

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

---o0o---



BÁO CÁO ĐỒ ÁN

Lớp: CS114.K21

Môn học: Máy học

Học kì II (2019 – 2020)

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

Giảng viên:

PGS.TS Lê Đình Duy - THS. Phạm Nguyễn Trường An

Sinh viên thực hiện:

Nguyễn Đức Thịnh – 18521442

Huỳnh Minh Tuấn – 18521596

Nguyễn Minh Thông – 18521459

MỤC LỤC

I.	NỘI DUNG	4
1.	Nội dung và mục đích đề tài	4
1.1.	Nội dung bài toán	4
1.2.	Mục đích	4
2.	Input và output	4
II.	CÁC BƯỚC THỰC HIỆN ĐỀ TÀI	5
1.	Xây dựng dataset	5
1.1.	Cách thức xây dựng	5
1.2.	Số lượng	5
2.	Xử lý dữ liệu	5
2.1.	Đọc dữ liệu	5
2.2.	Phân bố dữ liệu	6
2.3.	Thực hiện tiền xử lý câu, từ	6
2.4.	Tách từ	7
2.5.	Số lượng từ trong mỗi câu	8
2.6.	Word Embedding	8
3.	Chọn mô hình	10
3.1.	Phân chia dữ liệu	10
3.2.	Chọn mô hình	11
4.	Training mô hình	11
5.	Đánh giá mô hình	12
5.1.	Logistic regression	12
5.2.	Random forest	13
5.3.	Decision tree	13
5.4.	Naive Bayes	14
6.	Tinh chỉnh tham số	15
7.	Dự đoán và kết quả	16
III.	TÀI LIỆU THAM KHẢO	16

PHÂN CÔNG CÔNG VIỆC

Họ và tên	Công việc
Nguyễn Đức Thịnh	Thu thập dữ liệu, viết code, làm giao diện
Huỳnh Minh Tuấn	Lọc dữ liệu, làm powerpoint, viết báo cáo
Nguyễn Minh Thông	Lọc dữ liệu, làm powerpoint, viết báo cáo

I. NỘI DUNG

1. Nội dung và mục đích đề tài

1.1. Nội dung bài toán

Hiện nay nhu cầu mua hàng qua mạng của người dùng ngày càng trở nên phát triển mạnh hơn do những lợi ích mà nó mang lại, như 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 để xem, ... Tuy nhiên, việc mua hàng qua mạng cũng có những nhược điểm, trong đó có việc người dùng không thể tận mắt đánh giá sản phẩm của mình như mua trực tiếp tại các cửa hàng được. Vì vậy, các mục bình luận về sản phẩm 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.

1.2. Mục đích

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 các cơ sở kinh doanh nào cũng đều có trang web riêng để phù hợp với 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.

2. Input và output

- Input: nhập vào một câu, từ ngữ bất kì.

- Output: xuất ra “**Tích cực**” nếu câu, từ đó mang nghĩa tích cực, hoặc xuất ra “**Tiêu cực**” nếu câu, từ đó mang nghĩa tiêu cực.

II. CÁC BƯỚC THỰC HIỆN ĐỀ TÀI

1. Xây dựng dataset

1.1. 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).

1.2. 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. Tham khảo dataset tại https://github.com/thinh18521442/Course_Project_AI/blob/master/dataset1.csv

2. Xử lý dữ liệu

2.1. Đọc dữ liệu

Tiến hành đọc dữ liệu đã tiền xử lý ở các file. Ở đây ta lưu mỗi nhãn là 1 file CSV nên sẽ đọc bằng pandas.

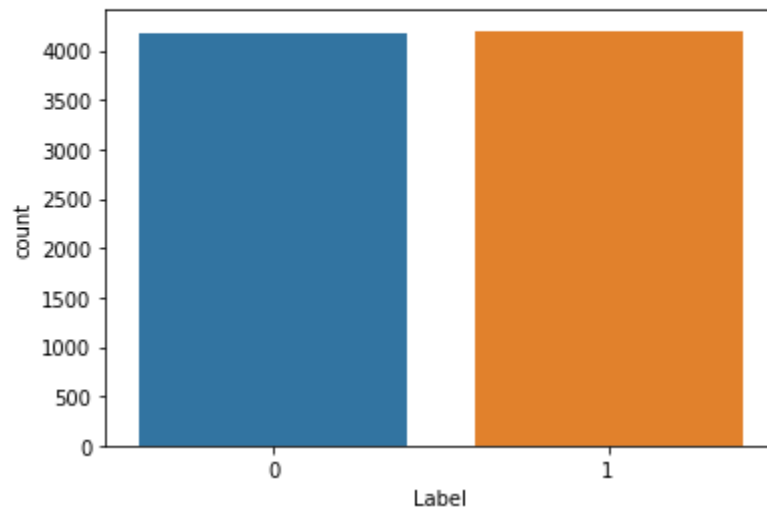
```
emb = None  
df = pd.read_csv('/content/drive/My Drive/Course_Project_AI/dataset1.csv')  
df.head(10)
```

	Text	Label
0	Mình đã dùng anywhere thế hệ đầu, quả là đầy t...	0
1	Quan tâm nhất là độ trễ có cao không, dùng thi...	0
2	đag xài con cùi bắp 98k....pin trâu, mỗi tội đ...	0
3	logitech chắc hàng phải tiền triệu trở lên dùn...	0
4	Đang xài con m175 cùi mía , nhà xài nhiều chuộ...	0
5	Đang xài 2 con M185, nút chuột giữa hai con đề...	0
6	Con Anywhere 1 mình dùng bị double click cũng ...	0
7	Hàng cty cấp, cấp xong rút ở nhà, xài con Xorn...	0
8	Magic mouse mà ngon hơn mới lạ, Magic mouse ch...	0
9	em giống y bác luôn, chân bluetooth kính hỏn, ...	0

2.2. Phân bố dữ liệu

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

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



Dựa vào biểu đồ có thể thấy data cân bằng

2.3. Thực hiện tiền xử lý câu, từ

Văn bản đầu vào của chúng ta có thể chứa nhiều ký tự thừa, dấu câu thừa, 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])\d+', lambda m: m.group(1).upper(), row, flags=re.IGNORECASE)  
    row = row.lower()  
    replace_list = {  
        '0a': '0á', '0á': 'oá', '0á': 'oà', '0a': 'oã', '0a': 'oê', '0e': 'oè', '0é': 'oé', '0e': 'oè',  
        '0e': 'oè', '0e': 'oé', '0y': 'úy', 'úy': 'uy', 'úy': 'uy', '0y': 'uy', 'uy': 'uy', 'uá': 'úa',  
        'á': 'à', 'ó': 'ò', 'í': 'ì', 'õ': 'ô', 'ô': 'ö', 'ë': 'ö', 'ä': 'ä', 'ä': 'ä', 'ä': 'ä',  
        'ä': 'ä', 'ó': 'ò', 'è': 'è', 'ä': 'ä', 'ü': 'ü', 'ë': 'ë', 'ö': 'ö', 'í': 'í',  
        'è': 'è', 'ä': 'ä', 'ä': 'ä', 'ü': 'ü', 'ë': 'ë', 'y': 'ý', 'ä': 'ä',  
        'ò kè!': ' ok ', 'okie!': ' ok ', ' o kê ': ' ok ',  
        'oke!': ' ok ', 'òkè!': ' ok ', 'oki!': ' ok ', ' oke ': ' ok ', 'okay!': ' ok ', 'okè!': ' ok ',  
        ' tks ': u' cảm ơn ', 'tnks': u' cảm ơn ', 'thanks': u' cảm ơn ', 'ths': u' cảm ơn ', 'thank!': u' cảm ơn ',  
        ☆: 'star', '★': 'star', ★: u' positive ',  
        'kg': u' không ', 'not!': u' không ', 'u kg': u' không ', 'k ': u' không ', 'kh ': u' không ', 'kò': u' không ', 'hok!': u' không ', 'kp ': u' không phải ', 'u kò ': u' không  
        'he he!': 'positive ', 'hehe!': 'positive ', 'hhhl!': 'positive ', 'haha!': 'positive ', 'hjhg!': 'positive ',  
        'lol!': 'negative ', 'cc ': 'negative ', 'cute!': u' dễ thương ', 'huhu!': 'negative ', ' vs ': u' với ', 'wa': ' quá ', 'wá': u' quá ', 'j ': u' gì ', 'm ': ' ',  
        ' sz ': u' cỡ ', 'size': u' cỡ ', 'u dx ': u' được ', 'dk!': u' được ', 'dc!': u' được ', 'dk!': u' được ',  
        'dc!': u' được ', 'authentic!': u' chuẩn chỉnh hàng ', 'u aut ': u' chuẩn chỉnh hàng ', 'u auth ': u' chuẩn chỉnh hàng ', 'thick!': u' positive ', 'store!': u' của hàng ',  
        'shop!': u' của 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 ',  
        'é': 'é', 'product': 'sản phẩm', 'quality': 'chất lượng', 'chat!': 'chât ', '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 sip ': u' giao hàng ',  
        'beautiful!': u' đẹp tuyệt vời ', 'u tl ': u' trả lời ', 'u r ': u' rồi ', 'u shopE ': u' của hàng ', 'u order ': u' đặt hàng ',  
        'chât lưg!': u' chât lưg ', '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 ',  
        'thích!': u' thích ', 'u sop ': u' của hàng ', 'fb ': 'facebook ', 'face ': 'facebook ', 'very ': u' rất ', 'u quâ ng ': u' quảng ',  
        'dep!': u' đẹp ', 'u xau ': u' xấu ', 'delicious!': u' ngon ', 'u hâg ': u' hàng ', 'u quâ ': u' quá ',  
        '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 ',  
        #đudi 3* quy về 1*, trên 3* quy về 5*  
        '6 sao!': '5star', '6 star!': '5tan', '5stan!': '5star', '5 sao!': '5star', '5sao!': '5tan',  
        'stanstanstanstan!': '5tan', '1 sao!': '1star', '1sao!': '1star', '2 sao!': '1tan', '2sao!': '1star',  
        '2 stanstan!': '1tan', '1stan!': '1star', '0 sao!': '1star', '0stan!': '1tan',  
    }  
    for k,v in replace_list.items():  
        row = row.replace(k,v)  
    row = row.replace(" ", " ").replace(".", " ") \  
        .replace(";", " ").replace("=", " ") \  
        .replace(":", " ").replace("&n", " ") \  
        .replace("'", " ").replace(""", " ") \  
        .replace("!?", " ").replace(">", " ") \  
        .replace("<?", " ").replace(">", " ") \  
        .replace("<?", " ").replace(">", " ") \  
    row = row.strip()  
    return row
```

2.4. 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
```

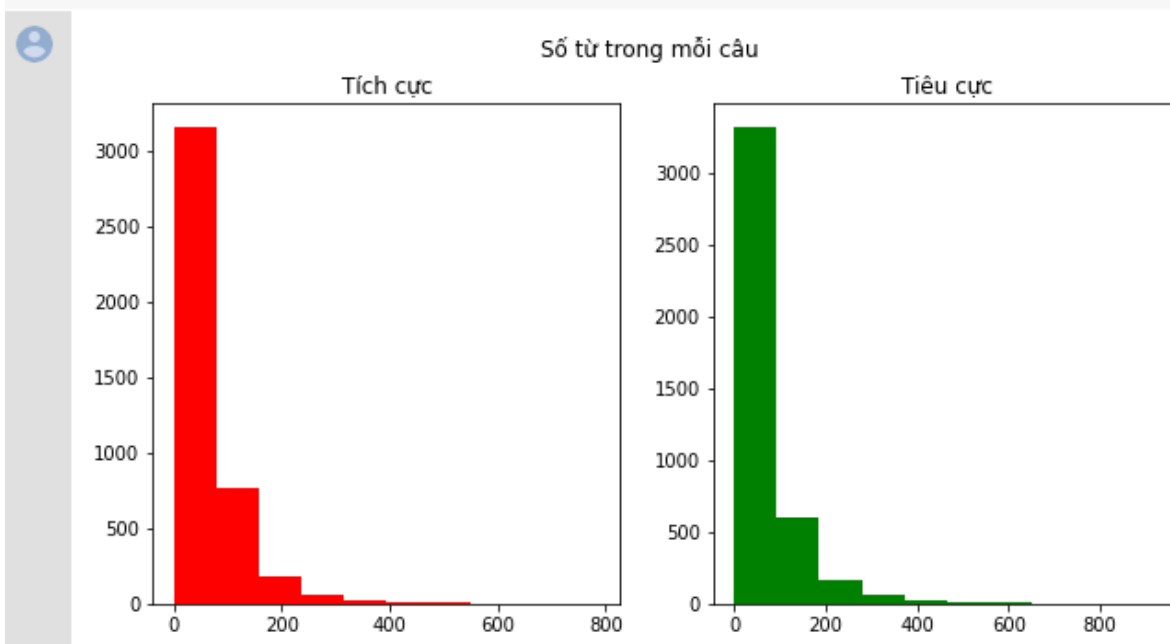
```
[ ] def tok(row):  
    return ViTokenizer.tokenize(row)
```

2.5. 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()
```



2.6. Word Embedding

- Word Embedding là 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-IDF, 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, link tải model tại <https://github.com/sonvx/word2vecVN>
- Mô tả đôi nét về model: dữ liệu train được lấy từ trang báo lớn ở Việt Nam (baomoi.vn), gồm 439056 từ shape(400,)
- Load model word2vec: sử dụng thư viện gensim để load model

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

'local/lib/python3.6/dist-packages/smart_open/smart_open_lib.py:254: UserWarning: This function is deprecated, use smart_open.open instead.
See the migration notes for details: %s' % _MIGRATION_NOTES_URL

[ ] vocab = wv_from_bin.wv.vocab

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: Call to deprecated `wv` (Attribute will be removed in
the future. Use the migration notes for details: %s' % _MIGRATION_NOTES_URL.

[ ] len(vocab)

439056
```

- Quá trình 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. 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.

```
[ ] 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)
```

- 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. Chọn mô hình

3.1. *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,)
```

3.2. Chọn mô hình

Chọn model Support Vector Machine (SVM). SVM cho một tập hợp các phương pháp học có giám sát liên quan đến nhau để phân loại và phân tích hồi quy. SVM dạng chuẩn nhận dữ liệu vào và phân loại chúng vào hai lớp khác nhau. Do đó SVM là một thuật toán phân loại nhị phân.

```
[ ] model = SVC(kernel='linear', C=1)
```

4. Training mô hình

- Tiến hành train sơ bộ trên bộ data cho ra base model.

```
model.fit(X_train, Y_train)
joblib.dump(model, 'saved_model.pkl')
```

- 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

```

5. Đánh giá mô hình

Để đánh giá model, ta có thể thử train trên các model phân lớp khác, từ đó có thể đưa ra đánh giá dễ dàng hơn. Một số model khác có thể dùng để train:

5.1. *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

5.2. 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

5.3. 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

5.4. 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 = bayes_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: 0.5500522154259286
- 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
 macro avg         0.71      0.71      0.71
 weighted avg      0.71      0.71      0.71
```

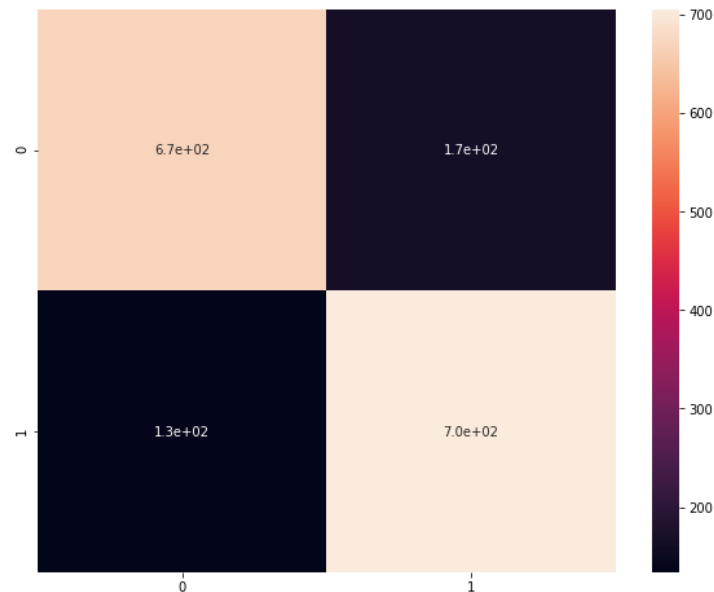
➔ Sau khi thử các mô hình phân lớp khác nhau, nhận thấy rằng SVM là mô hình cho kết quả ổn định nhất nên sẽ được dùng làm model chính thức.

```
Accuracy score of model: 0.8210023866348448
      precision    recall  f1-score   support

     0       0.83      0.80      0.82       837
     1       0.81      0.84      0.82       839

 accuracy          0.82
 macro avg         0.82      0.82      0.82
 weighted avg      0.82      0.82      0.82
```

```
[[671 166]
 [134 705]]
```




6. 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)
```


- Ta tìm được tham số tốt nhất sau khi tuning:

```
[ ] 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 sau khi fine tuning trên tập train là 99,82%:

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

 0.9982097568253021

- Accuracy sau khi fine tuning trên tập test là 76,79%:

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

 0.7678997613365155

7. Dự đoán và kết quả

Thực hiện dự đoán với một câu feedback của khách hàng trên một trang mua sắm bất kỳ:

```
[ ] text = input("Input a text to predict: ")
    if predict(text):
        print("Tích cực")
    else:
        print("Tiêu cực")
```

Input a text to predict: Điện Thoại dùng tốt! So với giá ngoài các cửa hàng điện thoại rẻ hơn được 1 triệu! Mua cho Bố dùng rất là thích. Mấy nữa mình sẽ mua thêm 1 cái Tặng mẹ ck
Tích cực

III. TÀI LIỆU THAM KHẢO

- Thư viện machine learning sklearn <https://scikit-learn.org/>
- Đánh giá ý kiến khách hàng lazada miai group
<https://www.miai.vn/2020/05/04/nlp-series-1-thu-lam-he-thong-danh-gia-san-pham-lazada/>
- Xử lý ngôn ngữ tự nhiên wikipedia
https://vi.wikipedia.org/wiki/X%E1%BB%AD_l%C3%BD_ng%C3%B4n_ng%E1%BB%AF_t%E1%BB%B1_nhi%C3%AAn
https://en.wikipedia.org/wiki/Natural_language_processing