In [1]:
```
import pandas as pd
import numpy as np
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('averaged_perceptron_tagger')
import re
from bs4 import BeautifulSoup
import contractions
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.linear_model import Perceptron
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
```

```
/opt/anaconda3/lib/python3.8/site-packages/scipy/__init__.py:138: UserWarning:
A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (de
tected version 1.24.1)
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion} is re
quired for this version of "
[nltk_data] Downloading package wordnet to /Users/boom/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /Users/boom/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /Users/boom/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

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

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

1. Dataset Preparation

Read Data

In [3]:
```
df = pd.read_csv('data.tsv', sep='\t', on_bad_lines='skip')
print(df.head())
```

```
<ipython-input-3-a057dc4512f1>:1: DtypeWarning: Columns (7) have mixed types.
Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('data.tsv', sep='\t', on_bad_lines='skip')
  marketplace  customer_id       review_id  product_id  product_parent  \
0          US      1797882  R3I2DHQBR577SS  B001ANOOOE         2102612
1          US     18381298  R1QNE9NQFJC2Y4  B0016J22EQ       106393691
2          US     19242472  R3LIDG2Q4LJBAO  B00HU6UQAG       375449471
3          US     19551372  R3KSZHPAEVPEAL  B002HWS7RM       255651889
4          US     14802407   RAI2OIG50KZ43  B00SM99KWU       116158747

                                       product_title product_category  \
0  The Naked Bee Vitmin C Moisturizing Sunscreen ...           Beauty
1      Alba Botanica Sunless Tanning Lotion, 4 Ounce           Beauty
2          Elysee Infusion Skin Therapy Elixir, 2oz.           Beauty
3  Diane D722 Color, Perm And Conditioner Process...           Beauty
```

```
4  Biore UV Aqua Rich Watery Essence SPF50+/PA+++...           Beauty

  star_rating  helpful_votes  total_votes vine verified_purchase  \
0           5            0.0          0.0    N                 Y
1           5            0.0          0.0    N                 Y
2           5            0.0          0.0    N                 Y
3           5            0.0          0.0    N                 Y
4           5            0.0          0.0    N                 Y

                                    review_headline  \
0                                        Five Stars
1                          Thank you Alba Bontanica!
2                                        Five Stars
3                                        GOOD DEAL!
4  this soaks in quick and provides a nice base f...

                                        review_body review_date
0                  Love this, excellent sun block!!  2015-08-31
1  The great thing about this cream is that it do...  2015-08-31
2  Great Product, I'm 65 years old and this is al...  2015-08-31
3  I use them as shower caps & conditioning caps....  2015-08-31
4  This is my go-to daily sunblock. It leaves no ...  2015-08-31
```

Keep Reviews and Ratings

In [4]:
```
df = df.loc[:, ['review_body', 'star_rating']]
df['star_rating'] = pd.to_numeric(df['star_rating'], errors='coerce')
df = df[~df['star_rating'].isna()]
df['star_rating'] = df['star_rating'].astype(int)
```

In [5]:
```
df = df.loc[:, ['review_body', 'star_rating']]
print(df.head())
```

```
                                        review_body  star_rating
0                  Love this, excellent sun block!!            5
1  The great thing about this cream is that it do...            5
2  Great Product, I'm 65 years old and this is al...            5
3  I use them as shower caps & conditioning caps....            5
4  This is my go-to daily sunblock. It leaves no ...            5
```

Group ratings to 3 classes

In [6]:
```
mapping = {1: 1, 2: 1, 3: 2, 4: 3, 5: 3}
df = df.replace({'star_rating': mapping})
print(df.head())
```

```
                                        review_body  star_rating
0                  Love this, excellent sun block!!            3
1  The great thing about this cream is that it do...            3
2  Great Product, I'm 65 years old and this is al...            3
3  I use them as shower caps & conditioning caps....            3
4  This is my go-to daily sunblock. It leaves no ...            3
```

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

In [7]:
```
# drop duplicates
df = df.drop_duplicates()
```

In [8]:
```
r_state = 555
list_sample = [df[df.star_rating == 1].sample(n=20000, random_state=r_state),
               df[df.star_rating == 2].sample(n=20000, random_state=r_state),
               df[df.star_rating == 3].sample(n=20000, random_state=r_state)]
df_sample = pd.concat(list_sample)
print(df_sample.head())
```

```
                                                review_body  star_rating
1294234  Handle provided had some rusted internal threa...            1
4186939  MY WIFE SAYS THAT THIS REALLY WASNT USEFUL FOR...            1
4493244  This came with no glue, and the gel was hard a...            1
3048924  I bought this product because of all the 5-sta...            1
4149821  i got this product expecting the wow factor, b...            1
```

In [9]:
```
print(df_sample.groupby('star_rating').count())
```

```
             review_body
star_rating
1                  20000
2                  20000
3                  20000
```

2. Data Cleaning

Convert to lowercase

In [10]:
```
df_clean = df_sample.copy()
df_clean.columns = ['review', 'stars']

df_clean['review'] = df_clean['review'].apply(str.lower)
print(df_clean.head())
```

```
                                                review  stars
1294234  handle provided had some rusted internal threa...      1
4186939  my wife says that this really wasnt useful for...      1
4493244  this came with no glue, and the gel was hard a...      1
3048924  i bought this product because of all the 5-sta...      1
4149821  i got this product expecting the wow factor, b...      1
```

remove the HTML and URLs from the reviews

In [11]:
```
# remove HTML
df_clean['review'] = df_clean['review'].str.replace(r'<[^<>]*>', '', regex=Tr

# remove URLs
def remove_urls(text):
    return re.sub(r'''(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{

df_clean['review'] = df_clean['review'].apply(remove_urls)
print(df_clean.head())
```

```
                                                review  stars
1294234  handle provided had some rusted internal threa...      1
4186939  my wife says that this really wasnt useful for...      1
4493244  this came with no glue, and the gel was hard a...      1
3048924  i bought this product because of all the 5-sta...      1
4149821  i got this product expecting the wow factor, b...      1
```

perform contractions on the reviews

In [12]:
```
def perform_contractions(text):
    return ' '.join([contractions.fix(word) for word in text.split()])

df_clean['review'] = df_clean['review'].apply(perform_contractions)
print(df_clean.head())
```

```
                                                    review  stars
1294234  handle provided had some rusted internal threa...      1
4186939  my wife says that this really was not useful f...      1
4493244  this came with no glue, and the gel was hard a...      1
3048924  i bought this product because of all the 5-sta...      1
4149821  i got this product expecting the wow factor, b...      1
```

remove non-alphabetical characters

In [13]:
```
def remove_non_alpha_chars(text):
    return re.sub(r'[^a-zA-Z0-9\s]', ' ', text)

df_clean['review'] = df_clean['review'].apply(remove_non_alpha_chars)
print(df_clean.head())
```

```
                                                    review  stars
1294234  handle provided had some rusted internal threa...      1
4186939  my wife says that this really was not useful f...      1
4493244  this came with no glue  and the gel was hard a...      1
3048924  i bought this product because of all the 5 sta...      1
4149821  i got this product expecting the wow factor  b...      1
```

remove extra spaces

In [14]:
```
def remove_extra_spaces(text):
    return re.sub(r'\s+', ' ', text)

df_clean['review'] = df_clean['review'].apply(remove_extra_spaces)
print(df_clean.head())
```

```
                                                    review  stars
1294234  handle provided had some rusted internal threa...      1
4186939  my wife says that this really was not useful f...      1
4493244  this came with no glue and the gel was hard as...      1
3048924  i bought this product because of all the 5 sta...      1
4149821  i got this product expecting the wow factor bu...      1
```

In [15]:
```
# printing average lengths before/after data cleaning
print('Average length of reviews before and after data cleaning: ', \
      df_sample['review_body'].str.len().mean(), ', ', df_clean['review'].str
```

```
Average length of reviews before and after data cleaning: 281.41571666666664,
273.19905
```

3. Pre-processing

Remove the stop words

In [16]:
```
stop_words = set(stopwords.words('english'))

def remove_stop_words(text):
    return ' '.join([word for word in text.split() if word not in stop_words]
```

```
df_preproc = df_clean.copy()
df_preproc['review'] = df_preproc['review'].apply(remove_stop_words)
print(df_preproc.head())
```

```
                                                   review  stars
1294234  handle provided rusted internal threading incl...      1
4186939  wife says really useful nail polish plate rubb...      1
4493244  came glue gel hard rock never buying product d...      1
3048924  bought product 5 star ratings received wonderi...      1
4149821  got product expecting wow factor found priced ...      1
```

Perform lemmatization

In [17]:

```
lemmatizer = WordNetLemmatizer()

def perform_lemmatization(lemmatizer, text):
    lemmatized_list = []
    for word, pos_tag in nltk.pos_tag(text.split()):
        if pos_tag.startswith('V'):
            lemmatized_list.append(lemmatizer.lemmatize(word, 'v'))
        elif pos_tag.startswith('J'):
            lemmatized_list.append(lemmatizer.lemmatize(word, 'a'))
        else:
            lemmatized_list.append(lemmatizer.lemmatize(word))
    return ' '.join(lemmatized_list)

df_lemma = df_preproc.copy()
df_lemma['review'] = df_lemma['review'].apply(lambda x: perform_lemmatization
print(df_lemma.head())
```

```
                                                   review  stars
1294234  handle provide rust internal threading include...      1
4186939  wife say really useful nail polish plate rub s...      1
4493244  come glue gel hard rock never buy product diss...      1
3048924  buy product 5 star rating receive wonder produ...      1
4149821  get product expect wow factor find price size ...      1
```

In [18]:

```
# printing average lengths before/after data preprocessing
print('Average length of reviews before and after data preprocessing: ', \
      df_clean['review'].str.len().mean(), ', ', df_lemma['review'].str.len()
```

```
Average length of reviews before and after data preprocessing: 273.19905, 158.
77876666666666
```

4. TF-IDF Feature Extraction

In [19]:

```
vectorizer = TfidfVectorizer(max_features=10000)

df_model = df_lemma.copy()
tfidf = vectorizer.fit_transform(df_model['review'])
df_X = pd.DataFrame(tfidf.toarray(), columns=vectorizer.get_feature_names())
```

In [20]:

```
print(df_X.head())
df_y = df_model['stars'].reset_index(drop=True)
df_y = df_y.astype(int)
print(df_y.head())
```

```
    00  000   01   02   03   04   05   06   07   08  ...  ziploc  ziplock  \
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0      0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0      0.0
```

```
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.0       0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.0       0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.0       0.0

   zipper  zirconium  zit  zits  zombie  zone  zoom  zoya
0     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
1     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
2     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
3     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
4     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0

[5 rows x 10000 columns]
0    1
1    1
2    1
3    1
4    1
Name: stars, dtype: int64
```

Split training/testing set - 80/20

In [21]:

```
X_train, X_test, y_train, y_test = train_test_split(df_X, df_y, test_size=0.2
print(X_train, X_test, y_train, y_test)
```

```
         00  000   01   02   03   04   05   06   07   08  ...  ziploc  ziplock
\
5521   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
38538  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
30253  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
45733  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
2593   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
...    ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...     ...       ...
39184  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
18365  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
33132  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
22889  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
10375  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0

       zipper  zirconium  zit  zits  zombie  zone  zoom  zoya
5521      0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
38538     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
30253     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
45733     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
2593      0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
...       ...        ...  ...   ...     ...   ...   ...   ...
39184     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
18365     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
33132     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
22889     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
10375     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0

[48000 rows x 10000 columns]         00  000   01   02   03   04   05   06   0
7   08  ...  ziploc  ziplock  \
31788  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
16832  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
8267   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
55958  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
11472  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
...    ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...     ...       ...
5042   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
35683  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
52589  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
6437   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0
33910  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.0       0.0

       zipper  zirconium  zit  zits  zombie  zone  zoom  zoya
31788     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
16832     0.0        0.0  0.0   0.0     0.0   0.0   0.0   0.0
```

```
8267      0.0         0.0   0.0   0.0      0.0   0.0   0.0   0.0
55958     0.0         0.0   0.0   0.0      0.0   0.0   0.0   0.0
11472     0.0         0.0   0.0   0.0      0.0   0.0   0.0   0.0
...       ...         ...   ...   ...      ...   ...   ...   ...
5042      0.0         0.0   0.0   0.0      0.0   0.0   0.0   0.0
35683     0.0         0.0   0.0   0.0      0.0   0.0   0.0   0.0
52589     0.0         0.0   0.0   0.0      0.0   0.0   0.0   0.0
6437      0.0         0.0   0.0   0.0      0.0   0.0   0.0   0.0
33910     0.0         0.0   0.0   0.0      0.0   0.0   0.0   0.0

[12000 rows x 10000 columns] 5521     1
38538    2
30253    2
45733    3
2593     1
        ..
39184    2
18365    1
33132    2
22889    2
10375    1
Name: stars, Length: 48000, dtype: int64 31788    2
16832    1
8267     1
55958    3
11472    1
        ..
5042     1
35683    2
52589    3
6437     1
33910    2
Name: stars, Length: 12000, dtype: int64
```

5. Perceptron

In [22]:
```
def report_scores(clf, X_test, y_test, clf_name):
    y_pred = clf.predict(X_test)
    dict_report = classification_report(y_test, y_pred, output_dict=True)
    key_list = ['1', '2', '3', 'weighted avg']
    report_name_list = ['(Class 1)', '(Class 2)', '(Class 3)', '(Average)']
    for i, k in enumerate(key_list):
        temp_report = dict_report[k]
        precision = str(temp_report['precision'])
        recall = str(temp_report['recall'])
        f1_score = str(temp_report['f1-score'])
        print('Precision, Recall, and f1-score for the testing split for ' +
              precision + ',' + recall + ',' + f1_score)
```

In [23]:
```
clf_perceptron = Perceptron(random_state=r_state, penalty='elasticnet')
clf_perceptron.fit(X_train, y_train)
```

Out[23]:
```
Perceptron(penalty='elasticnet', random_state=555)
```

In [24]:
```
report_scores(clf_perceptron, X_test, y_test, 'Perceptron')
```

```
Precision, Recall, and f1-score for the testing split for Perceptron (Class
1):  0.6639344262295082,0.6075,0.6344647519582245
Precision, Recall, and f1-score for the testing split for Perceptron (Class
2):  0.4931712191872085,0.74025,0.5919632147141144
```

```
Precision, Recall, and f1-score for the testing split for Perceptron (Class
3):  0.8394691780821918,0.49025,0.6190025252525253
Precision, Recall, and f1-score for the testing split for Perceptron (Averag
e):  0.6655249411663028,0.6126666666666667,0.6151434973082881
```

6. SVM

In [25]:
```
clf_SVC = LinearSVC(random_state=r_state, multi_class='ovr', dual=True, max_i
clf_SVC.fit(X_train, y_train)
```

Out[25]:
```
LinearSVC(max_iter=50000, random_state=555)
```

In [26]:
```
report_scores(clf_SVC, X_test, y_test, 'SVM')
```

```
Precision, Recall, and f1-score for the testing split for SVM (Class 1):  0.67
18175128771156,0.68475,0.6782221121703603
Precision, Recall, and f1-score for the testing split for SVM (Class 2):  0.57
54189944134078,0.54075,0.5575460755251966
Precision, Recall, and f1-score for the testing split for SVM (Class 3):  0.71
54178674351584,0.74475,0.7297893189612935
Precision, Recall, and f1-score for the testing split for SVM (Average):  0.65
42181249085607,0.65675,0.6551858355522835
```

7. Logistic Regression

In [27]:
```
clf_logreg = LogisticRegression(random_state=r_state, multi_class='ovr', max_
clf_logreg.fit(X_train, y_train)
```

Out[27]:
```
LogisticRegression(max_iter=5000, multi_class='ovr', random_state=555)
```

In [28]:
```
report_scores(clf_logreg, X_test, y_test, 'Logistic Regression')
```

```
Precision, Recall, and f1-score for the testing split for Logistic Regression
(Class 1):  0.6900655817342726,0.71025,0.7000123198225946
Precision, Recall, and f1-score for the testing split for Logistic Regression
(Class 2):  0.5955862802446158,0.56,0.5772452003607782
Precision, Recall, and f1-score for the testing split for Logistic Regression
(Class 3):  0.7331392527899078,0.7555,0.744151686776656
Precision, Recall, and f1-score for the testing split for Logistic Regression
(Average):  0.6729303715895987,0.67525,0.6738030689866763
```

8. Naive Bayes

In [29]:
```
clf_nb = MultinomialNB()
clf_nb.fit(X_train, y_train)
```

Out[29]:
```
MultinomialNB()
```

In [30]:
```
report_scores(clf_nb, X_test, y_test, 'Naive Bayes')
```

```
Precision, Recall, and f1-score for the testing split for Naive Bayes (Class
1):  0.6969292389853138,0.6525,0.6739832149774048
Precision, Recall, and f1-score for the testing split for Naive Bayes (Class
2):  0.5639562529719448,0.593,0.5781135754326103
Precision, Recall, and f1-score for the testing split for Naive Bayes (Class
```

```
3):  0.7135095085206223,0.72225,0.7178531494595601
Precision, Recall, and f1-score for the testing split for Naive Bayes (Averag
e):  0.6581316668259604,0.6559166666666667,0.6566499799565251
```

References

- https://stackoverflow.com/questions/45999415/removing-html-tags-in-pandas
- https://stackoverflow.com/questions/11331982/how-to-remove-any-url-within-a-string-in-python
- https://www.machinelearningplus.com/nlp/lemmatization-examples-python/