CSCI-544 Applied NLP - HW 1

MRINAL KADAM
USC ID: 3135945534

Three sample reviews along with their corresponding ratings:

```
df = df[["review_body","star_rating"]]
df.sample(n=3, random_state=100)
```

	review_body	star_rating
3433978	I loved it even though the pokemon sticker fel...	5
3020629	We chose this particular model mainly because ...	4
4629921	We loved our oven until the dome broke. The ma...	3

To obtain any three sample reviews along with their corresponding ratings from the data frame, I used the **sample** function of the **pandas** library and set n=3 with a random state of 100 to reciprocate the same results every time that the jupyter notebook is run.

Statistics of the ratings:

```
df['star_rating'].value_counts()
```

Rating	Number of Reviews
5	3128564
4	732471
1	427306
3	349929
2	242196

To get the statistics of all the different ratings, I used the **value_counts** function of the **pandas** li brary. This function listed the number of reviews falling under each distinct rating.

Number of reviews for each of the three classes:

```
df[((df['star_rating']==4.0) | (df['star_rating']==5.0))]['star_rating'].c
ount()
```

```
df[((df['star_rating']==1.0) | (df['star_rating']==2.0))]['star_rating'].c
ount()
df[df['star_rating']==3.0]['star_rating'].count()
```

Class	Number of Reviews
1(Positive Reviews)	3860839
0(Negative Reviews)	669463
Neutral Reviews	349921

I filtered the data frame by setting conditions on the rating column – (rating = 4 or 5(positive reviews-class 1), 1 or 2(negative reviews-class 0), 3(neutral reviews)) and used the **count** function of the **pandas** library to get the number of reviews falling under each of the three classes.

Average length of the reviews in terms of character length in the dataset before and after cleaning:

```
len_before_data_cleaning = df['review_body'].apply(len).mean()
```

Before cleaning: 322.24942

```
len_after_data_cleaning = df['review_body'].apply(len).mean()
```

After cleaning: 307.713255

To find the average length of the reviews in terms of character length in the dataset before and af ter cleaning, I used the **len** function and the **apply** function of the **pandas** library to apply the le n function on the review column of the data frame. Once I calculated the length of all the records in the review column in terms of characters, I used the **mean** function of the **pandas** library final ly to calculate the average length of all the records in that column.

Three sample reviews before (data cleaning + preprocessing):

```
df['review_body'].sample(n=3, random_state=100).values
```

```
180481    returned.  the handles are huge and stick way out.
8971      hard cold facts: total weight 15 oz. (digital postage scales), 8
inches long by 3.5 inches wide and 0.10 inches thick (almost one eight of
an inch) with an out-of-the cardboard sheath polish which is a mirror fini
sh.  metal is slightly magnetic (indicates stainless steel with some carbo
n steel).  [stainless steel was never intended to hold an edge, but you ca
n not find a true carbon steel blade anymore.  they rust if you do not tak
e care of them and turn a dark grey if you do...just not pretty.]  the cle
aver has a full tang with two wooden side pieces for a handle.  the wood i
s held to the full tang with three brass rivets.  should the rivets become
loose, as others have commented, i plan to use a hammer and punch set to t
```

```
ighten them.  that action failing, i would drill out the rivets and replac
e them with machine screws.<br />the wood is finished with a light reddish
tan stain which i find &#34;cheap&#34; looking. (reminds me of the stock c
olor on a chinese sks rifle copy of an ak-47.)<br />all that said, i am pl
eased with my purchase and have a fix for the small issues (like the two n
icks in the edge).  at these prices i do not mind taking a little time to
make this my own.
77027     great scale! great price. i really appreciate the free shipping!
```

To obtain any three sample reviews before (data cleaning + preprocessing) from the data frame, I used the **sample** function of the **pandas** library and set n=3 with a random state of 100 to reciprocate the same results every time that the jupyter notebook is run. **values** function of the **pandas** library was used at the end to give the entire record.

Those sample reviews after (data cleaning + preprocessing):

```
df['review_body'].sample(n=3, random_state=100).values
```

```
180481                          return handle huge stick way
8971      hard cold fact total weight oz digital postage scale inch long i
nch wide inch thick almost one eight inch cardboard sheath polish mirror f
inish metal slightly magnetic indicates stainless steel carbon steel stain
less steel never intend hold edge find true carbon steel blade anymore rus
t take care turn dark grey pretty cleaver full tang two wooden side piece
handle wood held full tang three brass rivet rivet become loose others com
ment plan use hammer punch set tighten action fail would drill rivet repla
ce machine screw wood finish light reddish tan stain find cheap look remin
ds stock color chinese sks rifle copy ak say pleased purchase fix small is
sue like two nick edge price mind take little time make
77027     great scale great price really appreciate free shipping
```

To obtain those three sample reviews after (data cleaning + preprocessing) from the data frame, I used the **sample** function of the **pandas** library and set n=3 with a random state of 100 again to reciprocate the same results every time that the jupyter notebook is run. **values** function of the **pandas** library was used at the end to give the entire record.

Average length of the reviews in terms of character length in the dataset before and after pre-processing:

```
len_before_pre_processing = df['review_body'].apply(len).mean()
```

Before cleaning: 307.713255

```
len_after_pre_processing = df['review_body'].apply(len).mean()
```

After cleaning: 183.03818

To find the average length of the reviews in terms of character length in the dataset before and af ter pre-processing, I used the **len** function and the **apply** function of the **pandas** library to apply the len function on the review column of the data frame. Once I calculated the length of all the re cords in the review column in terms of characters, I used the **mean** function of the **pandas** librar y finally to calculate the average length of all the records in that column.

Performance Metrics for different models on training and test sets:

```
----------------------Perceptron----------------------
----------------------Train----------------------


Accuracy of Perceptron Model: 0.8917875
Precision of Perceptron Model: 0.9285792364659907
Recall of Perceptron Model: 0.8489993253879019
F1-score of Perceptron Model: 0.8870079356792314

----------------------Test----------------------


Accuracy of Perceptron Model: 0.850325
Precision of Perceptron Model: 0.8881663053749097
Recall of Perceptron Model: 0.800791821188734
F1-score of Perceptron Model: 0.842219001185927



----------------------SVM----------------------
----------------------Train----------------------


Accuracy of SVM Model: 0.8802625
Precision of SVM Model: 0.881522883352549
Recall of SVM Model: 0.8787697074182346
F1-score of SVM Model: 0.8801441423405614

----------------------Test----------------------


Accuracy of SVM Model: 0.814075
Precision of SVM Model: 0.8171912219350261
Recall of SVM Model: 0.8080585346296482
F1-score of SVM Model: 0.8125992188484314



----------------------Logistic Regression----------------------
----------------------Train----------------------


Accuracy of Logistic Regression Model: 0.911
```

Precision of Logistic Regression Model: 0.9134997235208365
Recall of Logistic Regression Model: 0.9080903480498713
F1-score of Logistic Regression Model: 0.9107870039719831

----------Test----------

Accuracy of Logistic Regression Model: 0.896075
Precision of Logistic Regression Model: 0.8996205413609917
Recall of Logistic Regression Model: 0.891099528916508
F1-score of Logistic Regression Model: 0.8953397618268336

----------Naive Bayes----------
----------Train----------

Accuracy of Multinomial Naive Bayes Model: 0.88308125
Precision of Multinomial Naive Bayes Model: 0.89005328822699
Recall of Multinomial Naive Bayes Model: 0.8742972790645379
F1-score of Multinomial Naive Bayes Model: 0.8821049314636837

----------Test----------

Accuracy of Multinomial Naive Bayes Model: 0.8676
Precision of Multinomial Naive Bayes Model: 0.873852275045909
Recall of Multinomial Naive Bayes Model: 0.8585246065951689
F1-score of Multinomial Naive Bayes Model: 0.8661206329945903

HW1-CSCI544

September 9, 2021

[1]:
```
# import required libraries and methods from them

from platform import python_version

import pandas as pd
import numpy as np

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

import re

from bs4 import BeautifulSoup

import contractions

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Perceptron, LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/mrinalkadam/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/mrinalkadam/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /Users/mrinalkadam/nltk_data…
```

```
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

[2]:
```
# check the python version being used by the jupyter notebook

python_version()
```

[2]:
```
'3.8.5'
```

0.1 Read Data

[3]:
```
# read the input dataset into a dataframe

df = pd.read_csv("data.tsv", sep='\t', quoting=3)
df
```

[3]:

	marketplace	customer_id	review_id	product_id	product_parent	product_title	product_category	star_rating	helpful_votes	total_votes	vine	verified_purchase
0	US	37000337	R3DT59XH7HXR9K	B00303FI0G	529320574	Arthur Court Paper Towel Holder	Kitchen	5	0	0	N	Y
1	US	15272914	R1LFS11BNASSU8	B00JCZKZN6	274237558	Olde Thompson Bavaria Glass Salt and Pepper Mi…	Kitchen	5	0	1	N	Y
2	US	36137863	R296RT05AG0AF6	B00JLIKA5C	544675303	Progressive International PL8 Professional Man…	Kitchen	5	0	0	N	Y
3	US	43311049	R3V37XDZ7ZCI3L	B000GBNB8G	491599489	Zyliss Jumbo Garlic Press	Kitchen	5	0	1	N	Y
4	US	13763148	R14GU232NQFYX2	B00VJ5KX9S	353790155	1 X Premier Pizza Cutter - Stainless Steel 14"…	Kitchen	5	0	0	N	Y
…	…	…	…	…	…	…	…	…	…	…	…	…
4880461	US	51094108	R22DLC2P26MUMR	B00004SBGS	732420532	Le Creuset Enameled Cast-Iron 6-3/4-Quart Oval…	Kitchen	5	0	0	N	Y
4880462	US	50562512	R1N6KLTENLQOMT	B00004SBIA	261705371	Le Creuset Enameled Cast-Iron 2-Quart Heart Ca…	Kitchen	5	0	0	N	Y
4880463	US	52469742	R10TW4QXDV8KJC	B00004SPEF	191184892	Krups 358-70 La Glaciere Ice Cream Maker	Kitchen	5	0	0	N	Y
4880464	US	51865238	R41RL2U1FSQ4V	B00004RHR6	912491903	Hoffritz Stainless-Steel Manual Can Opener	Kitchen	5	0	0	N	Y
4880465	US	52900320	R1NHMPKSJG2E37	B0000021VO	41913389	Tammy Rogers	Kitchen	5	0	0	N	Y

```
…              …             …           …  …              …
4880461            4            30          41    N              N
4880462            5            84          92    N              N
4880463            4            55          60    N              N
4880464            4            30          42    N              N
4880465            5             5           5    N              N

                                 review_headline  \
0                Beautiful. Looks great on counter
1                             Awesome & Self-ness
2                Fabulous and worth every penny
3                                      Five Stars
4                                  Better than sex
…                                              …
4880461            Not as sturdy as you'd think.
4880462                    A Sweetheart of A Pan
4880463                  Ice Cream Like a Dream
4880464            Opens anything and everything
4880465  The more you listen, the more you hear…

                                          review_body review_date
0                   Beautiful.  Looks great on counter.  2015-08-31
1        I personally have 5 days sets and have also bo…  2015-08-31
2        Fabulous and worth every penny. Used for clean…  2015-08-31
3        A must if you love garlic on tomato marinara s…  2015-08-31
4        Worth every penny! Buy one now and be a pizza …  2015-08-31
…                                                    …           …
4880461  After a month of heavy use, primarily as a chi…  2000-04-28
4880462  I've used my Le Creuset enameled cast iron coo…  2000-04-28
4880463  According to my wife, this is \\"the best birt…  2000-04-28
4880464  Hoffritz has a name of producing a trendy and …  2000-04-24
4880465  OK. I was late to snap to the Dead Reckoners. …  2000-01-20

[4880466 rows x 15 columns]
```

0.2 Keep Reviews and Ratings

[4]:
```
# keep only reviews and ratings columns

df = df[["review_body","star_rating"]]
df.sample(n=3, random_state=100)
```

[4]:
```
                                          review_body  star_rating
3433978  I loved it even though the pokemon sticker fel…            5
3020629  We chose this particular model mainly because …            4
4629921  We loved our oven until the dome broke. The ma…            3
```

[5]:
```
# find out the number of reviews falling under each distinct rating

df['star_rating'].value_counts()
```

[5]:
```
5    3128564
4     732471
1     427306
3     349929
2     242196
Name: star_rating, dtype: int64
```

1 Labelling Reviews:

1.1 The reviews with rating 4,5 are labelled to be 1 and 1,2 are labelled as 0. Discard the reviews with rating 3'

[6]:
```
# check for null values in the reviews column

df['review_body'].isnull().sum()
```

[6]:
```
243
```

[7]:
```
# check for null values in the ratings column

df['star_rating'].isnull().sum()
```

[7]:
```
0
```

[8]:
```
# drop null value records from the dataframe

df.dropna(inplace=True)
```

```
<ipython-input-8-ba0c96652bb5>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.dropna(inplace=True)
```

[9]:
```
# find out the number of reviews falling under distinct ratings

print("Positive, Negative, Neutral Reviews Count:")
print(df[((df['star_rating']==4.0) | (df['star_rating']==5.0))]['star_rating'].
 ↪count(),",",df[((df['star_rating']==1.0) | (df['star_rating']==2.
 ↪0))]['star_rating'].count(),",",df[df['star_rating']==3.0]['star_rating'].
 ↪count())
```

```
Positive, Negative, Neutral Reviews Count:
3860839 , 669463 , 349921
```

[10]:
```
# label reviews falling under ratings 4 and 5 as 1, under ratings 1 and 2 as 0,␣
↪and remove the rewiews with rating 3

df['class'] = np.where(((df['star_rating']==4.0) | (df['star_rating']==5.
↪0)),1,0)
df = df[df['star_rating']!=3.0]

# drop the rating column once you have the label('class') column

df.drop(['star_rating'],axis=1,inplace=True)
df
```

```
<ipython-input-10-feda2984de6f>:3: 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#returning-a-view-versus-a-copy
  df['class'] = np.where(((df['star_rating']==4.0) |
(df['star_rating']==5.0)),1,0)
```

[10]:

	review_body	class
0	Beautiful. Looks great on counter.	1
1	I personally have 5 days sets and have also bo…	1
2	Fabulous and worth every penny. Used for clean…	1
3	A must if you love garlic on tomato marinara s…	1
4	Worth every penny! Buy one now and be a pizza …	1
…	…	…
4880461	After a month of heavy use, primarily as a chi…	1
4880462	I've used my Le Creuset enameled cast iron coo…	1
4880463	According to my wife, this is \\"the best birt…	1
4880464	Hoffritz has a name of producing a trendy and …	1
4880465	OK. I was late to snap to the Dead Reckoners. …	1

[4530302 rows x 2 columns]

We select 200000 reviews randomly with 100,000 positive and 100,000 negative reviews.

[11]:
```
# select a total of 200000 reviews randomly with 100000 positive and 100000␣
↪negative reviews

# find out classes with label 1(positive) and with label 0(negative)

df_positive = df[df['class']==1]
df_negative = df[df['class']==0]
```

```
# select 100000 records randomly from both the positive and negative classes

df_positive = df_positive.sample(n=100000, random_state=100)
df_negative = df_negative.sample(n=100000, random_state=100)

# concat the above records together to get a sample of 200000 reviews␣
 ↪consisting of 100000 random positive and 100000 random negative reviews

df = pd.concat([df_positive,df_negative]).reset_index()
df.drop(['index'],axis=1,inplace=True)
df
```

[11]:

	review_body	class
0	Works very well, had to figure out about turni…	1
1	I have no idea how or why this thing works, bu…	1
2	Got this as s gift they loved it!!	1
3	It is what it is!	1
4	sunshine daydream. no hard liquor just cold c…	1
…	…	…
199995	This is the second one of these I have purchas…	0
199996	Not what I expected very flimsy.	0
199997	too cheap	0
199998	I was expecting much better quality from Corel…	0
199999	Based on the video displayed on the sale site …	0

[200000 rows x 2 columns]

2 Data Cleaning

[12]:
```
# convert the reviews column to string type

df['review_body'] = df['review_body'].astype(str)

# find out the average length of the reviews in terms of character length in␣
 ↪the dataset before cleaning

len_before_data_cleaning = df['review_body'].apply(len).mean()
```

2.1 Convert the all reviews into the lower case.

[13]:
```
# convert the reviews column to lower case

df['review_body'] = df['review_body'].str.lower()
df
```

[13]:
```
                                              review_body  class
0       works very well, had to figure out about turni…      1
1       i have no idea how or why this thing works, bu…      1
2                      got this as s gift they loved it!!      1
3                                      it is what it is!      1
4       sunshine daydream.  no hard liquor just cold c…      1
…                                                     …    …
199995  this is the second one of these i have purchas…      0
199996                  not what i expected very flimsy.      0
199997                                        too cheap      0
199998  i was expecting much better quality from corel…      0
199999  based on the video displayed on the sale site …      0

[200000 rows x 2 columns]
```

[14]:
```
# find out three sample reviews before (data cleaning + pre-processing)

df['review_body'].sample(n=3, random_state=100).values
```

[14]:
```
array(['returned.  the handles are huge and stick way out.',
       'hard cold facts: total weight 15 oz. (digital postage scales), 8 inches
long by 3.5 inches wide and 0.10 inches thick (almost one eight of an inch) with
an out-of-the cardboard sheath polish which is a mirror finish.  metal is
slightly magnetic (indicates stainless steel with some carbon steel).
[stainless steel was never intended to hold an edge, but you can not find a true
carbon steel blade anymore.  they rust if you do not take care of them and turn
a dark grey if you do…just not pretty.]  the cleaver has a full tang with two
wooden side pieces for a handle.  the wood is held to the full tang with three
brass rivets.  should the rivets become loose, as others have commented, i plan
to use a hammer and punch set to tighten them.  that action failing, i would
drill out the rivets and replace them with machine screws.<br />the wood is
finished with a light reddish tan stain which i find &#34;cheap&#34; looking.
(reminds me of the stock color on a chinese sks rifle copy of an ak-47.)<br
/>all that said, i am pleased with my purchase and have a fix for the small
issues (like the two nicks in the edge).  at these prices i do not mind taking a
little time to make this my own.',
       'great scale! great price. i really appreciate the free shipping!'],
      dtype=object)
```

2.2 remove the HTML and URLs from the reviews

[15]:
```
# using BeautifulSoup, remove HTML tags from the reviews column

# function to remove HTML tags
def remove_html(string):

    # parse through html content
```

```
    bs = BeautifulSoup(string, "html.parser")

    for text in bs(['style', 'script']):
        # remove the tags
        text.decompose()

    # return data by retrieving the tag content
    return ' '.join(bs.stripped_strings)

# apply the remove_html function to the reviews column

df['review_body']=df['review_body'].apply(lambda x : remove_html(x))
df
```

```
/opt/anaconda3/lib/python3.8/site-packages/bs4/__init__.py:417:
MarkupResemblesLocatorWarning: "http://www.amazon.com/review/create-
review/ref=cm_cr_ryp_old_pipeline?ie=utf8&asin=b00kx9xfcs&channel=ryp-force-old-
pipeline&forceoldpipeline=1" looks like a URL. Beautiful Soup is not an HTTP
client. You should probably use an HTTP client like requests to get the document
behind the URL, and feed that document to Beautiful Soup.
  warnings.warn(
/opt/anaconda3/lib/python3.8/site-packages/bs4/__init__.py:332:
MarkupResemblesLocatorWarning: "." looks like a filename, not markup. You should
probably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
```

[15]:

	review_body	class
0	works very well, had to figure out about turni…	1
1	i have no idea how or why this thing works, bu…	1
2	got this as s gift they loved it!!	1
3	it is what it is!	1
4	sunshine daydream. no hard liquor just cold c…	1
…	…	…
199995	this is the second one of these i have purchas…	0
199996	not what i expected very flimsy.	0
199997	too cheap	0
199998	i was expecting much better quality from corel…	0
199999	based on the video displayed on the sale site …	0

[200000 rows x 2 columns]

[16]:
```
# using RegEx, remove URLs from the reviews column

# function to remove URLS
def remove_url(string):
    result = re.sub(r'^https?:\/\/.*[\r\n]*',r' ', string, flags=re.MULTILINE)
    return result
```

```
# apply the remove_url function to the reviews column

df['review_body']=df['review_body'].apply(lambda x : remove_url(x))
df
```

[16]:

	review_body	class
0	works very well, had to figure out about turni…	1
1	i have no idea how or why this thing works, bu…	1
2	got this as s gift they loved it!!	1
3	it is what it is!	1
4	sunshine daydream. no hard liquor just cold c…	1
…	…	…
199995	this is the second one of these i have purchas…	0
199996	not what i expected very flimsy.	0
199997	too cheap	0
199998	i was expecting much better quality from corel…	0
199999	based on the video displayed on the sale site …	0

[200000 rows x 2 columns]

2.3 remove non-alphabetical characters

[17]:
```
# using RegEx, remove the characters apart from alphabets and single␣
↪apostrophe(required for contractions later) from the reviews column and␣
↪replace them with a single space

df['review_body'] = df['review_body'].replace(r"[^a-zA-Z' ]\s?"," ",regex=True)

# replace the single apostrophe with no space

df['review_body'] = df['review_body'].replace("'","",regex=True)
df
```

[17]:

	review_body	class
0	works very well had to figure out about turnin…	1
1	i have no idea how or why this thing works but…	1
2	got this as s gift they loved it	1
3	it is what it is	1
4	sunshine daydream no hard liquor just cold co…	1
…	…	…
199995	this is the second one of these i have purchas…	0
199996	not what i expected very flimsy	0
199997	too cheap	0
199998	i was expecting much better quality from corel…	0
199999	based on the video displayed on the sale site …	0

[200000 rows x 2 columns]

2.4 Remove the extra spaces between the words

[18]:
```
# using RegEx, remove the extra spaces between words from the reviews column

df['review_body'] = df['review_body'].replace('\s+', ' ', regex=True)
df
```

[18]:

	review_body	class
0	works very well had to figure out about turnin…	1
1	i have no idea how or why this thing works but…	1
2	got this as s gift they loved it	1
3	it is what it is	1
4	sunshine daydream no hard liquor just cold coffee	1
…	…	…
199995	this is the second one of these i have purchas…	0
199996	not what i expected very flimsy	0
199997	too cheap	0
199998	i was expecting much better quality from corel…	0
199999	based on the video displayed on the sale site …	0

[200000 rows x 2 columns]

2.5 perform contractions on the reviews.

[19]:
```
# using the contractions library, perform contractions on the reviews

df['review_body'] = df['review_body'].apply(lambda x: [contractions.fix(word)␣
 ↪for word in x.split()])
df['review_body'] = [' '.join(map(str, d)) for d in df['review_body']]
df
```

[19]:

	review_body	class
0	works very well had to figure out about turnin…	1
1	i have no idea how or why this thing works but…	1
2	got this as s gift they loved it	1
3	it is what it is	1
4	sunshine daydream no hard liquor just cold coffee	1
…	…	…
199995	this is the second one of these i have purchas…	0
199996	not what i expected very flimsy	0
199997	too cheap	0
199998	i was expecting much better quality from corel…	0
199999	based on the video displayed on the sale site …	0

[200000 rows x 2 columns]

[20]:
```
# find out the average length of the reviews in terms of character length in the dataset after cleaning

len_after_data_cleaning = df['review_body'].apply(len).mean()
```

[21]:
```
# print the average length of the reviews in terms of character length in the dataset before and after cleaning

print("Average length of the reviews in terms of character length in the dataset before and after cleaning:")
print(len_before_data_cleaning,",",len_after_data_cleaning)
```

```
Average length of the reviews in terms of character length in the dataset before
and after cleaning:
322.24942 , 307.713255
```

3 Pre-processing

[22]:
```
# find out the average length of the reviews in terms of character length in the dataset before pre-processing

len_before_pre_processing = df['review_body'].apply(len).mean()
```

3.1 remove the stop words

[23]:
```
# remove all general stop words from the reviews column

stop_words = stopwords.words('english')
df['review_body'] = df['review_body'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_words)]))
df
```

[23]:

	review_body	class
0	works well figure turning glass bowl first see…	1
1	idea thing works gives clean precise edge dip …	1
2	got gift loved	1
3		1
4	sunshine daydream hard liquor cold coffee	1
…	…	…
199995	second one purchased first one stopped brewing…	0
199996	expected flimsy	0
199997	cheap	0
199998	expecting much better quality corelle glasses …	0
199999	based video displayed sale site purchased repl…	0

[200000 rows x 2 columns]

3.2 perform lemmatization

[24]:
```
# perform lemmatization with POS tagging

whitespace_tokenizer = nltk.tokenize.WhitespaceTokenizer()
wordnet_lemmatizer = nltk.stem.WordNetLemmatizer()

# funtion to return a POS form of a word
def pos(word):
    """Map POS tag to first character lemmatize() accepts"""
    pos_tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dictionary = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}

    return tag_dictionary.get(pos_tag, wordnet.NOUN)

# function to lemmatize the text
def lemmatize_text(string):
    return [wordnet_lemmatizer.lemmatize(w,pos(w)) for w in␣
 ↪whitespace_tokenizer.tokenize(string)]

df['review_body'] = df['review_body'].apply(lemmatize_text)
df['review_body'] = [' '.join(map(str, l)) for l in df['review_body']]
df
```

[24]:

	review_body	class
0	work well figure turn glass bowl first seem wo…	1
1	idea thing work give clean precise edge dip to…	1
2	get gift love	1
3		1
4	sunshine daydream hard liquor cold coffee	1
…	…	…
199995	second one purchase first one stop brewing cof…	0
199996	expect flimsy	0
199997	cheap	0
199998	expect much well quality corelle glass thin ch…	0
199999	base video displayed sale site purchase replac…	0

[200000 rows x 2 columns]

[25]:
```
# find out the three sample reviews after (data cleaning + pre-processing)

df['review_body'].sample(n=3, random_state=100).values
```

[25]:
```
array(['return handle huge stick way',
       'hard cold fact total weight oz digital postage scale inch long inch wide
inch thick almost one eight inch cardboard sheath polish mirror finish metal
slightly magnetic indicates stainless steel carbon steel stainless steel never
intend hold edge find true carbon steel blade anymore rust take care turn dark
grey pretty cleaver full tang two wooden side piece handle wood held full tang
three brass rivet rivet become loose others comment plan use hammer punch set
tighten action fail would drill rivet replace machine screw wood finish light
reddish tan stain find cheap look reminds stock color chinese sks rifle copy ak
say pleased purchase fix small issue like two nick edge price mind take little
time make',
       'great scale great price really appreciate free shipping'],
      dtype=object)
```

[26]:
```
# find out the average length of the reviews in terms of character length in␣
↪the dataset after pre-processing

len_after_pre_processing = df['review_body'].apply(len).mean()
```

[27]:
```
# print the average length of the reviews in terms of character length in the␣
↪dataset before and after pre-processing

print("Average length of the reviews in terms of character length in the␣
↪dataset before and after pre-processing:")
print(len_before_pre_processing,",",len_after_pre_processing)
```

```
Average length of the reviews in terms of character length in the dataset before
and after pre-processing:
307.713255 , 183.03818
```

4 TF-IDF Feature Extraction

[28]:
```
# transform the features into tf-idf features using TfidfVectorizer

vectorizer = TfidfVectorizer()

x = df['review_body']
y = df['class']

x_final = vectorizer.fit_transform(x)
```

[29]:
```
# Split the dataset into 80% training dataset and 20% testing dataset

x_train, x_test, y_train, y_test = train_test_split(x_final, y, test_size=0.20,␣
↪random_state=100)
```

5 Perceptron

[30]:
```
# train a Perceptron model on the training dataset

perceptron = Perceptron(n_jobs=-1, random_state=100)
perceptron.fit(x_train,y_train)
```

[30]:
```
Perceptron(n_jobs=-1, random_state=100)
```

[31]:
```
# predict the labels of train values

y_train_pred = perceptron.predict(x_train)

# find the accuracy, precision, recall and f1_score of the Perceptron model on␣
 ↪the training set

print("-----------------------Perceptron-----------------------")
print("-----------------------Train-----------------------")
print('\n')
print("Accuracy of Perceptron Model:",accuracy_score(y_train, y_train_pred))
print("Precision of Perceptron Model:",precision_score(y_train, y_train_pred))
print("Recall of Perceptron Model:",recall_score(y_train, y_train_pred))
print("F1-score of Perceptron Model:",f1_score(y_train, y_train_pred))
```

```
-----------------------Perceptron-----------------------
-----------------------Train-----------------------


Accuracy of Perceptron Model: 0.8917875
Precision of Perceptron Model: 0.9285792364659907
Recall of Perceptron Model: 0.8489993253879019
F1-score of Perceptron Model: 0.8870079356792314
```

[32]:
```
# predict the labels of test values

y_test_pred = perceptron.predict(x_test)

# find the accuracy, precision, recall and f1_score of the Perceptron model on␣
 ↪the test set

print("-----------------------Test-----------------------")
print('\n')
print("Accuracy of Perceptron Model:",accuracy_score(y_test, y_test_pred))
print("Precision of Perceptron Model:",precision_score(y_test, y_test_pred))
print("Recall of Perceptron Model:",recall_score(y_test, y_test_pred))
print("F1-score of Perceptron Model:",f1_score(y_test, y_test_pred))
```

```
-----------------------Test-----------------------
```

```
Accuracy of Perceptron Model: 0.850325
Precision of Perceptron Model: 0.8881663053749097
Recall of Perceptron Model: 0.800791821188734
F1-score of Perceptron Model: 0.842219001185927
```

6 SVM

[33]:
```
# standardize the features using StandardScaler

scalar = StandardScaler(with_mean=False)
x_train_std = scalar.fit_transform(x_train)
x_test_std = scalar.transform(x_test)

# train an SVM model on the training dataset

lin_svc = LinearSVC(random_state=100)
lin_svc.fit(x_train_std,y_train)
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/svm/_base.py:976:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

[33]:
```
LinearSVC(random_state=100)
```

[34]:
```
# predict the labels of train values

y_train_pred = lin_svc.predict(x_train_std)

# find the accuracy, precision, recall and f1_score of the SVM model on the␣
 ↪training set

print("-----------------------SVM-----------------------")
print("-----------------------Train-----------------------")
print('\n')
print("Accuracy of SVM Model:",accuracy_score(y_train, y_train_pred))
print("Precision of SVM Model:",precision_score(y_train, y_train_pred))
print("Recall of SVM Model:",recall_score(y_train, y_train_pred))
print("F1-score of SVM Model:",f1_score(y_train, y_train_pred))
```

```
-----------------------SVM-----------------------
-----------------------Train-----------------------


Accuracy of SVM Model: 0.8802625
Precision of SVM Model: 0.881522883352549
```

```
Recall of SVM Model: 0.8787697074182346
F1-score of SVM Model: 0.8801441423405614
```

[35]:
```
# predict the labels of test values

y_test_pred = lin_svc.predict(x_test_std)

# find the accuracy, precision, recall and f1_score of the SVM model on the␣
 ↪test set

print("-----------------------Test-----------------------")
print('\n')
print("Accuracy of SVM Model:",accuracy_score(y_test, y_test_pred))
print("Precision of SVM Model:",precision_score(y_test, y_test_pred))
print("Recall of SVM Model:",recall_score(y_test, y_test_pred))
print("F1-score of SVM Model:",f1_score(y_test, y_test_pred))
```

```
-----------------------Test-----------------------


Accuracy of SVM Model: 0.814075
Precision of SVM Model: 0.8171912219350261
Recall of SVM Model: 0.8080585346296482
F1-score of SVM Model: 0.8125992188484314
```

7 Logistic Regression

[36]:
```
# train a Logistic Regression model on the training dataset

log_reg = LogisticRegression(n_jobs=-1, random_state=100)
log_reg.fit(x_train,y_train)
```

[36]:
```
LogisticRegression(n_jobs=-1, random_state=100)
```

[37]:
```
# predict the labels of train values

y_train_pred = log_reg.predict(x_train)

# find the accuracy, precision, recall and f1_score of the Logistic Regression␣
 ↪model on the training set

print("-----------------------Logistic Regression-----------------------")
print("-----------------------Train-----------------------")
print('\n')
print("Accuracy of Logistic Regression Model:",accuracy_score(y_train,␣
 ↪y_train_pred))
```

```
print("Precision of Logistic Regression Model:",precision_score(y_train,␣
 ↪y_train_pred))
print("Recall of Logistic Regression Model:",recall_score(y_train,␣
 ↪y_train_pred))
print("F1-score of Logistic Regression Model:",f1_score(y_train, y_train_pred))
```

```
-----------------------Logistic Regression-----------------------
-----------------------Train-----------------------


Accuracy of Logistic Regression Model: 0.911
Precision of Logistic Regression Model: 0.9134997235208365
Recall of Logistic Regression Model: 0.9080903480498713
F1-score of Logistic Regression Model: 0.9107870039719831
```

[38]:
```
# predict the labels of test values

y_test_pred = log_reg.predict(x_test)

# find the accuracy, precision, recall and f1_score of the Logistic Regression␣
 ↪model on the test set

print("-----------------------Test-----------------------")
print('\n')
print("Accuracy of Logistic Regression Model:",accuracy_score(y_test,␣
 ↪y_test_pred))
print("Precision of Logistic Regression Model:",precision_score(y_test,␣
 ↪y_test_pred))
print("Recall of Logistic Regression Model:",recall_score(y_test, y_test_pred))
print("F1-score of Logistic Regression Model:",f1_score(y_test, y_test_pred))
```

```
-----------------------Test-----------------------


Accuracy of Logistic Regression Model: 0.896075
Precision of Logistic Regression Model: 0.8996205413609917
Recall of Logistic Regression Model: 0.891099528916508
F1-score of Logistic Regression Model: 0.8953397618268336
```

8 Naive Bayes

[39]:
```
# train a Multinomial Naive Bayes model on the training dataset

multi_nb = MultinomialNB()
multi_nb.fit(x_train,y_train)
```

[39]:
```
MultinomialNB()
```

[40]:
```
# predict the labels of train values

y_train_pred = multi_nb.predict(x_train)

# find the accuracy, precision, recall and f1_score of the Multinomial Naive␣
 ↪Bayes model on the training set

print("----------------------Naive Bayes----------------------")
print("----------------------Train----------------------")
print('\n')
print("Accuracy of Multinomial Naive Bayes Model:",accuracy_score(y_train,␣
 ↪y_train_pred))
print("Precision of Multinomial Naive Bayes Model:",precision_score(y_train,␣
 ↪y_train_pred))
print("Recall of Multinomial Naive Bayes Model:",recall_score(y_train,␣
 ↪y_train_pred))
print("F1-score of Multinomial Naive Bayes Model:",f1_score(y_train,␣
 ↪y_train_pred))
```

```
----------------------Naive Bayes----------------------
----------------------Train----------------------


Accuracy of Multinomial Naive Bayes Model: 0.88308125
Precision of Multinomial Naive Bayes Model: 0.89005328822699
Recall of Multinomial Naive Bayes Model: 0.8742972790645379
F1-score of Multinomial Naive Bayes Model: 0.8821049314636837
```

[41]:
```
# predict the labels of test values

y_test_pred = multi_nb.predict(x_test)

# find the accuracy, precision, recall and f1_score of the Multinomial Naive␣
 ↪Bayes model on the test set

print("----------------------Test----------------------")
print('\n')
print("Accuracy of Multinomial Naive Bayes Model:",accuracy_score(y_test,␣
 ↪y_test_pred))
print("Precision of Multinomial Naive Bayes Model:",precision_score(y_test,␣
 ↪y_test_pred))
print("Recall of Multinomial Naive Bayes Model:",recall_score(y_test,␣
 ↪y_test_pred))
print("F1-score of Multinomial Naive Bayes Model:",f1_score(y_test,␣
 ↪y_test_pred))
```

```
----------------------Test----------------------
```

```
Accuracy of Multinomial Naive Bayes Model: 0.8676
Precision of Multinomial Naive Bayes Model: 0.873852275045909
Recall of Multinomial Naive Bayes Model: 0.8585246065951689
F1-score of Multinomial Naive Bayes Model: 0.8661206329945903
```

[ ]: