

3725520208_HW1_CSCI544

January 21, 2024

```
[1]: # Importing necessary libraries
import pandas as pd # For data manipulation and analysis
import numpy as np  # For numerical operations
import re           # Regular expressions for text processing
from bs4 import BeautifulSoup # For HTML parsing
from sklearn.model_selection import train_test_split # For splitting data into
    ↪ training and testing sets
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score, precision_score, recall_score,
    ↪ f1_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression

import nltk          # Natural Language Toolkit for text processing
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet') # Download WordNet data
nltk.download('stopwords') # Download StopWords data

import warnings      # To handle warnings
warnings.filterwarnings("ignore") # Ignore warnings for the remainder of the
    ↪ code
warnings.filterwarnings("default") # Set warnings back to default behavior
```

```
[nltk_data] Downloading package wordnet to
[nltk_data] /Users/sakethanne/nltk_data...
[nltk_data] Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data] /Users/sakethanne/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

```
[2]: # ! pip install bs4 # in case you don't have it installed
     # ! pip install contractions # in case contractions are not already installed
```

```
# # Dataset: https://s3.amazonaws.com/amazon-reviews-pds/tsv/
↳ amazon_reviews_us_Beauty_v1_00.tsv.gz
```

0.1 Read Data

```
[3]: # Reading the data from the tsv (Amazon Kitchen dataset) file as a Pandas frame
full_data = pd.read_csv("./amazon_reviews_us_Office_Products_v1_00.tsv",
↳ delimiter='\t', encoding='utf-8', error_bad_lines=False)
```

```
/var/folders/xx/d1tzxzhj3fzbmswz4z7m_40w0000gn/T/ipykernel_12645/1150709939.py:2
: FutureWarning: The error_bad_lines argument has been deprecated and will be
removed in a future version. Use on_bad_lines in the future.
```

```
full_data = pd.read_csv("./amazon_reviews_us_Office_Products_v1_00.tsv",
delimiter='\t', encoding='utf-8', error_bad_lines=False)
```

```
Skipping line 20773: expected 15 fields, saw 22
Skipping line 39834: expected 15 fields, saw 22
Skipping line 52957: expected 15 fields, saw 22
Skipping line 54540: expected 15 fields, saw 22
```

```
Skipping line 80276: expected 15 fields, saw 22
Skipping line 96168: expected 15 fields, saw 22
Skipping line 96866: expected 15 fields, saw 22
Skipping line 98175: expected 15 fields, saw 22
Skipping line 112539: expected 15 fields, saw 22
Skipping line 119377: expected 15 fields, saw 22
Skipping line 120065: expected 15 fields, saw 22
Skipping line 124703: expected 15 fields, saw 22
```

```
Skipping line 134024: expected 15 fields, saw 22
Skipping line 153938: expected 15 fields, saw 22
Skipping line 156225: expected 15 fields, saw 22
Skipping line 168603: expected 15 fields, saw 22
Skipping line 187002: expected 15 fields, saw 22
```

```
Skipping line 200397: expected 15 fields, saw 22
Skipping line 203809: expected 15 fields, saw 22
Skipping line 207680: expected 15 fields, saw 22
Skipping line 223421: expected 15 fields, saw 22
Skipping line 244032: expected 15 fields, saw 22
```

```
Skipping line 270329: expected 15 fields, saw 22
Skipping line 276484: expected 15 fields, saw 22
Skipping line 304755: expected 15 fields, saw 22
```

Skipping line 379449: expected 15 fields, saw 22
Skipping line 386191: expected 15 fields, saw 22
Skipping line 391811: expected 15 fields, saw 22

Skipping line 414348: expected 15 fields, saw 22
Skipping line 414773: expected 15 fields, saw 22
Skipping line 417572: expected 15 fields, saw 22
Skipping line 419496: expected 15 fields, saw 22
Skipping line 430528: expected 15 fields, saw 22
Skipping line 442230: expected 15 fields, saw 22
Skipping line 450931: expected 15 fields, saw 22

Skipping line 465377: expected 15 fields, saw 22
Skipping line 467685: expected 15 fields, saw 22
Skipping line 485055: expected 15 fields, saw 22
Skipping line 487220: expected 15 fields, saw 22
Skipping line 496076: expected 15 fields, saw 22
Skipping line 512269: expected 15 fields, saw 22

Skipping line 529505: expected 15 fields, saw 22
Skipping line 531286: expected 15 fields, saw 22
Skipping line 535424: expected 15 fields, saw 22
Skipping line 569898: expected 15 fields, saw 22
Skipping line 586293: expected 15 fields, saw 22

Skipping line 593880: expected 15 fields, saw 22
Skipping line 599274: expected 15 fields, saw 22
Skipping line 607961: expected 15 fields, saw 22
Skipping line 612413: expected 15 fields, saw 22
Skipping line 615913: expected 15 fields, saw 22

Skipping line 677580: expected 15 fields, saw 22
Skipping line 687191: expected 15 fields, saw 22
Skipping line 710819: expected 15 fields, saw 22

Skipping line 728692: expected 15 fields, saw 22
Skipping line 730216: expected 15 fields, saw 22
Skipping line 758397: expected 15 fields, saw 22
Skipping line 760061: expected 15 fields, saw 22
Skipping line 768935: expected 15 fields, saw 22
Skipping line 769483: expected 15 fields, saw 22

Skipping line 822725: expected 15 fields, saw 22
Skipping line 823621: expected 15 fields, saw 22

Skipping line 857041: expected 15 fields, saw 22
Skipping line 857320: expected 15 fields, saw 22
Skipping line 858565: expected 15 fields, saw 22

Skipping line 860629: expected 15 fields, saw 22
Skipping line 864033: expected 15 fields, saw 22
Skipping line 868673: expected 15 fields, saw 22
Skipping line 869189: expected 15 fields, saw 22

Skipping line 938605: expected 15 fields, saw 22
Skipping line 940100: expected 15 fields, saw 22
Skipping line 975137: expected 15 fields, saw 22
Skipping line 976314: expected 15 fields, saw 22

Skipping line 985597: expected 15 fields, saw 22
Skipping line 990873: expected 15 fields, saw 22
Skipping line 991806: expected 15 fields, saw 22
Skipping line 1019808: expected 15 fields, saw 22
Skipping line 1021526: expected 15 fields, saw 22
Skipping line 1023905: expected 15 fields, saw 22
Skipping line 1044207: expected 15 fields, saw 22

Skipping line 1084683: expected 15 fields, saw 22
Skipping line 1093288: expected 15 fields, saw 22

Skipping line 1136430: expected 15 fields, saw 22
Skipping line 1139815: expected 15 fields, saw 22

Skipping line 1179821: expected 15 fields, saw 22
Skipping line 1195351: expected 15 fields, saw 22
Skipping line 1202007: expected 15 fields, saw 22
Skipping line 1224868: expected 15 fields, saw 22
Skipping line 1232490: expected 15 fields, saw 22
Skipping line 1238697: expected 15 fields, saw 22

Skipping line 1258654: expected 15 fields, saw 22
Skipping line 1279948: expected 15 fields, saw 22
Skipping line 1294360: expected 15 fields, saw 22
Skipping line 1302240: expected 15 fields, saw 22

Skipping line 1413654: expected 15 fields, saw 22

Skipping line 1687095: expected 15 fields, saw 22

Skipping line 1805966: expected 15 fields, saw 22

Skipping line 1892134: expected 15 fields, saw 22

```
/var/folders/xx/d1tzxzhj3fzbmswz4z7m_40w0000gn/T/ipykernel_12645/1150709939.py:2
: DtypeWarning: Columns (7) have mixed types. Specify dtype option on import or
set low_memory=False.
```

```
full_data = pd.read_csv("./amazon_reviews_us_Office_Products_v1_00.tsv",
```

```
delimiter='\t', encoding='utf-8', error_bad_lines=False)
```

```
[4]: # Printing the data frame that contains the entire dataset from the tsv file
print(full_data)
```

| | marketplace | customer_id | review_id | product_id | product_parent | \ |
|---------|-------------|-------------|----------------|------------|----------------|---|
| 0 | US | 43081963 | R18RVCKGH1SSI9 | B001BM2MAC | 307809868 | |
| 1 | US | 10951564 | R3L4L6LW1PUOFY | B00DZYEXPQ | 75004341 | |
| 2 | US | 21143145 | R2J8AWXWTDX2TF | B00RTMUHDW | 529689027 | |
| 3 | US | 52782374 | R1PR37BR7G3M6A | B00D7H8XB6 | 868449945 | |
| 4 | US | 24045652 | R3BDDDZMZBZDPU | B001XCWP34 | 33521401 | |
| ... | ... | ... | ... | ... | ... | |
| 2640249 | US | 53005790 | RLI7EI10S7SN0 | B00000DM9M | 223408988 | |
| 2640250 | US | 52188548 | R1F3SRK9MHE6A3 | B00000DM9M | 223408988 | |
| 2640251 | US | 52090046 | R23VOC4NRJL8EM | 0807865001 | 307284585 | |
| 2640252 | US | 52503173 | R13ZAE1ATEUC1T | 1572313188 | 870359649 | |
| 2640253 | US | 52585611 | RE8J502GY04NN | 1572313188 | 870359649 | |

| | product_title | product_category | \ |
|---------|---|------------------|---|
| 0 | Scotch Cushion Wrap 7961, 12 Inches x 100 Feet | Office Products | |
| 1 | Dust-Off Compressed Gas Duster, Pack of 4 | Office Products | |
| 2 | Amram Tagger Standard Tag Attaching Tagging Gu... | Office Products | |
| 3 | AmazonBasics 12-Sheet High-Security Micro-Cut ... | Office Products | |
| 4 | Derwent Colored Pencils, Inktense Ink Pencils,... | Office Products | |
| ... | ... | ... | |
| 2640249 | PalmOne III Leather Belt Clip Case | Office Products | |
| 2640250 | PalmOne III Leather Belt Clip Case | Office Products | |
| 2640251 | Gods and Heroes of Ancient Greece | Office Products | |
| 2640252 | Microsoft EXCEL 97/ Visual Basic Step-by-Step ... | Office Products | |
| 2640253 | Microsoft EXCEL 97/ Visual Basic Step-by-Step ... | Office Products | |

| | star_rating | helpful_votes | total_votes | vine | verified_purchase | \ |
|---------|-------------|---------------|-------------|------|-------------------|---|
| 0 | 5 | 0.0 | 0.0 | N | Y | |
| 1 | 5 | 0.0 | 1.0 | N | Y | |
| 2 | 5 | 0.0 | 0.0 | N | Y | |
| 3 | 1 | 2.0 | 3.0 | N | Y | |
| 4 | 4 | 0.0 | 0.0 | N | Y | |
| ... | ... | ... | ... | ... | ... | |
| 2640249 | 4 | 26.0 | 26.0 | N | N | |
| 2640250 | 4 | 18.0 | 18.0 | N | N | |
| 2640251 | 4 | 9.0 | 16.0 | N | N | |
| 2640252 | 5 | 0.0 | 0.0 | N | N | |
| 2640253 | 5 | 0.0 | 0.0 | N | N | |

| | review_headline | \ |
|---|--|---|
| 0 | Five Stars | |
| 1 | Phfffffft, Phfffffft. Lots of air, and it's C... | |
| 2 | but I am sure I will like it. | |

```

3         and the shredder was dirty and the bin was par...
4                                     Four Stars
...
2640249 Great value! A must if you hate to carry thing...
2640250         Attaches the Palm Pilot like an appendage
2640251 Excellent information, pictures and stories, I...
2640252                                     class text
2640253                                     Microsoft's Finest

                                     review_body review_date
0                                     Great product. 2015-08-31
1         What's to say about this commodity item except... 2015-08-31
2         Haven't used yet, but I am sure I will like it. 2015-08-31
3         Although this was labeled as &#34;new&#34; the... 2015-08-31
4                                     Gorgeous colors and easy to use 2015-08-31
...
2640249 I can't live anymore whithout my Palm III. But... 1998-12-07
2640250 Although the Palm Pilot is thin and compact it... 1998-11-30
2640251 This book had a lot of great content without b... 1998-10-15
2640252 I am teaching a course in Excel and am using t... 1998-08-22
2640253 A very comprehensive layout of exactly how Vis... 1998-07-15

[2640254 rows x 15 columns]

```

0.2 Keep Reviews and Ratings

```

[5]: # Keep only the Reviews and Ratings fields from the full data
data = full_data[['review_body', 'star_rating', 'review_headline']]

# Converting 'star_rating' to numeric values
data['star_rating'] = pd.to_numeric(data['star_rating'], errors='coerce')

# Displaying three sample reviews along with ratings
sample_reviews = data.sample(3)
print("=====Sample Reviews:=====")
print(sample_reviews)

```

```

=====Sample Reviews:=====

                                     review_body  star_rating \
825367  I feel like such a pro with this presenter! Pr...      5.0
345381                                     Great purchase      5.0
2450776  I purchased the Canon MP620 to replace an Epso...      4.0

                                     review_headline
825367          Great presenter!
345381          Five Stars
2450776  Excellent buy for the money

```

```
/var/folders/xx/dltzxzhj3fzbmswz4z7m_40w0000gn/T/ipykernel_12645/297036716.py:5:
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

```
data['star_rating'] = pd.to_numeric(data['star_rating'], errors='coerce')
```

```
[6]: # Reporting statistics of the ratings
ratings_statistics = data['star_rating'].value_counts().sort_index()
print("\n=====Ratings Statistics:
      ↳=====")
print("Ratings Count:")
print(ratings_statistics)
```

```
=====Ratings Statistics:=====
Ratings Count:
1.0      306979
2.0      138384
3.0      193691
4.0       418371
5.0      1582812
Name: star_rating, dtype: int64
```

Form two classes and select 100000 reviews randomly from each class.

```
[7]: # Creating binary labels for sentiment analysis
data['sentiment'] = data['star_rating'].apply(lambda x: 1 if x > 3 else 0 if x_
      ↳<= 2 else None)

# Discarding neutral reviews (rating 3)
data = data.dropna(subset=['sentiment'])

# Selecting 100,000 positive and 100,000 negative reviews
positive_reviews = data[data['sentiment'] == 1].sample(100000, random_state=42)
negative_reviews = data[data['sentiment'] == 0].sample(100000, random_state=42)

# Concatenating positive and negative reviews into a single data set for further_
      ↳test and train set split
selected_reviews = pd.concat([positive_reviews, negative_reviews])

# Printing the reviews that have been selected for further processing randomly
print(selected_reviews)
```

```
/var/folders/xx/dltzxzhj3fzbmswz4z7m_40w0000gn/T/ipykernel_12645/18226587.py:2:
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

```
data['sentiment'] = data['star_rating'].apply(lambda x: 1 if x > 3 else 0 if x
<= 2 else None)
```

| | review_body | star_rating \ |
|---------|---|---------------|
| 980067 | Yes they're thin, but they're sturdy and do th... | 5.0 |
| 655902 | order came in just fine, would order again | 5.0 |
| 1249308 | exactly as it appears great for work | 4.0 |
| 2190006 | I just set this up in my classroom. It is actu... | 5.0 |
| 935188 | good product | 5.0 |
| ... | ... | ... |
| 289745 | I have a Brother MFC -J6720DW. Unfortunately ... | 2.0 |
| 702019 | I was extremely disappointed with this product... | 1.0 |
| 2530891 | I was excited about this all-in-one based on r... | 2.0 |
| 1200675 | Alright, so you think to yourself this is goin... | 1.0 |
| 2214545 | These clips do not work on stainless steel ref... | 1.0 |

| | review_headline | sentiment |
|---------|---|-----------|
| 980067 | Thin but Sturdy | 1.0 |
| 655902 | Name Badge magnets | 1.0 |
| 1249308 | Four Stars | 1.0 |
| 2190006 | Great product! | 1.0 |
| 935188 | Five Stars | 1.0 |
| ... | ... | ... |
| 289745 | Doesn't work wiith Brother MFC -J6720DW | 0.0 |
| 702019 | Don't buy if you are a teacher! | 0.0 |
| 2530891 | Ink Guzzler | 0.0 |
| 1200675 | Failed, the magnet fell off after a few days w... | 0.0 |
| 2214545 | Clips don't work! | 0.0 |

[200000 rows x 4 columns]

Split the dataset into training and testing dataset

```
[8]: # Splitting the dataset into 80% training and 20% testing
X_train, X_test, y_train, y_test = \
    train_test_split(selected_reviews['review_body'],
                    selected_reviews['sentiment'],
                    test_size=0.2,
                    random_state=42)
```

```
[9]: # Printing the Features of the training set
print(X_train)
```



```

2472632    I'm sorry I bought this printer.  I guess it's...
1568399    Its a bit crude i must admit, but i still thou...
766793     Prints are blurry no matter how many times the...
1501214    Not worth $5,00 folder is made of cheap plasti...
59127      I got two. They both didn't work-
...
1824293    It has a variety of colors, but there's a thin...
968776     Fell apart in less than 2 weeks.
1126758    I bought this product a little over a year ago...
1745055    From the beginning, the pages printed terribly...
2237191    With the MagicJack Plus I have successfully ma...
Name: review_body, Length: 160000, dtype: object

```

```

[10]: # Printing the Features of the testing set
      print(X_test)

```

```

2292297    worked fine for a week. then auto feed malfunc...
1067920                                           GREAT
2626155    This toy was not created well for babies stand...
574517     They work as expected.
704740     very high quality like original ink. Love it a...
...
1933307    It doesn't pay to buy off name folders. The Pe...
928345                                           excellent
1898808    when working numbers in a fast paced environme...
307872     None of them worked. I'm going back to purchas...
1935724    I worked as a quality manager in paper manufac...
Name: review_body, Length: 40000, dtype: object

```

```

[11]: # Printing the Target(s) of the training set
      print(y_train)

```

```

2472632    0.0
1568399    1.0
766793     0.0
1501214    0.0
59127      0.0
...
1824293    0.0
968776     0.0
1126758    0.0
1745055    0.0
2237191    0.0
Name: sentiment, Length: 160000, dtype: float64

```

```

[12]: # Printing the Target(s) of the testing set
      print(y_test)

```

```

2292297    0.0
1067920    1.0
2626155    0.0
574517     1.0
704740     1.0
...
1933307    1.0
928345     1.0
1898808    0.0
307872     0.0
1935724    1.0
Name: sentiment, Length: 40000, dtype: float64

```

1 Data Cleaning

```

[13]: # Define a contraction map
CONTRACTION_MAP = {
    "won't": "will not",
    "can't": "cannot",
    "i'm": "i am",
    "you're": "you are",
    "he's": "he is",
    "she's": "she is",
    "it's": "it is",
    "that's": "that is",
    "we're": "we are",
    "they're": "they are",
    "isn't": "is not",
    "aren't": "are not",
    "haven't": "have not",
    "hasn't": "has not",
    "didn't": "did not",
    "doesn't": "does not",
    "don't": "do not",
    "wasn't": "was not",
    "weren't": "were not",
    "haven't": "have not",
    "hasn't": "has not",
    "won't've": "will not have",
    "can't've": "cannot have",
    "i'll": "i will",
    "you'll": "you will",
    "he'll": "he will",
    "she'll": "she will",
    "it'll": "it will",
    "that'll": "that will",

```

```

    "we'll": "we will",
    "they'll": "they will",
    "i'd": "i would",
    "you'd": "you would",
    "he'd": "he would",
    "she'd": "she would",
    "it'd": "it would",
    "that'd": "that would",
    "we'd": "we would",
    "they'd": "they would",
    "i've": "i have",
    "you've": "you have",
    "we've": "we have",
    "they've": "they have",
    "shouldn't": "should not",
    "couldn't": "could not",
    "wouldn't": "would not",
    "mightn't": "might not",
    "mustn't": "must not",
    "shan't": "shall not",
    "oughtn't": "ought not",
    "who's": "who is",
    "what's": "what is",
    "where's": "where is",
    "when's": "when is",
    "why's": "why is",
    "how's": "how is",
    "it's": "it is",
    "let's": "let us"
}

# Function to expand contractions
def expand_contractions(text):
    for contraction, expansion in CONTRACTION_MAP.items():
        text = re.sub(contraction, expansion, text)
    return text

# Preprocess the reviews
def preprocess_reviews(reviews):
    # Convert to lowercase and handle NaN values
    reviews = reviews.apply(lambda x: str(x).lower() if pd.notna(x) else '')

    # Remove HTML and URLs
    reviews = reviews.apply(lambda x: BeautifulSoup(x, 'html.parser').get_text())
    reviews = reviews.apply(lambda x: re.sub(r'http\S+', '', x))

    # Remove non-alphabetical characters

```

```

reviews = reviews.apply(lambda x: re.sub(r'[~a-zA-Z\s]', '', x))

# Remove extra spaces
reviews = reviews.apply(lambda x: re.sub(' +', ' ', x))

# Perform contractions
reviews = reviews.apply(expand_contractions)

# Return the processed text of the review
return reviews

# Preprocess the training set
X_train_preprocessed = preprocess_reviews(X_train)

# Print average length of reviews before and after cleaning
avg_length_before = X_train.apply(lambda x: len(str(x))).mean()
avg_length_after = X_train_preprocessed.apply(len).mean()
print("=====Printing the Average lenght of Reviews Before and_
↳After Cleaning=====")
print(f"\nAverage Length of Reviews (Before Cleaning): {int(avg_length_before)}_
↳characters")
print(f"Average Length of Reviews (After Cleaning): {int(avg_length_after)}_
↳characters")

```

/var/folders/xx/dltzxzhj3fzbmswz4z7m_40w0000gn/T/ipykernel_12645/2837347252.py:7
5: MarkupResemblesLocatorWarning: The input looks more like a filename than
markup. You may want to open this file and pass the filehandle into Beautiful
Soup.

```

reviews = reviews.apply(lambda x: BeautifulSoup(x, 'html.parser').get_text())
/var/folders/xx/dltzxzhj3fzbmswz4z7m_40w0000gn/T/ipykernel_12645/2837347252.py:7
5: MarkupResemblesLocatorWarning: The input looks more like a URL than markup.
You may want to use an HTTP client like requests to get the document behind the
URL, and feed that document to BeautifulSoup.

```

```

reviews = reviews.apply(lambda x: BeautifulSoup(x, 'html.parser').get_text())
=====Printing the Average lenght of Reviews Before and After
Cleaning=====

```

Average Length of Reviews (Before Cleaning): 318 characters
Average Length of Reviews (After Cleaning): 300 characters

2 Pre-processing

2.0.1 – remove the stop words

2.0.2 – perform lemmatization

```
[14]: # Initialize NLTK's stopwords and WordNet lemmatizer
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

# Function to remove stop words and perform lemmatization
def preprocess_nltk(review):
    if pd.notna(review):
        words = nltk.word_tokenize(str(review).lower()) # Convert to lowercase
        words = [lemmatizer.lemmatize(word) for word in words if word.isalpha()]
        ↪and word not in stop_words]
        return ' '.join(words)
    else:
        return ''

# Preprocess the training set using NLTK
X_train_nltk_preprocessed = X_train_preprocessed.apply(preprocess_nltk)

# Print three sample reviews before and after NLTK preprocessing
sample_reviews_indices = X_train_preprocessed.sample(3).index

print("===== Printing Sample Reviews Before and After Pre-processing_
↪=====")
for index in sample_reviews_indices:
    print(f"\nSample Review {index} Before Pre-processing:")
    print(X_train_preprocessed.loc[index])

    print(f"\nSample Review {index} After NLTK Pre-processing:")
    print(X_train_nltk_preprocessed.loc[index])

# Print average length of reviews before and after NLTK processing
avg_length_before_nltk = X_train_preprocessed.apply(len).mean()
avg_length_after_nltk = X_train_nltk_preprocessed.apply(len).mean()
print("\n=====Printing the Average length of Reviews Before and_
↪After Pre-processing=====")
print(f"\nAverage Length of Reviews (Before NLTK Processing):_
↪{int(avg_length_before_nltk)} characters")
print(f"Average Length of Reviews (After NLTK Processing):_
↪{int(avg_length_after_nltk)} characters")
```

```
===== Printing Sample Reviews Before and After Pre-processing
=====
```

Sample Review 591443 Before Pre-processing:
printer did not work at all as the carriage was stuck in the far right position

Sample Review 591443 After NLTK Pre-processing:
printer work carriage stuck far right position

Sample Review 1498236 Before Pre-processing:
product was as advertised and is a great teaching tool tool set provides large
visuals for students and offers a varity

Sample Review 1498236 After NLTK Pre-processing:
product advertised great teaching tool tool set provides large visuals student
offer varity

Sample Review 2161966 Before Pre-processing:
ive had this for about a year i had my nd staples mailmate die on me in years
with pretty light home usei bought this amazon basics but have been disappointed
with performance its underpowered jams often gets stuck runningtoo bad ive liked
all the other amazon basics products ive purchased

Sample Review 2161966 After NLTK Pre-processing:
ive year nd staple mailmate die year pretty light home usei bought amazon basic
disappointed performance underpowered jam often get stuck runningtoo bad ive
liked amazon basic product ive purchased
=====Printing the Average lenght of Reviews Before and After Pre-
processing=====

Average Length of Reviews (Before NLTK Processing): 300 characters
Average Length of Reviews (After NLTK Processing): 190 characters

3 TF-IDF Feature Extraction

```
[15]: # Initialize the TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=2000000)

# Fit and transform the training set
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train_nltk_preprocessed)

# Transform the test set
X_test_tfidf = tfidf_vectorizer.transform(X_test.apply(preprocess_nltk))

# Print the shape of the TF-IDF matrices
print(f"\nShape of X_train_tfidf: {X_train_tfidf.shape}")
print(f"Shape of X_test_tfidf: {X_test_tfidf.shape}")
```

Shape of X_train_tfidf: (160000, 108488)
Shape of X_test_tfidf: (40000, 108488)

4 Perceptron

```
[16]: # Initialize the Perceptron model
perceptron_model = Perceptron(random_state=42)

# Train the Perceptron model on the TF-IDF features
perceptron_model.fit(X_train_tfidf, y_train)

# Predictions on the training set
y_train_pred = perceptron_model.predict(X_train_tfidf)

# Predictions on the test set
y_test_pred = perceptron_model.predict(X_test_tfidf)

# Calculate metrics for the training set
accuracy_train = accuracy_score(y_train, y_train_pred)
precision_train = precision_score(y_train, y_train_pred)
recall_train = recall_score(y_train, y_train_pred)
f1_train = f1_score(y_train, y_train_pred)

# Calculate metrics for the test set
accuracy_test = accuracy_score(y_test, y_test_pred)
precision_test = precision_score(y_test, y_test_pred)
recall_test = recall_score(y_test, y_test_pred)
f1_test = f1_score(y_test, y_test_pred)

# Print the results
print(f"\n===== Training Set Metrics: (Perceptron)␣
↪=====")
print(f"Accuracy: {accuracy_train}")
print(f"Precision: {precision_train}")
print(f"Recall: {recall_train}")
print(f"F1-score: {f1_train}")

print(f"\n===== Testing Set Metrics: (Perceptron)␣
↪=====")
print(f"Accuracy: {accuracy_test}")
print(f"Precision: {precision_test}")
print(f"Recall: {recall_test}")
print(f"F1-score: {f1_test}")
```

```

===== Training Set Metrics: (Perceptron) =====
Accuracy: 0.91620625
Precision: 0.9258094215129661
Recall: 0.9049458172409914
F1-score: 0.9152587367502892

===== Testing Set Metrics: (Perceptron) =====
Accuracy: 0.83635
Precision: 0.8197555523850287
Recall: 0.8621517531135897
F1-score: 0.8404193076548025

```

5 SVM

```

[17]: # Initialize the SVM model
svm_model = SVC(random_state=42)

# Train the SVM model on the TF-IDF features
svm_model.fit(X_train_tfidf, y_train)

# Predictions on the training set
y_train_pred_svm = svm_model.predict(X_train_tfidf)

# Predictions on the test set
y_test_pred_svm = svm_model.predict(X_test_tfidf)

# Calculate metrics for the training set
accuracy_train_svm = accuracy_score(y_train, y_train_pred_svm)
precision_train_svm = precision_score(y_train, y_train_pred_svm)
recall_train_svm = recall_score(y_train, y_train_pred_svm)
f1_train_svm = f1_score(y_train, y_train_pred_svm)

# Calculate metrics for the test set
accuracy_test_svm = accuracy_score(y_test, y_test_pred_svm)
precision_test_svm = precision_score(y_test, y_test_pred_svm)
recall_test_svm = recall_score(y_test, y_test_pred_svm)
f1_test_svm = f1_score(y_test, y_test_pred_svm)

# Print the results
print(f"\n===== Training Set Metrics: (SVM) =====")
print(f"Accuracy: {accuracy_train_svm}")
print(f"Precision: {precision_train_svm}")
print(f"Recall: {recall_train_svm}")
print(f"F1-score: {f1_train_svm}")

print(f"\n===== Testing Set Metrics: (SVM) =====")

```



```

print(f"Accuracy: {accuracy_test_svm}")
print(f"Precision: {precision_test_svm}")
print(f"Recall: {recall_test_svm}")
print(f"F1-score: {f1_test_svm}")

```

```

===== Training Set Metrics: (SVM) =====
Accuracy: 0.97399375
Precision: 0.974595149300428
Recall: 0.9733648305773245
F1-score: 0.9739796014082657

```

```

===== Testing Set Metrics: (SVM) =====
Accuracy: 0.903925
Precision: 0.8963773807186334
Recall: 0.9133696793877857
F1-score: 0.90479375696767

```

6 Logistic Regression

```

[18]: # Initialize the Logistic Regression model
logreg_model = LogisticRegression(random_state=42)

# Train the Logistic Regression model on the TF-IDF features
logreg_model.fit(X_train_tfidf, y_train)

# Predictions on the training set
y_train_pred_logreg = logreg_model.predict(X_train_tfidf)

# Predictions on the test set
y_test_pred_logreg = logreg_model.predict(X_test_tfidf)

# Calculate metrics for the training set
accuracy_train_logreg = accuracy_score(y_train, y_train_pred_logreg)
precision_train_logreg = precision_score(y_train, y_train_pred_logreg)
recall_train_logreg = recall_score(y_train, y_train_pred_logreg)
f1_train_logreg = f1_score(y_train, y_train_pred_logreg)

# Calculate metrics for the test set
accuracy_test_logreg = accuracy_score(y_test, y_test_pred_logreg)
precision_test_logreg = precision_score(y_test, y_test_pred_logreg)
recall_test_logreg = recall_score(y_test, y_test_pred_logreg)
f1_test_logreg = f1_score(y_test, y_test_pred_logreg)

# Print the results

```

```

print(f"\n===== Training Set Metrics: (Logistic Regression)␣
↪=====")
print(f"Accuracy: {accuracy_train_logreg}")
print(f"Precision: {precision_train_logreg}")
print(f"Recall: {recall_train_logreg}")
print(f"F1-score: {f1_train_logreg}")

print(f"\n===== Testing Set Metrics: (Logistic Regression)␣
↪=====")
print(f"Accuracy: {accuracy_test_logreg}")
print(f"Precision: {precision_test_logreg}")
print(f"Recall: {recall_test_logreg}")
print(f"F1-score: {f1_test_logreg}")

```

/Users/sakethanne/anaconda3/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
n_iter_i = _check_optimize_result(
```

```
===== Training Set Metrics: (Logistic Regression)
```

```
=====
```

```
Accuracy: 0.9125625
```

```
Precision: 0.9156665953079548
```

```
Recall: 0.9088454760208482
```

```
F1-score: 0.912243284949002
```

```
===== Testing Set Metrics: (Logistic Regression)
```

```
=====
```

```
Accuracy: 0.8929
```

```
Precision: 0.887627695800227
```

```
Recall: 0.899614865202821
```

```
F1-score: 0.893581081081081
```

7 Naive Bayes

```
[19]: # Initialize the Multinomial Naive Bayes model
nb_model = MultinomialNB()

# Train the Multinomial Naive Bayes model on the TF-IDF features
nb_model.fit(X_train_tfidf, y_train)

# Predictions on the training set
y_train_pred_nb = nb_model.predict(X_train_tfidf)

# Predictions on the test set
y_test_pred_nb = nb_model.predict(X_test_tfidf)

# Calculate metrics for the training set
accuracy_train_nb = accuracy_score(y_train, y_train_pred_nb)
precision_train_nb = precision_score(y_train, y_train_pred_nb)
recall_train_nb = recall_score(y_train, y_train_pred_nb)
f1_train_nb = f1_score(y_train, y_train_pred_nb)

# Calculate metrics for the test set
accuracy_test_nb = accuracy_score(y_test, y_test_pred_nb)
precision_test_nb = precision_score(y_test, y_test_pred_nb)
recall_test_nb = recall_score(y_test, y_test_pred_nb)
f1_test_nb = f1_score(y_test, y_test_pred_nb)

# Print the results
print(f"\n===== Training Set Metrics: (Multinomial Naive Bayes)␣
↪=====")
print(f"Accuracy: {accuracy_train_nb}")
print(f"Precision: {precision_train_nb}")
print(f"Recall: {recall_train_nb}")
print(f"F1-score: {f1_train_nb}")

print(f"\n===== Testing Set Metrics: (Multinomial Naive Bayes)␣
↪=====")
print(f"Accuracy: {accuracy_test_nb}")
print(f"Precision: {precision_test_nb}")
print(f"Recall: {recall_test_nb}")
print(f"F1-score: {f1_test_nb}")
```

```
===== Training Set Metrics: (Multinomial Naive Bayes)
=====
Accuracy: 0.88323125
Precision: 0.9019769789454364
Recall: 0.8599372554901447
```

F1-score: 0.8804555779505391

===== Testing Set Metrics: (Multinomial Naive Bayes)

=====

Accuracy: 0.860325

Precision: 0.8706390861376968

Recall: 0.846296203671285

F1-score: 0.8582950769777057