CSCI544: Homework Assignment No 1
Nehal Muthukumar
1677301672

1) Dataset Preparation

- The data is downloaded from the given link (https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_ us_Beauty_v1_00.tsv.gz) and unzipped to get the .tsv file.
- It is loaded into a data frame using the pandas framework in python with the delimiter "\t" since it is a tsv file.
- Then only the Reviews and rating fields are extracted. We used both review_title and review_body as review.
- The star_ rating columns contained values other than 1-5. So it was handled by
  - removing rows having Nan reviews or label
  - Mapping all ratings to the same data type. Eg: string ratings like '2' was changed to 2.
- The ratings are relabelled as given
- 1&2 -> class 1, 3 -> class 2, 4&5 -> class 3
- Then we divide them into 3 data frames one for each class, sampled 20000 data points from each class/data frame, and merged them leading to 60000 data points.

2) Data Cleaning

- The reviews are first converted to lowercase using the str.lower() method on the data frame.
- HTML tags are removed using the beautiful soup library. We use the default html.parser and get_text() method to remove all tags.
- URLs are removed using regular expressions. Any substring that starts with "http", "https", or "www" is removed.
- Non-alphabetic characters are removed using regular expressions, keeping only characters a-z and A-Z.
- Extra spaces are removed using regular expressions, replacing any pattern of one or more spaces with a single space.
- Leading and trailing spaces in the reviews are trimmed using the strip() method.
- Contractions are applied using the contractions library in python to standardize the text.

Average Length of reviews before cleaning: 293
Average Length of reviews after cleaning: 281

3) Preprocessing

Stop Word Removal: We use the NLTK library to import a set of stop words and remove them from the reviews by tokenizing the text and matching the tokens against the set of stop words.

Lemmatization: We use the NLTK library to perform parts of speech tagging on the tokenized reviews. This allows us to identify the base form of the words as nouns, verbs, adjectives, etc. We then use the WordNetLemmatizer from the NLTK library to perform lemmatization and reduce the words to their base form.

Average Length of reviews before pre processing: 281
Average Length of reviews after pre processing: 165

4) Feature Extraction

- We perform a 80-20 split of the data into training and testing sets, with 4800 datapoints for training and 12000 data points for testing.
- To ensure an equal representation of each class in both sets, we use a stratified sampling method.
- To capture the context of the words in the reviews, we use both bigrams and trigrams in our feature extraction process.
- We use the Tf-Idf vectorization method in sklearn to create the Tf-Idf vectors for both the training and testing data.

5) Perceptron

Perceptron model from sklearn is used

	precision	recall	f1-score	support
1	0.78	0.74	0.76	4000
2	0.69	0.68	0.69	4000
3	0.79	0.85	0.82	4000
accuracy			0.76	12000
macro avg	0.76	0.76	0.76	12000
weighted avg	0.76	0.76	0.76	12000

```
0.7841 , 0.7365 , 0.7596
0.6928 , 0.6815 , 0.6871
0.7925 , 0.8535 , 0.8219
0.7565 , 0.7572 , 0.7562
```

6) SVM

LinearSVC model is used

	precision	recall	f1-score	support
1	0.81	0.75	0.78	4000
2	0.71	0.71	0.71	4000
3	0.80	0.86	0.83	4000
accuracy			0.77	12000
macro avg	0.77	0.77	0.77	12000
weighted avg	0.77	0.77	0.77	12000

```
0.8096 , 0.7482 , 0.7777
0.7075 , 0.7053 , 0.7064
0.7998 , 0.8630 , 0.8302
0.7723 , 0.7722 , 0.7714
```

7) Logistic Regression

LogisticRegression model from sklearn is used

	precision	recall	f1-score	support
1	0.81	0.74	0.77	4000
2	0.70	0.71	0.70	4000
3	0.80	0.85	0.82	4000
accuracy			0.77	12000
macro avg	0.77	0.77	0.77	12000
weighted avg	0.77	0.77	0.77	12000

0.8064	0.7392	0.7714
0.6952	0.7110	0.7030
0.7980	0.8462	0.8214
0.7665	0.7655	0.7653

8) Multinomial Naive Bayes

MultinomialNB model from sklearn is used

	precision	recall	f1-score	support
1	0.78	0.76	0.77	4000
2	0.68	0.71	0.70	4000
3	0.84	0.82	0.83	4000
accuracy			0.76	12000
macro avg	0.77	0.76	0.77	12000
weighted avg	0.77	0.76	0.77	12000

0.7807	0.7572	0.7688
0.6826	0.7145	0.6982
0.8365	0.8225	0.8294
0.7666	0.7647	0.7655

9) Improving Performance

We ran 3 variants for the experiment

1) Stop word removal + lemmatization
2) No stop word removal + lemmatization
3) No stop word removal + No lemmatization

Perceptron

Stopword+lemmatization

	precision	recall	f1-score	support
1	0.78	0.74	0.76	4000
2	0.69	0.68	0.69	4000
3	0.79	0.85	0.82	4000
accuracy			0.76	12000
macro avg	0.76	0.76	0.76	12000
weighted avg	0.76	0.76	0.76	12000

No stopword removal+lemmatization

	precision	recall	f1-score	support
1	0.78	0.74	0.76	4000
2	0.69	0.68	0.69	4000
3	0.79	0.85	0.82	4000
accuracy			0.76	12000
macro avg	0.76	0.76	0.76	12000
weighted avg	0.76	0.76	0.76	12000

No stopword removal+ No lemmatization

	precision	recall	f1-score	support
1	0.81	0.79	0.80	4000
2	0.73	0.74	0.73	4000
3	0.86	0.88	0.87	4000
accuracy			0.80	12000
macro avg	0.80	0.80	0.80	12000
weighted avg	0.80	0.80	0.80	12000

SVM

Stopword+lemmatization

	precision	recall	f1-score	support
1	0.81	0.75	0.78	4000
2	0.71	0.71	0.71	4000
3	0.80	0.86	0.83	4000
accuracy			0.77	12000
macro avg	0.77	0.77	0.77	12000
weighted avg	0.77	0.77	0.77	12000

No stopword removal+lemmatization

	precision	recall	f1-score	support
1	0.81	0.75	0.78	4000
2	0.71	0.71	0.71	4000
3	0.80	0.86	0.83	4000
accuracy			0.77	12000
macro avg	0.77	0.77	0.77	12000
weighted avg	0.77	0.77	0.77	12000

No stopword removal+ No lemmatization

	precision	recall	f1-score	support
1	0.83	0.80	0.82	4000
2	0.75	0.76	0.75	4000
3	0.88	0.88	0.88	4000
accuracy			0.82	12000
macro avg	0.82	0.82	0.82	12000
weighted avg	0.82	0.82	0.82	12000

Logistic regression

Stopword+lemmatization

	precision	recall	f1-score	support
1	0.81	0.74	0.77	4000
2	0.70	0.71	0.70	4000
3	0.80	0.85	0.82	4000
accuracy			0.77	12000
macro avg	0.77	0.77	0.77	12000
weighted avg	0.77	0.77	0.77	12000

No stopword removal+lemmatization

	precision	recall	f1-score	support
1	0.81	0.74	0.77	4000
2	0.70	0.71	0.70	4000
3	0.80	0.85	0.82	4000
accuracy			0.77	12000
macro avg	0.77	0.77	0.77	12000
weighted avg	0.77	0.77	0.77	12000

No stopword removal+ No lemmatization

	precision	recall	f1-score	support
1	0.82	0.80	0.81	4000
2	0.73	0.77	0.75	4000
3	0.88	0.86	0.87	4000
accuracy			0.81	12000
macro avg	0.81	0.81	0.81	12000
weighted avg	0.81	0.81	0.81	12000

Multinomial NB

Stopword+lemmatization

	precision	recall	f1-score	support
1	0.78	0.76	0.77	4000
2	0.68	0.71	0.70	4000
3	0.84	0.82	0.83	4000
accuracy			0.76	12000
macro avg	0.77	0.76	0.77	12000
weighted avg	0.77	0.76	0.77	12000

No stopword removal+lemmatization

	precision	recall	f1-score	support
1	0.78	0.76	0.77	4000
2	0.68	0.71	0.70	4000
3	0.84	0.82	0.83	4000
accuracy			0.76	12000
macro avg	0.77	0.76	0.77	12000
weighted avg	0.77	0.76	0.77	12000

No stopword removal+ No lemmatization

	precision	recall	f1-score	support
1	0.81	0.80	0.81	4000
2	0.69	0.80	0.74	4000
3	0.93	0.79	0.85	4000
accuracy			0.80	12000
macro avg	0.81	0.80	0.80	12000
weighted avg	0.81	0.80	0.80	12000

In all 4 models No stop word removal and No lemmatization give best results.

hw1-csci544

January 25, 2023

[2]:
```
import pandas as pd
import numpy as np
import nltk
nltk.download('wordnet')
import re
from bs4 import BeautifulSoup
```

```
[nltk_data] Downloading package wordnet to C:\Users\MUTHUKUMAR
[nltk_data]     S\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

[3]:
```
! pip install bs4 # in case you don't have it installed
! pip install contractions
```

```
ERROR: Invalid requirement: '#'

[notice] A new release of pip available: 22.3 -> 22.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
Requirement already satisfied: contractions in
e:\usc\spring23\nlp\submit\venv\lib\site-packages (0.1.73)
Requirement already satisfied: textsearch>=0.0.21 in
e:\usc\spring23\nlp\submit\venv\lib\site-packages (from contractions) (0.0.24)
Requirement already satisfied: anyascii in
e:\usc\spring23\nlp\submit\venv\lib\site-packages (from
textsearch>=0.0.21->contractions) (0.3.1)
Requirement already satisfied: pyahocorasick in
e:\usc\spring23\nlp\submit\venv\lib\site-packages (from
textsearch>=0.0.21->contractions) (2.0.0)


[notice] A new release of pip available: 22.3 -> 22.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

0.1 Read Data

[4]:
```
data = pd.read_csv('./data.tsv', sep='\t',on_bad_lines='skip',low_memory=False)
```

[5]:
```
data.head()
```

[5]:
```
  marketplace  customer_id       review_id  product_id  product_parent  \
0          US      1797882  R3I2DHQBR577SS  B001ANOOOE         2102612
1          US     18381298  R1QNE9NQFJC2Y4  B0016J22EQ       106393691
2          US     19242472  R3LIDG2Q4LJBAO  B00HU6UQAG       375449471
3          US     19551372  R3KSZHPAEVPEAL  B002HWS7RM       255651889
4          US     14802407   RAI2OIG50KZ43  B00SM99KWU       116158747

                                       product_title product_category  \
0  The Naked Bee Vitmin C Moisturizing Sunscreen …           Beauty
1      Alba Botanica Sunless Tanning Lotion, 4 Ounce           Beauty
2          Elysee Infusion Skin Therapy Elixir, 2oz.           Beauty
3  Diane D722 Color, Perm And Conditioner Process…           Beauty
4  Biore UV Aqua Rich Watery Essence SPF50+/PA+++…           Beauty

  star_rating  helpful_votes  total_votes vine verified_purchase  \
0           5            0.0          0.0    N                 Y
1           5            0.0          0.0    N                 Y
2           5            0.0          0.0    N                 Y
3           5            0.0          0.0    N                 Y
4           5            0.0          0.0    N                 Y

                                     review_headline  \
0                                         Five Stars
1                          Thank you Alba Bontanica!
2                                         Five Stars
3                                         GOOD DEAL!
4  this soaks in quick and provides a nice base f…

                                         review_body review_date
0                  Love this, excellent sun block!!  2015-08-31
1  The great thing about this cream is that it do…  2015-08-31
2  Great Product, I'm 65 years old and this is al…  2015-08-31
3  I use them as shower caps & conditioning caps…  2015-08-31
4  This is my go-to daily sunblock. It leaves no …  2015-08-31
```

link text## Keep Reviews and Ratings

[6]:
```python
list(data["star_rating"].unique())
```

[6]:
```
['5',
 '4',
 '1',
 '3',
 '2',
 '2015-08-28',
 '2015-08-16',
 '2015-08-14',
```

```
 '2015-07-27',
 '2015-07-26',
 '2015-07-23',
 '2015-07-22',
 nan,
 '2015-06-14',
 '2015-06-02',
 '2015-04-14',
 '2015-04-09',
 '2015-04-08',
 '2015-04-03',
 '2015-04-02',
 '2015-04-01',
 '2015-03-31',
 '2015-03-30',
 '2015-03-18',
 '2015-02-28',
 '2015-02-10',
 '2014-12-30',
 '2014-12-03',
 '2014-10-09']
```

0.1.1 1) review = review_title + review body - body is appended with the title

0.1.2 2) Removing Nan reviews and label

0.1.3 3) Extracting only review body and rating

[7]:
```
data["review_body"]=data["review_headline"]+" "+data["review_body"]
reviews=data[["review_body","star_rating"]].copy()
reviews = reviews[reviews['review_body'].notna()]
reviews
```

[7]:
```
                                                review_body star_rating
0                Five Stars Love this, excellent sun block!!           5
1         Thank you Alba Bontanica! The great thing abou…           5
2         Five Stars Great Product, I'm 65 years old and…           5
3         GOOD DEAL! I use them as shower caps & conditi…           5
4         this soaks in quick and provides a nice base f…           5
…                                                     …         …
5094302   Great Little Grooming Tool After watching my D…           5
5094303   Not bad for the price Like most sound machines…           3
5094304   Best Curling Iron Ever I bought this product b…           5
5094305   The best electric toothbrush ever, REALLY! We …           5
5094306   Smooth and shiny teeth! I love this toothbrush…           5

[5093876 rows x 2 columns]
```

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

[8]:
```
list(reviews["star_rating"].unique())
```

[8]: ['5', '4', '1', '3', '2']

0.1.4 relabelling the classed

0.1.5 1&2 -> class1

0.1.6 3 -> class2

0.1.7 4&5 -> class3

[9]:
```
reviews["star_rating"]=reviews["star_rating"].replace('1',1)
reviews["star_rating"]=reviews["star_rating"].replace(2,1)
reviews["star_rating"]=reviews["star_rating"].replace('2',1)
reviews["star_rating"]=reviews["star_rating"].replace('3',2)
reviews["star_rating"]=reviews["star_rating"].replace(4,3)
reviews["star_rating"]=reviews["star_rating"].replace('4',3)
reviews["star_rating"]=reviews["star_rating"].replace(5,3)
reviews["star_rating"]=reviews["star_rating"].replace('5',3)
```

[10]:
```
reviews["star_rating"].unique()
```

[10]: array([3, 1, 2], dtype=int64)

0.1.8 sampling 20000 datapoints from each class

[11]:
```
class1_df = reviews[reviews["star_rating"]==1]
sample1=class1_df.sample(n = 20000,random_state=47)
sample1 = sample1.reset_index(drop=True)
class2_df=reviews[reviews["star_rating"]==2]
sample2=class2_df.sample(n = 20000,random_state=47)
sample2 = sample2.reset_index(drop=True)
class3_df = reviews[reviews["star_rating"]==3]
sample3=class3_df.sample(n = 20000,random_state=47)
sample3 = sample3.reset_index(drop=True)

#reviews_df = (sample1.append(sample2, ignore_index = True)).append(sample3,␣
↪ignore_index = True)
reviews_df=pd.concat([sample1,sample2,sample3],axis=0,ignore_index=True)
```

[12]:
```
for i in list(reviews_df["star_rating"].unique()):
  print(i , "  ",len(reviews_df[reviews_df["star_rating"]==i]))
```

```
1     20000
2     20000
3     20000
```

[13]:
```
reviews_df['review_body'].str.len().mean()
```

[13]: 293.23768333333334

[14]:
```
len_before_cleaning=reviews_df['review_body'].str.len().mean()
```

1 Data Cleaning

[15]:
```
import contractions
```

1.0.1 lowercase

1.0.2 remove html tags

1.0.3 remove url

1.0.4 remove non-alphabetical

1.0.5 remove extra spaces

1.0.6 contraction

[16]:
```
# lower case
reviews_df['review_body']=reviews_df['review_body'].str.lower()
#remove html tags
reviews_df['review_body'] = reviews_df['review_body'].apply(lambda x:␣
↪BeautifulSoup(x, 'html.parser').get_text())
# remove url
reviews_df['review_body'] =  reviews_df['review_body'].apply(lambda x: re.
↪sub(r'http\S+','', str(x)))
reviews_df['review_body'] =  reviews_df['review_body'].apply(lambda x: re.
↪sub(r'https\S+','', str(x)))
reviews_df['review_body'] =  reviews_df['review_body'].apply(lambda x: re.
↪sub(r'www\.\S+','', str(x)))
#reviews_df['review_body'] =  reviews_df['review_body'].apply(lambda x: re.
↪sub(r'\S+\.com','', str(x)))
#reviews_df['review_body'] =  reviews_df['review_body'].apply(lambda x: re.
↪sub(r'\S+\@gmail','', str(x)))
# remove speci ascii
#reviews_df['review_body'] =  reviews_df['review_body'].apply(lambda x: re.
↪sub(r'&\S+','', str(x)))
# keeping only alphabets
reviews_df['review_body'] =  reviews_df['review_body'].apply(lambda x: re.
↪sub(r'[^a-z A-Z]+',' ', str(x)))
# removes words of length less than equal to 2
#reviews_df['review_body'] =  reviews_df['review_body'].apply(lambda x: re.
↪sub(r'\b\w{1,2}\b','', str(x)))
# removing extra spaces
```

```
reviews_df['review_body'] =  reviews_df['review_body'].apply(lambda x: re.
 ↪sub(r'\s+',' ', str(x)))
# strping spaces at start and end
reviews_df['review_body']=reviews_df['review_body'].str.strip()
# contractions
reviews_df['review_body'] =  reviews_df['review_body'].apply(lambda x:␣
 ↪contractions.fix(x))
```

```
e:\USC\spring23\nlp\submit\venv\Lib\site-packages\bs4\__init__.py:435:
MarkupResemblesLocatorWarning: The input looks more like a filename than markup.
You may want to open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
```

[17]:
```
reviews_df['review_body'].str.len().mean()
```

[17]:
```
281.0772166666667
```

[18]:
```
len_after_cleaning=reviews_df['review_body'].str.len().mean()
```

1.1 Avg length of review before and after cleaning

[19]:
```
print(int(len_before_cleaning),",",int(len_after_cleaning))
```

```
293 , 281
```

2 Pre-processing

2.1 remove the stop words

[20]:
```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

[21]:
```
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to C:\Users\MUTHUKUMAR
[nltk_data]     S\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to C:\Users\MUTHUKUMAR
[nltk_data]     S\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\MUTHUKUMAR S\AppData\Roaming\nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

```
[nltk_data] Downloading package omw-1.4 to C:\Users\MUTHUKUMAR
[nltk_data]     S\AppData\Roaming\nltk_data…
[nltk_data]   Package omw-1.4 is already up-to-date!
```

[21]:
```
True
```

[22]:
```
len_before_preprocess=len_after_cleaning
```

2.1.1 tokenization and stop word removal

[23]:
```
reviews_df['tokenized'] = reviews_df['review_body'].apply(word_tokenize)
stop_words = set(stopwords.words('english'))
reviews_df['stopwords_removed'] = reviews_df['tokenized'].apply(lambda x: [word␣
↪for word in x if word not in stop_words])
reviews_df['stopwords_removed']
```

[23]:
```
0        [rancid, smell, threw, away, smelled, like, go…
1        [flavor, gross, nasty, flavor, product, amazin…
2        [fan, product, fan, product, thick, dry, prope…
3        [worth, investment, using, months, seen, benef…
4        [wow, mean, rude, wow, peice, censored, jesus,…
                               …
59995    [vi, tae, shea, butter, soap, second, purchase…
59996    [four, stars, working, buy, handled, return, buy]
59997    [smell, awesome, leaves, hair, smooth, smell, …
59998    [pretty, hair, really, loved, hair, would, rec…
59999    [great, natural, product, past, experiences, n…
Name: stopwords_removed, Length: 60000, dtype: object
```

2.2 perform lemmatization

[24]:
```
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords, wordnet
```

2.2.1 POS tagging

[25]:
```
reviews_df['pos_tags'] = reviews_df['stopwords_removed'].apply(nltk.tag.pos_tag)
reviews_df['pos_tags']
```

[25]:
```
0        [(rancid, NN), (smell, NN), (threw, VBD), (awa…
1        [(flavor, NN), (gross, JJ), (nasty, JJ), (flav…
2        [(fan, JJ), (product, NN), (fan, NN), (product…
3        [(worth, JJ), (investment, NN), (using, VBG), …
4        [(wow, JJ), (mean, JJ), (rude, NN), (wow, NN),…
                               …
59995    [(vi, NN), (tae, NN), (shea, VBP), (butter, NN…
```

```
59996    [(four, CD), (stars, NNS), (working, VBG), (bu…
59997    [(smell, NN), (awesome, JJ), (leaves, VBZ), (h…
59998    [(pretty, RB), (hair, NN), (really, RB), (love…
59999    [(great, JJ), (natural, JJ), (product, NN), (p…
Name: pos_tags, Length: 60000, dtype: object
```

[26]:
```python
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
reviews_df['wordnet_pos'] = reviews_df['pos_tags'].apply(lambda x: [(word,␣
 ↪get_wordnet_pos(pos_tag)) for (word, pos_tag) in x])
```

2.2.2 mapping word+pos to their corresponding base form using WordNetLemmatizer

[27]:
```python
wnl = WordNetLemmatizer()
reviews_df['lemmatized'] = reviews_df['wordnet_pos'].apply(lambda x: [wnl.
 ↪lemmatize(word, tag) for word, tag in x])
reviews_df["reviews_processed"]=reviews_df['lemmatized'].apply(lambda x: ' '.
 ↪join(x))
reviews_df['reviews_processed']
```

[27]:
```
0        rancid smell throw away smell like go rancid h…
1        flavor gross nasty flavor product amaze though…
2        fan product fan product thick dry properly fel…
3        worth investment use month see benefit adverti…
4        wow mean rude wow peice censor jesus christ ma…
                                   …
59995    vi tae shea butter soap second purchase soap t…
59996                 four star work buy handle return buy
59997    smell awesome leave hair smooth smell awesome …
59998    pretty hair really love hair would recommend a…
59999    great natural product past experience never lu…
Name: reviews_processed, Length: 60000, dtype: object
```

[28]:
```python
reviews_df['reviews_processed'].str.len().mean()
```

[28]:
```
165.82413333333332
```

2.3 Avg length of review before and after Pre - Processing

[29]:
```
len_after_preprocess = reviews_df['reviews_processed'].str.len().mean()

print(int(len_before_preprocess),",",int(len_after_preprocess))
```

```
281 , 165
```

3 TF-IDF Feature Extraction

[30]:
```
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
from sklearn.model_selection import train_test_split
x_train␣
↪,x_test,y_train,y_test=train_test_split(reviews_df["reviews_processed"],reviews_df["star_ra
                                                                test_size=0.2 ,
                                                               ␣
↪random_state=15,stratify=reviews_df['star_rating'])

tfIdfVectorizer=TfidfVectorizer(use_idf=True,ngram_range=(1,␣
↪3),sublinear_tf=True)
train_tfIdf = tfIdfVectorizer.fit_transform(x_train)
test_tfIdf = tfIdfVectorizer.transform(x_test)
```

3.1 Modelling Variant 1 - with stopword removal and lemmatization

4 Perceptron

[31]:
```
from sklearn.metrics import confusion_matrix,classification_report

def printMetrics(y_test,pred):
    report = classification_report(y_test, pred, output_dict=True)
    print("{:0.4f}".format(report['1']['precision']),",","{:0.4f}".
↪format(report['1']['recall']),",","{:0.4f}".format(report['1']['f1-score']))
    print("{:0.4f}".format(report['2']['precision']),",","{:0.4f}".
↪format(report['2']['recall']),",","{:0.4f}".format(report['2']['f1-score']))
    print("{:0.4f}".format(report['3']['precision']),",","{:0.4f}".
↪format(report['3']['recall']),",","{:0.4f}".format(report['3']['f1-score']))
    print("{:0.4f}".format(report['macro avg']['precision']),",","{:0.4f}".
↪format(report['macro avg']['recall']),",","{:0.4f}".format(report['macro␣
↪avg']['f1-score']))
```

[32]:
```
from sklearn.linear_model import Perceptron
model = Perceptron()
model.fit(train_tfIdf,y_train)
pred=model.predict(test_tfIdf)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           1       0.78      0.74      0.76      4000
           2       0.69      0.68      0.69      4000
           3       0.79      0.85      0.82      4000

    accuracy                           0.76     12000
   macro avg       0.76      0.76      0.76     12000
weighted avg       0.76      0.76      0.76     12000
```

[33]:
```
printMetrics(y_test,pred)
```

```
0.7841 , 0.7365 , 0.7596
0.6928 , 0.6815 , 0.6871
0.7925 , 0.8535 , 0.8219
0.7565 , 0.7572 , 0.7562
```

5 SVM

[34]:
```
from sklearn.svm import LinearSVC
model = LinearSVC(random_state=0, tol=1e-4)
model.fit(train_tfIdf,y_train)
pred=model.predict(test_tfIdf)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           1       0.81      0.75      0.78      4000
           2       0.71      0.71      0.71      4000
           3       0.80      0.86      0.83      4000

    accuracy                           0.77     12000
   macro avg       0.77      0.77      0.77     12000
weighted avg       0.77      0.77      0.77     12000
```

[35]:
```
printMetrics(y_test,pred)
```

```
0.8096 , 0.7482 , 0.7777
0.7075 , 0.7053 , 0.7064
0.7998 , 0.8630 , 0.8302
0.7723 , 0.7722 , 0.7714
```

6 Logistic Regression

[36]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(train_tfIdf,y_train)
pred=model.predict(test_tfIdf)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           1       0.81      0.74      0.77      4000
           2       0.70      0.71      0.70      4000
           3       0.80      0.85      0.82      4000

    accuracy                           0.77     12000
   macro avg       0.77      0.77      0.77     12000
weighted avg       0.77      0.77      0.77     12000


e:\USC\spring23\nlp\submit\venv\Lib\site-
packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[37]:
```
printMetrics(y_test,pred)
```

```
0.8064 , 0.7392 , 0.7714
0.6952 , 0.7110 , 0.7030
0.7980 , 0.8462 , 0.8214
0.7665 , 0.7655 , 0.7653
```

7 Naive Bayes

[38]:
```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(train_tfIdf,y_train)
pred=model.predict(test_tfIdf)
print(classification_report(y_test,pred))
```

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

```
           1       0.78      0.76      0.77      4000
           2       0.68      0.71      0.70      4000
           3       0.84      0.82      0.83      4000

    accuracy                           0.76     12000
   macro avg       0.77      0.76      0.77     12000
weighted avg       0.77      0.76      0.77     12000
```

[39]:
```
printMetrics(y_test,pred)
```

```
0.7807 , 0.7572 , 0.7688
0.6826 , 0.7145 , 0.6982
0.8365 , 0.8225 , 0.8294
0.7666 , 0.7647 , 0.7655
```

8 Model Variant 2 - Without removing stop words and with lemmatization

[40]:
```
# applying lemmatization for tokens without stopwords
reviews_df['pos_tags'] = reviews_df['tokenized'].apply(nltk.tag.pos_tag)
wnl = WordNetLemmatizer()
reviews_df['lemmatized'] = reviews_df['wordnet_pos'].apply(lambda x: [wnl.
 ↪lemmatize(word, tag) for word, tag in x])
reviews_df["reviews_processed_without_stop"]=reviews_df['lemmatized'].
 ↪apply(lambda x: ' '.join(x))
reviews_df['reviews_processed_without_stop'].str.len().mean()
```

[40]: 165.82413333333332

[41]:
```
x_train␣
 ↪,x_test,y_train,y_test=train_test_split(reviews_df["reviews_processed_without_stop"],review
                                                                test_size=0.2 ,
                                                               ␣
 ↪random_state=15,stratify=reviews_df['star_rating'])
tfIdfVectorizer=TfidfVectorizer(use_idf=True,ngram_range=(1,␣
 ↪3),sublinear_tf=True)
train_tfIdf = tfIdfVectorizer.fit_transform(x_train)
test_tfIdf = tfIdfVectorizer.transform(x_test)
```

Perceptron

[42]:
```
model = Perceptron()
model.fit(train_tfIdf,y_train)
pred=model.predict(test_tfIdf)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           1       0.78      0.74      0.76      4000
           2       0.69      0.68      0.69      4000
           3       0.79      0.85      0.82      4000

    accuracy                           0.76     12000
   macro avg       0.76      0.76      0.76     12000
weighted avg       0.76      0.76      0.76     12000
```

9 SVM

[43]:
```
model = LinearSVC(random_state=0, tol=1e-4)
model.fit(train_tfIdf,y_train)
pred=model.predict(test_tfIdf)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           1       0.81      0.75      0.78      4000
           2       0.71      0.71      0.71      4000
           3       0.80      0.86      0.83      4000

    accuracy                           0.77     12000
   macro avg       0.77      0.77      0.77     12000
weighted avg       0.77      0.77      0.77     12000
```

10 Logistic Regression

[44]:
```
model=LogisticRegression()
model.fit(train_tfIdf,y_train)
pred=model.predict(test_tfIdf)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           1       0.81      0.74      0.77      4000
           2       0.70      0.71      0.70      4000
           3       0.80      0.85      0.82      4000

    accuracy                           0.77     12000
   macro avg       0.77      0.77      0.77     12000
weighted avg       0.77      0.77      0.77     12000
```

```
e:\USC\spring23\nlp\submit\venv\Lib\site-
packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

11 Naive Bayes

[45]:
```
model = MultinomialNB()
model.fit(train_tfIdf,y_train)
pred=model.predict(test_tfIdf)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           1       0.78      0.76      0.77      4000
           2       0.68      0.71      0.70      4000
           3       0.84      0.82      0.83      4000

    accuracy                           0.76     12000
   macro avg       0.77      0.76      0.77     12000
weighted avg       0.77      0.76      0.77     12000
```

12 Mode lvariant 3 - without stopword removal and without lemmatization

[46]:
```
x_train␣
 ↪,x_test,y_train,y_test=train_test_split(reviews_df["review_body"],reviews_df["star_rating"]
                                                  test_size=0.2 ,
                                                 ␣
 ↪random_state=15,stratify=reviews_df['star_rating'])
tfIdfVectorizer=TfidfVectorizer(use_idf=True,ngram_range=(1,␣
 ↪3),sublinear_tf=True)
train_tfIdf = tfIdfVectorizer.fit_transform(x_train)
test_tfIdf = tfIdfVectorizer.transform(x_test)
```

13 Perceptron

[47]:
```
model = Perceptron()
model.fit(train_tfIdf,y_train)
pred=model.predict(test_tfIdf)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           1       0.81      0.79      0.80      4000
           2       0.73      0.74      0.73      4000
           3       0.86      0.88      0.87      4000

    accuracy                           0.80     12000
   macro avg       0.80      0.80      0.80     12000
weighted avg       0.80      0.80      0.80     12000
```

14 SVM

[48]:
```
model = LinearSVC(random_state=0, tol=1e-4)
model.fit(train_tfIdf,y_train)
pred=model.predict(test_tfIdf)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           1       0.83      0.80      0.82      4000
           2       0.75      0.76      0.75      4000
           3       0.88      0.88      0.88      4000

    accuracy                           0.82     12000
   macro avg       0.82      0.82      0.82     12000
weighted avg       0.82      0.82      0.82     12000
```

15 Logistic Regression

[49]:
```
model=LogisticRegression(max_iter=100)
model.fit(train_tfIdf,y_train)
pred=model.predict(test_tfIdf)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           1       0.82      0.80      0.81      4000
```

```
           2       0.73      0.77      0.75      4000
           3       0.88      0.86      0.87      4000

    accuracy                           0.81     12000
   macro avg       0.81      0.81      0.81     12000
weighted avg       0.81      0.81      0.81     12000
```

```
e:\USC\spring23\nlp\submit\venv\Lib\site-
packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

16 Naive Bayes

[50]:

```python
model = MultinomialNB()
model.fit(train_tfIdf,y_train)
pred=model.predict(test_tfIdf)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           1       0.81      0.80      0.81      4000
           2       0.69      0.80      0.74      4000
           3       0.93      0.79      0.85      4000

    accuracy                           0.80     12000
   macro avg       0.81      0.80      0.80     12000
weighted avg       0.81      0.80      0.80     12000
```