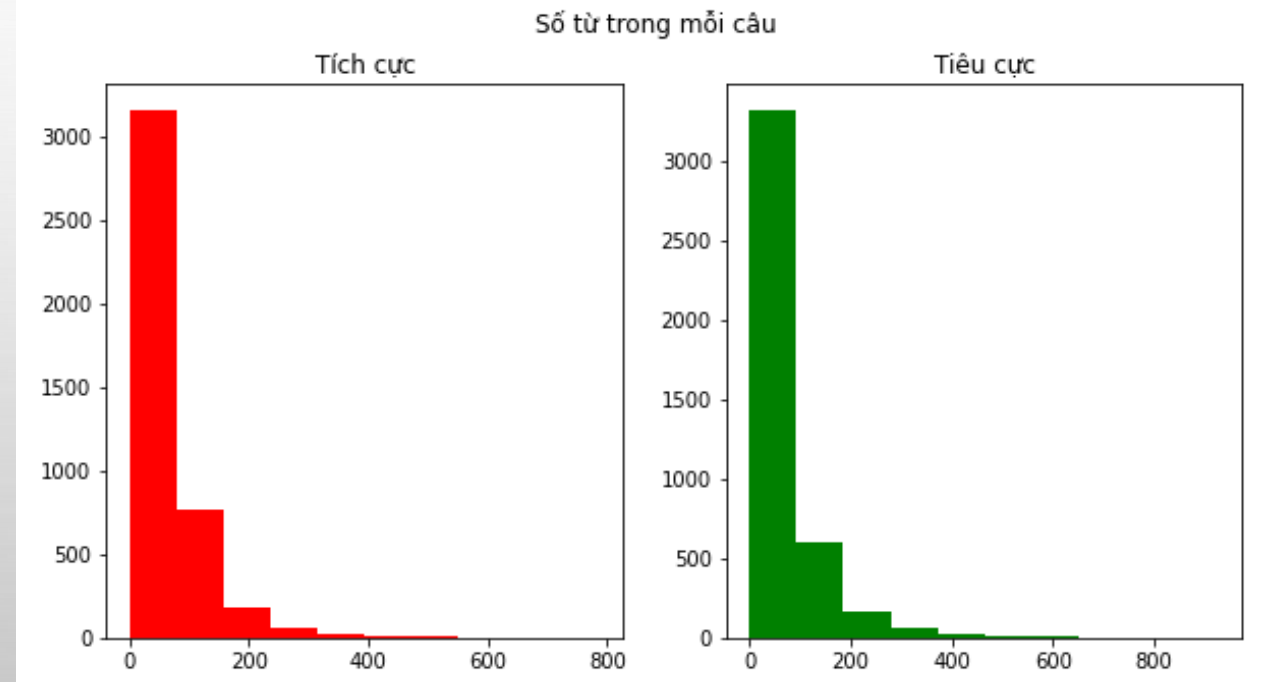
# 3. THỰC HIỆN BÀI TOÁN

## 3.3. Trục quan hóa dữ liệu và tiền xử lý

- ❖ Số lượng từ trong mỗi câu: giờ đây, chúng ta cần kiểm tra sự phân bố của dữ liệu bằng số lượng từ trong mỗi câu
- ❖ Phần lớn các câu trong data chứa ít hơn 100 từ. Một số câu còn lại khá dài từ 100 đến 200 từ (đánh giá sơ bộ số từ trong câu bằng cách tách theo khoảng trắng, sẽ không chính xác cho đến khi bước tách từ thực sự được thực hiện).

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))
text_len = df[df['Label'] == 1]['Text'].str.split().map(lambda x: len(x))
ax1.hist(text_len, color='red')
ax1.set_title('Tích cực')

text_len = df[df['Label'] == 0]['Text'].str.split().map(lambda x: len(x))
ax2.hist(text_len, color='green')
ax2.set_title('Tiêu cực')
fig.suptitle('Số từ trong mỗi câu')
plt.show()
```



# 3. THỰC HIỆN BÀI TOÁN

## 3.4. Word embedding

### ❖ *Word embedding là gì?*

Word Embedding làm một phương pháp biểu diễn một từ cụ thể ở dạng vector. Có 2 phương pháp word embedding là: dựa vào tần số xuất hiện của từ để tạo ra các vector từ (TF-IPF, Count Vector, Co-occurrence Matrix,...), xây dựng các vector từ các mô hình dự đoán (Word2Vec,...).

### ❖ Sử dụng model word2vec tiếng Việt đã được train sẵn của tác giả Vũ Xuân Sơn, dữ liệu train được lấy từ trang báo lớn ở Việt Nam (baomoi.vn), gồm 439056 từ shape(400,). Để load model này ta sử dụng thư viện gensim:

```
from gensim.test.utils import datapath  
wv_from_bin = KeyedVectors.load_word2vec_format(datapath("/content/drive/My Drive/Course_Project_ML/baomoi.model.bin"),
```

# 3. THỰC HIỆN BÀI TOÁN

## 3.4. Word embedding

- ❖ *Tách từ*: Câu trước khi word embedding cần được tách ra thành từng từ. Sử dụng thư viện **pyvi** hỗ trợ tách từ Tiếng Việt

```
[ ] from pyvi import ViTokenizer
```

```
[ ] def tok(row):  
    return ViTokenizer.tokenize(row)
```



# 3. THỰC HIỆN BÀI TOÁN

## 3.4. Word embedding

- ❖ *Word embedding*: Model Word2vec vector hóa mỗi từ thành một vector với dim = 400. Phương pháp của nhóm mình đó chính là lấy các từ trong câu sau khi đã thực hiện tách từ, sau đó dùng mô hình Word2Vec đã được training cho tiếng Việt, chuyển đổi các từ đó sang các vector số thực có chiều dài cố định. Cuối cùng, vector của 1 câu mà nhóm mình chuyển đổi sang sẽ là TỔNG của các vector đại diện cho các từ trong câu đó !

```
X = []
def embedding(row):
    # words_train = X_train.str.split(" ")
    # words_test = X_test.str.split(" ")
    # X_train_train = np.zeros((400))
    # for word_train in words_train:
    #     if word_train in vocab:
    #         X_train_train += w2v.wv[word_train]
    # X_test_test = np.zeros((400))
    # for word_test in words_test:
    #     if word_test in vocab:
    #         X_test_test += w2v.wv[word_test]
    # return X_train_train, X_test_test
    words = row.split(' ')
    vec_row = np.zeros((400))
    for word in words:
        if word in vocab:
            vec_row += wv_from_bin.wv[word]
    vec_row.reshape(1, -1)
    l = vec_row.tolist()
    X.append(l)
```

# 3. THỰC HIỆN BÀI TOÁN

## 3.4. Word embedding

Hàm embedding nhận đầu vào là một text thực hiện tính tổng các vector từ sau đó reshape vector đó về dạng (1,400) rồi đưa vào list global X.

```
[ ] df['Text'] = df.Text.apply(standardize_data)
```

```
[ ] df['Text'] = df.Text.apply(tok)
```

```
[ ] df['Text'] = df.Text.apply(embedding)
```

Sau khi tiến hành thực hiện các hàm trên với bộ dataset ta thu được một list X là các vector đại diện cho bộ dataset của chúng ta.

Nhưng model nhận vào là một numpy array (n\_samples, m\_features) nên một lần nữa ta chuyển list X sang dạng np.array. Khá công kênh !

```
[ ] X = np.array(X)
X.shape
```



(8379, 400)



# 3. THỰC HIỆN BÀI TOÁN

## 3.5. Phân chia dữ liệu

Dữ liệu được chia thành 2 phần train và test với `test_size = 20%`.

```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X, df['Label'], test_size = 0.2, random_state = 42)
```

```
[ ] X_train.shape
```

```
(6703, 400)
```

```
[ ] Y_train.shape
```


```
(6703,)
```

# 3. THỰC HIỆN BÀI TOÁN

## 3.6. Xây dựng model, cài đặt

Chọn model Support Vector Machine. Tiến hành train sơ bộ trên bộ data cho ra base model.

```
[ ] model = SVC(kernel='linear', C=1)
    model.fit(X_train, Y_train)
```

```
 SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Model train mất gần 3h nên để tiện việc sử dụng lần sau ta lưu lại file trọng số của model sau khi train

```
[ ] joblib.dump(model, 'saved_model.pkl')
```

```
 ['saved_model.pkl']
```

# 3. THỰC HIỆN BÀI TOÁN

## 3.6. Xây dựng model, cài đặt

Accuracy của model trên tập train là 86,86%

```
[ ] score = model.score(X_train, Y_train)
    print(score)
```



0.8685663135909294

Accuracy của model trên tập test là 82,1%

```
[ ] score = model.score(X_test, Y_test)
    print(score)
```



0.8210023866348448

# 3. THỰC HIỆN BÀI TOÁN

## 3.6. Xây dựng model, cài đặt

*Thử train trên các mô hình phân lớp khác:*

❖ Logistic regression:

```
[ ] from sklearn.metrics import accuracy_score
    from sklearn.metrics import classification_report
    from sklearn.linear_model import LogisticRegression
    logistic_model = LogisticRegression()
    logistic_model.fit(X_train, Y_train)
    print("- The accuracy of model on training set: {}".format(logistic_model.score(X_train, Y_train)))
    y_pred = logistic_model.predict(X_test)
    print("- The accuracy of model on test set: {}".format(accuracy_score(y_true=Y_test, y_pred=y_pred)))
    print("- The classification report: \n{}".format(classification_report(y_true=Y_test, y_pred=y_pred)))
```



```
- The accuracy of model on training set: 0.8624496494107117
- The accuracy of model on test set: 0.8239856801909308
- The classification report:
```

	precision	recall	f1-score	support
0	0.83	0.81	0.82	837
1	0.81	0.84	0.83	839
accuracy			0.82	1676
macro avg	0.82	0.82	0.82	1676
weighted avg	0.82	0.82	0.82	1676

# 3. THỰC HIỆN BÀI TOÁN

## 3.6. Xây dựng model, cài đặt

*Thử train trên các mô hình phân lớp khác:*

❖ Random forest:

```
[ ] from sklearn.ensemble import RandomForestClassifier
randomForest_model = RandomForestClassifier(random_state=0)
randomForest_model.fit(X_train, Y_train)
print("- The accuracy of model on training set: {}".format(randomForest_model.score(X_train, Y_train)))
y_pred = randomForest_model.predict(X_test)
print("- The accuracy of model on test set: {}".format(accuracy_score(y_true=Y_test, y_pred=y_pred)))
print("- The classification report: \n{}".format(classification_report(y_true=Y_test, y_pred=y_pred)))
```



```
- The accuracy of model on training set: 0.9995524392063255
- The accuracy of model on test set: 0.8156324582338902
- The classification report:
```

	precision	recall	f1-score	support
0	0.81	0.82	0.82	837
1	0.82	0.81	0.81	839
accuracy			0.82	1676
macro avg	0.82	0.82	0.82	1676
weighted avg	0.82	0.82	0.82	1676

# 3. THỰC HIỆN BÀI TOÁN

## 3.6. Xây dựng model, cài đặt

*Thử train trên các mô hình phân lớp khác:*

❖ Decision Tree:

```
[ ] from sklearn import tree
    decision_model = tree.DecisionTreeClassifier()
    decision_model.fit(X_train, Y_train)
    print("- The accuracy of model on training set: {}".format(decision_model.score(X_train, Y_train)))
    y_pred = decision_model.predict(X_test)
    print("- The accuracy of model on test set: {}".format(accuracy_score(y_true=Y_test, y_pred=y_pred)))
    print("- The classification report: \n{}".format(classification_report(y_true=Y_test, y_pred=y_pred)))
```



- The accuracy of model on training set: 0.9995524392063255
- The accuracy of model on test set: 0.7106205250596659
- The classification report:

	precision	recall	f1-score	support
0	0.73	0.67	0.70	837
1	0.70	0.75	0.72	839
accuracy			0.71	1676
macro avg	0.71	0.71	0.71	1676
weighted avg	0.71	0.71	0.71	1676

# 3. THỰC HIỆN BÀI TOÁN

## 3.6. Xây dựng model, cài đặt

*Thử train trên các mô hình phân lớp khác:*

❖ Naive Bayes:

```
[ ] from sklearn.naive_bayes import GaussianNB
    bayes_model = GaussianNB()
    bayes_model.fit(X_train, Y_train)
    print("- The accuracy of model on training set: {}".format(bayes_model.score(X_train, Y_train)))
    y_pred = decision_model.predict(X_test)
    print("- The accuary of model on test set: {}".format(accuracy_score(y_true=Y_test, y_pred=y_pred)))
    print("- The classification report: \n{}".format(classification_report(y_true=Y_test, y_pred=y_pred)))
```



```
- The accuracy of model on training set: 0.5500522154259286
- The accuary of model on test set: 0.7106205250596659
- The classification report:
```

	precision	recall	f1-score	support
0	0.73	0.67	0.70	837
1	0.70	0.75	0.72	839
accuracy			0.71	1676
macro avg	0.71	0.71	0.71	1676
weighted avg	0.71	0.71	0.71	1676



## 3. THỰC HIỆN BÀI TOÁN

### 3.6. Xây dựng model, cài đặt

*Sau khi thử các mô hình phân lớp khác nhau, nhận thấy rằng SVC là mô hình cho kết quả ổn định nhất nên sẽ được dùng làm model chính thức*



# 3. THỰC HIỆN BÀI TOÁN

## 3.7. Tinh chỉnh tham số

Sử dụng GridsearchCV để lựa chọn các hyperparameter cho model SVC (gamma, C,...)

```
[ ] param_grid = {'C': [0.1, 1, 10, 100, 1000],  
                  'gamma': [1, 0.1, 0.01, 0.001, 0.0001],  
                  'kernel': ['rbf']}  
  
grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)  
  
grid.fit(X_train, Y_train)
```

# 3. THỰC HIỆN BÀI TOÁN

## 3.7. Tinh chỉnh tham số

```
[ ] print('Parameter tốt nhất sau khi turning: ', grid.best_params_)  
    print('Trạng thái của model: ', grid.best_estimator_)
```



```
Parameter tốt nhất sau khi turning: {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}  
Trạng thái của model: SVC(C=10, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,  
    decision_function_shape='ovr', degree=3, gamma=0.0001, kernel='rbf',  
    max_iter=-1, probability=False, random_state=None, shrinking=True,  
    tol=0.001, verbose=False)
```

Accuracy của model sau khi fine turning trên tập train là 99,82%

```
[ ] score1 = grid.score(X_train, Y_train)  
    print(score1)
```



0.9982097568253021

Accuracy của model sau khi fine turning trên tập test là 76.8%

```
[ ] score1 = grid.score(X_test, Y_test)  
    print(score1)
```



0.7678997613365155

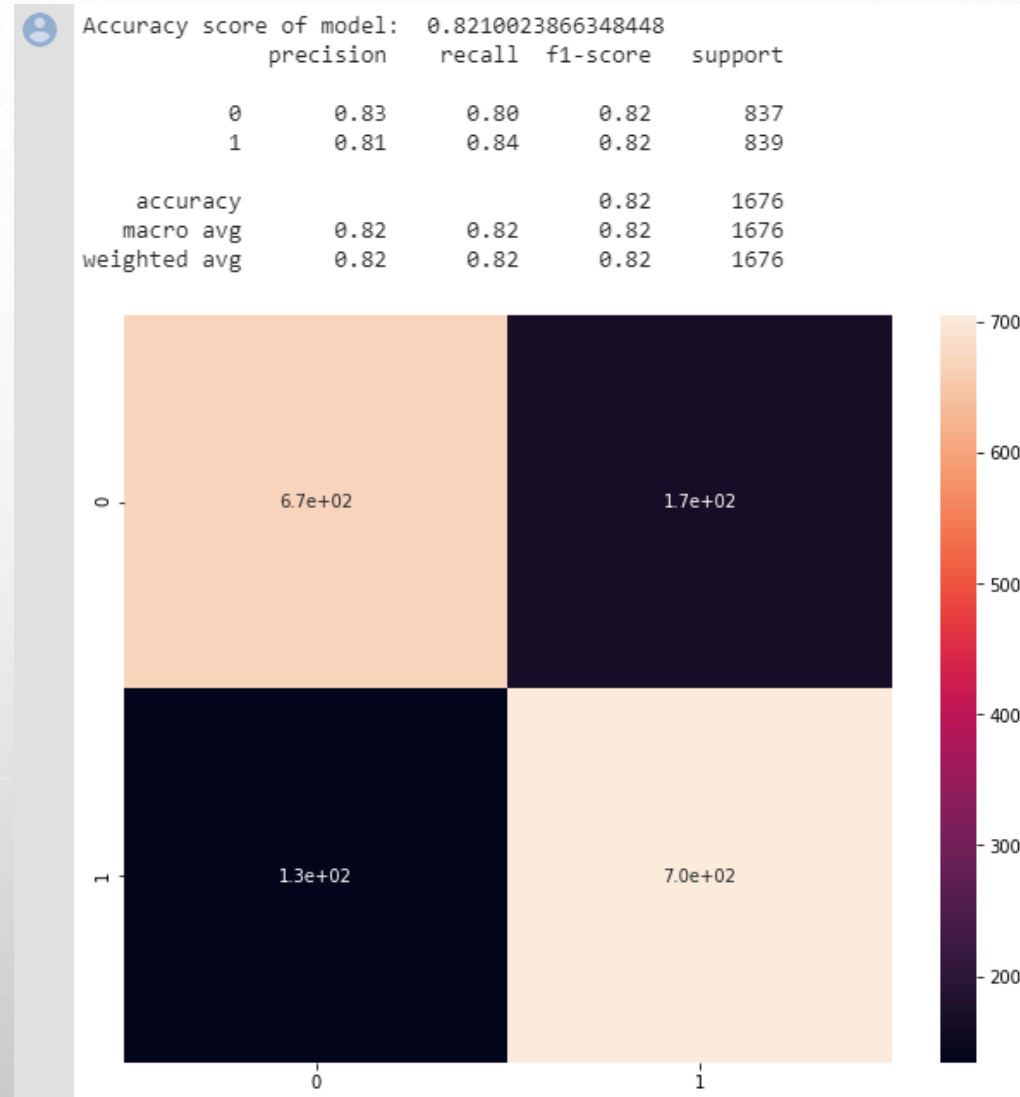
# 3. THỰC HIỆN BÀI TOÁN

## 3.8. Đánh giá model

Trước khi fine tuning:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
import seaborn as sn

pre = model.predict(X_test)
df_cm = pd.DataFrame(confusion_matrix(Y_test, pre), index = [i for i in [0, 1]],
                    columns = [i for i in [0,1]])
plt.figure(figsize = (10,8))
sn.heatmap(df_cm, annot=True)
print("Accuracy score of model: ",accuracy_score(Y_test, pre))
print(classification_report(Y_test, pre))
```



# 3. THỰC HIỆN BÀI TOÁN

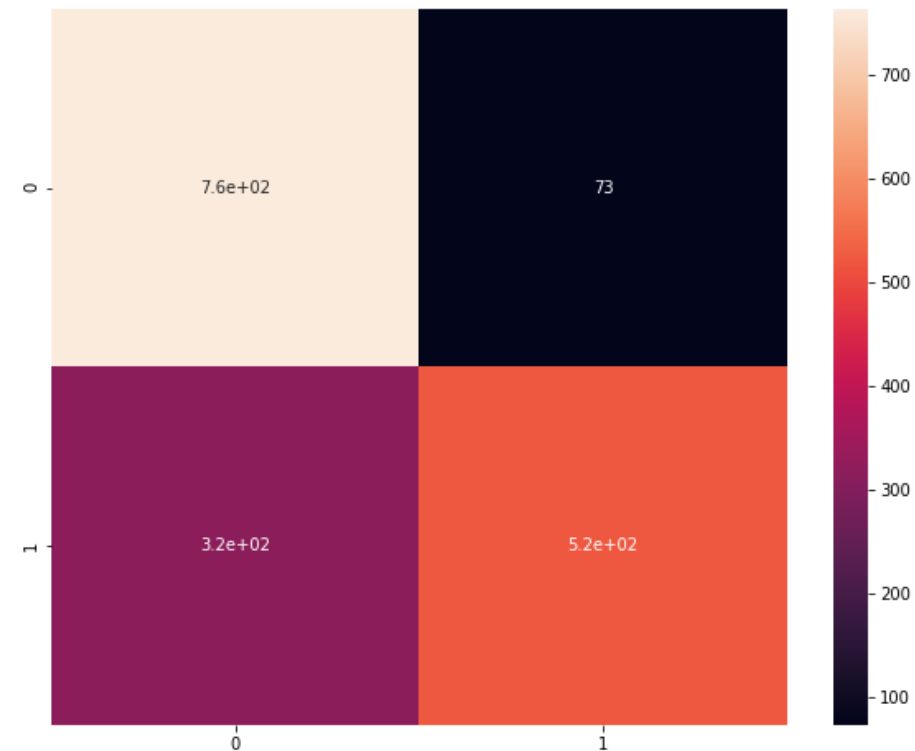
## 3.8. Đánh giá model

Sau khi fine tuning:

```
[ ] from sklearn.metrics import confusion_matrix
pre1 = grid.predict(X_test)
df_cm = pd.DataFrame(confusion_matrix(Y_test, pre1), index = [i for i in [0, 1]],
                     columns = [i for i in [0,1]])
plt.figure(figsize = (10,8))
sn.heatmap(df_cm, annot=True)
print("Accuracy score of model: ",accuracy_score(Y_test, pre1))
print(classification_report(Y_test, pre1))
```

Accuracy score of model: 0.7678997613365155

	precision	recall	f1-score	support
0	0.71	0.91	0.80	837
1	0.88	0.62	0.73	839
accuracy			0.77	1676
macro avg	0.79	0.77	0.76	1676
weighted avg	0.79	0.77	0.76	1676



## 4. DỰ ĐOÁN CHO MỘT CÂU MỚI ĐƯỢC NHẬP VÀO

```
[ ] def predict(text):
    data = text

    data = re.sub(r'([A-Z])\1+', lambda m: m.group(1).upper(), data, flags=re.IGNORECASE)
    data = data.lower()
    replace_list = {
        'òa': 'oà', 'óa': 'oá', 'òà': 'oà', 'õa': 'oã', 'ọà': 'ọạ', 'òè': 'oè', 'óè': 'oé', 'òè': 'oè',
        'õe': 'oẽ', 'ọè': 'oẹ', 'ùy': 'uỳ', 'úy': 'uý', 'ùý': 'uỳ', 'ũy': 'uỹ', 'uy': 'uỵ', 'uà': 'uà',
        'à': 'à', 'ố': 'ố', 'u': 'ố', 'ổ': 'ổ', 'ò': 'ò', 'ố': 'ố', 'ã': 'ã', 'ă': 'ă', 'ă': 'ă',
        'à': 'à', 'ò': 'ò', 'è': 'è', 'ế': 'ế', 'ă': 'ă', 'ù': 'ù', 'ế': 'ế', 'ò': 'ò', 'i': 'i',
        'è': 'è', 'àk': 'u' à ', 'a': 'à', 'i': 'i', 'ă': 'ă', 'ừ': 'ừ', 'e': 'ế', 'y': 'ỹ', 'a': 'á',
        'ô kê': ' ok ', 'okie': ' ok ', ' o kê ': ' ok ', '(:)': 'positive', '(:)': 'negative',
        'okey': ' ok ', 'ôkê': ' ok ', 'oki': ' ok ', ' oke ': ' ok ', ' okay ': ' ok ', 'okê': ' ok ',
        ' tks ': 'u' cảm ơn ', 'thks': 'u' cảm ơn ', 'thanks': 'u' cảm ơn ', 'ths': 'u' cảm ơn ', 'thank': 'u' cảm ơn ',
        '★': 'star ', '*': 'star ', '🌟': 'star ', '👍': 'u' positive ',
        'kg': 'u' không ', 'not': 'u' không ', 'u' kg ': 'u' không ', 'k': 'u' không ', 'kh': 'u' không ', 'kô': 'u' không ', 'hok': 'u' không ', 'kp': 'u' không phải ', 'u' kô ': 'u' không ',
        'he he': ' positive ', 'hehe': ' positive ', 'hihi': ' positive ', 'haha': ' positive ', 'hjhj': ' positive ',
        'lol': ' negative ', 'cc': ' negative ', 'cute': 'u' dễ thương ', 'huhu': ' negative ', 'vs': 'u' với ', 'wa': ' quá ', 'wá': 'u' quá ', 'j': 'u' gì ', '": ' ',
        'sz': 'u' cỡ ', 'size': 'u' cỡ ', 'u' đx ': 'u' được ', 'dk': 'u' được ', 'dc': 'u' được ', 'đk': 'u' được ',
        'đc': 'u' được ', 'authentic': 'u' chuẩn chính hãng ', 'u' aut ': 'u' chuẩn chính hãng ', 'u' auth ': 'u' chuẩn chính hãng ', 'thick': 'u' positive ', 'store': 'u' cửa hàng ',
        'shop': 'u' cửa hàng ', 'sp': 'u' sản phẩm ', 'gud': 'u' tốt ', 'god': 'u' tốt ', 'wel done': ' tốt ', 'good': 'u' tốt ', 'gút': 'u' tốt ',
        'sầu': 'u' xấu ', 'gut': 'u' tốt ', 'u' tot ': 'u' tốt ', 'u' nice ': 'u' tốt ', 'perfect': 'rất tốt', 'bt': 'u' bình thường ',
        'time': 'u' thời gian ', 'qá': 'u' quá ', 'u' ship ': 'u' giao hàng ', 'u' m ': 'u' mình ', 'u' mik ': 'u' mình ',
        'ế': 'ế', 'product': 'sản phẩm', 'quality': 'chất lượng', 'chat': 'chất ', 'exelent': 'hoàn hảo', 'bad': 'tệ', 'fresh': ' tươi ', 'sad': ' tệ ',
        'date': 'u' hạn sử dụng ', 'hsd': 'u' hạn sử dụng ', 'quickly': 'u' nhanh ', 'quick': 'u' nhanh ', 'fast': 'u' nhanh ', 'delivery': 'u' giao hàng ', 'u' súp ': 'u' giao hàng ',
        'beautiful': 'u' đẹp tuyệt vời ', 'u' tl ': 'u' trả lời ', 'u' r ': 'u' rồi ', 'u' shopE ': 'u' cửa hàng ', 'u' order ': 'u' đặt hàng ',
        'chất lg': 'u' chất lượng ', 'u' sd ': 'u' sử dụng ', 'u' dt ': 'u' điện thoại ', 'u' nt ': 'u' nhắn tin ', 'u' tl ': 'u' trả lời ', 'u' sài ': 'u' xài ', 'u' bjo': 'u' bao giờ ',
        'thik': 'u' thích ', 'u' sop ': 'u' cửa hàng ', 'fb': ' facebook ', 'face': ' facebook ', 'very': 'u' rất ', 'u' quả ng ': 'u' quãng ',
        'dep': 'u' đẹp ', 'u' xau ': 'u' xấu ', 'delicious': 'u' ngon ', 'u' hầg': 'u' hàng ', 'u' quả': 'u' quả ',
        'iu': 'u' yêu ', 'fake': 'u' giả mạo ', 'trl': 'trả lời', '>': 'u' positive ',
        'por': 'u' tệ ', 'poor': 'u' tệ ', 'ib': 'u' nhắn tin ', 'rep': 'u' trả lời ', 'u' fback': ' feedback ', 'fedback': ' feedback ',
        #dưới 3* quy về 1*, trên 3* quy về 5*
        '6 sao': ' 5star ', '6 star': ' 5star ', '5star': ' 5star ', '5 sao': ' 5star ', '5sao': ' 5star ',
        'starstarstarstarstar': ' 5star ', '1 sao': ' 1star ', '1sao': ' 1star ', '2 sao': ' 1star ', '2sao': ' 1star ',
        '2 starstar': ' 1star ', '1star': ' 1star ', '0 sao': ' 1star ', '0star': ' 1star '
    }
}
```

```

for k ,v in replace_list.items():
    data = data.replace(k,v)
data = data.replace(","," ").replace(".", " ") \
    .replace(";"," ").replace("'", " ") \
    .replace(":", " ").replace('"'," ") \
    .replace('`'," ").replace("`'," ") \
    .replace("!", " ").replace("?", " ") \
    .replace("-", " ").replace("?", " ")
data = data.strip()

data = ViTokenizer.tokenize(data)
T = []
words = data.split(' ')
vec_data = np.zeros((400))
for word in words:
    if word in vocab:
        vec_data += wv_from_bin.wv[word]
vec_data.reshape(1,-1)
l = vec_data.tolist()
T.append(l)
T = np.array(T)
with open('/content/drive/My Drive/Course_Project_ML/saved_model.pkl', 'rb') as f:
    model = joblib.load(f)
prediction = model.predict(T)
return int(prediction[0])

```

The end!

