CSCI:544 Assignment- 1 25- Jan-2025

Name: Atharva Bhide USC ID: 2223451631

---

1) Dataset Preparation:

Total "**Star_rating**" in the dataset:

Star Rating	Total Count
5	1582877
4	418381
3	193694
2	138391
1	306992

Number of 4 and 5 star ratings: 2001258
Number of 3 star ratings: 193694
Number of 1 and 2 star ratings: 445383

Review body	Star rating	sentiment
Haven't used yet, but I am sure I will like it.	5	1
Although this was labeled as "new" the...	1	0
Gorgeous colors and easy to use	4	1

Review Body	Star Rating
I purchased this system with 5 handsets to rep…	5.0
This is great, love the magnetic aspect. It h...	4.0
I ordered two boxes of these. My biggest compl...	2.0

Review Body	Sentiment
purchased system handset replace older att han...	1
great love magnetic aspect hold well fridge pr...	1
ordered two box biggest complaint color distri...	0

In this step only required columns of the table are kept. First I have identified the columns and then removed the values which are not from 1 to 5 using various operations like dropping NAN, other values like dates and extracting 100000 values from each of the classes.

2) Data Cleaning:
    a) **Avg. Length of reviews Before Cleaning: 318.24 characters**
    b) **Avg. Length of reviews After Cleaning: 301.33 characters**

Removed all of the non necessary words as per the requirements.

3) Preprocessing:

1. Original Review: I already love the Uni-ball Vision Elite pens. When I found about the BLK
series pens, I had to drop some cash and see how they were.

I'm quite happy with them. They have even lines and the colors are a nice,
dark color. I don't really care for the brown, but it's just a personal choice,
not a technical issue.
    a. After Cleaning: i already love the uniball vision elite pens when i found about the blkseries pens i had to drop some cash and see how they wereim quite happy with them they have even lines and the colors are a nicedark color i dont really care for the brown but its just a personal choicenot a technical issue
    b. After Preprocessing: already love uniball vision elite pen found blkseries pen drop cash see wereim quite happy even line color nicedark color dont really care brown personal choicenot technical issue
2. Original Review: metric/ English dual system. Very practical.
    a. After Cleaning: metric english dual system very practical
    b. After Preprocessing: metric english dual system practical

3. Original Review: Work like a charm as expected and will recommend it
    a. After Cleaning: work like a charm as expected and will recommend it
    b. After Preprocessing: work like charm expected recommend

    a) **Avg. Length of reviews before preprocessing 301.33 characters**
    b) **Avg. Length of reviews after preprocessing 191.93 characters**

Used nltk to process the information like removing stop words and lemmatization. After extracted features from the dataset.

4) TFIDF: Split the data into train: **80%- test: 20%**

Max Feature count is set to 10000.

5) Perceptron:
    a) Perceptron Training Metrics:
        i) Accuracy: 0.9062
        ii) Precision: 0.90363
        iii) Recall: 0.9062
        iv) F1 Score: 0.9062
    b) Perceptron Testing Metrics:

i) Accuracy: 0.8850
ii) Precision: 0.8851
iii) Recall: 0.8850
iv) F1 Score: 0.8850

6) SVM:

a) SVM Training Metrics:
i) Accuracy: 0.9341
ii) Precision: 0.9342
iii) Recall: 0.9341
iv) F1 Score: 0.9341

b) SVM Testing Metrics:
i) Accuracy: 0.9153
ii) Precision: 0.9153
iii) Recall: 0.9153
iv) F1 Score: 0.9153

7) Logistic Regression:

a) Logistic Regression Training Metrics:
i) Accuracy: 0.9236
ii) Precision: 0.9237
iii) Recall: 0.9236
iv) F1 Score: 0.9236

b) Logistic Regression Testing Metrics:
i) Accuracy: 0.9161
ii) Precision: 0.9161
iii) Recall: 0.9161
iv) F1 Score: 0.9160

8) Multinomial Naive Bayes:

a) Training Metrics:
i) Accuracy: 0.8816
ii) Precision: 0.8818
iii) Recall: 0.8816
iv) F1 Score: 0.8816

b) Testing Metrics:
i) Accuracy: 0.8742
ii) Precision: 0.8743
iii) Recall: 0.8742
iv) F1 Score: 0.8742

In [43]:
```
import re
import pandas as pd
import numpy as np
import nltk
nltk.download('punkt_tab')
nltk.download('wordnet')
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
nltk.download('wordnet')
nltk.download('stopwords')
import re
import unicodedata
from sklearn.model_selection import train_test_split
import scipy
from sklearn.svm import LinearSVC
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import Perceptron, LogisticRegression
from sklearn.naive_bayes import MultinomialNB
import pickle
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Read Data

In [44]:
```
data= pd.read_csv('E:/544/amazon_reviews_us_Office_Products_v1_00.tsv/amazon_review
                  sep='\t', on_bad_lines='skip',
                  memory_map=True,
                  nrows=10)

print(data.head())
```

```
  marketplace  ...  review_date
0          US  ...   2015-08-31
1          US  ...   2015-08-31
2          US  ...   2015-08-31
3          US  ...   2015-08-31
4          US  ...   2015-08-31

[5 rows x 15 columns]
```

Keep Reviews and Ratings

In [45]:
```
df1 = pd.read_csv('E:/544/amazon_reviews_us_Office_Products_v1_00.tsv/amazon_review
                  sep='\t', on_bad_lines='skip',
                    usecols=['star_rating','review_headline', 'review_body'],
                  memory_map=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_19628\1650238296.py:1: DtypeWarning: Column
s (7) have mixed types. Specify dtype option on import or set low_memory=False.
  df1 = pd.read_csv('E:/544/amazon_reviews_us_Office_Products_v1_00.tsv/amazon_revie
ws_us_Office_Products_v1_00.tsv',
```

In [46]:
```
df1.head()
```

Out[46]:

	star_rating	review_headline	review_body
0	5	Five Stars	Great product.
1	5	Phfffffffft, Phfffffft. Lots of air, and it's C...	What's to say about this commodity item except...
2	5	but I am sure I will like it.	Haven't used yet, but I am sure I will like it.
3	1	and the shredder was dirty and the bin was par...	Although this was labeled as "new" the...
4	4	Four Stars	Gorgeous colors and easy to use

In [47]:
```
df_req1 = df1.loc[:,['review_body', 'star_rating']]
df_req1.head()
```

Out[47]:

	review_body	star_rating
0	Great product.	5
1	What's to say about this commodity item except...	5
2	Haven't used yet, but I am sure I will like it.	5
3	Although this was labeled as "new" the...	1
4	Gorgeous colors and easy to use	4

In [48]:
```
df_req1['star_rating'].unique()
```

Out[48]:
```
array([5, 1, 4, 2, 3, '5', '1', '3', '4', '2', '2015-06-05', '2015-02-11',
       nan, '2014-02-14'], dtype=object)
```

In [49]:
```
df_req1['star_rating']= pd.to_numeric(df1['star_rating'], errors='coerce')
df_req1['star_rating'].unique()
```

Out[49]:
```
array([ 5.,  1.,  4.,  2.,  3., nan])
```

In [50]:
```
df_req1.dropna(subset=["star_rating"], inplace=True)
df_req1['star_rating'].unique()
```

Out[50]:
```
array([5., 1., 4., 2., 3.])
```

In [51]:
```
df_req1["star_rating"].value_counts()
```

Out[51]:
```
star_rating
5.0    1582877
4.0     418381
1.0     306992
3.0     193694
2.0     138391
Name: count, dtype: int64
```

In [52]:
```
counts_4_5 = df_req1[df_req1['star_rating'].isin([4, 5])].shape[0]
counts_3 = df_req1[df_req1['star_rating'] == 3].shape[0]
counts_1_2 = df_req1[df_req1['star_rating'].isin([1, 2])].shape[0]

print("Number of 4 and 5 star ratings:", counts_4_5)
print("Number of 3 star ratings:", counts_3)
print("Number of 1 and 2 star ratings:", counts_1_2)
```

```
Number of 4 and 5 star ratings: 2001258
Number of 3 star ratings: 193694
Number of 1 and 2 star ratings: 445383
```

In [53]:
```
df_req1 = df_req1[df_req1['star_rating']!=3]
df_req1['sentiment']= df_req1['star_rating'].apply(lambda x:0 if x<3 else 1 )
print("Shape of unfiltered dataframe:", df1.shape)
print("Shape of the filtered dataframe:", df_req1.shape)

df_req1.head()
```

```
Shape of unfiltered dataframe: (2640352, 3)
Shape of the filtered dataframe: (2446641, 3)
```

Out[53]:

	review_body	star_rating	sentiment
0	Great product.	5.0	1
1	What's to say about this commodity item except...	5.0	1
2	Haven't used yet, but I am sure I will like it.	5.0	1
3	Although this was labeled as "new" the...	1.0	0
4	Gorgeous colors and easy to use	4.0	1

In [54]:
```
# Taking 100000 values as per the requirement
filtered_df = pd.concat(
    [
        df_req1.query('sentiment==0').sample(
            n=100000, random_state=3
        ),
        df_req1.query('sentiment==1').sample(
            n=100000, random_state =3
```

```
        )
    ],
    ignore_index=True
).sample(frac =1, random_state = 3)
filtered_df.head()
```

Out[54]:

	review_body	star_rating	sentiment
158707	I purchased this system with 5 handsets to rep...	5.0	1
135082	This is great, love the magnetic aspect. It h...	4.0	1
190077	These pens are handy, and I love the fact that...	5.0	1
20142	DO NOT BUY!!!!!!!! I wanted to surprise my nep...	1.0	0
21163	I ordered two boxes of these. My biggest compl...	2.0	0

In [55]:
```
filtered_df = filtered_df[['review_body', 'sentiment']]
filtered_df
filtered_df['sentiment'].value_counts()
```

Out[55]:
```
sentiment
1    100000
0    100000
Name: count, dtype: int64
```

Data Cleaning

In [56]:
```
#Expanding short forms

contractions_dict = { 'arent': 'are not', 'wont': 'will not', 'cant': 'can not', 'd
                      'wouldnt': 'would not', 'couldnt': 'could not', 'shouldnt': 's
                      'didnt': 'did not', 'doesnt': 'does not',
                      'theyre': "they are"}
#character with diacritics
def unicode_to_ascii(s):
    return ''.join(c for c in unicodedata.normalize('NFD', s)
                   if unicodedata.category(c)!='Mn')
#Using the contraction function
def expand_contractions(text):
    if isinstance(text, str):
        for word in contractions_dict:
            text = text.replace(word, contractions_dict[word])
        return text



def cleaned_text(text):
    text = unicode_to_ascii(text.lower().strip())

    #URLs
    text = re.sub(r"\bhttps?:\/\/\S+|www\.\S+", " ", text)
    text = re.sub(
```

```
        r"[a-zA-Z0-9_\-\.]+@[a-zA-Z0-9_\-\.]+\.[a-zA-Z]{2,5}", " ", text
    )
    # Remove HTML tags with empty string
    text = re.sub(r"<.*?>", "", text)
    text = expand_contractions(text)
    # removes all non-alphabetical characters

    text = re.sub(r"[^a-zA-Z\s]+", "", text)
    # creating a space between a word and the punctuation following it
    text = re.sub(r"([?.!,¿])", r" \1 ", text)
    text = re.sub(r'[" "]+', " ", text)
    # remove extra spaces
    text = re.sub(" +", " ", text)

    text = text.strip()
    return text

final_clean = np.vectorize(cleaned_text)
```

In [57]:
```
filtered_df["review_body"] = filtered_df["review_body"].fillna("")
avg_len_bef_clean = filtered_df["review_body"].apply(len).mean()
filtered_df["review_body"] =filtered_df["review_body"].apply(final_clean)

avg_len_aft_clean = filtered_df["review_body"].apply(len).mean()

print(f'Avg. Length of reviews Before Cleaning: {avg_len_bef_clean:.2f} characters'
print(f'Avg. Length of reviews After Cleaning: {avg_len_aft_clean:.2f} characters')
```

```
Avg. Length of reviews Before Cleaning: 318.24 characters
Avg. Length of reviews After Cleaning: 301.33 characters
```

In [58]:
```
pos_rev =filtered_df[filtered_df['sentiment']==1]
neg_rev = filtered_df[filtered_df['sentiment']==0]
```

In [59]:
```
pos_rev.sample(3)
```

Out[59]:

	review_body	sentiment
143142	i havent needed to install and use this produc...	1
100570	got a b never seen before in my life but it wo...	1
114375	compatible and prints nicely i cannot attest t...	1

In [60]:
```
neg_rev.sample(3)
```

Out[60]:

	review_body	sentiment
23621	the absolute worse printing experience i do no...	0
27845	i was very disappointed in the picture quality...	0
28797	it might work fine in other printers but not o...	0

Pre-processing

remove the stop words

In [61]:
```
req_stpwds = set(stopwords.words('english'))
#stop words removal and lemmatization
pattern = re.compile(r'\b('+r'|'.join(req_stpwds)+r')\b\s*')

def lem_text(text):
    lemmatizer = WordNetLemmatizer()
    words = word_tokenize(text)
    lemmatized_words= [lemmatizer.lemmatize(word) for word in words]
    return ' '.join(lemmatized_words)

def preprocess_text(text):
    text = pattern.sub('', text)
    text = lem_text(text)
    return text
preprocess_text_vect = np.vectorize(preprocess_text)
```

perform lemmatization

In [62]:
```
#Calculate avg length of reviews in terms of character length

avg_len_bef_preprocess = avg_len_aft_clean
filtered_df = filtered_df.dropna(subset=["review_body"])
filtered_df["review_body"] = filtered_df["review_body"].astype(str)

filtered_df["review_body"]= filtered_df["review_body"].apply(preprocess_text_vect)

avg_len_after_preprocess = filtered_df['review_body'].apply(len).mean()

print(f"Avg. Length of reviews before preprocessing {avg_len_bef_preprocess:.2f} ch
print(f"Avg. Length of reviews after preprocessing {avg_len_after_preprocess:.2f} c
```

```
Avg. Length of reviews before preprocessing 301.33 characters
Avg. Length of reviews after preprocessing 191.93 characters
```

TF-IDF Feature Extraction

In [63]:
```
tfidf_vectorizer = TfidfVectorizer(max_features=10000)
docs = pos_rev['review_body'].values.tolist() + neg_rev['review_body'].values.tolis
X = tfidf_vectorizer.fit_transform(docs)

with open('tfidf_vectorizer.pkl', 'wb') as f:
    pickle.dump(tfidf_vectorizer, f)

pos_x, neg_x= X[:100000], X[100000:]
```

SPLIT

In [64]:
```
# Split positive reviews
pos_x_train, pos_x_test = train_test_split(pos_x, test_size=0.2, random_state=3)

# Split negative reviews
neg_x_train, neg_x_test = train_test_split(neg_x, test_size=0.2, random_state=3)

print(f"Positive reviews - Training set: {pos_x_train.shape}, Testing set: {pos_x_t
print(f"Negative reviews - Training set: {neg_x_train.shape}, Testing set: {neg_x_t
X_train = scipy.sparse.vstack([pos_x_train, neg_x_train])

Y_train = [1] * pos_x_train.shape[0] + [0] * neg_x_train.shape[0]

X_test = scipy.sparse.vstack([pos_x_test, neg_x_test])
Y_test = [1] * pos_x_test.shape[0] + [0] * neg_x_test.shape[0]
```

```
Positive reviews - Training set: (80000, 10000), Testing set: (20000, 10000)
Negative reviews - Training set: (80000, 10000), Testing set: (20000, 10000)
```

Perceptron

In [65]:
```
# Perceptron Model with different variable names
perceptron_model = Perceptron(random_state=3)

# Fit the model
perceptron_model.fit(X_train, Y_train)

# Predict on training and testing data
Y_train_pred_perceptron = perceptron_model.predict(X_train)
Y_test_pred_perceptron = perceptron_model.predict(X_test)
#Following function can be used in all of the below models
# Evaluate the model
def evaluate_model(Y_true, Y_pred):
    print(f"Accuracy: {accuracy_score(Y_true, Y_pred):.4f}")
    print(f"Precision: {precision_score(Y_true, Y_pred, average='weighted'):.4f}")
    print(f"Recall: {recall_score(Y_true, Y_pred, average='weighted'):.4f}")
    print(f"F1 Score: {f1_score(Y_true, Y_pred, average='weighted'):.4f}")

# Eval on training data
print("Perceptron Training Metrics:")
evaluate_model(Y_train, Y_train_pred_perceptron)

# Eval on testing data
print("\nPerceptron Testing Metrics:")
evaluate_model(Y_test, Y_test_pred_perceptron)
```

```
Perceptron Training Metrics:
Accuracy: 0.9062
Precision: 0.9063
Recall: 0.9062
F1 Score: 0.9062

Perceptron Testing Metrics:
Accuracy: 0.8850
Precision: 0.8851
Recall: 0.8850
F1 Score: 0.8850
```

SVM

In [66]:
```
# SVM

svm_model = LinearSVC(random_state=3, verbose=False)
svm_model.fit(X_train, Y_train)
#Perdict
Y_train_pred_svm = svm_model.predict(X_train)
Y_test_pred_svm = svm_model.predict(X_test)
#Eval
print("SVM Training Metrics:")
evaluate_model(Y_train, Y_train_pred_svm)
print("\nSVM Testing Metrics:")
evaluate_model(Y_test, Y_test_pred_svm)
```

```
SVM Training Metrics:
Accuracy: 0.9341
Precision: 0.9342
Recall: 0.9341
F1 Score: 0.9341

SVM Testing Metrics:
Accuracy: 0.9153
Precision: 0.9153
Recall: 0.9153
F1 Score: 0.9153
```

Logistic Regression

In [67]:
```
##
# Logistic Regression
logreg_model = LogisticRegression(max_iter=1000, random_state=3)
logreg_model.fit(X_train, Y_train)

# Predictions
Y_train_pred_logreg = logreg_model.predict(X_train)
Y_test_pred_logreg = logreg_model.predict(X_test)

# Evaluation
print("Logistic Regression Training Metrics:")
```

```
evaluate_model(Y_train, Y_train_pred_logreg)

print("\nLogistic Regression Testing Metrics:")
evaluate_model(Y_test, Y_test_pred_logreg)
```

```
Logistic Regression Training Metrics:
Accuracy: 0.9236
Precision: 0.9237
Recall: 0.9236
F1 Score: 0.9236

Logistic Regression Testing Metrics:
Accuracy: 0.9161
Precision: 0.9161
Recall: 0.9161
F1 Score: 0.9160
```

Naive Bayes

In [68]:
```
# Naive Bayes
nb_model = MultinomialNB()
nb_model.fit(X_train, Y_train)

# Predictions
Y_train_pred_nb = nb_model.predict(X_train)
Y_test_pred_nb = nb_model.predict(X_test)
#Eval
print("Naive Bayes Training Matrics:")
evaluate_model(Y_train, Y_train_pred_nb)
print("\nNaive Bayes Testing Matrics:")
evaluate_model(Y_test, Y_test_pred_nb)
```

```
Naive Bayes Training Matrics:
Accuracy: 0.8816
Precision: 0.8818
Recall: 0.8816
F1 Score: 0.8816

Naive Bayes Testing Matrics:
Accuracy: 0.8742
Precision: 0.8743
Recall: 0.8742
F1 Score: 0.8742
```

In [69]:
```
# Sample 3 rows from the original dataframe with the cleaned text and preprocessed
sample_data = df_req1.sample(3, random_state=5).copy()

for idx, row in sample_data.iterrows():
    original_text = row["review_body"]
    cleaned = cleaned_text(original_text)       # uses the 'cleaned_text' function
    preprocessed = preprocess_text(cleaned)     # uses the 'preprocess_text' functi
    print("Original Review:", original_text)
    print("After Cleaning:", cleaned)
    print("After Preprocessing:", preprocessed, "\n")
```

Original Review: I already love the Uni-ball Vision Elite pens. When I found about t
he BLK
series pens, I had to drop some cash and see how they were.

I'm quite happy with them. They have even lines and the colors are a nice,
dark
color. I don't really care for the brown, but it's just a personal choice,
not
a technical issue.
After Cleaning: i already love the uniball vision elite pens when i found about the
blkseries pens i had to drop some cash and see how they wereim quite happy with them
they have even lines and the colors are a nicedark color i dont really care for the
brown but its just a personal choicenot a technical issue
After Preprocessing: already love uniball vision elite pen found blkseries pen drop
cash see wereim quite happy even line color nicedark color dont really care brown pe
rsonal choicenot technical issue

Original Review: metric/ English dual system. Very practical.
After Cleaning: metric english dual system very practical
After Preprocessing: metric english dual system practical

Original Review: Work like a charm as expected and will recommend it
After Cleaning: work like a charm as expected and will recommend it
After Preprocessing: work like charm expected recommend