Text Data - Sentiment Analysis

In [2]:
```
import zipfile
import os
import pandas as pd
```

In [3]:
```
from tqdm import tqdm
```

Extraction of file using Zipfile

In [ ]:
```
with zipfile.ZipFile("txt_reviews.zip",'r') as zip_data:
    tqdm(zip_data.extractall())
```

In [11]:
```
zip_data
```

Out[11]:
```
<zipfile.ZipFile [closed]>
```

In [4]:
```
path=r"C:\Users\pc\Downloads\Innomatics classes\Machine learning 14 Nov 2022\ML NLI
for files in os.listdir(path):
    pass
```

In [1]:
```
 with open("txt_reviews/review_10001.txt",'r',errors='ignore') as f:
        print(f.read())
```

```
ProductId: B000P41A28
UserId: A3K3GNZWUYXRUT
ProfileName: L. Bream
HelpfulnessNumerator: 3
HelpfulnessDenominator: 6
Score: 5
Time: 1214697600
ReviewSummary: If you're worried about consitpation....
ReviewText: I purchased this formula but was worried after reading the comments he
re that my 5 month old baby would suffer from constipation.  He did.  However, I r
eally wanted to use organic formula so I added a few teaspoons of prunes to his ce
real and within 12 hours - problem solved.  No constipation since and he has been
on this formula for about 2 weeks. I give him some prune/cereal every 4 days.  If
your baby is not yet on solids you might consider giving him a little apple or pea
r juice mixed with water.  This should do the trick also.  Don't let the constipat
ion issue scare you off.
```

In [8]:
```
ProductId=[]
UserId=[]
ProfileName=[]
HelpfulnessNumerator=[]
HelpfulnessDenominator=[]
Score=[]
Time=[]
ReviewSummary=[]
ReviewText=[]
```

In [10]:
```
for files in tqdm(os.listdir(path)):
    with open(path+files,'r',errors='ignore') as f:
        lines=f.readlines()
        ProductId.append(lines[0].strip("\n"))
        UserId.append(lines[1].strip("\n"))
        ProfileName.append(lines[2].strip('\n'))
        HelpfulnessNumerator.append(lines[3].strip('\n'))
        HelpfulnessDenominator.append(lines[4].strip('\n'))
        Score.append(lines[5].strip('\n'))
        Time.append(lines[6].strip('\n'))
        ReviewSummary.append(lines[7].strip('\n'))
        ReviewText.append(lines[8].strip('\n'))
```

```
100%|██████████████████████████████████████████████████████████████| 56
8454/568454 [42:11<00:00, 224.58it/s]
```

In [11]:

```
df=pd.DataFrame({'ProductId':ProductId,'UserId':UserId,'ProfileName':ProfileName,'
df
```

Out[11]:

	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenomi
0	ProductId: B001E4KFG0	UserId: A3SGXH7AUHU8GW	ProfileName: delmartian	HelpfulnessNumerator: 1	HelpfulnessDenom
1	ProductId: B00171APVA	UserId: A21BT40VZCCYT4	ProfileName: Carol A. Reed	HelpfulnessNumerator: 0	HelpfulnessDenom
2	ProductId: B0019CW0HE	UserId: A2P6ACFZ8FTNVV	ProfileName: Melissa Benjamin	HelpfulnessNumerator: 0	HelpfulnessDenom
3	ProductId: B006F2NYI2	UserId: A132DJVI37RB4X	ProfileName: Scottdrum	HelpfulnessNumerator: 2	HelpfulnessDenom
4	ProductId: B000P41A28	UserId: A82WIMR4RSVLI	ProfileName: Emrose mom	HelpfulnessNumerator: 0	HelpfulnessDenom
...	...	...	...	...	
568449	ProductId: B000LQORDE	UserId: AL22WN8RBBOW7	ProfileName: LifeInTheCity "trogg"	HelpfulnessNumerator: 2	HelpfulnessDenom
568450	ProductId: B000LQORDE	UserId: A2P7HIRYYWVOBD	ProfileName: Mason	HelpfulnessNumerator: 2	HelpfulnessDenom
568451	ProductId: B000LQORDE	UserId: A1K0ZH5MQFBA77	ProfileName: jennilight	HelpfulnessNumerator: 2	HelpfulnessDenom
568452	ProductId: B000LQORDE	UserId: A29FRN2O7LWINL	ProfileName: T. Tsai	HelpfulnessNumerator: 2	HelpfulnessDenom
568453	ProductId: B000LQORDE	UserId: A9Q950IPXJR1D	ProfileName: Lynda "casual customer"	HelpfulnessNumerator: 2	HelpfulnessDenom

568454 rows × 9 columns

In [14]:
```
len(ProductId)
```

```
Out[14]: 568454
```

```
In [12]: saving the dataframe
         df.to_csv('datauncleaned.csv')
```

In [1]:
```
import pandas as pd
import re
import seaborn as sns
```

In [2]:
```
from datetime import datetime
```

In [3]:
```
df=pd.read_csv("datauncleaned.csv")
df.head()
```

Out[3]:

	Unnamed: 0	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessD
0	0	ProductId: B001E4KFG0	UserId: A3SGXH7AUHU8GW	ProfileName: delmartian	HelpfulnessNumerator: 1	HelpfulnessD
1	1	ProductId: B00171APVA	UserId: A21BT40VZCCYT4	ProfileName: Carol A. Reed	HelpfulnessNumerator: 0	HelpfulnessD
2	2	ProductId: B0019CW0HE	UserId: A2P6ACFZ8FTNVV	ProfileName: Melissa Benjamin	HelpfulnessNumerator: 0	HelpfulnessD
3	3	ProductId: B006F2NYI2	UserId: A132DJVI37RB4X	ProfileName: Scottdrum	HelpfulnessNumerator: 2	HelpfulnessD
4	4	ProductId: B000P41A28	UserId: A82WIMR4RSVLI	ProfileName: Emrose mom	HelpfulnessNumerator: 0	HelpfulnessD

In [4]:
```
del df['Unnamed: 0']
```

In [5]:
```
lst=['ProductId',
'UserId',
'ProfileName',
'HelpfulnessNumerator',
'HelpfulnessDenominator',
'Score',
'Time',
'ReviewSummary',
'ReviewText']
```

In [6]:
```
for i in lst:
    df[i].replace(to_replace= i+":", value='', regex=True,inplace= True)
```

In [7]:
```
df.to_csv('datalittlecleaned.csv')
```

In [8]:
```
df.head()
```

Out[8]:

	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator
0	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1
1	B00171APVA	A21BT40VZCCYT4	Carol A. Reed	0	0
2	B0019CW0HE	A2P6ACFZ8FTNVV	Melissa Benjamin	0	1
3	B006F2NYI2	A132DJVI37RB4X	Scottdrum	2	5
4	B000P41A28	A82WIMR4RSVLI	Emrose mom	0	1

In [9]:
```
df['Score']=pd.to_numeric(df['Score'], errors='coerce')
df['HelpfulnessNumerator']=pd.to_numeric(df['HelpfulnessNumerator'], errors='coerc
df['HelpfulnessDenominator']=pd.to_numeric(df['HelpfulnessDenominator'], errors='c
df['Time']=pd.to_numeric(df['Time'], errors='coerce')
```

In [10]:
```
df.dtypes
```

Out[10]:
```
ProductId                 object
UserId                    object
ProfileName               object
HelpfulnessNumerator       int64
HelpfulnessDenominator     int64
Score                      int64
Time                       int64
ReviewSummary             object
ReviewText                object
dtype: object
```

In [11]:
```
data=pd.read_csv('reviews.csv')
```

In [12]:
```
data.head()
```

Out[12]:

	Unnamed: 0.1	Unnamed: 0	ProductId	UserId	ProfileName	HelpfulnessNumerator
0	0	0	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1
1	1	1	B00171APVA	A21BT40VZCCYT4	Carol A Reed	0
2	2	2	B0019CW0HE	A2P6ACFZ8FTNVV	Melissa Benjamin	0
3	3	3	B006F2NYI2	A132DJVI37RB4X	Scottdrum	2
4	4	4	B000P41A28	A82WIMR4RSVLI	Emrose mom	0

In [13]:
```
del data['Unnamed: 0']
del data['Unnamed: 0.1']
```

In [14]:
```
data.head()
```

Out[14]:

	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator
0	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1
1	B00171APVA	A21BT40VZCCYT4	Carol A Reed	0	0
2	B0019CW0HE	A2P6ACFZ8FTNVV	Melissa Benjamin	0	1
3	B006F2NYI2	A132DJVI37RB4X	Scottdrum	2	5
4	B000P41A28	A82WIMR4RSVLI	Emrose mom	0	1

In [15]:
```
data['Time'] = data['Time'].apply(lambda timestamp: datetime.fromtimestamp(timesta
```

In [ ]:

In [16]:
```
data['Time'].max()
```

Out[16]:
```
Timestamp('2012-10-26 05:30:00')
```

In [17]:
```
data['Time'].min()
```

Out[17]:
```
Timestamp('1999-10-08 05:30:00')
```

In [18]:
```
data.dtypes
```

Out[18]:
```
ProductId                         object
UserId                            object
ProfileName                       object
HelpfulnessNumerator               int64
HelpfulnessDenominator             int64
Score                              int64
Time                      datetime64[ns]
ReviewSummary                     object
ReviewText                        object
dtype: object
```

In [19]:
```
data.head()
```

Out[19]:

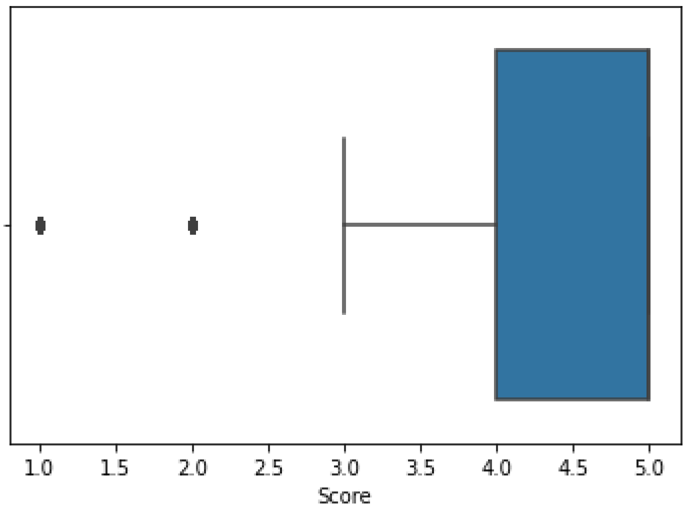
	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator
0	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1
1	B00171APVA	A21BT40VZCCYT4	Carol A Reed	0	0
2	B0019CW0HE	A2P6ACFZ8FTNVV	Melissa Benjamin	0	1
3	B006F2NYI2	A132DJVI37RB4X	Scottdrum	2	5
4	B000P41A28	A82WIMR4RSVLI	Emrose mom	0	1

In [20]:
```
#data['Helpfulness']=(data['HelpfulnessNumerator']/data['HelpfulnessDenominator'])
```

In [21]:
```
#data1=data['Helpfulness']
```

In [22]:
```
sns.boxplot(x=data["Score"])
```

Out[22]:
```
<AxesSubplot:xlabel='Score'>
```



In [23]:
```
data.duplicated().sum()
```

Out[23]:
```
294
```

In [24]:
```
data.drop_duplicates(inplace = True)
```

```
In [25]: data.isnull().sum()

Out[25]: ProductId                   0
         UserId                      0
         ProfileName               462
         HelpfulnessNumerator        0
         HelpfulnessDenominator      0
         Score                       0
         Time                        0
         ReviewSummary             263
         ReviewText                  1
         dtype: int64

In [26]: data.dropna(inplace=True)

In [27]: data.isnull().sum()

Out[27]: ProductId                 0
         UserId                    0
         ProfileName               0
         HelpfulnessNumerator      0
         HelpfulnessDenominator    0
         Score                     0
         Time                      0
         ReviewSummary             0
         ReviewText                0
         dtype: int64

In [28]: data.dropna(subset=['Time'], inplace = True)

In [29]: data.to_csv('datafinalcleaned.csv')
```

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv(r"C:\Users\pc\Downloads\Innomatics classes\Machine learning 14 Nov
```

In [3]:
```
df.head()
```

Out[3]:

	Unnamed: 0	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessD
0	0	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
1	1	B00171APVA	A21BT40VZCCYT4	Carol A Reed	0	
2	2	B0019CW0HE	A2P6ACFZ8FTNVV	Melissa Benjamin	0	
3	3	B006F2NYI2	A132DJVI37RB4X	Scottdrum	2	
4	4	B000P41A28	A82WIMR4RSVLI	Emrose mom	0	

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 567438 entries, 0 to 567437
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype 
---  ------                  --------------   ----- 
 0   Unnamed: 0              567438 non-null  int64 
 1   ProductId               567438 non-null  object
 2   UserId                  567438 non-null  object
 3   ProfileName             567438 non-null  object
 4   HelpfulnessNumerator    567438 non-null  int64 
 5   HelpfulnessDenominator  567438 non-null  int64 
 6   Score                   567438 non-null  int64 
 7   Time                    567438 non-null  object
 8   ReviewSummary           567438 non-null  object
 9   ReviewText              567438 non-null  object
dtypes: int64(4), object(6)
memory usage: 43.3+ MB
```

```
In [5]: df.describe()
```

Out[5]:

	Unnamed: 0	HelpfulnessNumerator	HelpfulnessDenominator	Score
count	567438.000000	567438.000000	567438.000000	567438.000000
mean	284242.433418	1.743912	2.228321	4.183669
std	164102.171805	7.631681	8.284970	1.309995
min	0.000000	0.000000	0.000000	1.000000
25%	142136.250000	0.000000	0.000000	4.000000
50%	284242.500000	0.000000	1.000000	5.000000
75%	426372.750000	2.000000	2.000000	5.000000
max	568453.000000	866.000000	923.000000	5.000000

Arranging the Columns and keeping the Target Variable as Last Column

```
In [6]: df.columns
```

```
Out[6]: Index(['Unnamed: 0', 'ProductId', 'UserId', 'ProfileName',
               'HelpfulnessNumerator', 'HelpfulnessDenominator', 'Score', 'Time',
               'ReviewSummary', 'ReviewText'],
              dtype='object')
```

```
In [7]: df = df[['ProductId', 'UserId', 'ProfileName',
               'HelpfulnessNumerator', 'HelpfulnessDenominator', 'Time',
               'ReviewSummary', 'ReviewText', 'Score']]
```

```
In [8]: df
```

Out[8]:

	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenomi
0	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
1	B00171APVA	A21BT40VZCCYT4	Carol A Reed	0	
2	B0019CW0HE	A2P6ACFZ8FTNVV	Melissa Benjamin	0	
3	B006F2NYI2	A132DJVI37RB4X	Scottdrum	2	
4	B000P41A28	A82WIMR4RSVLI	Emrose mom	0	
...	...	...	...	...	
567433	B000LQORDE	AL22WN8RBBOW7	LifeInTheCity	2	
567434	B000LQORDE	A2P7HIRYYWVOBD	Mason	2	
567435	B000LQORDE	A1K0ZH5MQFBA77	jennilight	2	
567436	B000LQORDE	A29FRN2O7LWINL	T Tsai	2	
567437	B000LQORDE	A9Q950IPXJR1D	Lynda customer	2	

567438 rows × 9 columns

In [ ]:

1.Text Data Visualisation

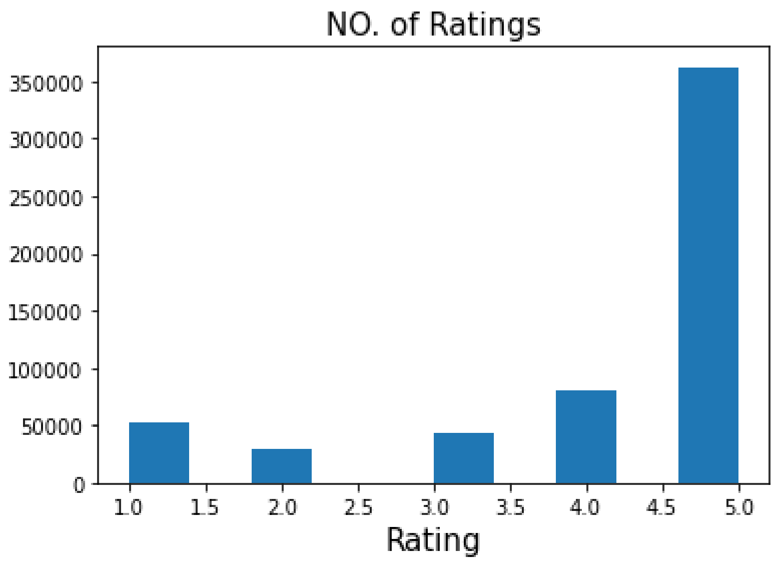
In [9]:
```
df.Score.value_counts()
```

Out[9]:
```
5    362528
4     80537
1     52105
3     42544
2     29724
Name: Score, dtype: int64
```

In [20]:
```
df["Score"].count()
```

Out[20]:
```
567438
```

In [12]:
```
plt.hist(df['Score'])
plt.title("NO. of Ratings",fontsize=15)
plt.xlabel('Rating',fontsize=15)
plt.figure(figsize=(20,5))
plt.show();
```



```
<Figure size 1440x360 with 0 Axes>
```

In [ ]:
```
sns.distplot(df["Score"])
plt.show()
```

In [13]:
```
df.head()
```

Out[13]:

	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator
0	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1
1	B00171APVA	A21BT40VZCCYT4	Carol A Reed	0	0
2	B0019CW0HE	A2P6ACFZ8FTNVV	Melissa Benjamin	0	1
3	B006F2NYI2	A132DJVI37RB4X	Scottdrum	2	5
4	B000P41A28	A82WIMR4RSVLI	Emrose mom	0	1

In [14]:
```
df.ProductId.value_counts()
```

Out[14]:
```
 B007JFMH8M    911
 B002QWP8H0    629
 B002QWHJOU    629
 B0026RQTGE    629
 B002QWP89S    629
              ...
 B002UG9N6Y      1
 B0040IZN4Q      1
 B002GWQ3AQ      1
 B003MNOBMU      1
 B001E4KFG0      1
Name: ProductId, Length: 74218, dtype: int64
```

In [15]:
```
df.ProductId.unique()
```

Out[15]:
```
array([' B001E4KFG0', ' B00171APVA', ' B0019CW0HE', ..., ' B000LLHNV2',
       ' B0028GY8U2', ' B000KGOTO2'], dtype=object)
```

In [16]:
```
df.ProfileName.value_counts()
```

Out[16]:
```
 J                     515
 C F Hill              449
 O Brown O Khannah     418
 Chris                 404
 M                     391
                      ...
 G Knight                1
 Robert A Balslev        1
 TeamTQ                  1
 M Polikoff              1
 Lynda customer          1
Name: ProfileName, Length: 208893, dtype: int64
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [70]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
df = pd.read_csv('datafinalcleaned1.csv')
```

In [5]:
```
del df['Unnamed: 0']
```

Data Preparation - Split the data into train and test set

In [6]:
```
y=df['Score']
X=df[['ReviewText']]
```

In [7]:
```
X.head()
```

Out[7]:

	ReviewText
0	I have bought several of the Vitality canned ...
1	This is a very healthy dog food Good for thei...
2	I fed this to my Golden Retriever and he hate...
3	I have to admit I was a sucker for the large ...
4	We have a 7 week old He had gas and constipat...

In [8]:
```
#spliting into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_s
```

In [9]:
```
X_train.head()
```

Out[9]:

	ReviewText
447215	It is good and soothing to drink I have not t...
253653	Whenever I need something a little special fo...
566123	I bought this almost a month ago and my dog a...
381706	Extract is listed as an ingredient Sounds har...
547077	I purchased these nuts as a gift and was disa...

Data Prepration - Text Preprocessing

In [10]:
```
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
```

In [11]:
```
## initialise the inbuilt Stemmer
stemmer = PorterStemmer()
## We can also use Lemmatizer instead of Stemmer
lemmatizer = WordNetLemmatizer()
```

Step by Step explanation of Preprocessing

In [12]:
```
raw_text = 'I was buying my white pepper from a gourmet store and 6 ounces cost Thi
raw_text
```

Out[12]:
```
'I was buying my white pepper from a gourmet store and 6 ounces cost This pepper i
s just as good at a fraction of the price'
```

In [13]:
```
sentence = re.sub("[^a-zA-Z]", " ", raw_text)
print(sentence)
```

```
I was buying my white pepper from a gourmet store and   ounces cost This pepper is
just as good at a fraction of the price
```

In [14]:
```
sentence = sentence.lower()
print(sentence)
```

```
i was buying my white pepper from a gourmet store and   ounces cost this pepper is
just as good at a fraction of the price
```

In [15]:
```
tokens = sentence.split()
print(tokens)
```

```
['i', 'was', 'buying', 'my', 'white', 'pepper', 'from', 'a', 'gourmet', 'store',
'and', 'ounces', 'cost', 'this', 'pepper', 'is', 'just', 'as', 'good', 'at', 'a',
'fraction', 'of', 'the', 'price']
```

In [16]:
```
clean_tokens = [t for t in tokens if t not in stopwords.words("english")]
print(clean_tokens)
```

```
['buying', 'white', 'pepper', 'gourmet', 'store', 'ounces', 'cost', 'pepper', 'goo
d', 'fraction', 'price']
```

In [17]:
```
clean_tokens_stem = [stemmer.stem(word) for word in clean_tokens]
print(clean_tokens_stem)
```

```
['buy', 'white', 'pepper', 'gourmet', 'store', 'ounc', 'cost', 'pepper', 'good',
'fraction', 'price']
```

In [18]:
```
clean_tokens_lem = [lemmatizer.lemmatize(word) for word in clean_tokens]
print(clean_tokens_lem)
```

```
['buying', 'white', 'pepper', 'gourmet', 'store', 'ounce', 'cost', 'pepper', 'goo
d', 'fraction', 'price']
```

In [88]:
```
def preprocess(raw_text, flag):
    # Removing special characters and digits
    sentence = re.sub("[^a-zA-Z]", " ", raw_text)

    # change sentence to lower case
    sentence = sentence.lower()

    # tokenize into words
    tokens = sentence.split()
    # remove stop words
    clean_tokens = [t for t in tokens if not t in stopwords.words("english")]

    # Stemming/Lemmatization
    if(flag == 'stem'):
        clean_tokens = [stemmer.stem(word) for word in clean_tokens]
    else:
        clean_tokens = [lemmatizer.lemmatize(word) for word in clean_tokens]

    return pd.Series([" ".join(clean_tokens), len(clean_tokens)])
```

In [20]:
```
from tqdm import tqdm, tqdm_notebook

tqdm.pandas()
```

temp_df = X_train["ReviewText"].progress_apply(lambda x: preprocess(x, 'stem'))

temp_df.head()

temp_df.columns = ['clean_text_stem', 'text_length_stem']

temp_df.head()

X_train = pd.concat([X_train, temp_df], axis=1)

X_train.head()

X_train.to_csv('X_train_stem.csv')

In [21]:
```
X_train_stem=pd.read_csv("X_train_stem.csv")
X_train_stem
```

Out[21]:

	Unnamed: 0	ReviewText	clean_text_stem	text_length_stem
0	447215	It is good and soothing to drink I have not t...	good sooth drink tri mani time yet share sister	9
1	253653	Whenever I need something a little special fo...	whenev need someth littl special dinner mario ...	37
2	566123	I bought this almost a month ago and my dog a...	bought almost month ago dog small corgi mix st...	27
3	381706	Extract is listed as an ingredient Sounds har...	extract list ingredi sound harmless right spec...	33
4	547077	I purchased these nuts as a gift and was disa...	purchas nut gift disappoint arriv small brown ...	15
...	...	...	...	...
425573	385156	I had no problem eating these bars they taste...	problem eat bar tast ok thought wow nice addit...	33
425574	321502	All 3 of my dogs just love them I think they ...	dog love think good cost whole lot less	8
425575	441633	I love all of the Happy Baby products but thi...	love happi babi product one favorit highlight ...	34
425576	239499	Great price and a really good hot addition to...	great price realli good hot addit meal like th...	16
425577	103904	Dear America is amazing stuff You won all lik...	dear america amaz stuff like though either say...	19

425578 rows × 4 columns

X_train_lemma

temp_df = X_train['ReviewText'].progress_apply(lambda x: preprocess(x, 'lemma'))

temp_df.head()

temp_df.columns = ['clean_text_lemma', 'text_length_lemma']

temp_df.head()

X_train = pd.concat([X_train, temp_df], axis=1)

X_train.head()

X_train.to_csv("X_train_lemma.csv")

In [22]:
```
X_train=pd.read_csv("X_train_lemma.csv")

X_train
```

Out[22]:

	Unnamed: 0	ReviewText	clean_text_lemma	text_length_lemma
0	447215	It is good and soothing to drink I have not t...	good soothing drink tried many time yet sharin...	9
1	253653	Whenever I need something a little special fo...	whenever need something little special dinner ...	37
2	566123	I bought this almost a month ago and my dog a...	bought almost month ago dog small corgi mix st...	27
3	381706	Extract is listed as an ingredient Sounds har...	extract listed ingredient sound harmless right...	33
4	547077	I purchased these nuts as a gift and was disa...	purchased nut gift disappointed arrived small ...	15
...	...	...	...	...
425573	385156	I had no problem eating these bars they taste...	problem eating bar tasted ok thought wow nice ...	33
425574	321502	All 3 of my dogs just love them I think they ...	dog love think good cost whole lot le	8
425575	441633	I love all of the Happy Baby products but thi...	love happy baby product one favorite highlight...	34
425576	239499	Great price and a really good hot addition to...	great price really good hot addition meal like...	16
425577	103904	Dear America is amazing stuff You won all lik...	dear america amazing stuff like though either ...	19

425578 rows × 4 columns

In [23]:
```
del X_train['Unnamed: 0']
```

In [24]:
```
X_train
```

Out[24]:

	ReviewText	clean_text_lemma	text_length_lemma
0	It is good and soothing to drink I have not t...	good soothing drink tried many time yet sharin...	9
1	Whenever I need something a little special fo...	whenever need something little special dinner ...	37
2	I bought this almost a month ago and my dog a...	bought almost month ago dog small corgi mix st...	27
3	Extract is listed as an ingredient Sounds har...	extract listed ingredient sound harmless right...	33
4	I purchased these nuts as a gift and was disa...	purchased nut gift disappointed arrived small ...	15
...	...	...	...
425573	I had no problem eating these bars they taste...	problem eating bar tasted ok thought wow nice ...	33
425574	All 3 of my dogs just love them I think they ...	dog love think good cost whole lot le	8
425575	I love all of the Happy Baby products but thi...	love happy baby product one favorite highlight...	34
425576	Great price and a really good hot addition to...	great price really good hot addition meal like...	16
425577	Dear America is amazing stuff You won all lik...	dear america amazing stuff like though either ...	19

425578 rows × 3 columns

Using Bag Of Words(BOW)

In [25]:
```
from sklearn.feature_extraction.text import CountVectorizer

vocab = CountVectorizer()

X_train_bow= vocab.fit_transform(X_train['clean_text_lemma'])
```

In [26]:
```
X_train_bow
```

Out[26]:
```
<425578x83171 sparse matrix of type '<class 'numpy.int64'>'
	with 13510900 stored elements in Compressed Sparse Row format>
```

In [27]:
```
X_train_bow[0]
```

Out[27]:
```
<1x83171 sparse matrix of type '<class 'numpy.int64'>'
	with 9 stored elements in Compressed Sparse Row format>
```

Preprocessing the test data

In [28]:
```
X_test.head()
```

```
Out[28]:
```

	ReviewText
346436	Ive tried tons of cheap cat litter brands and...
132237	My 35 pound pit bull mix chewed the rope in h...
261415	This product I love However the product pictu...
76796	The Switch Kiwi Berry tastes metallic and fak...
498830	i love ordering on line esp from amazon just ...

temp_df = X_test['ReviewText'].progress_apply(lambda x: preprocess(x, 'lemma'))

temp_df.head()

temp_df.columns = ['clean_text_lemma', 'text_length_lemma']

temp_df.head()

X_test = pd.concat([X_test, temp_df], axis=1)

X_test.head()

X_test.to_csv("X_test.csv")

```
In [29]: X_test=pd.read_csv("X_test.csv")
```

```
In [30]: del X_test['Unnamed: 0']
```

```
In [31]: X_test
```

Out[31]:

	ReviewText	clean_text_lemma	text_length_lemma
0	Ive tried tons of cheap cat litter brands and...	ive tried ton cheap cat litter brand cheap yea...	25
1	My 35 pound pit bull mix chewed the rope in h...	pound pit bull mix chewed rope half le minute ...	54
2	This product I love However the product pictu...	product love however product pictured amazon s...	25
3	The Switch Kiwi Berry tastes metallic and fak...	switch kiwi berry taste metallic fake neither ...	67
4	i love ordering on line esp from amazon just ...	love ordering line esp amazon run coffee get e...	21
...	...	...	...
141855	These are the best of the mauna loa collectio...	best mauna loa collection u like coffee one so...	30
141856	I really like this cookie A bit dry but good ...	really like cookie bit dry good granddaughter ...	9
141857	These bones are awesome My dogs love them it ...	bone awesome dog love great purchased went bac...	19
141858	Very tasty No sugar or junk that I don want J...	tasty sugar junk want real food make great sna...	10
141859	This must be a typeo or this is the most expe...	must typeo expensive pod coffee ever	6

141860 rows × 3 columns

In [32]:
```
X_test_bow = vocab.transform(X_test['clean_text_lemma'])
```

In [ ]:
```

```

In [33]:
```
X_test_bow
```

Out[33]:
```
<141860x83171 sparse matrix of type '<class 'numpy.int64'>'
	with 4486102 stored elements in Compressed Sparse Row format>
```

In [137…
```
# TF-IDF

from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()

X_train_vector = vectorizer.fit_transform(X_train['clean_text_lemma'])
```

In [35]:
```
X_test_vector = vectorizer.transform(X_test['clean_text_lemma'])
```

In [36]:
```
X_test_vector
```

Out[36]:
```
<141860x83171 sparse matrix of type '<class 'numpy.float64'>'
	with 4486102 stored elements in Compressed Sparse Row format>
```

In [ ]:
```

```

In [ ]:
```

```

Using Models

Logistic Regression Using TF-IDF

In [138…

```
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression()
classifier.fit(X_train_vector,y_train)
```

```
C:\Users\pc\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:444: Con
vergenceWarning: 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(
```

Out[138]:

```
▾ LogisticRegression
LogisticRegression()
```

In [139…

```
y_test_pred = classifier.predict(X_test_vector)
```

In [140…

```
from sklearn.metrics import accuracy_score, classification_report

print(accuracy_score(y_test, y_test_pred))

print(classification_report(y_test, y_test_pred))
```

```
0.7351191315381362
              precision    recall  f1-score   support

           1       0.65      0.67      0.66     13085
           2       0.45      0.20      0.28      7401
           3       0.45      0.28      0.35     10672
           4       0.51      0.26      0.34     20264
           5       0.79      0.95      0.86     90438

    accuracy                           0.74    141860
   macro avg       0.57      0.47      0.50    141860
weighted avg       0.70      0.74      0.70    141860
```

In [141…

```
import joblib
from joblib import dump, load
```

In [142…

```
joblib.dump(classifier,'logistic_Regression_TFIDF')
```

Out[142]:

```
['logistic_Regression_TFIDF']
```

Logistic Regression using BOW

In [40]:

```
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression()
classifier.fit(X_train_bow,y_train)
```

```
C:\Users\pc\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:444: Con
vergenceWarning: 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(
```

Out[40]:
```
▾ LogisticRegression
LogisticRegression()
```

In [41]:
```
y_test_pred = classifier.predict(X_test_bow)
```

In [42]:
```
from sklearn.metrics import accuracy_score, classification_report

print(accuracy_score(y_test, y_test_pred))

print(classification_report(y_test, y_test_pred))
```

```
0.7395883265191033
              precision    recall  f1-score   support

           1       0.67      0.67      0.67     13085
           2       0.44      0.26      0.33      7401
           3       0.47      0.30      0.37     10672
           4       0.53      0.27      0.35     20264
           5       0.80      0.95      0.87     90438

    accuracy                           0.74    141860
   macro avg       0.58      0.49      0.52    141860
weighted avg       0.70      0.74      0.71    141860
```

In [43]:
```
import joblib
from joblib import dump, load
```

In [44]:
```
joblib.dump(classifier,'logistic_Regression')
```

Out[44]:
```
['logistic_Regression']
```

Decision Tree Classifier

In [45]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train_bow, y_train)
```

Out[45]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [46]:
```
y_test_pred = classifier.predict(X_test_bow)
```

In [47]:
```
from sklearn.metrics import accuracy_score, classification_report

print(accuracy_score(y_test, y_test_pred))
print(classification_report(y_test, y_test_pred))
```

```
0.7526011560693642
              precision    recall  f1-score   support

           1       0.64      0.62      0.63     13085
           2       0.54      0.45      0.49      7401
           3       0.54      0.48      0.51     10672
           4       0.56      0.51      0.53     20264
           5       0.84      0.88      0.86     90438

    accuracy                           0.75    141860
   macro avg       0.62      0.59      0.60    141860
weighted avg       0.74      0.75      0.75    141860
```

In [48]:
```
joblib.dump(classifier,'Decision_Tree')
```

Out[48]:
```
['Decision_Tree']
```

SVC

In [49]:
```
#from sklearn.svm import SVC
#classifier = SVC()
#classifier.fit(X_train_bow, y_train)
```

In [50]:
```
#y_test_pred = classifier.predict(X_test_bow)
#print(accuracy_score(y_test, y_test_pred))

#print(classification_report(y_test, y_test_pred))
```

MODEL DEPLOYMENT

In [56]:
```
model=joblib.load("logistic_Regression")
```

In [66]:
```
x=model.predict(X_test_bow)
```

In [77]:
```
np. unique(x,return_counts=True)
```

Out[77]:
```
(array([1, 2, 3, 4, 5], dtype=int64),
 array([ 12993,   4409,   6902,  10197, 107359], dtype=int64))
```

In [85]:
```
X_test_bow[0]
```

Out[85]:
```
<1x83171 sparse matrix of type '<class 'numpy.int64'>'
	with 23 stored elements in Compressed Sparse Row format>
```

Model Texting

In [145…
```
new_input=input("Enter your Review:")
new_input_pro=preprocess(new_input,'lemma')
df=pd.DataFrame(new_input_pro)
df.columns=["Review"]
new_input_vec=vocab.transform(df)
new_output =model.predict(new_input_vec)
```

```
print(new_input)
print("Score:",new_output)
```

```
Enter your Review:It was amazing, never tried it.
It was amazing, never tried it.
Score: [5]
```

In [143…

```
model2=joblib.load("logistic_Regression_TFIDF")
```

In [144…

```
new_input=input("Enter your Review:")
new_input_pro=preprocess(new_input,'lemma')
df=pd.DataFrame(new_input_pro)
df.columns=["Review"]
new_input_vec=vocab.transform(df)
new_output =model2.predict(new_input_vec)

print(new_input)
print("Score:",new_output)
```

```
Enter your Review:It was a bad food
It was a bad food
Score: [1]
```

In [ ]: