Homework 1

Ankita Samanta

USC ID - 8019258016

Python Version: Python 3.9.12

Used '**Contractions**' Library to perform contractions on the reviews

Used '**sklearn**' Library to import Perceptron, LinaerSVC, LogisticRegression, MultinomialNB, classification_report and precision_recall_fscore_support

Reading Data

1. Read the given dataset by using pandas
2. To keep the reviews and ratings, I have used iloc() function to keep those 2 columns
3. Then assign classes as per the ratings by using apply() method which works as map() function in python, and also used lambda function to define the expression
4. To return 20000 rows of each class, I used sample() method to get specified number of random rows

Data Cleaning (Used different types of python functions to clean the data)

1. Converting the reviews into lower case by using lower() function
2. Removing the extra space by using strip() function
3. Removing the URLs by using re.split() to create the list of the reviews seperating by URLs, as well as lambda function which is defined to put blank space on the place of URLs
4. For the HTML tags, I have used replace() function which is simply replacing the HTML tags to ''
5. To eliminate the non-alphabetical characters, I have used re.sub() which is basically putting blank space everywhere except for the alphabetical characters and space.
6. Lastly, removing all the extra spaces again

Pre-processing (Downloaded nltk 'averaged_perceptron_tagger' and 'punkt' to run the lemmatization step)

1. Removed the stop words by implementing nltk.corpus english stopwords
2. For lemmatization, I have used pos_tag(), word_tokenize() and WordNetLemmatizer, wordnet to perform the lemmatization. pos_tag will help us in processing to mark up the words in text format based on it's a noun, verb, adjective or adverb I have used word_tokenize as it's helping me to convert the review into words otherwise it's getting converted into array of characters after lemmatization

TF-IDF Feature Extraction

Implemented TfidfVectorizer to extract the features along with that used the ngram_range parameter to fix the range i.e., only unigrams, bigrams and trigrams

Models

Used sklearn library to implement all the 4 models, along with that used classification_report and precision_recall_fscore_support to get the scores for each class. By using precision_recall_fscore_support, I'm calculating Precission, Recall and F1 Score for the testing data.

Note - Lemmatization, TF-IDF Feature Extraction and Logistic Regression is taking time

Citations

1. Machinelearningplus.com. (2018). [online] Available at: https://www.machinelearningplus.com/nlp/lemmatization-examples-python/ (https://www.machinelearningplus.com/nlp/lemmatization-examples-python/) [Accessed 25 Jan. 2023].
2. Stack Overflow. (2017). Lemmatization of all pandas cells. [online] Available at: https://stackoverflow.com/questions/47557563/lemmatization-of-all-pandas-cells (https://stackoverflow.com/questions/47557563/lemmatization-of-all-pandas-cells) [Accessed 19 Jan. 2023].
3. Smith, J. (2018). Removing URL from a column in Pandas Dataframe. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/51994254/removing-url-from-a-column-in-pandas-dataframe (https://stackoverflow.com/questions/51994254/removing-url-from-a-column-in-pandas-dataframe) [Accessed 25 Jan. 2023].
4. scikit-learn. (2023). sklearn.metrics.precision_recall_fscore_support. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html) [Accessed 25 Jan. 2023].
5. scikit-learn. (2023). API Reference. [online] Available at: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model (https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model) [Accessed 25 Jan. 2023].
6. collarblind (2015). Python remove stop words from pandas dataframe. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/29523254/python-remove-stop-words-from-pandas-dataframe (https://stackoverflow.com/questions/29523254/python-remove-stop-words-from-pandas-dataframe) [Accessed 25 Jan. 2023].

In [1]:

```
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
[nltk_data]     /Users/ankitasamanta/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

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_Beauty_v1_00.tsv.gz
```

Read Data

Reading the datasets by using pandas

header = 0 as the name of the column is in the first row

Printing the head of the given dataset

In [2]:

```
dataset=pd.read_csv('amazon_reviews_us_Beauty_v1_00.tsv',header=0,on_bad_lines='skip',sep='\t')
dataset.head()
```

0	US	1797882	R3I2DHQBR577SS	B001ANOOOE	2102612	Vitmin C Moisturizing Sunscreen ...	Beauty	5	0.0	0.0	N
1	US	18381298	R1QNE9NQFJC2Y4	B0016J22EQ	106393691	Alba Botanica Sunless Tanning Lotion, 4 Ounce	Beauty	5	0.0	0.0	N
2	US	19242472	R3LIDG2Q4LJBAO	B00HU6UQAG	375449471	Elysee Infusion Skin Therapy Elixir, 2oz.	Beauty	5	0.0	0.0	N
3	US	19551372	R3KSZHPAEVPEAL	B002HWS7RM	255651889	Diane D722 Color, Perm And Conditioner Process...	Beauty	5	0.0	0.0	N
4	US	14802407	RAI2OIG50KZ43	B00SM99KWU	116158747	Biore UV Aqua Rich Watery Essence SPF50+/PA+++	Beauty	5	0.0	0.0	N

Keep Reviews and Ratings

Used iloc() function to keep those 'star_rating' and 'review_body' columns

Printing the head and tail of the dataset which contains only 'star_rating' and 'review_body'

In [3]:

```
df=dataset.iloc[:,[7,13]]
df
```

Out[3]:

	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 ...
...	...	...
5094302	5	After watching my Dad struggle with his scisso...
5094303	3	Like most sound machines, the sounds choices a...
5094304	5	I bought this product because it indicated 30 ...
5094305	5	We have used Oral-B products for 15 years; thi...
5094306	5	I love this toothbrush. It's easy to use, and ...

5094307 rows × 2 columns

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

Dividing the entire dataframe into three classes and then randomly selecting 20000 reviews from ecah class

Printing the dataframe after randomly selecting 20000 reviews from ecah class

In [4]:

```
df['star_rating']=df['star_rating'].apply(lambda x:1 if x in [1,2] else (2 if x==3 else 3))

data=df.groupby('star_rating').sample(n=20000,random_state=0)

data
```

```
/var/folders/s2/pq7zyj816gv48bpm56xcfj3c0000gn/T/ipykernel_82980/1928051500.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retur
ning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
-versus-a-copy)
  df['star_rating']=df['star_rating'].apply(lambda x:1 if x in [1,2] else (2 if x==3 else 3))
```

Out[4]:

	star_rating	review_body
2514046	1	Still smudges. Cant seem to find a product th...
4504842	1	The shears cut reasonably well near the tip. ...
3934587	1	Ace combs used to be the best you could buy. B...
4291138	1	The perfume was discolored and smelt off, very...
1574024	1	i failed to noticed the power of that one.. it...
...	...	...
2971181	3	The only stuff I will ever use on my body, it ...
1981300	3	Item was as described. Thank you!
1041273	3	Definitely too expensive for the price. Bought...
1615062	3	Exactly what I wanted. They are great!
3766002	3	These arrived from Asia and I am very pleased ...

60000 rows × 2 columns

Data Cleaning

Calculating the average length before cleaning

In [5]:

```
len_1=data['review_body'].str.len()
avg_len_1=len_1.mean()
```

Cleaning the data:

In [6]:

```
#remove lower case
data['review_body']=data['review_body'].str.lower()

#remove extra space
data['review_body']=data['review_body'].str.strip()


#remove URL
data['review_body']=data['review_body'].apply(lambda x:re.split('https:\/\/.*',str(x))[0])

#remove HTML texts
data['review_body']=data['review_body'].str.replace(r'<[^<>]*>','',regex=True)

#using contractions
import contractions
data['review_body']=data['review_body'].apply(lambda x: contractions.fix(x))

#remove non-alphabetical characters
data['review_body'] = data['review_body'].apply(lambda x: re.sub(r'[^a-z\s]+', ' ', x))

#again remove extra space
data['review_body'] = data['review_body'].str.strip()

#drop null values
data.dropna()


data
```

	star_rating	review_body
2514046	1	still smudges cannot seem to find a product ...
4504842	1	the shears cut reasonably well near the tip ...
3934587	1	ace combs used to be the best you could buy b...
4291138	1	the perfume was discolored and smelt off very...
1574024	1	i failed to noticed the power of that one it ...
...	...	...
2971181	3	the only stuff i will ever use on my body it ...
1981300	3	item was as described thank you
1041273	3	definitely too expensive for the price bought...
1615062	3	exactly what i wanted they are great
3766002	3	these arrived from asia and i am very pleased ...

Printing the average length of the reviews before and after cleaning

In [7]:

```
len_2=data['review_body'].str.len()
avg_len_2=len_2.mean()

print("Average length of reviews before and after data cleaning - ",avg_len_1, ',', avg_len_2)
```

```
Average length of reviews before and after data cleaning -  280.61780392679754 , 276.8863666666667
```

Pre-processing

Calculating the average length before pre-processing (it'll be the same as the one after cleaning)

In [8]:

```
length_1=data['review_body'].str.len()
avg_length_1=length_1.mean()
```

remove the stop words

Printing the head and tail of the dataframe after removing the stop words

In [9]:

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

Out[9]:

	star_rating	review_body	new
2514046	1	still smudges cannot seem to find a product ...	still smudges cannot seem find product spent f...
4504842	1	the shears cut reasonably well near the tip ...	shears cut reasonably well near tip cut well a...
3934587	1	ace combs used to be the best you could buy b...	ace combs used best could buy longer made hard...
4291138	1	the perfume was discolored and smelt off very...	perfume discolored smelt disappointing meant c...
1574024	1	i failed to noticed the power of that one it ...	failed noticed power one powerful needs time d...
...	...	...	...
2971181	3	the only stuff i will ever use on my body it ...	stuff ever use body amazing leaves skin soft s...
1981300	3	item was as described thank you	item described thank
1041273	3	definitely too expensive for the price bought...	definitely expensive price bought driver seat ...
1615062	3	exactly what i wanted they are great	exactly wanted great
3766002	3	these arrived from asia and i am very pleased ...	arrived asia pleased price service sharpness b...

60000 rows × 3 columns

perform lemmatization

Printing the head and tail of the dataframe after performing the lemmatization

In [10]:

```
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt')

lemmatizer=WordNetLemmatizer()

# Machinelearningplus.com. (2018). [online] Available at: https://www.machinelearningplus.com/nlp/lemmatization-examples-pytho

def get_wordnet_pos(word):
    tag=nltk.pos_tag([word])[0][1][0].upper()
    tag_dict={"J": wordnet.ADJ,"N": wordnet.NOUN,"V": wordnet.VERB,"R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.NOUN)

def lemmatize_text(text):
    return ' '.join([lemmatizer.lemmatize(w,get_wordnet_pos(w)) for w in nltk.word_tokenize(text)])

data['new'] = data.new.apply(lemmatize_text)
data



# Stack Overflow. (2017). Lemmatization of all pandas cells. [online] Available at: https://stackoverflow.com/questions/475575
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /Users/ankitasamanta/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/ankitasamanta/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[10]:

	star_rating	review_body	new
2514046	1	still smudges cannot seem to find a product ...	still smudge can not seem find product spent f...
4504842	1	the shears cut reasonably well near the tip ...	shear cut reasonably well near tip cut well aw...
3934587	1	ace combs used to be the best you could buy b...	ace comb use best could buy longer make hard r...
4291138	1	the perfume was discolored and smelt off very...	perfume discolor smelt disappoint meant christ...
1574024	1	i failed to noticed the power of that one it ...	fail notice power one powerful need time dry hair
...	...	...	...
2971181	3	the only stuff i will ever use on my body it ...	stuff ever use body amaze leaf skin soft smooth
1981300	3	item was as described thank you	item described thank
1041273	3	definitely too expensive for the price bought...	definitely expensive price bought driver seat ...
1615062	3	exactly what i wanted they are great	exactly want great
3766002	3	these arrived from asia and i am very pleased ...	arrive asia pleased price service sharpness bl...

60000 rows × 3 columns

Printing the average length of the reviews before and after pre-processing

In [11]:

```
length_2=data['new'].str.len()
avg_length_2=length_2.mean()

print("Average length of reviews before and after pre-processing - ",avg_length_1, ',', avg_length_2)
```

```
Average length of reviews before and after pre-processing -  276.8863666666667 , 157.07915
```

TF-IDF Feature Extraction

Creating the dataframe after the TF-IDF features are extracted, and then spliting it into 80% training dataset and 20% testing dataset

Printing the shape of the new dataframe that is 'df_new'

In [12]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
v = TfidfVectorizer(ngram_range=(1,3))

x = v.fit_transform(data['new'])

#creating the new dataframe for training and testing data
df_new=pd.DataFrame(data=x.toarray(),columns=v.get_feature_names_out())


from sklearn.model_selection import train_test_split
print(df_new.shape)

#spliting the datasets into 80% training dataset and 20% testing dataset
X_train, X_test, y_train, y_test = train_test_split(x, data['star_rating'], test_size=0.2, random_state=42,shuffle="false")
```

```
(60000, 1856971)
```

Perceptron

Printing the classification report as well as the respective precision, recall af1 score and the average for each class for the Perceptron model

In [13]:

```
from sklearn.linear_model import Perceptron
from sklearn.metrics import classification_report
from sklearn.metrics import precision_recall_fscore_support as score

p=Perceptron()
p.fit(X_train,y_train)
y_pred_prec=p.predict(X_test)



precision_prec,recall_prec,f1score_prec,support_prec=score(y_test, y_pred_prec)
print("Perceptron")
b=0
for i,j,k in zip(precision_prec, recall_prec, f1score_prec):
    b+=1
    print('Class',b,': Precision, Recall, F1 Score - ' ,i,',',j,',',k )

precision_prec_avg,recall_prec_avg,f1score_prec_avg,support_prec_avg=score(y_test, y_pred_prec, average='weighted')
print('Average : Precision, Recall, F1 Score - ',precision_prec_avg,',',recall_prec_avg,',',f1score_prec_avg)

print(classification_report(y_test, y_pred_prec))
```

```
Perceptron
Class 1 : Precision, Recall, F1 Score -  0.6626943005181347 , 0.6454706030784759 , 0.6539690655758661
Class 2 : Precision, Recall, F1 Score -  0.562467997951869 , 0.5473343298455406 , 0.5547979797979798
Class 3 : Precision, Recall, F1 Score -  0.6896551724137931 , 0.7258264976385782 , 0.7072786726413951
Average : Precision, Recall, F1 Score -  0.6382072346127384 , 0.6395833333333333 , 0.6386683831518818
              precision    recall  f1-score   support

           1       0.66      0.65      0.65      3963
           2       0.56      0.55      0.55      4014
           3       0.69      0.73      0.71      4023

    accuracy                           0.64     12000
   macro avg       0.64      0.64      0.64     12000
weighted avg       0.64      0.64      0.64     12000
```

SVM

Printing the classification report as well as the respective precision, recall, f1 score and the average for each class for the SVM model

In [14]:

```
from sklearn.svm import LinearSVC

svm=LinearSVC()
svm.fit(X_train, y_train)
y_pred_svm=svm.predict(X_test)



precision_svm,recall_svm,f1score_svm,support_svm=score(y_test, y_pred_svm)
print("SVM")
b=0
for i,j,k in zip(precision_svm, recall_svm, f1score_svm):
    b+=1
    print('Class',b,': Precision, Recall, F1 Score - ' ,i,',',j,',',k )

precision_svm_avg,recall_svm_avg,f1score_svm_avg,support_svm_avg=score(y_test, y_pred_svm, average='weighted')
print('Average : Precision, Recall, F1 Score - ',precision_svm_avg,',',recall_svm_avg,',',f1score_svm_avg)

print(classification_report(y_test, y_pred_svm))
```

```
SVM
Class 1 : Precision, Recall, F1 Score -  0.6676029962546817 , 0.7196568256371436 , 0.6926533090467517
Class 2 : Precision, Recall, F1 Score -  0.6029016657710908 , 0.5590433482810164 , 0.5801447776628749
Class 3 : Precision, Recall, F1 Score -  0.7416375436844733 , 0.7385036042754164 , 0.7400672561962884
Average : Precision, Recall, F1 Score -  0.6707804832337582 , 0.67225 , 0.6709147310807271
              precision    recall  f1-score   support

           1       0.67      0.72      0.69      3963
           2       0.60      0.56      0.58      4014
           3       0.74      0.74      0.74      4023

    accuracy                           0.67     12000
   macro avg       0.67      0.67      0.67     12000
weighted avg       0.67      0.67      0.67     12000
```

Logistic Regression

Printing the classification report as well as the respective precision, recall, f1 score and the average for each class for the Logistic Regression model

In [15]:

```
from sklearn.linear_model import LogisticRegression

lr=LogisticRegression()
lr.fit(X_train, y_train)
y_pred_logreg=lr.predict(X_test)

print("Logistic Regression")
precision_logreg,recall_logreg,f1score_logreg,support_logreg=score(y_test, y_pred_logreg)
c=0
for i,j,k in zip(precision_logreg, recall_logreg, f1score_logreg):
    c+=1
    print('Class',c,': Precision, Recall, F1 Score - ' ,i,',',j,',',k )
precision_logreg_avg,recall_logreg_avg,f1score_logreg_avg,support_logreg_avg=score(y_test, y_pred_logreg, average='weighted')
print('Average : Precision, Recall, F1 Score - ',precision_logreg_avg,',',recall_logreg_avg,',',f1score_logreg_avg)

print(classification_report(y_test, y_pred_logreg))
```

```
/Users/ankitasamanta/opt/miniconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:458: Convergenc
eWarning: 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 (https://scikit-learn.org/stable/modules/preproces
sing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stabl
e/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(

Logistic Regression
Class 1 : Precision, Recall, F1 Score -  0.6714851946840755 , 0.7267221801665406 , 0.6980126030053321
Class 2 : Precision, Recall, F1 Score -  0.6077376377145786 , 0.5909317389138017 , 0.5992168750789441
Class 3 : Precision, Recall, F1 Score -  0.7620798319327731 , 0.7213522247079294 , 0.7411569403652151
Average : Precision, Recall, F1 Score -  0.6805334890154047 , 0.6795 , 0.6794295711138562
              precision    recall  f1-score   support

           1       0.67      0.73      0.70      3963
           2       0.61      0.59      0.60      4014
           3       0.76      0.72      0.74      4023

    accuracy                           0.68     12000
   macro avg       0.68      0.68      0.68     12000
weighted avg       0.68      0.68      0.68     12000
```

Naive Bayes

Printing the classification report as well as the respective precision, recall, f1 score and the average for each class for the Naive Bayes model

In [16]:

```
from sklearn.naive_bayes import MultinomialNB

nb=MultinomialNB()
nb.fit(X_train, y_train)
y_pred_NB=nb.predict(X_test)

print("Naive Bayes")
precision_nb,recall_nb,f1score_nb,support_nb=score(y_test, y_pred_NB)
d=0
for i,j,k in zip(precision_nb, recall_nb, f1score_nb):
    d+=1
    print('Class',d,': Precision, Recall, F1 Score - ' ,i,',',j,',',k )
precision_nb_avg,recall_nb_avg,f1score_nb_avg,support_nb_avg=score(y_test, y_pred_NB, average='weighted')
print('Average : Precision, Recall, F1 Score - ',precision_nb_avg,',',recall_nb_avg,',',f1score_nb_avg)

print(classification_report(y_test, y_pred_NB))
```

```
Naive Bayes
Class 1 : Precision, Recall, F1 Score -  0.6737247353224254 , 0.7065354529396921 , 0.6897401157778051
Class 2 : Precision, Recall, F1 Score -  0.5823461091753774 , 0.6245640259093174 , 0.6027166726770045
Class 3 : Precision, Recall, F1 Score -  0.7838372421588019 , 0.6895351727566492 , 0.7336683417085426
Average : Precision, Recall, F1 Score -  0.6800738027931331 , 0.6734166666666667 , 0.6753577118038671
              precision    recall  f1-score   support

           1       0.67      0.71      0.69      3963
           2       0.58      0.62      0.60      4014
           3       0.78      0.69      0.73      4023

    accuracy                           0.67     12000
   macro avg       0.68      0.67      0.68     12000
weighted avg       0.68      0.67      0.68     12000
```