CSCI 544- Applied Natural Language Processing

HW 1

Question 1)

- Average length of reviews before and after data cleaning (with a comma between them)

```
In [18]: print("Average length of reviews before and after data cleaning:",dt_before_clean,',',dt_after_clean)

Average length of reviews before and after data cleaning: 269.4664 , 261.4241666666667
```

Question 2)

Average length of reviews before and after data preprocessing (with comma between them)

```
In [25]: print("Average length of reviews before and after data preprocessing:",dt_before_preprocess,',',dt_after_preprocess)

Average length of reviews before and after data preprocessing: 261.4241666666667 , 251.40098333333333
```

Question3)

Precision, Recall, and f1-score for the testing split in 4 lines (in the order of rating classes and then the average) for Perceptron (with comma between the three values)

```
---------------------------------Perceptron Classification---------------------------------------------------------------------------
class 1 Precision: 0.7201382033563672 , Recall: 0.7256901268341208 , F1 score: 0.7229035055122012 , average: 0.7229106119008963
class 2 Precision: 0.6285034373347436 , Recall: 0.6080839089281146 , F1 score: 0.6181250812638149 , average: 0.6182374758422243
class 3 Precision: 0.7959673547767643 , Recall: 0.8147420147420148 , F1 score: 0.805245264691598 , average: 0.8053182114034589
```

Question 4)

Precision, Recall, and f1-score for the testing split in 4 lines (in the order of rating classes and then the average) for SVM (with comma between the three values)

```
--------------------------------Support Vector Machine Classification-----------------------------------------------------------
class 1 Precision: 0.7509930486593843 , Recall: 0.7523004227804029 , F1 score: 0.7516461672257424 , average: 0.7516465462218432
class 2 Precision: 0.6531625718766335 , Recall: 0.639293937068304 , F1 score: 0.6461538461538462 , average: 0.6462034516995945
class 3 Precision: 0.8210323203087313 , Recall: 0.8363636363636363 , F1 score: 0.8286270691333981 , average: 0.8286743419352552
```

Question 5)

Precision, Recall, and f1-score for the testing split in 4 lines (in the order of rating classes and then the average) for Logistic Regression (with comma between the three values)

```
------------------------------------------Logistic Regression Classification-----------------------------------------------
class 1 Precision: 0.7474368592148037 , Recall: 0.7433474260134295 , F1 score: 0.7453865336658354 , average: 0.7453902729646895
class 2 Precision: 0.6449257114077058 , Recall: 0.655154771041187 , F1 score: 0.65 , average: 0.6500268274829643
class 3 Precision: 0.8267990074441688 , Recall: 0.8186732186732186 , F1 score: 0.8227160493827161 , average: 0.8227294251667011
```

Question 6)

Precision, Recall, and f1-score for the testing split in 4 lines (in the order of rating classes and then the average) for Naive Bayes (with comma between the three values)

```
---------------------------------------------Naive Bayes Classification---------------------------------------------------
class 1 Precision: 0.7722342733188721 , Recall: 0.7082815220094504 , F1 score: 0.7388766376961993 , average: 0.7397974776748405
class 2 Precision: 0.5786831434639892 , Recall: 0.7666922486569455 , F1 score: 0.6595510563380282 , average: 0.6683088161529875
class 3 Precision: 0.8962655601659751 , Recall: 0.6899262899262899 , F1 score: 0.7796751353602667 , average: 0.7886223284841772
```

During the Data Cleaning phase, I have implemented the following actions on the dataset-

Lower case all the strings in the dataset, while strip is used to remove the extra spaces before and after the strings, Contractions is used to remove all the words which are shortened by dropping letters and replacing them by apostrophe.

The contraction step comes before the punctuation removal step because if we use the punctuation removal step first, the apostrophe, which is a characteristic feature of removing contractions, will be lost, and the contraction step will be useless and will not affect the dataset, resulting in random words in the strings that make no sense.

After contractions are removed, the main non-contributing things are the HTML and URL tags, which are removed using Regex library of Python and the extra white spaces are also removed between words.

After the Data Cleaning step, Data Pre-processing is used to remove the non-essential words which do not contribute much to the sentiment of the review,

For this purpose, firstly I removed the stop-words, but the precision come out to be less, around 63% but when I tried to perform the models without removing the stop words, the precision came out to be better compared to with removing stop words around 70%.

After that, Lemmatization is used to extract the base words, which contribute more towards the sentiments rather than using the actual words which can be deceiving towards extracting the sentiments of the sentences.

After the Data Pre-processing step, TF-IDF feature extraction is used get the important features out of all the sentences, which directly contribute getting the exact sentiment of the sentences.

Then comes the step to implement all the Machine Learning models namely- Perceptron, Support Vector Machine, Logistic regression and Naïve Bayes, and the result are as follows-

Perceptron-

	precision	recall	f1-score	support
class 1	0.72	0.73	0.72	4021
class 2	0.63	0.61	0.62	3909
class 3	0.80	0.81	0.81	4070
accuracy			0.72	12000
macro avg	0.71	0.72	0.72	12000
weighted avg	0.72	0.72	0.72	12000

Support Vector Machine-

	precision	recall	f1-score	support
class 1	0.75	0.75	0.75	4021
class 2	0.65	0.64	0.65	3909
class 3	0.82	0.84	0.83	4070
accuracy			0.74	12000
macro avg	0.74	0.74	0.74	12000
weighted avg	0.74	0.74	0.74	12000

Logistic Regression-

	precision	recall	f1-score	support
class 1	0.75	0.74	0.75	4021
class 2	0.64	0.66	0.65	3909
class 3	0.83	0.82	0.82	4070
accuracy			0.74	12000
macro avg	0.74	0.74	0.74	12000
weighted avg	0.74	0.74	0.74	12000

Naïve Bayes-

	precision	recall	f1-score	support
class 1	0.77	0.71	0.74	4021
class 2	0.58	0.77	0.66	3909
class 3	0.90	0.69	0.78	4070
accuracy			0.72	12000
macro avg	0.75	0.72	0.73	12000
weighted avg	0.75	0.72	0.73	12000

While Doing the Homework, I have learnt how to do real-life sentiment analysis on the reviews which are on some business platform like Amazon reviews.

The main task of any sentiment analysis is to remove the unnecessary words which doesn't contribute much to the task in hand, while keeping the important words or part of sentences which do contribute. Performing tasks like removing stop words, lemmatization really help me understand the intricacies of how to perform the sentiment analysis perfectly and handle real-life dataset.

While going about the Precision, Recall, and F1-score, I learned about the different metrics which help evaluate the performance of the models and also learned how to increase those metrics values.

In [1]:

```
#pip install contractions
```

In [2]:

```
import pandas as pd
import numpy as np
import nltk
nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('omw-1.4')
import re
from bs4 import BeautifulSoup
import contractions
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_recall_fscore_support as score
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\USER\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\USER\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\USER\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

In [ ]:

Read Data

In [3]:

```
df= pd.read_table('amazon_reviews_us_Beauty_v1_00.tsv',on_bad_lines='skip',low_memory=False)
```

Keep Reviews and Ratings

In [4]:

```
df_fnl= df[['review_body','star_rating']]
```

In [5]:

```
df_fnl
```

Out[5]:

	review_body	star_rating
0	Love this, excellent sun block!!	5
1	The great thing about this cream is that it do...	5
2	Great Product, I'm 65 years old and this is al...	5
3	I use them as shower caps & conditioning caps....	5
4	This is my go-to daily sunblock. It leaves no ...	5
...	...	...
5094302	After watching my Dad struggle with his scisso...	5
5094303	Like most sound machines, the sounds choices a...	3
5094304	I bought this product because it indicated 30 ...	5
5094305	We have used Oral-B products for 15 years; thi...	5

We form three classes and select 20000 reviews randomly from each class.

In [6]:

```
df_final=df_fnl.replace({'star_rating':{2:1,3:2,4:3,5:3,'5':3,'2':1,'3':2,'4':3,'1':1}})
```

In [7]:

```
df_final.isnull().sum()
```

Out[7]:

```
review_body    400
star_rating     10
dtype: int64
```

In [8]:

```
df_final.dropna(inplace=True)
```

In [9]:

```
s0 = df_final[df_final['star_rating'].eq(1)].sample(20000).index
s1 = df_final[df_final['star_rating'].eq(2)].sample(20000).index
s2 = df_final[df_final['star_rating'].eq(3)].sample(20000).index
```

In [10]:

```
df_fi = df_final.loc[s0.union(s1).union(s2)]
```

In [11]:

```
df_fi
```

Out[11]:

	review_body	star_rating
78	Like all of Dove's Men+ line these are good bu...	2
233	I love this product! It always leaves my hair ...	3
367	Wonderful product	3
527	I ordered this almost a year ago to replace my...	1
539	I dont like this product.	1
...	...	...
5094018	The bubble spa is very very loud, the air comi...	2
5094035	the Quiet mode was very loud!....and I still h...	1
5094066	This hair dryer is maybe only slightly quieter...	1
5094081	It works for less than a minute and then the b...	1

In [12]:

```
df_fi.isnull().sum()
```

Out[12]:

```
review_body    0
star_rating    0
dtype: int64
```

Data Cleaning

Pre-processing

In [13]:

```
dt_before_clean=df_fi['review_body'].apply(len).mean()
```

In [15]:

```
def remove_alphanumeric(s):
    s=s.lower()
    s=s.strip()
    s=contractions.fix(s)
    s=s.replace(r'<[^<>]*>', '')
    s=s.replace(r'http\S+', '').replace(r'www\S+', '')
    s=re.sub(r'[^a-zA-Z]',' ',s)
    return re.sub(' +', ' ',s)
```

In [16]:

```
df_fi['review_body']=df_fi['review_body'].apply(remove_alphanumeric)
```

In [17]:

```
dt_after_clean=df_fi['review_body'].apply(len).mean()
```

In [18]:

```
print("Average length of reviews before and after data cleaning:",dt_before_clean,',',dt_after_clean)
```

```
Average length of reviews before and after data cleaning: 269.4664 , 261.4241666666667
```

In [ ]:

remove the stop words

In [19]:

```
dt_before_preprocess=df_fi['review_body'].apply(len).mean()
```

In [20]:

```
# from nltk.corpus import stopwords
# stop = stopwords.words('english')
# df_fi['review_body'] = df_fi['review_body'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
```

In [21]:

```
df_fi
```

Out[21]:

	review_body	star_rating
78	like all of dove s men line these are good but...	2
233	i love this product it always leaves my hair s...	3
367	wonderful product	3
527	i ordered this almost a year ago to replace my...	1
539	i do not like this product	1
...	...	...
5094018	the bubble spa is very very loud the air comin...	2
5094035	the quiet mode was very loud and i still had a...	1
5094066	this hair dryer is maybe only slightly quieter...	1
5094081	it works for less than a minute and then the b...	1

perform lemmatization

In [22]:

```
from nltk.stem import WordNetLemmatizer
lemmatizer=WordNetLemmatizer()
def lemmatize_words(text):
        words = text.split()
        words = [lemmatizer.lemmatize(word,pos='v') for word in words]
        return ' '.join(words)

df_fi['review_body']=df_fi['review_body'].apply(lemmatize_words)
```

In [23]:

```
df_fi
```

	review_body	star_rating
78	like all of dive s men line these be good but ...	2
233	i love this product it always leave my hair so...	3
367	wonderful product	3
527	i order this almost a year ago to replace my f...	1
539	i do not like this product	1
...	...	...
5094018	the bubble spa be very very loud the air come ...	2
5094035	the quiet mode be very loud and i still have a...	1
5094066	this hair dryer be maybe only slightly quieter...	1
5094081	it work for less than a minute and then the bl...	1
5094261	the three speed be fine and the unit generate ...	1

In [24]:

```
dt_after_preprocess=df_fi['review_body'].apply(len).mean()
```

In [25]:

```
print("Average length of reviews before and after data preprocessing:",dt_before_preprocess,',',dt_after_preprocess)
```

```
Average length of reviews before and after data preprocessing: 261.4241666666667 , 251.40098333333333
```

TF-IDF Feature Extraction

In [26]:

```
imp_features = TfidfVectorizer(ngram_range=(1,3))
x = imp_features.fit_transform(df_fi['review_body'])
```

In [27]:

```
X_train, X_test, Y_train, Y_test = train_test_split(x, df_fi['star_rating'], test_size=0.2, random_state=42)
Y_train=Y_train.astype('int')
Y_test=Y_test.astype('int')
```

Perceptron

In [28]:

```
from sklearn.linear_model import Perceptron
clf_percep=Perceptron(tol=1e-3, random_state=0)
clf_percep.fit(X_train,Y_train)
```

Out[28]:

```
▾ Perceptron
Perceptron()
```

In [29]:

```
predicted_perceptron = clf_percep.predict(X_test)
```

In [30]:

```
target_names = ['class 1', 'class 2', 'class 3']
print(classification_report(Y_test, predicted_perceptron, target_names=target_names))
```

```
              precision    recall  f1-score   support

     class 1       0.72      0.73      0.72      4021
     class 2       0.63      0.61      0.62      3909
     class 3       0.80      0.81      0.81      4070

    accuracy                           0.72     12000
   macro avg       0.71      0.72      0.72     12000
weighted avg       0.72      0.72      0.72     12000
```

In [31]:

```
precision, recall, fscore, support = score(Y_test, predicted_perceptron)

# print('precision: {}'.format(precision))
# print('recall: {}'.format(recall))
# print('fscore: {}'.format(fscore))
# print('support: {}'.format(support))
```

In [32]:

```
print('----------------------------------Perceptron Classification--------------------------------------------------------------------')
for i in range(3):
    print('class '+ str(i+1),'Precision:',precision[i],',','Recall:', recall[i],',', 'F1 score:',fscore[i],',','average:',((precision[i]+
```

```
----------------------------------Perceptron Classification-----------------------------------------------------------------
----
class 1 Precision: 0.7201382033563672 , Recall: 0.7256901268341208 , F1 score: 0.7229035055122012 , average: 0.722910611900
8963
class 2 Precision: 0.6285034373347436 , Recall: 0.6080839089281146 , F1 score: 0.6181250812638149 , average: 0.618237475842
2243
class 3 Precision: 0.7959673547767643 , Recall: 0.8147420147420148 , F1 score: 0.805245264691598 , average: 0.8053182114034
589
```

SVM

In [33]:

```
from sklearn.svm import LinearSVC
clf_SVM= LinearSVC(random_state=42, tol=1e-5)
clf_SVM.fit(X_train,Y_train)
```

Out[33]:

```
▾ LinearSVC
LinearSVC(random_state=42, tol=1e-05)
```

In [34]:

```
predicted_SVM=clf_SVM.predict(X_test)
```

In [35]:

```
target_names = ['class 1', 'class 2', 'class 3']
print(classification_report(Y_test, predicted_SVM, target_names=target_names))
```

```
              precision    recall  f1-score   support

     class 1       0.75      0.75      0.75      4021
     class 2       0.65      0.64      0.65      3909
     class 3       0.82      0.84      0.83      4070

    accuracy                           0.74     12000
   macro avg       0.74      0.74      0.74     12000
weighted avg       0.74      0.74      0.74     12000
```

In [36]:

```
precision, recall, fscore, support = score(Y_test, predicted_SVM)

print('precision: {}'.format(precision))
print('recall: {}'.format(recall))
print('fscore: {}'.format(fscore))
print('support: {}'.format(support))
```

```
precision: [0.75099305 0.65316257 0.82103232]
recall: [0.75230042 0.63929394 0.83636364]
fscore: [0.75164617 0.64615385 0.82862707]
support: [4021 3909 4070]
```

In [37]:

```
print('---------------------------------Support Vector Machine Classification--------------------------------------------------------')
for i in range(3):
    print('class '+ str(i+1),'Precision:',precision[i],',','Recall:', recall[i],',', 'F1 score:',fscore[i],',','average:',((precision[i]+
```

```
---------------------------------Support Vector Machine Classification--------------------------------------------------
----
class 1 Precision: 0.7509930486593843 , Recall: 0.7523004227804029 , F1 score: 0.7516461672257424 , average: 0.751646546221
8432
class 2 Precision: 0.6531625718766335 , Recall: 0.639293937068304 , F1 score: 0.6461538461538462 , average: 0.6462034516995
945
class 3 Precision: 0.8210323203087313 , Recall: 0.8363636363636363 , F1 score: 0.8286270691333981 , average: 0.828674341935
2552
```

Logistic Regression

In [38]:

```
from sklearn.linear_model import LogisticRegression

clf_Logistic= LogisticRegression(random_state=42,max_iter=1000000)
clf_Logistic.fit(X_train,Y_train)
```

Out[38]:

```
▾ LogisticRegression
LogisticRegression(max_iter=1000000, random_state=42)
```

In [39]:

```
predicted_Log=clf_Logistic.predict(X_test)
```

In [40]:

```
target_names = ['class 1', 'class 2', 'class 3']
print(classification_report(Y_test, predicted_Log, target_names=target_names))
```

```
              precision    recall  f1-score   support

     class 1       0.75      0.74      0.75      4021
     class 2       0.64      0.66      0.65      3909
     class 3       0.83      0.82      0.82      4070

    accuracy                           0.74     12000
   macro avg       0.74      0.74      0.74     12000
weighted avg       0.74      0.74      0.74     12000
```

In [41]:

```
precision, recall, fscore, support = score(Y_test, predicted_Log)

print('precision: {}'.format(precision))
print('recall: {}'.format(recall))
print('fscore: {}'.format(fscore))
print('support: {}'.format(support))
```

```
precision: [0.74743686 0.64492571 0.82679901]
recall: [0.74334743 0.65515477 0.81867322]
fscore: [0.74538653 0.65       0.82271605]
support: [4021 3909 4070]
```

In [42]:

```
print('--------------------------------------------Logistic Regression Classification-------------------------------------------------')
for i in range(3):
    print('class '+ str(i+1),'Precision:',precision[i],',','Recall:', recall[i],',', 'F1 score:',fscore[i],',','average:',((precision[i]+
```

```
--------------------------------------------Logistic Regression Classification-------------------------------------------
----
class 1 Precision: 0.7474368592148037 , Recall: 0.7433474260134295 , F1 score: 0.7453865336658354 , average: 0.745390272964
6895
class 2 Precision: 0.6449257114077058 , Recall: 0.655154771041187 , F1 score: 0.65 , average: 0.6500268274829643
class 3 Precision: 0.8267990074441688 , Recall: 0.8186732186732186 , F1 score: 0.8227160493827161 , average: 0.822729425166
7011
```

Naive Bayes

In [43]:

```
from sklearn.naive_bayes import MultinomialNB

clf_naive = MultinomialNB()
clf_naive.fit(X_train, Y_train)
```

Out[43]:

```
▾ MultinomialNB
MultinomialNB()
```

In [44]:

```
predicted_naive = clf_naive.predict(X_test)
```

In [45]:

```
conf=confusion_matrix(Y_test,predicted_naive)
```

In [46]:

```
target_names = ['class 1', 'class 2', 'class 3']
print(classification_report(Y_test, predicted_naive, target_names=target_names))
```

```
              precision    recall  f1-score   support

     class 1       0.77      0.71      0.74      4021
     class 2       0.58      0.77      0.66      3909
     class 3       0.90      0.69      0.78      4070

    accuracy                           0.72     12000
   macro avg       0.75      0.72      0.73     12000
weighted avg       0.75      0.72      0.73     12000
```

In [ ]:

In [47]:

```
precision, recall, fscore, support = score(Y_test, predicted_naive)

print('precision: {}'.format(precision))
print('recall: {}'.format(recall))
print('fscore: {}'.format(fscore))
print('support: {}'.format(support))
```

```
precision: [0.77223427 0.57868314 0.89626556]
recall: [0.70828152 0.76669225 0.68992629]
fscore: [0.73887664 0.65955106 0.77967514]
support: [4021 3909 4070]
```

In [ ]:

In [48]:

```
print('-----------------------------------------------Naive Bayes Classification-----------------------------------------------------')
for i in range(3):
    print('class '+ str(i+1),'Precision:',precision[i],',','Recall:', recall[i],',', 'F1 score:',fscore[i],',','average:',((precision[i]+
```

```
-----------------------------------------------Naive Bayes Classification-----------------------------------------------
---
class 1 Precision: 0.7722342733188721 , Recall: 0.7082815220094504 , F1 score: 0.7388766376961993 , average: 0.739797477674
8405
class 2 Precision: 0.5786831434639892 , Recall: 0.7666922486569455 , F1 score: 0.6595510563380282 , average: 0.668308816152
9875
class 3 Precision: 0.8962655601659751 , Recall: 0.6899262899262899 , F1 score: 0.7796751353602667 , average: 0.788622328484
1772
```

In [ ]: