CSCI 544- Homework 1 Report
Asmita Chotani

1. **Dataset Preparation**

For the Dataset preparation, the dataset was loaded as 'df2'. Since we are only considering the rating and reviews for the assignment, a separate dataframe "df3" was created by extracting the "star_rating" and "review_body" columns only.

For easier accessibility, a copy of the new dataframe(df3) was created as "df". This dataframe was then used for the rest of the process.

Further a "class" column was added to the dataframe to assign the class label based on the rating. Considering the class values, separate dataframes were created for each class.

Since we have to work with 20000 reviews for each class, 20000 entries were considered randomly from each of the 3 dataframes, using the "sample" function and combined to create the working dataframe to be used furtheron.

2. **Data Cleaning**

For cleaning the process, the below tasks were performed.

- Since random 20000 entries were added from the three dataframes, in order to prevent repetition of indexes, the indexes were reset.
- Rows with missing values were determined.
- Since the missing values were only in the "review" column, the null value was replaced by an empty string.
- The reviews were made into lowerecase using the lower() function available for strings.
- The HTML tags and URLs were removed from the reviews
- The punctuations were removed by comparing characters with the ones part of string.punctuation and joinig the ones that are not present to create a new string.
- The non-alphabetical characters were removed from the reviews by tokenizeing the words, and then removing characters that are not between A-Z or a-z.
- The review sentences was split into words and Word Contractions was performed on those that needed it by using the "contraction" library. The series of words were then again joined to create a sentence.

```
The length of the reviews initially-   267.9975333333333
The length of the reviews after cleaning-   256.52843333333334
```

CSCI 544- Homework 1 Report
Asmita Chotani

3. Preprocessing

- NTLK package was used to remove stop words and lemmatize.

```
The length of the reviews post cleaning-   256.52843333333334
The length of the reviews after pre-processing-   158.23988333333332
```

4. Feature Extraction

Performed TF-IDF using sklearn to extract the features and labels that are to be used in the models in the future steps.

The distribution of classes obtained was –

```
Class 3    16152
Class 1    15935
Class 2    15913
```

The distribution was almost balanced. Hence Smote/upsampling or downsampling was not applied.

5. Perceptron

Grid Search was used to determine the most efficient “alpha” value and tolerance value “tol” along with using the L2 norm as the penalty. The most efficient alpha was found out to be**- 'alpha': 1e-05, 'tol': 0.001**

The Result of the model was-

```
1 :  0.5988321799307958 , 0.6811808118081181 , 0.637357578547589
2 :  0.5351048951048951 , 0.46806948862246145 , 0.4993474288697468
3 :  0.691397000789266 , 0.682952182952183 , 0.6871486468819453
macro avg :  0.6084446919416523 , 0.6107341611275875 , 0.6079512180997604
weighted avg :  0.6068101813957906 , 0.6091666666666666 , 0.6063199576490275
```

6. SVM

Grid Search was used to determine the most efficient “C” value and tolerance value ‘tol’ along with using the L2 norm as the penalty. The most efficient “C” was found out to be- **'C': 0.35, 'tol': 0.001.**

The Result of the model was-

```
1 :  0.6864902833060174 , 0.7212792127921279 , 0.7034548944337812
2 :  0.6138920134983127 , 0.5341326156104722 , 0.5712416590344105
3 :  0.7277737838485502 , 0.7892411642411642 , 0.7572621867597555
macro avg :  0.6760520268842933 , 0.6815509975479214 , 0.6773195800759825
weighted avg :  0.6750027650879821 , 0.6793333333333333 , 0.675679475083208
```

7. Logistic Regression
   Grid Search was used to determine the most efficient "C" value, "solver" and tolerance value i.e "tol" along with using the L2 norm as the penalty. The most efficient values were found to be **'C': 0.4, 'solver': 'saga', 'tol': 0.01**

   The Result of the model was-

```
1 :  0.6902781079153791 , 0.7143911439114391 , 0.7021276595744681
2 :  0.6078174186778594 , 0.5669195008563739 , 0.58665653880238
3 :  0.7430293896006028 , 0.7687110187110187 , 0.7556520628432749
macro avg :  0.6803749720646138 , 0.6833405544929438 , 0.681478753740041
weighted avg :  0.6791089491662956 , 0.6815833333333333 , 0.679963612339705
```

8. **Naïve Bayes**
   Grid Search was used to determine the most efficient "alpha" value. The most efficient alpha was found out to be **- 6**

   The "class_prior" parameter of MultinomialNB was experimented with, but it did not have much of an impact on the performance of the model, hence it was not considered.

   The Result of the model was-

```
1 :  0.6996966632962589 , 0.6809348093480935 , 0.6901882558284503
2 :  0.5944359367023991 , 0.5698556398336188 , 0.5818863210493441
3 :  0.722249151720795 , 0.7744282744282744 , 0.747429144720341
macro avg :  0.6721272505731509 , 0.6750729078699956 , 0.6731679071993785
weighted avg :  0.671078445451968 , 0.6730833333333334 , 0.6716576669129326
```

CSCI 544 HOMEWORK 1

NAME: Asmita Chotani

USC ID: 3961468036

In [1]:

```
import pandas as pd
import numpy as np
import nltk
nltk.download('wordnet')
nltk.download('punkt')  # for word tokenizing
nltk.download('stopwords') # for determining stop words taht have to be removed
nltk.download('omw-1.4') # for lemmatizing

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Perceptron
from sklearn.metrics import classification_report
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC

import re
from bs4 import BeautifulSoup
import warnings
warnings.filterwarnings('ignore')
import string
import contractions
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/asmitachotani/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/asmitachotani/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/asmitachotani/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     /Users/asmitachotani/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

In [2]:

```
!pip install contractions
```

```
Requirement already satisfied: contractions in /Users/asmitachotani/opt/minico
nda3/lib/python3.9/site-packages (0.1.73)
Requirement already satisfied: textsearch>=0.0.21 in /Users/asmitachotani/opt/
miniconda3/lib/python3.9/site-packages (from contractions) (0.0.24)
Requirement already satisfied: anyascii in /Users/asmitachotani/opt/miniconda
3/lib/python3.9/site-packages (from textsearch>=0.0.21->contractions) (0.3.1)
Requirement already satisfied: pyahocorasick in /Users/asmitachotani/opt/minic
onda3/lib/python3.9/site-packages (from textsearch>=0.0.21->contractions) (2.
0.0)
```

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

# Dataset: https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Be
```

```
Requirement already satisfied: bs4 in /Users/asmitachotani/opt/miniconda3/lib/
python3.9/site-packages (0.0.1)
Requirement already satisfied: beautifulsoup4 in /Users/asmitachotani/opt/mini
conda3/lib/python3.9/site-packages (from bs4) (4.11.1)
Requirement already satisfied: soupsieve>1.2 in /Users/asmitachotani/opt/minic
onda3/lib/python3.9/site-packages (from beautifulsoup4->bs4) (2.3.2.post1)
```

Read Data

In [4]:
```
df2 = pd.read_csv('./amazon_reviews_us_Beauty_v1_00.tsv',
                                sep='\t',
                                error_bad_lines=False
                               )
display(df2)
print(df2.columns)
```

```
b'Skipping line 10093: expected 15 fields, saw 22\nSkipping line 31965: expect
ed 15 fields, saw 22\nSkipping line 49886: expected 15 fields, saw 22\nSkippin
g line 49905: expected 15 fields, saw 22\n'
b'Skipping line 67579: expected 15 fields, saw 22\nSkipping line 75367: expect
ed 15 fields, saw 22\nSkipping line 92462: expected 15 fields, saw 22\nSkippin
g line 105041: expected 15 fields, saw 22\nSkipping line 109697: expected 15 f
ields, saw 22\nSkipping line 121931: expected 15 fields, saw 22\n'
b'Skipping line 139492: expected 15 fields, saw 22\nSkipping line 158729: expe
cted 15 fields, saw 22\nSkipping line 165784: expected 15 fields, saw 22\nSkip
ping line 176996: expected 15 fields, saw 22\nSkipping line 182928: expected 1
5 fields, saw 22\nSkipping line 195841: expected 15 fields, saw 22\n'
b'Skipping line 196938: expected 15 fields, saw 22\nSkipping line 202535: expe
cted 15 fields, saw 22\nSkipping line 261147: expected 15 fields, saw 22\n'
b'Skipping line 265777: expected 15 fields, saw 22\nSkipping line 277693: expe
cted 15 fields, saw 22\nSkipping line 280010: expected 15 fields, saw 22\nSkip
ping line 296315: expected 15 fields, saw 22\nSkipping line 299043: expected 1
5 fields, saw 22\n'
b'Skipping line 334564: expected 15 fields, saw 22\nSkipping line 337801: expe
cted 15 fields, saw 22\nSkipping line 341391: expected 15 fields, saw 22\nSkip
ping line 354940: expected 15 fields, saw 22\nSkipping line 366330: expected 1
5 fields, saw 22\nSkipping line 367649: expected 15 fields, saw 22\n'
b'Skipping line 399174: expected 15 fields, saw 22\nSkipping line 414439: expe
cted 15 fields, saw 22\n'
b'Skipping line 473579: expected 15 fields, saw 22\nSkipping line 483540: expe
cted 15 fields, saw 22\nSkipping line 499744: expected 15 fields, saw 22\nSkip
ping line 505775: expected 15 fields, saw 22\n'
b'Skipping line 547693: expected 15 fields, saw 22\nSkipping line 561254: expe
cted 15 fields, saw 22\n'
b'Skipping line 609329: expected 15 fields, saw 22\nSkipping line 642814: expe
cted 15 fields, saw 22\nSkipping line 643189: expected 15 fields, saw 22\nSkip
ping line 647075: expected 15 fields, saw 22\nSkipping line 647457: expected 1
5 fields, saw 22\n'
b'Skipping line 660868: expected 15 fields, saw 22\nSkipping line 668514: expe
cted 15 fields, saw 22\nSkipping line 673314: expected 15 fields, saw 22\nSkip
ping line 700416: expected 15 fields, saw 22\n'
b'Skipping line 723492: expected 15 fields, saw 22\nSkipping line 725052: expe
cted 15 fields, saw 22\nSkipping line 726222: expected 15 fields, saw 22\nSkip
ping line 744078: expected 15 fields, saw 22\nSkipping line 753129: expected 1
5 fields, saw 22\nSkipping line 758347: expected 15 fields, saw 22\nSkipping l
ine 759076: expected 15 fields, saw 22\nSkipping line 759139: expected 15 fiel
ds, saw 22\nSkipping line 768106: expected 15 fields, saw 22\nSkipping line 77
7835: expected 15 fields, saw 22\nSkipping line 779763: expected 15 fields, sa
w 22\nSkipping line 781395: expected 15 fields, saw 22\n'
b'Skipping line 787023: expected 15 fields, saw 22\nSkipping line 811679: expe
cted 15 fields, saw 22\nSkipping line 811739: expected 15 fields, saw 22\n'
b'Skipping line 855784: expected 15 fields, saw 22\nSkipping line 878325: expe
cted 15 fields, saw 22\nSkipping line 886822: expected 15 fields, saw 22\nSkip
ping line 890742: expected 15 fields, saw 22\n'
b'Skipping line 919607: expected 15 fields, saw 22\nSkipping line 920655: expe
cted 15 fields, saw 22\nSkipping line 923107: expected 15 fields, saw 22\nSkip
ping line 930890: expected 15 fields, saw 22\nSkipping line 932841: expected 1
5 fields, saw 22\n'
b'Skipping line 1003214: expected 15 fields, saw 22\nSkipping line 1007588: ex
pected 15 fields, saw 22\nSkipping line 1018374: expected 15 fields, saw 22\nS
kipping line 1022909: expected 15 fields, saw 22\nSkipping line 1030983: expec
ted 15 fields, saw 22\nSkipping line 1048441: expected 15 fields, saw 22\n'
b'Skipping line 1056292: expected 15 fields, saw 22\nSkipping line 1056518: ex
pected 15 fields, saw 22\nSkipping line 1073064: expected 15 fields, saw 22\nS
kipping line 1088887: expected 15 fields, saw 22\nSkipping line 1103881: expec
ted 15 fields, saw 22\nSkipping line 1111021: expected 15 fields, saw 22\nSkip
```

```
ping line 1111314: expected 15 fields, saw 22\n'
b'Skipping line 1119421: expected 15 fields, saw 22\nSkipping line 1119549: ex
pected 15 fields, saw 22\nSkipping line 1130122: expected 15 fields, saw 22\nS
kipping line 1132767: expected 15 fields, saw 22\nSkipping line 1143315: expec
ted 15 fields, saw 22\nSkipping line 1151947: expected 15 fields, saw 22\nSkip
ping line 1154207: expected 15 fields, saw 22\nSkipping line 1154616: expected
15 fields, saw 22\nSkipping line 1155875: expected 15 fields, saw 22\nSkipping
line 1164714: expected 15 fields, saw 22\nSkipping line 1164959: expected 15 f
ields, saw 22\nSkipping line 1169410: expected 15 fields, saw 22\n'
b'Skipping line 1184604: expected 15 fields, saw 22\nSkipping line 1203964: ex
pected 15 fields, saw 22\nSkipping line 1211287: expected 15 fields, saw 22\nS
kipping line 1217834: expected 15 fields, saw 22\nSkipping line 1235346: expec
ted 15 fields, saw 22\nSkipping line 1238073: expected 15 fields, saw 22\nSkip
ping line 1238439: expected 15 fields, saw 22\n'
b'Skipping line 1246837: expected 15 fields, saw 22\nSkipping line 1263235: ex
pected 15 fields, saw 22\nSkipping line 1265620: expected 15 fields, saw 22\n'
b'Skipping line 1312400: expected 15 fields, saw 22\nSkipping line 1314122: ex
pected 15 fields, saw 22\nSkipping line 1319707: expected 15 fields, saw 22\nS
kipping line 1337672: expected 15 fields, saw 22\nSkipping line 1343961: expec
ted 15 fields, saw 22\nSkipping line 1346372: expected 15 fields, saw 22\nSkip
ping line 1358447: expected 15 fields, saw 22\nSkipping line 1370844: expected
15 fields, saw 22\n'
b'Skipping line 1406108: expected 15 fields, saw 22\nSkipping line 1435069: ex
pected 15 fields, saw 22\nSkipping line 1439866: expected 15 fields, saw 22\n'
b'Skipping line 1442123: expected 15 fields, saw 22\nSkipping line 1463237: ex
pected 15 fields, saw 22\nSkipping line 1469027: expected 15 fields, saw 22\nS
kipping line 1469598: expected 15 fields, saw 22\nSkipping line 1482636: expec
ted 15 fields, saw 22\nSkipping line 1484745: expected 15 fields, saw 22\nSkip
ping line 1499831: expected 15 fields, saw 22\n'
b'Skipping line 1508882: expected 15 fields, saw 22\nSkipping line 1514887: ex
pected 15 fields, saw 22\nSkipping line 1527564: expected 15 fields, saw 22\nS
kipping line 1569519: expected 15 fields, saw 22\n'
b'Skipping line 1583105: expected 15 fields, saw 22\nSkipping line 1604380: ex
pected 15 fields, saw 22\nSkipping line 1607380: expected 15 fields, saw 22\nS
kipping line 1631601: expected 15 fields, saw 22\n'
b'Skipping line 1642095: expected 15 fields, saw 22\nSkipping line 1646714: ex
pected 15 fields, saw 22\nSkipping line 1655248: expected 15 fields, saw 22\nS
kipping line 1657807: expected 15 fields, saw 22\nSkipping line 1667534: expec
ted 15 fields, saw 22\nSkipping line 1668489: expected 15 fields, saw 22\nSkip
ping line 1691733: expected 15 fields, saw 22\nSkipping line 1701102: expected
15 fields, saw 22\nSkipping line 1701499: expected 15 fields, saw 22\n'
b'Skipping line 1704450: expected 15 fields, saw 22\nSkipping line 1706154: ex
pected 15 fields, saw 22\nSkipping line 1712789: expected 15 fields, saw 22\n'
b'Skipping line 1773984: expected 15 fields, saw 22\n'
b'Skipping line 1846441: expected 15 fields, saw 22\nSkipping line 1848019: ex
pected 15 fields, saw 22\nSkipping line 1856015: expected 15 fields, saw 22\nS
kipping line 1858248: expected 15 fields, saw 22\nSkipping line 1859629: expec
ted 15 fields, saw 22\nSkipping line 1873117: expected 15 fields, saw 22\nSkip
ping line 1894414: expected 15 fields, saw 22\n'
b'Skipping line 1902421: expected 15 fields, saw 22\nSkipping line 1909201: ex
pected 15 fields, saw 22\nSkipping line 1914394: expected 15 fields, saw 22\nS
kipping line 1936976: expected 15 fields, saw 22\nSkipping line 1940327: expec
ted 15 fields, saw 22\nSkipping line 1945664: expected 15 fields, saw 22\nSkip
ping line 1946171: expected 15 fields, saw 22\nSkipping line 1946284: expected
15 fields, saw 22\nSkipping line 1946835: expected 15 fields, saw 22\nSkipping
line 1952446: expected 15 fields, saw 22\nSkipping line 1953387: expected 15 f
ields, saw 22\n'
b'Skipping line 1979093: expected 15 fields, saw 22\nSkipping line 1982997: ex
pected 15 fields, saw 22\nSkipping line 1992924: expected 15 fields, saw 22\nS
kipping line 1996161: expected 15 fields, saw 22\nSkipping line 2003175: expec
```

```
ted 15 fields, saw 22\nSkipping line 2024153: expected 15 fields, saw 22\nSkip
ping line 2026345: expected 15 fields, saw 22\n'
b'Skipping line 2041159: expected 15 fields, saw 22\nSkipping line 2042954: ex
pected 15 fields, saw 22\nSkipping line 2044244: expected 15 fields, saw 22\nS
kipping line 2047949: expected 15 fields, saw 22\nSkipping line 2051022: expec
ted 15 fields, saw 22\nSkipping line 2052365: expected 15 fields, saw 22\nSkip
ping line 2064460: expected 15 fields, saw 22\nSkipping line 2077010: expected
15 fields, saw 22\nSkipping line 2083893: expected 15 fields, saw 22\n'
b'Skipping line 2097514: expected 15 fields, saw 22\nSkipping line 2100479: ex
pected 15 fields, saw 22\nSkipping line 2103183: expected 15 fields, saw 22\nS
kipping line 2108608: expected 15 fields, saw 22\nSkipping line 2116577: expec
ted 15 fields, saw 22\nSkipping line 2127375: expected 15 fields, saw 22\nSkip
ping line 2128053: expected 15 fields, saw 22\nSkipping line 2135954: expected
15 fields, saw 22\nSkipping line 2137154: expected 15 fields, saw 22\nSkipping
line 2140279: expected 15 fields, saw 22\nSkipping line 2150764: expected 15 f
ields, saw 22\nSkipping line 2151464: expected 15 fields, saw 22\nSkipping lin
e 2151588: expected 15 fields, saw 22\nSkipping line 2157049: expected 15 fiel
ds, saw 22\n'
b'Skipping line 2163762: expected 15 fields, saw 22\nSkipping line 2167939: ex
pected 15 fields, saw 22\nSkipping line 2172050: expected 15 fields, saw 22\nS
kipping line 2177960: expected 15 fields, saw 22\nSkipping line 2202813: expec
ted 15 fields, saw 22\nSkipping line 2207828: expected 15 fields, saw 22\nSkip
ping line 2211189: expected 15 fields, saw 22\nSkipping line 2211589: expected
15 fields, saw 22\nSkipping line 2214034: expected 15 fields, saw 22\nSkipping
line 2214264: expected 15 fields, saw 22\nSkipping line 2214462: expected 15 f
ields, saw 22\nSkipping line 2215027: expected 15 fields, saw 22\nSkipping lin
e 2215639: expected 15 fields, saw 22\nSkipping line 2216007: expected 15 fiel
ds, saw 22\nSkipping line 2217132: expected 15 fields, saw 22\nSkipping line 2
226703: expected 15 fields, saw 22\n'
b'Skipping line 2231683: expected 15 fields, saw 22\nSkipping line 2245222: ex
pected 15 fields, saw 22\nSkipping line 2256136: expected 15 fields, saw 22\nS
kipping line 2269399: expected 15 fields, saw 22\nSkipping line 2283979: expec
ted 15 fields, saw 22\n'
b'Skipping line 2340899: expected 15 fields, saw 22\nSkipping line 2342134: ex
pected 15 fields, saw 22\nSkipping line 2342748: expected 15 fields, saw 22\nS
kipping line 2348402: expected 15 fields, saw 22\nSkipping line 2355164: expec
ted 15 fields, saw 22\nSkipping line 2357020: expected 15 fields, saw 22\n'
b'Skipping line 2366077: expected 15 fields, saw 22\nSkipping line 2366997: ex
pected 15 fields, saw 22\nSkipping line 2367353: expected 15 fields, saw 22\nS
kipping line 2414691: expected 15 fields, saw 22\n'
b'Skipping line 2464571: expected 15 fields, saw 22\nSkipping line 2466302: ex
pected 15 fields, saw 22\nSkipping line 2487679: expected 15 fields, saw 22\nS
kipping line 2487771: expected 15 fields, saw 22\n'
b'Skipping line 2506605: expected 15 fields, saw 22\nSkipping line 2511369: ex
pected 15 fields, saw 22\n'
b'Skipping line 2558281: expected 15 fields, saw 22\nSkipping line 2607202: ex
pected 15 fields, saw 22\n'
b'Skipping line 2625718: expected 15 fields, saw 22\nSkipping line 2640978: ex
pected 15 fields, saw 22\nSkipping line 2650635: expected 15 fields, saw 22\nS
kipping line 2670724: expected 15 fields, saw 22\n'
b'Skipping line 2690954: expected 15 fields, saw 22\nSkipping line 2713810: ex
pected 15 fields, saw 22\nSkipping line 2715292: expected 15 fields, saw 22\nS
kipping line 2724453: expected 15 fields, saw 22\nSkipping line 2724458: expec
ted 15 fields, saw 22\nSkipping line 2735678: expected 15 fields, saw 22\nSkip
ping line 2740358: expected 15 fields, saw 22\nSkipping line 2751188: expected
15 fields, saw 22\n'
b'Skipping line 2763890: expected 15 fields, saw 22\nSkipping line 2766982: ex
pected 15 fields, saw 22\nSkipping line 2813747: expected 15 fields, saw 22\n'
b'Skipping line 2819306: expected 15 fields, saw 22\nSkipping line 2883075: ex
pected 15 fields, saw 22\n'
```

```
b'Skipping line 2975635: expected 15 fields, saw 22\n'
b'Skipping line 3391761: expected 15 fields, saw 22\n'
b'Skipping line 3474241: expected 15 fields, saw 22\n'
b'Skipping line 3690054: expected 15 fields, saw 22\nSkipping line 3720113: ex
pected 15 fields, saw 22\n'
b'Skipping line 3763182: expected 15 fields, saw 22\n'
b'Skipping line 4929700: expected 15 fields, saw 22\n'
```

	marketplace	customer_id	review_id	product_id	product_parent	prod
0	US	1797882	R3I2DHQBR577SS	B001ANOOOE	2102612	The N Mo Sun
1	US	18381298	R1QNE9NQFJC2Y4	B0016J22EQ	106393691	Alba Sunles Lotion,
2	US	19242472	R3LIDG2Q4LJBAO	B00HU6UQAG	375449471	Elyse Skir E
3	US	19551372	R3KSZHPAEVPEAL	B002HWS7RM	255651889	Di Color, I Co I
4	US	14802407	RAI2OIG50KZ43	B00SM99KWU	116158747	Biore Ric SPF50+
...	...	...	...	...	...	
5094302	US	50113639	RZ7RZ02MTP4SL	B000050B70	185454094	NE Cordl and Ear
5094303	US	52940456	R2IRC0IZ8YCE5T	B000050FF2	678848064	I En S Ala
5094304	US	47587881	R1U4ZSXOD228CZ	B000050B6U	862195513	Cona Heat Cu
5094305	US	53047750	R3SFJLZE09URWM	B000050FDE	195242894	Pro Care 10 To
5094306	US	51193940	R1MEWK4I7YS5XK	B000050AUD	190668305	Soni (47 To

5094307 rows × 15 columns

```
Index(['marketplace', 'customer_id', 'review_id', 'product_id',
       'product_parent', 'product_title', 'product_category', 'star_rating',
       'helpful_votes', 'total_votes', 'vine', 'verified_purchase',
       'review_headline', 'review_body', 'review_date'],
      dtype='object')
```

In [5]:
```
df3 = pd.read_csv('./amazon_reviews_us_Beauty_v1_00.tsv',
                                 sep='\t',
                                 error_bad_lines=False,
                                 usecols=["star_rating", "review_body"]
                                )
display(df3)
```

	star_rating	review_body
0	5	Love this, excellent sun block!!
1	5	The great thing about this cream is that it do...
2	5	Great Product, I'm 65 years old and this is al...
3	5	I use them as shower caps & conditioning caps....
4	5	This is my go-to daily sunblock. It leaves no ...
...	...	...
5094558	5	After watching my Dad struggle with his scisso...
5094559	3	Like most sound machines, the sounds choices a...
5094560	5	I bought this product because it indicated 30 ...
5094561	5	We have used Oral-B products for 15 years; thi...
5094562	5	I love this toothbrush. It's easy to use, and ...

5094563 rows × 2 columns

In [6]:
```
# understanding the different rating values that are possible
df3['star_rating'].unique()
```

Out[6]:
```
array(['5', '4', '1', '3', '2', '2015-08-28', '2015-08-16', '2015-08-14',
       5, 4, 3, 1, 2, '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'], dtype=object)
```

In [7]:
```
df3['star_rating'].value_counts()
```

Out[7]:
```
5             2594188
5              646886
4              604894
1              372548
3              324129
2              215756
4              133678
1               82544
3               72663
2               47240
2015-04-09          3
2015-04-03          2
2015-04-02          2
2014-12-03          1
2014-12-30          1
2015-02-10          1
2015-02-28          1
2015-03-18          1
2015-03-30          1
2015-03-31          1
2015-04-01          1
2015-07-22          1
2015-04-08          1
2015-04-14          1
2015-06-02          1
2015-06-14          1
2015-07-23          1
2015-07-26          1
2015-07-27          1
2015-08-14          1
2015-08-16          1
2015-08-28          1
2014-10-09          1
Name: star_rating, dtype: int64
```

In [8]:
```
# creating a copy of the dataframe to work with
df=df3.copy()
```

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

In [9]:
```
# 3 classes are formed for the 5 kinds of ratings possible.
def categorise(row):
    if row['star_rating'] ==  1 or row['star_rating']== '1' or row['star_rating
        return 1
    elif row['star_rating'] ==  2 or row['star_rating']== '2'or row['star_ratir
        return 1
    elif row['star_rating'] == 3 or row['star_rating']== '3'or row['star_rating
        return 2
    elif row['star_rating'] ==  4 or row['star_rating']== '4'or row['star_ratir
        return 3
    elif row['star_rating'] ==  5 or row['star_rating']== '5'or row['star_ratir
        return 3
    else:
        return 0   # the entries with invalid values in the rating column
```

In [10]:
```
df['star_rating'].unique()
```

Out[10]:
```
array(['5', '4', '1', '3', '2', '2015-08-28', '2015-08-16', '2015-08-14',
       5, 4, 3, 1, 2, '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'], dtype=object)
```

In [11]:
```
df['class'] = df.apply(lambda row: categorise(row), axis=1)
display(df)
```

	star_rating	review_body	class
0	5	Love this, excellent sun block!!	3
1	5	The great thing about this cream is that it do...	3
2	5	Great Product, I'm 65 years old and this is al...	3
3	5	I use them as shower caps & conditioning caps....	3
4	5	This is my go-to daily sunblock. It leaves no ...	3
...	...	...	...
5094558	5	After watching my Dad struggle with his scisso...	3
5094559	3	Like most sound machines, the sounds choices a...	2
5094560	5	I bought this product because it indicated 30 ...	3
5094561	5	We have used Oral-B products for 15 years; thi...	3
5094562	5	I love this toothbrush. It's easy to use, and ...	3

5094563 rows × 3 columns

In [12]:
```
# understanding the distribution of the classes
df['class'].value_counts()
```

Out[12]:
```
3    3979646
1     718088
2     396792
0         37
Name: class, dtype: int64
```

In [13]:
```
df['class'].unique()
```

Out[13]:
```
array([3, 1, 2, 0])
```

In [14]:
```
# Creating separate dataframes for separate classes
S1_dfa = df.loc[df['class'] == 1]
S2_dfa = df.loc[df['class'] == 2]
S3_dfa = df.loc[df['class'] == 3]

# COnsidering only 20000 data entries for each class
S1_df=S1_dfa.sample(n=20000)
S2_df=S2_dfa.sample(n=20000)
S3_df=S3_dfa.sample(n=20000)
```

In [15]:
```
# Concatenating 20000 reviews for each class into one dataframe that we will wo
review_df = pd.concat([S1_df, S2_df, S3_df])
```

```
display(review_df)
```

	star_rating	review_body	class
573704	1	Flimsy... Not what I expected. I returned it. ...	1
2930618	1	This product is overpriced for what it is. Ma...	1
3541803	1	The quality of these brushes are terrible. It ...	1
2949360	2	I love nude and this is not what I call nude. ...	1
1587697	1	I've used this for 35 years. Massiv...	1
...	...	...	...
4720721	4	I love my Konad kit! The black is a must have ...	3
2608653	4	The palette is durable. I fit around around 2...	3
1433774	5.0	We have used this conditioner on our hair for ...	3
1673223	5	It's the best alternative to aluminum I've eve...	3
2099528	5.0	Easy on and easy off. No need to harm my hair...	3

60000 rows × 3 columns

Data Cleaning

Reseting Index

In [16]:
```
# Since we have randomly chosen 20000 entries from each class, it is necessary
# repitition of entries.
review_df = review_df.reset_index(drop=True)
display(review_df)
```

	star_rating	review_body	class
0	1	Flimsy... Not what I expected. I returned it. ...	1
1	1	This product is overpriced for what it is. Ma...	1
2	1	The quality of these brushes are terrible. It ...	1
3	2	I love nude and this is not what I call nude. ...	1
4	1	I've used this for 35 years. Massiv...	1
...	...	...	...
59995	4	I love my Konad kit! The black is a must have ...	3
59996	4	The palette is durable. I fit around around 2...	3
59997	5.0	We have used this conditioner on our hair for ...	3
59998	5	It's the best alternative to aluminum I've eve...	3
59999	5.0	Easy on and easy off. No need to harm my hair...	3

60000 rows × 3 columns

Dealing with Null Values

In [17]:
```
# Checking for null values
review_df.isnull().values.any()
```

Out[17]:
```
True
```

In [18]:
```
# Checking number of null values in the two columns
review_df.isnull().sum()
```

Out[18]:
```
star_rating     0
review_body     4
class           0
dtype: int64
```

In [19]:
```
# Filling the null values with an empty string as only empty value is in the re
review_df = review_df.fillna('')
```

Creating New DataFrame to store the length of the reviews after different steps.

In [20]:
```
# Creating a separate dataframe to store the length of the reviews after every
# verify that the task was done successfully
display_df = pd.DataFrame()
display_df['before_cleaning'] = review_df['review_body'].str.len()
display(display_df)
```

	before_cleaning
0	65
1	145
2	499
3	185
4	96
...	...
59995	338
59996	147
59997	201
59998	91
59999	129

60000 rows × 1 columns

In [21]:
```
display(review_df)
```

	star_rating	review_body	class
0	1	Flimsy... Not what I expected. I returned it. ...	1
1	1	This product is overpriced for what it is. Ma...	1
2	1	The quality of these brushes are terrible. It ...	1
3	2	I love nude and this is not what I call nude. ...	1
4	1	I've used this for 35 years. Massiv...	1
...	...	...	...
59995	4	I love my Konad kit! The black is a must have ...	3
59996	4	The palette is durable. I fit around around 2...	3
59997	5.0	We have used this conditioner on our hair for ...	3
59998	5	It's the best alternative to aluminum I've eve...	3
59999	5.0	Easy on and easy off. No need to harm my hair...	3

60000 rows × 3 columns

Converting into Lower Case

In [22]:
```
#Converting the reviews into Lower Case
review_df['review_body'] = review_df['review_body'].str.lower()
display(review_df)
```

	star_rating	review_body	class
0	1	flimsy... not what i expected. i returned it. ...	1
1	1	this product is overpriced for what it is. ma...	1
2	1	the quality of these brushes are terrible. it ...	1
3	2	i love nude and this is not what i call nude. ...	1
4	1	i've used this for 35 years. massiv...	1
...	...	...	...
59995	4	i love my konad kit! the black is a must have ...	3
59996	4	the palette is durable. i fit around around 2...	3
59997	5.0	we have used this conditioner on our hair for ...	3
59998	5	it's the best alternative to aluminum i've eve...	3
59999	5.0	easy on and easy off. no need to harm my hair...	3

60000 rows × 3 columns

Removing HTML and URLs

In [23]:
```
def remove_mention_tag_fn(text):
    text = re.sub(r'@\S*', '', text)
    return re.sub(r'#\S*', '', text)
```

In [24]:
```
# Removing well-formed tags i.e the HTML and URLs
review_df['review_body'] = review_df['review_body'].str.replace(r'<[^<>]*>', ''
review_df['review_body'] = review_df['review_body'].apply(lambda x: re.split('h
```

In [25]:
```
review_df['review_body'] = review_df['review_body'].apply(remove_mention_tag_fn
display_df['tag_cleaning'] = review_df['review_body'].str.len()
display(display_df)
```

	before_cleaning	tag_cleaning
0	65	65
1	145	145
2	499	499
3	185	185
4	96	72
...	...	...
59995	338	338
59996	147	147
59997	201	201
59998	91	91
59999	129	129

60000 rows × 2 columns

Removing punctuations

In [26]:
```
def remove_punctuations(text):
    return ''.join(char for char in text if char not in string.punctuation)
```

In [27]:
```
# Remove puctuations
review_df['review_body'] = review_df['review_body'].apply(remove_punctuations)
display_df['punctuation_cleaning'] = review_df['review_body'].str.len()
display(review_df)
display(display_df)
```

	star_rating	review_body	class
0	1	flimsy not what i expected i returned it pure ...	1
1	1	this product is overpriced for what it is man...	1
2	1	the quality of these brushes are terrible it d...	1
3	2	i love nude and this is not what i call nude i...	1
4	1	ive used this for 35 yearsmassive compliments ...	1
...	...	...	...
59995	4	i love my konad kit the black is a must have f...	3
59996	4	the palette is durable i fit around around 21...	3
59997	5.0	we have used this conditioner on our hair for ...	3
59998	5	its the best alternative to aluminum ive ever ...	3
59999	5.0	easy on and easy off no need to harm my hair ...	3

60000 rows × 3 columns

	before_cleaning	tag_cleaning	punctuation_cleaning
0	65	65	59
1	145	145	141
2	499	499	488
3	185	185	178
4	96	72	69
...	...	...	...
59995	338	338	330
59996	147	147	143
59997	201	201	196
59998	91	91	87
59999	129	129	125

60000 rows × 3 columns

Remove Emojis

In [28]:
```
# def remove_emoji_fn(string):
#      emoji_pattern = re.compile('['u'U0001F600-U0001F64F' # emoticons
#      u'U0001F300-U0001F5FF' # symbols & pictographs
#      u'U0001F680-U0001F6FF' # transport & map symbols
#      u'U0001F1E0-U0001F1FF' # flags (iOS)
#      u'U00002702-U000027B0'
#      u'U000024C2-U0001F251'
#      ']+', flags=re.UNICODE)
#     return emoji_pattern.sub(r'', string)
```

In [29]:
```
# review_df['review_body'] = review_df['review_body'].apply(remove_emoji_fn)
# display_df['emoji_cleaning'] = review_df['review_body'].str.len()
# display(review_df)
# display(display_df)
```

Removing non-alphabets

In [30]:
```
def remove_alphanum(text):
    t= " ".join([re.sub('[^A-Za-z]+','', text) for text in nltk.word_tokenize(t
    return t
```

In [31]:
```
# Remove non-alpabetics
review_df['review_body']=review_df['review_body'].apply(remove_alphanum)
display_df['alphanum_cleaning'] = review_df['review_body'].str.len()
display(review_df)
display(display_df)
```

	star_rating	review_body	class
0	1	flimsy not what i expected i returned it pure ...	1
1	1	this product is overpriced for what it is manu...	1
2	1	the quality of these brushes are terrible it d...	1
3	2	i love nude and this is not what i call nude i...	1
4	1	ive used this for yearsmassive compliments th...	1
...	...	...	...
59995	4	i love my konad kit the black is a must have f...	3
59996	4	the palette is durable i fit around around sm...	3
59997	5.0	we have used this conditioner on our hair for ...	3
59998	5	its the best alternative to aluminum ive ever ...	3
59999	5.0	easy on and easy off no need to harm my hair t...	3

60000 rows × 3 columns

	before_cleaning	tag_cleaning	punctuation_cleaning	alphanum_cleaning
0	65	65	59	59
1	145	145	141	139
2	499	499	488	488
3	185	185	178	178
4	96	72	69	67
...	...	...	...	...
59995	338	338	330	329
59996	147	147	143	139
59997	201	201	196	196
59998	91	91	87	86
59999	129	129	125	123

60000 rows × 4 columns

Removing extra spaces

In [32]:
```
review_df['review_body'] = review_df['review_body'].apply(lambda x: re.sub(' +'
display_df['remove_spaces'] = review_df['review_body'].str.len()
display(review_df)
display(display_df)
```

	star_rating	review_body	class
0	1	flimsy not what i expected i returned it pure ...	1
1	1	this product is overpriced for what it is manu...	1
2	1	the quality of these brushes are terrible it d...	1
3	2	i love nude and this is not what i call nude i...	1
4	1	ive used this for yearsmassive compliments thr...	1
...	...	...	...
59995	4	i love my konad kit the black is a must have f...	3
59996	4	the palette is durable i fit around around sma...	3
59997	5.0	we have used this conditioner on our hair for ...	3
59998	5	its the best alternative to aluminum ive ever ...	3
59999	5.0	easy on and easy off no need to harm my hair t...	3

60000 rows × 3 columns

	before_cleaning	tag_cleaning	punctuation_cleaning	alphanum_cleaning	remove_space
0	65	65	59	59	5
1	145	145	141	139	13
2	499	499	488	488	48
3	185	185	178	178	17
4	96	72	69	67	6
...	...	...	...	...	.
59995	338	338	330	329	32
59996	147	147	143	139	13
59997	201	201	196	196	19
59998	91	91	87	86	8
59999	129	129	125	123	12

60000 rows × 5 columns

Contracting the words

In [33]:
```
def word_contractions(text):
    t=[]
    for i in text.split():
        t.append(contractions.fix(i))
    # Now that the review has been split into a list of words and contracted, t
    return ' '.join(t)
```

In [34]:
```
# Contracting the reviews
review_df['review_body']=review_df['review_body'].apply(word_contractions)
```

```
# display(review_df)

# # Now that the review has been split into a list of words and contracted, the
# review_df['review_body'] = review_df['review_body'].apply(word_contractions)

display_df['post_contractions'] = review_df['review_body'].str.len()

display(review_df)
display(display_df)
```

	star_rating	review_body	class
0	1	flimsy not what i expected i returned it pure ...	1
1	1	this product is overpriced for what it is manu...	1
2	1	the quality of these brushes are terrible it d...	1
3	2	i love nude and this is not what i call nude i...	1
4	1	i have used this for yearsmassive compliments ...	1
...	...	...	...
59995	4	i love my konad kit the black is a must have f...	3
59996	4	the palette is durable i fit around around sma...	3
59997	5.0	we have used this conditioner on our hair for ...	3
59998	5	its the best alternative to aluminum i have ev...	3
59999	5.0	easy on and easy off no need to harm my hair t...	3

60000 rows × 3 columns

	before_cleaning	tag_cleaning	punctuation_cleaning	alphanum_cleaning	remove_space
0	65	65	59	59	5
1	145	145	141	139	13
2	499	499	488	488	48
3	185	185	178	178	17
4	96	72	69	67	6
...	...	...	...	...	.
59995	338	338	330	329	32
59996	147	147	143	139	13
59997	201	201	196	196	19
59998	91	91	87	86	8
59999	129	129	125	123	12

60000 rows × 6 columns

In [35]:
```
display_df['after_cleaning'] = review_df['review_body'].str.len()
display(display_df)
```

	before_cleaning	tag_cleaning	punctuation_cleaning	alphanum_cleaning	remove_space
0	65	65	59	59	5
1	145	145	141	139	13
2	499	499	488	488	48
3	185	185	178	178	17
4	96	72	69	67	6
...	...	...	...	...	.
59995	338	338	330	329	32
59996	147	147	143	139	13
59997	201	201	196	196	19
59998	91	91	87	86	8
59999	129	129	125	123	12

60000 rows × 7 columns

In [36]:
```
print("The length of the reviews initially-  ",display_df['before_cleaning'].me
print("The length of the reviews after cleaning-  ", display_df['after_cleaning
```

```
The length of the reviews initially-   267.9975333333333
The length of the reviews after cleaning-   256.52843333333334
```

Pre-Processing

Removing the Stop Words

In [37]:
```
from nltk.corpus import stopwords
```

In [38]:
```
stop = set(stopwords.words('english'))
```

In [39]:
```
def stop_word_fn(text):
    return ' '.join(i for i in text.split() if i not in (stop))
```

In [40]:
```
review_df['review_body'] = review_df['review_body'].apply(stop_word_fn)
display(review_df)
```

	star_rating	review_body	class
0	1	flimsy expected returned pure cheap plastic	1
1	1	product overpriced manufacture making killing ...	1
2	1	quality brushes terrible even come neat box pa...	1
3	2	love nude call nude orangey would able tell st...	1
4	1	used yearsmassive compliments throughout timerwh	1
...	...	...	...
59995	4	love konad kit black must collection gave star...	3
59996	4	palette durable fit around around small eyesha...	3
59997	5.0	used conditioner hair decades makes easy comb ...	3
59998	5	best alternative aluminum ever used use even f...	3
59999	5.0	easy easy need harm hair get great colors love...	3

60000 rows × 3 columns

In [41]:
```python
display_df['stopword_removal'] = review_df['review_body'].str.len()
display(display_df)
```

	before_cleaning	tag_cleaning	punctuation_cleaning	alphanum_cleaning	remove_space
0	65	65	59	59	5
1	145	145	141	139	13
2	499	499	488	488	48
3	185	185	178	178	17
4	96	72	69	67	6
...	...	...	...	...	.
59995	338	338	330	329	32
59996	147	147	143	139	13
59997	201	201	196	196	19
59998	91	91	87	86	8
59999	129	129	125	123	12

60000 rows × 8 columns

Lemmatization

In [42]:
```python
from nltk.stem import WordNetLemmatizer
```

In [43]:
```python
wnl = WordNetLemmatizer()
review_df['review_body'] = review_df['review_body'].apply(wnl.lemmatize)
display_df['after_lemmatize'] = review_df['review_body'].str.len()
display(display_df)
```

	before_cleaning	tag_cleaning	punctuation_cleaning	alphanum_cleaning	remove_space
0	65	65	59	59	5
1	145	145	141	139	13
2	499	499	488	488	48
3	185	185	178	178	17
4	96	72	69	67	6
...	...	...	...	...	.
59995	338	338	330	329	32
59996	147	147	143	139	13
59997	201	201	196	196	19
59998	91	91	87	86	8
59999	129	129	125	123	12

60000 rows × 9 columns

In [44]:
```
display(review_df)
```

	star_rating	review_body	class
0	1	flimsy expected returned pure cheap plastic	1
1	1	product overpriced manufacture making killing ...	1
2	1	quality brushes terrible even come neat box pa...	1
3	2	love nude call nude orangey would able tell st...	1
4	1	used yearsmassive compliments throughout timerwh	1
...	...	...	...
59995	4	love konad kit black must collection gave star...	3
59996	4	palette durable fit around around small eyesha...	3
59997	5.0	used conditioner hair decades makes easy comb ...	3
59998	5	best alternative aluminum ever used use even f...	3
59999	5.0	easy easy need harm hair get great colors love...	3

60000 rows × 3 columns

In [45]:
```
review_df['star_rating'].unique()
```

Out[45]:
```
array(['1', 1, 2, '2', 3, '3', 5.0, 4, '5', '4'], dtype=object)
```

In [46]:
```
review_df['class'].value_counts()
```

```
Out[46]: 1    20000
2    20000
3    20000
Name: class, dtype: int64
```

```
In [47]: review_df.isnull().sum()
```

```
Out[47]: star_rating    0
review_body    0
class          0
dtype: int64
```

```
In [48]: print("The length of the reviews post cleaning-  ",display_df['after_cleaning']
print("The length of the reviews after pre-processing-  ", display_df['after_le
```

```
The length of the reviews post cleaning-   256.52843333333334
The length of the reviews after pre-processing-   158.23988333333332
```

TF-IDF Feature Extraction

```
In [49]: #Splitting the Data into train and test data (split should be of 80%-20%)
Xtrain, Xtest, ytrain, ytest = train_test_split(review_df['review_body'], revie

print("Training Data Size: ", Xtrain.shape)
print("Testing Data Size: ", Xtest.shape)
```

```
Training Data Size:  (48000,)
Testing Data Size:  (12000,)
```

```
In [50]: # Verifying the distribution of the classes in the training data
ytrain.value_counts()
```

```
Out[50]: 3    16152
1    15935
2    15913
Name: class, dtype: int64
```

```
In [51]: tfID_feat_extract = TfidfVectorizer(
    sublinear_tf=True,
    strip_accents='unicode',
    analyzer='word',
    token_pattern=r'\w{1,}',
    stop_words='english',
    ngram_range=(1, 2),
    max_features=12000
)
```

```
In [52]: Xtrain_tfid = tfID_feat_extract.fit_transform(Xtrain)

Xtest_tfid = tfID_feat_extract.transform(Xtest)

print("Training document-term matrix : ", Xtrain_tfid)
print("Training feature names for transformation : ", tfID_feat_extract.get_fea
```

```
Training document-term matrix :	  (0, 6223)	0.12866512916124828
  (0, 5158)	0.13453435367535402
  (0, 11688)	0.13037007261111924
  (0, 9869)	0.07041136868880836
  (0, 5066)	0.08595912050360374
  (0, 1584)	0.12279590464714253
  (0, 1982)	0.09768705352064395
  (0, 11339)	0.11508340522607381
  (0, 9811)	0.09359390423954994
  (0, 7205)	0.06798215936371434
  (0, 10416)	0.1174729822777536
  (0, 11542)	0.05715067099333924
  (0, 2878)	0.07713167305122111
  (0, 6212)	0.11989113838855016
  (0, 4434)	0.06373607111240538
  (0, 5641)	0.05177872695946962
  (0, 80)	0.06605077321612345
  (0, 1396)	0.12714000527524236
  (0, 10701)	0.07080478273354904
  (0, 2545)	0.1871452907505059
  (0, 1254)	0.11674906699989634
  (0, 6222)	0.11172800993205609
  (0, 1572)	0.07681501348240494
  (0, 11906)	0.07854795015070919
  (0, 10296)	0.19432605192742297
  :	:
  (47999, 6397)	0.22405505258445435
  (47999, 5798)	0.12243515042322203
  (47999, 1562)	0.2192955320406732
  (47999, 11481)	0.12132377184412131
  (47999, 5499)	0.1452180020475867
  (47999, 11499)	0.12503644133034675
  (47999, 6390)	0.09897347500899512
  (47999, 11498)	0.11676618039561945
  (47999, 7597)	0.20537202983390893
  (47999, 2623)	0.15630795522156135
  (47999, 3364)	0.13582298475767707
  (47999, 6843)	0.12699346138029993
  (47999, 9177)	0.14557174407357398
  (47999, 1561)	0.163140306867726
  (47999, 11176)	0.07655851861103888
  (47999, 3747)	0.11928469948112666
  (47999, 4094)	0.19732737615615942
  (47999, 750)	0.10258075248295706
  (47999, 1671)	0.15712766873954198
  (47999, 2611)	0.09690179517214602
  (47999, 5395)	0.10800140468308556
  (47999, 7624)	0.05860844641522749
  (47999, 2181)	0.09624521988747611
  (47999, 10701)	0.11990787773173055
  (47999, 7592)	0.11739020419688075
Training feature names for transformation :  ['aa' 'aa battery' 'aaa' ... 'zi
t' 'zits' 'zone']
```

In [53]:
```
Xtrain_tfid.todense()
```

Out[53]:
```
matrix([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]])
```

In [54]:
```
from sklearn.model_selection import GridSearchCV
```

Models

Grid Search has been used for determining the most efficient hyperparameters for the different models. The two types of penalties i.e Ridge and Lasso has been considered for the models and the best is chosen.

Perceptron

In [55]:
```
params2=  {'alpha':[0.00000001, 0.0000001,0.000001,0.00001,0.0001,0.001,0.01,0.
model_perceptron2 = Perceptron(
    penalty= 'l2',       #Penalty for wrong prediction
    max_iter=1500,       #Maximum number of iterations
    shuffle=True,
    random_state=16,
)
clf32 = GridSearchCV(model_perceptron2, params2)
clf32=clf32.fit(Xtrain_tfid , ytrain)
pred_percept2=clf32.predict(Xtest_tfid)
result2=classification_report(ytest, pred_percept2,output_dict=True)
print(result2)
```

```
{'1': {'precision': 0.5988321799307958, 'recall': 0.6811808118081181, 'f1-scor
e': 0.637357578547589, 'support': 4065}, '2': {'precision': 0.535104895104895
1, 'recall': 0.46806948862246145, 'f1-score': 0.4993474288697468, 'support': 4
087}, '3': {'precision': 0.691397000789266, 'recall': 0.682952182952183, 'f1-s
core': 0.6871486468819453, 'support': 3848}, 'accuracy': 0.6091666666666666,
'macro avg': {'precision': 0.6084446919416523, 'recall': 0.6107341611275875,
'f1-score': 0.6079512180997604, 'support': 12000}, 'weighted avg': {'precisio
n': 0.6068101813957906, 'recall': 0.6091666666666666, 'f1-score': 0.6063199576
490275, 'support': 12000}}
```

In [56]:
```
i=1
for keys,values in result2.items():
    if i==4:
        i=i+1
        continue
    else:
        print(keys,": ",values['precision'],",",values['recall'],",",values['f1
        i=i+1
```

```
1 :  0.5988321799307958 , 0.6811808118081181 , 0.637357578547589
2 :  0.5351048951048951 , 0.46806948862246145 , 0.4993474288697468
3 :  0.691397000789266 , 0.682952182952183 , 0.6871486468819453
macro avg :  0.6084446919416523 , 0.6107341611275875 , 0.6079512180997604
weighted avg :  0.6068101813957906 , 0.6091666666666666 , 0.6063199576490275
```

In [57]:
```
print(clf32.best_params_)
```

```
{'alpha': 1e-05, 'tol': 0.001}
```

SVM

In [58]:
```
parameters = {'C':[0.01, 0.05,0.1,0.15, 0.2,0.25,0.3,0.35,0.4],'tol':[0.000001,
```

In [59]:
```
svm_model = LinearSVC(
    max_iter=1000,                  #Total iterations
    random_state=16,                 #Control the random number generation to co
    penalty='l1',                   #Norm of Penalty
    class_weight="balanced",        #Provides the weight to each class
    loss='squared_hinge',           #Specifies the Loss Function
    dual=False,                     #Selects the algorithm to either the dual or
)
clf = GridSearchCV(svm_model, parameters)
clf.fit(Xtrain_tfid , ytrain)
pred_svm=clf.predict(Xtest_tfid)
svm_result=classification_report(ytest, pred_svm,output_dict=True)
print(svm_result)
```

```
{'1': {'precision': 0.6864902833060174, 'recall': 0.7212792127921279, 'f1-scor
e': 0.7034548944337812, 'support': 4065}, '2': {'precision': 0.613892013498312
7, 'recall': 0.5341326156104722, 'f1-score': 0.5712416590344105, 'support': 40
87}, '3': {'precision': 0.7277737838485502, 'recall': 0.7892411642411642, 'f1-
score': 0.7572621867597555, 'support': 3848}, 'accuracy': 0.6793333333333333,
'macro avg': {'precision': 0.6760520268842933, 'recall': 0.6815509975479214,
'f1-score': 0.6773195800759825, 'support': 12000}, 'weighted avg': {'precisio
n': 0.6750027650879821, 'recall': 0.6793333333333333, 'f1-score': 0.6756794750
83208, 'support': 12000}}
```

In [60]:
```
i=1
for keys,values in svm_result.items():
    if i==4:
        i=i+1
        continue
    else:
        print(keys,": ",values['precision'],",",values['recall'],",",values['f1
        i=i+1
```

```
1 :  0.6864902833060174 , 0.7212792127921279 , 0.7034548944337812
2 :  0.6138920134983127 , 0.5341326156104722 , 0.5712416590344105
3 :  0.7277737838485502 , 0.7892411642411642 , 0.7572621867597555
macro avg :  0.6760520268842933 , 0.6815509975479214 , 0.6773195800759825
weighted avg :  0.6750027650879821 , 0.6793333333333333 , 0.675679475083208
```

In [61]:
```
print(clf.best_params_)
```

```
{'C': 0.35, 'tol': 0.001}
```

Logistic Regression

In [62]:
```
parameters2 = {'C':[0.01, 0.05,0.1,0.15, 0.2,0.25,0.3,0.35,0.4],'solver':['saga
```

In [63]:
```
lr_model = LogisticRegression(
    max_iter=2000,                #Max iterations to be considered
    penalty='l2',                 #Penalty for wrong prediction
    multi_class='multinomial',
    random_state=16,
)
clf2 = GridSearchCV(lr_model, parameters2)
clf2.fit(Xtrain_tfid , ytrain)
pred_logistic=clf2.predict(Xtest_tfid)
lr_result=classification_report(ytest, pred_logistic,output_dict=True)

print(lr_result)
```

```
{'1': {'precision': 0.6902781079153791, 'recall': 0.7143911439114391, 'f1-scor
e': 0.7021276595744681, 'support': 4065}, '2': {'precision': 0.607817418677859
4, 'recall': 0.5669195008563739, 'f1-score': 0.58665653880238, 'support': 408
7}, '3': {'precision': 0.7430293896006028, 'recall': 0.7687110187110187, 'f1-s
core': 0.7556520628432749, 'support': 3848}, 'accuracy': 0.6815833333333333,
'macro avg': {'precision': 0.6803749720646138, 'recall': 0.6833405544929438,
'f1-score': 0.681478753740041, 'support': 12000}, 'weighted avg': {'precisio
n': 0.6791089491662956, 'recall': 0.6815833333333333, 'f1-score': 0.6799636123
39705, 'support': 12000}}
```

In [64]:
```
for keys,values in lr_result.items():
    print("Class",keys," ", values)
```

```
Class 1   {'precision': 0.6902781079153791, 'recall': 0.7143911439114391, 'f1-
score': 0.7021276595744681, 'support': 4065}
Class 2   {'precision': 0.6078174186778594, 'recall': 0.5669195008563739, 'f1-
score': 0.58665653880238, 'support': 4087}
Class 3   {'precision': 0.7430293896006028, 'recall': 0.7687110187110187, 'f1-
score': 0.7556520628432749, 'support': 3848}
Class accuracy   0.6815833333333333
Class macro avg   {'precision': 0.6803749720646138, 'recall': 0.68334055449294
38, 'f1-score': 0.681478753740041, 'support': 12000}
Class weighted avg   {'precision': 0.6791089491662956, 'recall': 0.68158333333
33333, 'f1-score': 0.679963612339705, 'support': 12000}
```

In [65]:
```
i=1
for keys,values in lr_result.items():
    if i==4:
        i=i+1
        continue
    else:
        print(keys,": ",values['precision'],",",values['recall'],",",values['f1
        i=i+1
```

```
1 :  0.6902781079153791 , 0.7143911439114391 , 0.7021276595744681
2 :  0.6078174186778594 , 0.5669195008563739 , 0.58665653880238
3 :  0.7430293896006028 , 0.7687110187110187 , 0.7556520628432749
macro avg :  0.6803749720646138 , 0.6833405544929438 , 0.681478753740041
weighted avg :  0.6791089491662956 , 0.6815833333333333 , 0.679963612339705
```

In [66]:
```
print(clf2.best_params_)
```

```
{'C': 0.4, 'solver': 'saga', 'tol': 0.01}
```

Naive Bayes

In [67]:
```
parameters3 = {'alpha':[1,2,3,4,5,6,7,8]}
```

In [68]:
```
nb_model = MultinomialNB()

clf3 = GridSearchCV(nb_model, parameters3)
clf3.fit(Xtrain_tfid , ytrain)
pred_nb=clf3.predict(Xtest_tfid)
nb_report=classification_report(ytest, pred_nb,output_dict=True)

print(nb_report)
```

```
{'1': {'precision': 0.6996966632962589, 'recall': 0.6809348093480935, 'f1-scor
e': 0.6901882558284503, 'support': 4065}, '2': {'precision': 0.594435936702399
1, 'recall': 0.5698556398336188, 'f1-score': 0.5818863210493441, 'support': 40
87}, '3': {'precision': 0.722249151720795, 'recall': 0.7744282744282744, 'f1-s
core': 0.747429144720341, 'support': 3848}, 'accuracy': 0.6730833333333334, 'm
acro avg': {'precision': 0.6721272505731509, 'recall': 0.6750729078699956, 'f1
-score': 0.6731679071993785, 'support': 12000}, 'weighted avg': {'precision':
0.671078445451968, 'recall': 0.6730833333333334, 'f1-score': 0.671657666912932
6, 'support': 12000}}
```

In [69]:
```
i=1
for keys,values in nb_report.items():
    if i==4:
        i=i+1
        continue
    else:
        print(keys,": ",values['precision'],",",values['recall'],",",values['f1
        i=i+1
```

```
1 :  0.6996966632962589 , 0.6809348093480935 , 0.6901882558284503
2 :  0.5944359367023991 , 0.5698556398336188 , 0.5818863210493441
3 :  0.722249151720795 , 0.7744282744282744 , 0.747429144720341
macro avg :  0.6721272505731509 , 0.6750729078699956 , 0.6731679071993785
weighted avg :  0.671078445451968 , 0.6730833333333334 , 0.6716576669129326
```

In [70]:
```
print(clf3.best_params_)
```

```
{'alpha': 6}
```

REFERENCES

In [71]:
```
# https://www.geeksforgeeks.org/how-to-randomly-select-rows-from-pandas-datafra
# https://www.statology.org/pandas-select-rows-based-on-column-values/
# https://stackoverflow.com/questions/45999415/removing-html-tags-in-pandas
# https://stackoverflow.com/questions/753052/strip-html-from-strings-in-python
# https://stackoverflow.com/questions/11331982/how-to-remove-any-url-within-a-s
# https://datatofish.com/lowercase-pandas-dataframe/
# https://stackoverflow.com/questions/39782418/remove-punctuations-in-pandas
# https://www.geeksforgeeks.org/python-map-function/
# https://stackoverflow.com/questions/29523254/python-remove-stop-words-from-pa
# https://aparnamishra144.medium.com/how-to-categorize-a-column-by-applying-a-f
# https://stackoverflow.com/questions/52279834/splitting-training-data-with-equ
# https://www.geeksforgeeks.org/python-remove-unwanted-spaces-from-string/
# https://towardsdatascience.com/primer-to-cleaning-text-data-7e856d6e5791
# https://michael-fuchs-python.netlify.app/2021/05/22/nlp-text-pre-processing-i
# https://www.programiz.com/python-programming/methods/string/join
```