In [1]:
```
import pandas as pd
import pandas as pd
import numpy as np

import regex as re
import emoji
from underthesea import word_tokenize
import string
import os
from pyvi import ViTokenizer
```

In [2]:
```
data = pd.read_csv('ks.csv')
```

In [3]:
```
textdata = data[['Rating', 'Content comment']]
```

In [4]:
```
label = []
for i in textdata['Rating']:
    if 30<i<=50:
        a = 1
    else:
        a = -1
    label.append(a)
```

In [5]:
```
textdata['Label'] = label
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_16500\3423520257.py:1: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  textdata['Label'] = label
```

In [6]:
```
textdata['Label'].value_counts()
```

Out[6]:
```
Label
 1    8953
-1     320
Name: count, dtype: int64
```

In [7]:
```
reviews = textdata.iloc[:,1].values
labels = textdata.iloc[:,2].values
```

In [8]:
```
#1. Loại bỏ URL
def remove_html(txt):
    return re.sub(r'<[^>]*>', '', txt)
```

In [9]:
```
#2. xoá dấu câu
def remove_dau(txt):
    txt = re.sub(r'[^\w\s]','',txt)
    return txt
```

In [10]:
```
#3. Xóa chữ số
import re
def remove_digit(txt):
    txt = re.sub('\d+', "", txt)
    return txt
```

In [11]:
```
#4. xoá các dấu xuống dòng
def remove_n(txt):
    return txt.replace('\r\n',' ')
```

In [12]:
```
#5. xoá các punctation
import string

def remove_punct(txt):
    table = str.maketrans ("","",string.punctuation)
    return txt.translate (table)
```

In [13]:
```
#6. xoá các từ nối dài vd : đẹpppp
def remove_word(txt):
    txt = re.sub(r'([A-Z])\1+', lambda m: m.group(1).upper(), txt, flags=re.I
    return txt
```

In [14]:
```
#7. xoá emoji
def remove_emoji(txt):
    return emoji.replace_emoji(txt,replace='')
```

In [15]:
```
#8. chuẩn hoá emoji
def normalize_emoji(txt):
    #Chuẩn hóa tiếng Việt, xử lý emoj, chuẩn hóa tiếng Anh, thuật ngữ
    replace_list = {
        'òa': 'oà', 'óa': 'oá', 'ỏa': 'oả', 'õa': 'oã', 'ọa': 'oạ', 'òe': 'oè
        'õe': 'oẽ', 'ọe': 'oẹ', 'ùy': 'uỳ', 'úy': 'uý', 'ủy': 'uỷ', 'ũy': 'uỹ
        'ả': 'ả', 'ố': 'ố', 'u´': 'ố','ỗ': 'ỗ', 'ồ': 'ồ', 'ổ': 'ổ', 'ấ': 'ấ',
        'ầ': 'ầ', 'ỏ': 'ỏ', 'ề': 'ề','ễ': 'ễ', 'ắ': 'ắ', 'ủ': 'ủ', 'ế': 'ế',
        'ẻ': 'ẻ', 'àk': u' à ','a`': 'à', 'i`': 'ì', 'ă´': 'ắ','ử': 'ử', 'e~'
        #Quy các icon về 2 loại emoj: Tích cực hoặc tiêu cực
        "👹": "nagative", "👻": "positive", "💃": "positive",'🤙': ' positi
        "💄": "positive", "💎": "positive", "💩": "positive","😳": "negativ
        "😾": "nagative", "🚫": "nagative",  "🤬": "nagative","🏌": "positi
        '👎': ' nagative ', '😒': ' nagative ','✨': ' positive ', '❣': ' po
        '♥': ' positive ', '😍': ' positive ', 'like': ' positive ', '💌': '
        '🤣': ' positive ', '🖤': ' positive ', '😁': ' positive ', ':(': '
        '❤️': ' positive ', '😻': ' positive ', '😘': ' positive ', '😓': '
        '?': ' ? ', '😁': ' positive ', '💔': ' positive ', '😳': ' nagative
        '💯': ' positive ', '💗': ' positive ', '♡': ' positive ', '💜': ' p
        '^^': ' positive ', '😨': ' nagative ', '☺': ' positive ', '💋': ' p
        '😖': ' nagative ', '😀': ' positive ', ':((': ' nagative ', '😡': '
        '😔': ' nagative ', '🥲': ' positive ', '😌': ' nagative ', '😝': '
        '😙': ' positive ', '😤': ' nagative ', '😎': ' positive ', '😆': '
        '✌': ' positive ', '💕': ' positive ', '😞': ' nagative ', '😓': '
        '😊': ' positive ', '😂': ' positive ', ':v':  ' positive ', '=))': '
        '❤️‍🔥': ' positive ', '😐': ' nagative ', ':3': ' positive ', '😫': '
        '😛': ' positive ', '😬': ' 😬 ', '😌': ' 😌 ', '💛': ' positive '
        '😗': ' positive ', '🤦': ' nagative ', '😑': ' nagative ', '🔥': '
        '🆗': ' positive ', '😻': ' positive ', '💙': ' positive ', '💟': '
        '😚': ' positive ', '❌': ' nagative ', '👏': ' positive ', ';)': '
        '😋': ' positive ',  '🌷': ' positive ', '🌸': ' positive ', '🌺': '
        '🌼': ' positive ', '🍓': ' positive ', '🐆': ' positive ', '🐾': '
        '💐': ' positive ', '💞': ' positive ', '💥': ' positive ', '💪': '
        '💰': ' positive ',  '😇': ' positive ', '😜': ' positive ', '😝': '
        '😳': ' positive ', '🤑': ' positive ', '🤪': ' positive ','☹': ' n
        '😔': ' nagative ', '😳': ' nagative ', '😩': ' nagative ', '😰': '
        '😵': ' nagative ', '😶': ' nagative ', '😧': ' nagative ',
        #Chuẩn hóa 1 số sentiment words/English words
        ':))': '  positive ', ':)': ' positive ', 'ô kêi': ' ok ', 'okie': ' 
        'okey': ' ok ', 'ôkê': ' ok ', 'oki': ' ok ', ' oke ':  ' ok ',' okay
        ' tks ': u' cám ơn ', 'thks': u' cám ơn ', 'thanks': u' cám ơn ', 'th
        '⭐': 'star ', '*': 'star ', '🌟': 'star ', '🎉': u' positive ',
        'kg ': u' không ','not': u' không ', u' kg ': u' không ', '"k ': u' k
        'he he': ' positive ','hehe': ' positive ','hihi': ' positive ', 'hah
        ' lol ': ' nagative ',' cc ': ' nagative ','cute': u' dễ thương ','hu
        ' sz ': u' cỡ ', 'size': u' cỡ ', u' đx ': u' được ', 'dk': u' được '
        'đc': u' được ','authentic': u' chuẩn chính hãng ',u' aut ': u' chuẩn
        'shop': u' cửa hàng ', 'sp': u' sản phẩm ', 'gud': u' tốt ','god': u'
        'sấu': u' xấu ','gut': u' tốt ', u' tot ': u' tốt ', u' nice ': u' tố
        'time': u' thời gian ', 'qá': u' quá ', u' ship ': u' giao hàng ', u'
        'ể': 'ể', 'product': 'sản phẩm', 'quality': 'chất lượng','chat':' chấ
        'date': u' hạn sử dụng ', 'hsd': u' hạn sử dụng ','quickly': u' nhanh
        'beautiful': u' đẹp tuyệt vời ', u' tl ': u' trả lời ', u' r ': u' rồ
        'chất lg': u' chất lượng ',u' sd ': u' sử dụng ',u' dt ': u' điện tho
        'thik': u' thích ',u' sop ': u' cửa hàng ', ' fb ': ' facebook ', ' f
        'dep': u' đẹp ',u' xau ': u' xấu ','delicious': u' ngon ', u'hàg': u'
        'iu': u' yêu ','fake': u' giả mạo ', 'trl': 'trả lời', '><': u' posit
```

```
        ' por ': u' tệ ',' poor ': u' tệ ', 'ib':u' nhắn tin ', 'rep':u' trả
        #dưới 3* quy về 1*, trên 3* quy về 5*
        '6 sao': ' 5star ','6 star': ' 5star ', '5star': ' 5star ','5 sao': '
        'starstarstarstarstar': ' 5star ', '1 sao': ' 1star ', '1sao': ' 1sta
        '2 starstar':' 1star ','1star': ' 1star ', '0 sao': ' 1star ', '0star

    for k, v in replace_list.items():
        txt = txt.replace(k, v)
    return txt
```

In [16]:
```
#9. remove stop words
f = open("vietnamese-stopwords-dash.txt",encoding="utf-8")
List_StopWords=f.read().split("\n")
def remove_stopword(txt):
    txt =" ".join(txt for txt in txt.split() if txt not in List_StopWords)
    return txt
```

In [17]:
```
#10. remove unexpected word
def remove_unexpected(txt):
    txtm =" ".join(txt for txt in txt.split() if 1< len(txt) < 9)
    return txtm
```

In [18]:
```
correct_mapping = {
    "ks": "khách sạn",
    "ksan" : "khách sạn",
    "nv" : "nhân viên",
    "nvien" : "nhân viên",
    "nvienn": "nhân viên",
    "m": "mình",
    "mik": "mình",
    "ko": "không",
    "k": " không ",
    "kh": "không",
    "khong": "không",
    "kg": "không",
    "khg": "không",
    "tl": "trả lời",
    "r": "rồi",
    "fb": "mạng xã hội", # facebook
    "face": "mạng xã hội",
    "thanks": "cảm ơn",
    "thank": "cảm ơn",
    "tks": "cảm ơn",
    "tk": "cảm ơn",
    "ok": "tốt",
    "dc": "được",
    "vs": "với",
    "đt": "điện thoại",
    "thjk": "thích",
    "qá": "quá",
    "trể": "trễ",
    "bgjo": "bao giờ",
    "zui" : "vui",
    "dui": "vui",
    "book":"đặt phòng",
    "villa": "biệt thự",
    "resort": ""
}
```

In [19]:
```
#11. Xoá các từ viết tắt phổ biến
def tokmap(cau):
    txt = ""
    for text in str(cau).split():
        if text in correct_mapping:
            txt = txt + correct_mapping[text] + ' '
        else:
            txt = txt + text + ' '
    return txt
```

In [20]:
```
def txt_processed(txt):
# txt = txt.lower()
    txt = tokmap(txt)
    txt = remove_digit(txt)
    txt = remove_n(txt)
    #txt = remove_dau(txt)
    txt = remove_word(txt)
    txt = txt.lower()
    txt = tokmap(txt)
    txt = remove_html(txt)
    txt = normalize_emoji(txt)
    txt = remove_emoji(txt)
    txt = remove_punct(txt)
    txt = re.sub(r'\s+', ' ', txt).strip()
    txt = remove_unexpected(txt)
    txt = ViTokenizer.tokenize(txt)
    txt = remove_stopword(txt)
    return txt
```

In [21]:
```
reviews_processed = []
for txt in reviews:
    txt = txt_processed(txt)
    reviews_processed.append(txt)
```

In [22]:
```
data = pd.DataFrame(reviews_processed, columns=['reviews'])
data['labels'] = labels
```

In [23]:
```
data
```

Out[23]:

	reviews	labels
0	dịch_vụ nhân_viên thân_thiện lịch_sự vila đẹp ...	1
1	furama vilas tiện_nghi dịp bạn_bè dừng chân ph...	1
2	phòng sạch_sẽ không_gian rộng_rãi trải nghiệm ...	1
3	ký_túc_xá đi ngắm hồ bơi phòng sạch_sẽ nhân_vi...	1
4	dịch_vụ cơ_sở vật_chất tuyệt_vời nhân_viên chu...	1
...	...	...
9268	phòng view biển phòng phòng hoàn tiền	-1
9269	đồ ngon nhân_viên giao đồ đi vệ_sinh kịp chuôn...	-1
9270	khách_sạn đem thú cưng đêm thú cưng nhân_viên ...	-1
9271	khách_sạn xem_xét_lại thái_độ nhân_viên bưng_b...	-1
9272	đừng giá chất_lượng tồi_tệ	-1

In [24]:
```
# data.to_csv(r'E:\Capstone\data_cleaned.csv', index=False)
```

LDA

In [25]:
```
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
```

In [26]:
```
from gensim.models import CoherenceModel
import pyLDAvis
import pyLDAvis.gensim
import matplotlib as plt
%matplotlib inline
```

In [27]:
```
data_reviews = data['reviews']
```

In [28]:
```
def sent_to_words(sentences, deacc=True):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence)))
```

In [29]:
```
data1= data_reviews.values.tolist()
data_words = list(sent_to_words(data1))
```

In [30]:
```
id2word = corpora.Dictionary(data_words)
corpus = [id2word.doc2bow(text) for text in data_words]
print(len(corpus))
```

```
9273
```

In [31]:
```
[[(id2word[id],freq) for id, freq in cp] for cp in corpus[:1]]
```

Out[31]:
```
[[('biệt_thự', 1),
  ('bơi', 1),
  ('dịch_vụ', 1),
  ('furama', 1),
  ('gia_đình', 2),
  ('hiểu_biết', 1),
  ('hồ', 1),
  ('kì', 1),
  ('lễ_tân', 1),
  ('lịch_sự', 1),
  ('nghỉ', 1),
  ('nhi', 1),
  ('nhiệt_tình', 1),
  ('nhân_viên', 1),
  ('quên', 1),
  ('rộng', 2),
  ('thu', 1),
  ('thân_thiện', 1),
  ('trẻ_em', 1),
  ('tầm', 1),
  ('vila', 1),
  ('đẹp', 2)]]
```

In [32]:
```
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=2,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha=100,
                                            per_word_topics=True)
```

In [33]:
```
print(lda_model.print_topics())
doc_lda = lda_model[corpus]
```

```
[(0, '0.039*"khách_sạn" + 0.036*"nhân_viên" + 0.029*"phòng" + 0.023*"đẹp" +
0.021*"nhiệt_tình" + 0.019*"positive" + 0.014*"thân_thiện" + 0.014*"ngon" +
0.013*"dịch_vụ" + 0.012*"sạch_sẽ"'), (1, '0.053*"khách_sạn" + 0.035*"phòng"
+ 0.025*"nhân_viên" + 0.020*"biển" + 0.018*"đẹp" + 0.017*"đi" + 0.012*"đà_nẵ
ng" + 0.011*"thân_thiện" + 0.010*"sạch_sẽ" + 0.010*"tuyệt_vời"')]
```

In [34]:
```
coherence_model_lda = CoherenceModel(model=lda_model,texts=data_words, dictio
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Coherence Score:  0.36892525379509067
```

Training model

In [35]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data.reviews, data.labels
```

In [36]:
```
y_train = np.array(y_train)
y_test = np.array(y_test)
```

In [37]:
```
vocab_size = 300
embedding_dim = 20
max_length = 35
```

In [38]:
```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

In [39]:
```
tokenizer = Tokenizer(num_words=vocab_size, oov_token="<OOV>")
tokenizer.fit_on_texts(X_train)
```

In [40]:
```
X_train = tokenizer.texts_to_sequences(X_train)
padded_train_text = pad_sequences(X_train, maxlen = max_length, truncating='p
```

In [41]:
```
X_test = tokenizer.texts_to_sequences(X_test)
padded_test_text = pad_sequences(X_test, maxlen = max_length, truncating='pos
```

In [42]:
```
padded_train_text
```

Out[42]:
```
array([[  2,   3,   7, ...,   0,   0,   0],
       [ 45,  44,   2, ...,  89,   0,   0],
       [ 70,  14,  18, ...,   0,   0,   0],
       ...,
       [  2,   3,  83, ...,   0,   0,   0],
       [ 21,   8,  17, ...,  43, 157, 155],
       [  2,   3,  55, ...,   3,   1, 201]])
```

In [43]:
```
from sklearn.metrics import confusion_matrix, f1_score, accuracy_score, preci
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.naive_bayes import MultinomialNB
import time
```

In [44]:
```
def Logis(X_train, y_train):
    model = LogisticRegression()
    model.fit(X_train, y_train)
    return model
```

In [45]:
```
def Rand(X_train, y_train):
    model = RandomForestRegressor()
    model.fit(X_train, y_train)
    return model
```

In [46]:
```
def DTree(X_train, y_train):
    model = DecisionTreeClassifier()
    model.fit(X_train, y_train)
    return model
```

In [47]:
```
def SGD(X_train, y_train):
    model = SGDClassifier()
    model.fit(X_train, y_train)
    return model
```

In [48]:
```
def NBayes(X_train, y_train):
    model = MultinomialNB()
    model.fit(X_train, y_train)
    return model
```

In [49]:
```
lr = Logis(padded_train_text, y_train)
dtree = DTree(padded_train_text, y_train)
rand = Rand(padded_train_text, y_train)
sgd = SGD(padded_train_text, y_train)
nb = NBayes(padded_train_text, y_train)
```

In [50]:
```
y_pred_lr = lr.predict(padded_test_text)
y_pred_dt = dtree.predict(padded_test_text)
y_pred_rd = rand.predict(padded_test_text)
y_pred_sgd = sgd.predict(padded_test_text)
y_pred_nb = nb.predict(padded_test_text)
```

In [51]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error,mean_abso
```

In [52]:
```
from imblearn.metrics import specificity_score
```

In [53]:
```
start_time = time.time()
entries = []
entries.append(['Naive Bayes',accuracy_score(y_test,y_pred_nb),
                    precision_score(y_test, y_pred_nb), recall_score(y_test, 
                    time.time() - start_time
                    ])

entries.append(['Logistic Regression',accuracy_score(y_test,y_pred_lr),
                    precision_score(y_test, y_pred_lr), recall_score(y_test, 
                    time.time() - start_time
                    ])

# entries.append(['RandomForest',accuracy_score(y_test,y_pred_rd),
#                     precision_score(y_test, y_pred_rd), recall_score(y_test,
#                     time.time() - start_time
#                     ])

entries.append(['Decision Tree',accuracy_score(y_test,y_pred_dt),
                    precision_score(y_test, y_pred_dt), recall_score(y_test, 
                    time.time() - start_time
                    ])

entries.append(['SGD Classifier',accuracy_score(y_test,y_pred_sgd),
                    precision_score(y_test, y_pred_sgd), recall_score(y_test,
                    time.time() - start_time
                    ])
```

In [54]:
```
cv_df = pd.DataFrame(entries,
                     columns=['Model', 'Accuracy', 'Precision', 'Recall', 'F1
                              ])
```

In [55]:
```
cv_df
```

Out[55]:

	Model	Accuracy	Precision	Recall	F1	Time
0	Naive Bayes	0.642241	0.973199	0.647715	0.777778	0.003003
1	Logistic Regression	0.966595	0.966595	1.000000	0.983014	0.005003
2	Decision Tree	0.940733	0.976244	0.962096	0.969118	0.008001
3	SGD Classifier	0.959052	0.971460	0.986622	0.978982	0.010000

Prediction

In [56]:
```
data_10 = pd.read_csv('data_test.csv')
```

In [57]:
```
data_10
```

Out[57]:

	Tên	Ngày comment	Ngày ở	Loại hình du lịch	Rating	Title comment	Content comment	
0	Phan T	Phan T thg 10 2019	tháng 10 năm 2019	NaN	50	Chuyến đi cùng ba mẹ !	Phòng Delux đầy đủ tiện nghi và rất sạch sẽ , ...	https://www.tripadvisor.com.vn
1	Quynh Chi	Quynh Chi thg 10 2019	tháng 10 năm 2019	Đã đi du lịch với gia đình	50	Khách sạn tuyệt vời	Đây là Kì nghỉ tuyệt vời nhất của gia đình tôi...	https://www.tripadvisor.com.vn
2	trần Thành	trần Thành thg 12 2019	tháng 12 năm 2019	Đã đi du lịch với bạn bè	50	Du lịch	Khách sạn danacity năm trên đường đông kinh ng...	https://www.tripadvisor.com.vn
3	Mai Hương T	Mai Hương T thg 7 2017	tháng 3 năm 2017	Đã đi du lịch theo đôi	50	Tuyệt vời !!!	Mình và người yêu đã nghỉ ở đây 1 tuần.\nBọn m...	https://www.tripadvisor.com.vn
4	Hoang Anh B	Hoang Anh B 9 thg 5	tháng 5 năm 2023	Đã đi du lịch với gia đình	50	Trải nghiệm tuyệt vời!	Tôi và gia đình đã có khoảng thgian tuyệt vời ...	https://www.tripadvisor.com.vn
...	...	...	...	...	...	...	...	
326	Intlgal007	Intlgal007 thg 7 2018	tháng 7 năm 2018	Đã đi du lịch một mình	50	Khách sạn tuyệt vời và nhân viên xuất sắc	Bốn điểm nằm xa hơn một chút xuống bãi biển. V...	https://www.tripadvisor.com.vn
327	Ngoc T T	Ngoc T T thg 1 2018	tháng 1 năm 2018	Đã đi du lịch theo đôi	40	Great Stay	Tôi đã có một kỳ nghỉ thực sự tuyệt vời ở đây....	https://www.tripadvisor.com.vn
328	Des D	Des D thg 7 2018	tháng 7 năm 2018	Đã đi du lịch theo dạng công tác	50	Vượt quá mong đợi	Đây là lần thứ 2 của tôi ở tại khách sạn này. ...	https://www.tripadvisor.com.vn

	Tên	Ngày comment	Ngày ở	Loại hình du lịch	Rating	Title comment	Content comment	
329	Lucy L	Lucy L thg 7 2018	tháng 7 năm 2018	Đã đi du lịch với gia đình	50	Kinh ngạc..	Chúng tôi đã có một thời gian khó khăn để tìm ...	https://www.tripadvisor.com.vn
330	Queenie	Queenie thg 7 2018	tháng 7 năm 2018	Đã đi du lịch với bạn bè	50	Phòng Tuyệt vời và Đồ ăn Tuyệt vời	Tôi ở lại một đêm ở đó và được cho phòng góc v...	https://www.tripadvisor.com.vn

331 rows × 9 columns

In [58]:
```
ks_10 = data_10[['Content comment']]
```

In [59]:
```
ks_10
```

Out[59]:

	Content comment
0	Phòng Delux đầy đủ tiện nghi và rất sạch sẽ , ...
1	Đây là Kì nghỉ tuyệt vời nhất của gia đình tôi...
2	Khách sạn danacity năm trên đường đông kinh ng...
3	Mình và người yêu đã nghỉ ở đây 1 tuần.\nBọn m...
4	Tôi và gia đình đã có khoảng thgian tuyệt vời ...
...	...
326	Bốn điểm nằm xa hơn một chút xuống bãi biển. V...
327	Tôi đã có một kỳ nghỉ thực sự tuyệt vời ở đây....
328	Đây là lần thứ 2 của tôi ở tại khách sạn này. ...
329	Chúng tôi đã có một thời gian khó khăn để tìm ...
330	Tôi ở lại một đêm ở đó và được cho phòng góc v...

331 rows × 1 columns

In [60]:
```
review10 = ks_10.values
```

In [61]:
```
review10
```

Out[61]:
```
array([['Phòng Delux đầy đủ tiện nghi và rất sạch sẽ , các bạn nhân viên
rất dễ thương! Cảm ơn các bạn!\nMình nghỉ lại khách sạn, cảm giác rất tho
ải mái, các bạn lễ tân giúp đỡ mình rất nhiều trong việc di chuyển cũng n
hư việc tư vấn các địa điểm đi chơi và ăn uống.'],
       ['Đây là Kì nghỉ tuyệt vời nhất của gia đình tôi tại Đà Nẵng. khác
h sạn rất nhỏ nhưng ấm cúng. Nhân viên vô cùng thân thiện dễ mến.giá cả k
hách sạn cũng rất phải chăng. Vị trí khách sạn rất gần biển nhiều đồ ăn n
gon và tôi thích nhất điều này vì không phải mất nhiều chi phí cho việc đ
i lại. nếu có dịp tôi chắc chắn quay lại ks. Cảm ơn Ann quản lí trẻ siêu
dễ thương.'],
       ['Khách sạn danacity năm trên đường đông kinh nghĩa thục là vị trí
trung tấm cách khoảng 2, 3phut đi bộ tới biển Mỹ Khê. Biển Mỹ Khê là bãi
biển nổi tiếp về vẻ đẹp và sự langc mạn, nhẹ nhàng. Ngoài ra khách sạn cũ
ng gân khu tâm ăn uống, tai đây có thể thoải mái khám phá về đêm, rất thu
ận tiện...'],
       ['Mình và người yêu đã nghỉ ở đây 1 tuần.\nBọn mình có đến từ sáng
sớm và được hỗ trợ checkin phòng sớm mà không cần trả thêm phụ phí. Thuê
xe máy nửa ngày tính giá nửa ngày luôn :) phòng sạch tuy nhiên hơi bí do
nhỏ và ở tầng 1, nhưng mục đích của mình cũng chỉ để có chỗ nghỉ còn đâu
```

In [62]:
```
reviews_processed_ks = []
for txt in review10:
    txt = txt_processed(txt)
    reviews_processed_ks.append(txt)
```

In [63]:
```
data_pred10 = pd.DataFrame(reviews_processed_ks, columns=['reviews'])
```

In [64]:
```
data_pred10
```

Out[64]:

	reviews
0	phòng delux đầy_đủ tiện_nghi sạch_sẽ nhân_viên...
1	kì nghỉ tuyệt_vời gia_đình đà_nẵng khách_sạn ấ...
2	khách_sạn danacity đường đông kinh_nghĩa thục ...
3	người_yêu nghỉ tuầnnbọn checkin phòng phụ_phí ...
4	gia_đình thgian tuyệt_vời khách_sạn nhân_viên ...
...	...
326	bốn nằm một_chút bãi biển tiếng ồn tiếng còi đ...
327	kỳ nghỉ tuyệt_vời phòng tuyệt_vời dịch_vụ khác...
328	khách_sạn khách_sạn mở_cửa mọc khách_sạn hoàn_...
329	tìm_kiếm khách_sạn tốt_đẹp đà_nẵng kết_thúc ph...
330	đêm phòng góc tầm ngoạn_mục biển phòng nội_thấ...

331 rows × 1 columns

In [65]:
```
X_pred10 = data_pred10['reviews']
```

In [66]:
```
tokenizer.fit_on_texts(X_pred10)
```

In [67]:
```
X_pred10 = tokenizer.texts_to_sequences(X_pred10)
padded_pred_text10 = pad_sequences(X_pred10, maxlen = max_length, truncating=
```

In [68]:
```
y_pred_dt10 = dtree.predict(padded_pred_text10)
```

In [69]:
```
df10 = pd.DataFrame(y_pred_dt10, columns=['Predict'])
```

In [70]:
```
df10['reviews'] = data_pred10['reviews']
```

In [71]:
```
data_reviews10 = data_pred10['reviews']
```

Get topics

In [72]:
```
data_pred10= data_reviews10.values.tolist()
data_words10 = list(sent_to_words(data_pred10))
```

In [73]:
```
id2word10 = corpora.Dictionary(data_words10)
corpus10 = [id2word10.doc2bow(text) for text in data_words10]
print(len(corpus10))
```

```
331
```

In [74]:
```
from gensim.models.coherencemodel import CoherenceModel

# Compute Perplexity
print('\nPerplexity: ', lda_model.log_perplexity(corpus))  # a measure of how

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model, texts=data_words, dicti
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Perplexity:  -6.818831952551743

Coherence Score:  0.36892525379509067
```

In [75]:
```
def format_topics_sentences(ldamodel=lda_model, corpus=corpus, texts=data):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row in enumerate(ldamodel[corpus]):
        row = sorted(row[0], key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each doc
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                new_row = pd.Series([int(topic_num), round(prop_topic, 4), to
                sent_topics_df = pd.concat([sent_topics_df, new_row.to_frame(

                #sent_topics_df = sent_topics_df.append(pd.Series([int(topic_
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_K

    # Add original text to the end of the output
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)
```

In [76]:
```
df_topic_sents_keywords = format_topics_sentences(ldamodel=lda_model, corpus=

# Format
df_dominant_topic = df_topic_sents_keywords.reset_index()
df_dominant_topic.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Con

# Show
df_dominant_topic
```

Out[76]:

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	0	0.5058	khách_sạn, nhân_viên, phòng, đẹp, nhiệt_tình, ...	[phòng, delux, đầy_đủ, tiện_nghi, sạch_sẽ, nhâ...
1	1	1	0.5072	khách_sạn, phòng, nhân_viên, biển, đẹp, đi, đà...	[kì, nghỉ, tuyệt_vời, gia_đình, đà_nẵng, khách...
2	2	1	0.5048	khách_sạn, phòng, nhân_viên, biển, đẹp, đi, đà...	[khách_sạn, danacity, đường, đông, kinh_nghĩa,...
3	3	0	0.505	khách_sạn, nhân_viên, phòng, đẹp, nhiệt_tình, ...	[người_yêu, nghỉ, tuầnnbọn, checkin, phòng, ph...
4	4	0	0.5018	khách_sạn, nhân_viên, phòng, đẹp, nhiệt_tình, ...	[gia_đình, thgian, tuyệt_vời, khách_sạn, nhân_...
...	...	...	...	...	...
326	326	0	0.5086	khách_sạn, nhân_viên, phòng, đẹp, nhiệt_tình, ...	[bốn, nằm, một_chút, bãi, biển, tiếng, ồn, tiế...
327	327	1	0.5082	khách_sạn, phòng, nhân_viên, biển, đẹp, đi, đà...	[kỳ, nghỉ, tuyệt_vời, phòng, tuyệt_vời, dịch_v...
328	328	1	0.5001	khách_sạn, phòng, nhân_viên, biển, đẹp, đi, đà...	[khách_sạn, khách_sạn, mở_cửa, mọc, khách_sạn,...
329	329	1	0.5136	khách_sạn, phòng, nhân_viên, biển, đẹp, đi, đà...	[tìm_kiếm, khách_sạn, tốt_đẹp, đà_nẵng, kết_th...
330	330	1	0.5161	khách_sạn, phòng, nhân_viên, biển, đẹp, đi, đà...	[đêm, phòng, góc, tầm, ngoạn_mục, biển, phòng,...

331 rows × 5 columns

In [77]:
```
df10['topics'] = df_dominant_topic['Dominant_Topic']
```

results

In [78]:
```
df10['hotel'] = data_10['Tên ks']
```

In [79]:
```
df10['predict'] = df10['Predict']
```

In [80]:
```
df10 = df10.drop('Predict', axis =1 )
```

In [81]:
```
df10
```

Out[81]:

	reviews	topics	hotel	predict
0	phòng delux đầy_đủ tiện_nghi sạch_sẽ nhân_viên...	0	Monsieur Diesel Hotel	1
1	kì nghỉ tuyệt_vời gia_đình đà_nẵng khách_sạn ấ...	1	Monsieur Diesel Hotel	1
2	khách_sạn danacity đường đông kinh_nghĩa thục ...	1	Danaciti Hotel	-1
3	người_yêu nghỉ tuầnnbọn checkin phòng phụ_phí ...	0	Lucky Bee Homestay	1
4	gia_đình thgian tuyệt_vời khách_sạn nhân_viên ...	0	Four Points by Sheraton	1
...	...	...	...	...
326	bốn nằm một_chút bãi biển tiếng ồn tiếng còi đ...	0	Four Points by Sheraton	-1
327	kỳ nghỉ tuyệt_vời phòng tuyệt_vời dịch_vụ khác...	1	Pavilion Hotel	1
328	khách_sạn khách_sạn mở_cửa mọc khách_sạn hoàn_...	1	Four Points by Sheraton	1
329	tìm_kiếm khách_sạn tốt_đẹp đà_nẵng kết_thúc ph...	1	Four Points by Sheraton	1
330	đêm phòng góc tầm ngoạn_mục biển phòng nội_thấ...	1	Four Points by Sheraton	1

331 rows × 4 columns

In [82]:
```
df10['hotel'].value_counts()
```

Out[82]:
```
hotel
Four Points by Sheraton       57
Pavilion Hotel                53
Yarra Ocean Suites            41
Sanouva Danang Hotel          38
TIA Wellness Resort Spa       36
Jolia Hotel Apartment         36
Lahome Apartment And Villa    23
Monsieur Diesel Hotel         16
Lucky Bee Homestay            16
Danaciti Hotel                15
Name: count, dtype: int64
```

In [83]:
```
def calculated(column):
    count1 = 0
    count0 = 0
    for i in column:
        if i == 1:
            count1 +=1
        else:
            count0  += 1
    a = round(((count1-count0)/(count1+count0)),2)
    return a
```

In [84]:
```
unique_values_hotel = df10['hotel'].unique().tolist()
```

In [85]:
```
unique_values_hotel
```

Out[85]:
```
['Monsieur Diesel Hotel',
 'Danaciti Hotel',
 'Lucky Bee Homestay',
 'Four Points by Sheraton',
 'Pavilion Hotel',
 'TIA Wellness Resort Spa',
 'Yarra Ocean Suites',
 'Sanouva Danang Hotel',
 'Lahome Apartment And Villa',
 'Jolia Hotel Apartment']
```

In [86]:
```
df = pd.DataFrame(unique_values_hotel, columns = ['Hotel'])
```

In [87]:
```
average_predict = df10[df10['topics'] == 0].groupby(['hotel', 'topics'])['pre
```

In [88]:
```
average_values = average_predict.values
```

In [89]:
```
df['Phòng'] = pd.DataFrame(average_values)
```

In [90]:
```
average_predict1 = df10[df10['topics'] == 1].groupby(['hotel', 'topics'])['pr
```

In [91]:
```
average_values1 = average_predict1.values
```

In [92]:
```
df['Nhân viên'] = pd.DataFrame(average_values1)
```

In [93]:
```
average_predict_score = df10.groupby(['hotel'])['predict'].apply(calculated)
```

In [94]:
```
average_values_score = average_predict_score.values
```

In [95]:
```
df['Điểm bình luận'] = pd.DataFrame(average_values_score)
```

In [96]:
```
df
```

Out[96]:

	Hotel	Phòng	Nhân viên	Điểm bình luận
0	Monsieur Diesel Hotel	1.00	0.67	0.73
1	Danaciti Hotel	0.68	0.94	0.82
2	Lucky Bee Homestay	0.75	0.93	0.89
3	Four Points by Sheraton	0.85	0.80	0.83
4	Pavilion Hotel	1.00	0.71	0.75
5	TIA Wellness Resort Spa	1.00	1.00	1.00
6	Yarra Ocean Suites	1.00	0.95	0.96
7	Sanouva Danang Hotel	1.00	0.91	0.95
8	Lahome Apartment And Villa	0.85	0.83	0.83
9	Jolia Hotel Apartment	0.87	1.00	0.95

score = 0,229871176*Phòng + 0,122181965*Nhân viên + 0,6479486* Điểm bình luận

In [97]:
```
X1=np.array(df['Phòng'])
```

In [98]:
```
X2=np.array(df['Nhân viên'])
```

In [99]:
```
X3=np.array(df['Điểm bình luận'])
```

In [100]:
```
score = 0.229871176*X1 + 0.122181965*X2 + 0.6479486*X3
```

In [101]:
```
score = np.round(score,3)
```

In [102]:
```
df['Score'] = pd.DataFrame(score)
```

In [103]:
```
df
```

Out[103]:

	Hotel	Phòng	Nhân viên	Điểm bình luận	Score
0	Monsieur Diesel Hotel	1.00	0.67	0.73	0.785
1	Danaciti Hotel	0.68	0.94	0.82	0.802
2	Lucky Bee Homestay	0.75	0.93	0.89	0.863
3	Four Points by Sheraton	0.85	0.80	0.83	0.831
4	Pavilion Hotel	1.00	0.71	0.75	0.803
5	TIA Wellness Resort Spa	1.00	1.00	1.00	1.000
6	Yarra Ocean Suites	1.00	0.95	0.96	0.968
7	Sanouva Danang Hotel	1.00	0.91	0.95	0.957
8	Lahome Apartment And Villa	0.85	0.83	0.83	0.835
9	Jolia Hotel Apartment	0.87	1.00	0.95	0.938

In [104]:
```
sorted_df = df.sort_values('Score', ascending=False)
```

In [105]:
```
sorted_df
```

Out[105]:

	Hotel	Phòng	Nhân viên	Điểm bình luận	Score
5	TIA Wellness Resort Spa	1.00	1.00	1.00	1.000
6	Yarra Ocean Suites	1.00	0.95	0.96	0.968
7	Sanouva Danang Hotel	1.00	0.91	0.95	0.957
9	Jolia Hotel Apartment	0.87	1.00	0.95	0.938
2	Lucky Bee Homestay	0.75	0.93	0.89	0.863
8	Lahome Apartment And Villa	0.85	0.83	0.83	0.835
3	Four Points by Sheraton	0.85	0.80	0.83	0.831
4	Pavilion Hotel	1.00	0.71	0.75	0.803
1	Danaciti Hotel	0.68	0.94	0.82	0.802
0	Monsieur Diesel Hotel	1.00	0.67	0.73	0.785

In [ ]:
```

```