Assignment 1 Report

Yuhang Xiao – 6913860906 – yxiao776@usc.edu

1. Dataset Preparation

Read the data from provided URL using read_csv in Pandas. Only kept the 'star_rating', 'review_headline' and 'review_body' columns in the dataframe and dropped rows with NaN values. Below are three sample reviews.

	star_rating	review_headline	review_body
246582	5	Fast Shipping!	Fast Shipping. Works Perfectly!
2639050	5	it works!	5 stars based on transmission clarity. compar...
697475	5	Five Stars	Love it, looks great on my wall!

The statistics of the ratings:
All reviews: 2640037
1 rating reviews: 306962, 11.627185528081615%
2 rating reviews: 138380, 5.241593204943719%
3 rating reviews: 193674, 7.336033548014668%
4 rating reviews: 418339, 15.845952159003833%
5 rating reviews: 1582682, 59.949235559956165%

Then, mapped the ratings more than 3 to 1, ratings less than or equal to 2 to 0 and ratings equal to 3 to neutral reviews. Below are the number of reviews for each of these three classes.

All reviews: 2640037
Positive reviews: 2001021
Negative reviews: 445342
Neutral reviews: 193674

Discarded neutral reviews. Sample 100,000 rows each from positive and negative reviews. Merged 'review_headline' and 'review_body' to 'review' column. Changed the column name from 'star_rating' to 'label'. For reproducibility, processes with randomness are set a fixed seed.

2. Data Cleaning

Used Pandas built-in function to transfer all reviews into lower case.
Removed the URLs using regex library via matching specific patterns.
Removed HTML using BeautifulSoup4 to parse the content and keep the text only.
Removed non-alphabetic characters and extra spaces using regex again.
Finally, performed contractions using contractions library.

Average length before and after data cleaning: 343.482135 and 326.426925

3. Preprocessing
Used English stopwords list and word_tokenize() function in NLTK package to remove all stopwords by traversing each word.
Used WordNetLemmatizer() in NLTK to lemmatize earch word by traversal.

Average length before and after preprocessing: 326.426925 and 204.287135

Three sample reviews before and after data cleaning + preprocessing:

	label	review
1448408	1	Works GREAT! So many great uses! Initially wa...
2070049	0	not good ink. Toner ink is not clear and look ...
2471912	1	An almost perfect product! The Bic Wite-Out Co...

	label	review
1448408	1	work great many great us initially bought barc...
2070049	0	good ink toner ink clear look like dry ink pri...
2471912	1	almost perfect product bic wite correction tap...

4. Feature Extraction
Used TfidfVectorizer() in sklearn package with default args other than ngram=(1,3) to include short phrase features as well. Then, used train_test_split() in sklearn to split training and test set with a ratio of 8:2. Set the arg stratify=labels to keep the same ratio of positive and negative samples in training and test set as the original dataset.

5. Perceptron
Used the Perceptron() in sklearn package with all default args.

The statistics of the model performance:
Train accuracy: 0.999125
Train precision: 0.9985765657776446
Train recall: 0.999675
Train f1 score: 0.9991254809854581
Test accuracy: 0.931675
Test precision: 0.9258199753390876
Test recall: 0.93855
Test f1 score: 0.9321415270018621

6. SVM
Used LinearSVC() in sklearn with all default args.

The statistics of the model performance:
Train accuracy: 0.99944375
Train precision: 0.999225338914225
Train recall: 0.9996625
Train f1 score: 0.9994438716530759
Test accuracy: 0.937675
Test precision: 0.9408713170486024
Test recall: 0.93405
Test f1 score: 0.9374482499059089

7. Logistic Regression
Used LogisticRegression() in sklearn with default args other than solver='saga' to accelerate training process.

The statistics of the model performance:
Train accuracy: 0.9618125
Train precision: 0.9673742536180548
Train recall: 0.9558625
Train f1 score: 0.9615839243498818
Test accuracy: 0.92725
Test precision: 0.9341089209510263
Test recall: 0.91935
Test f1 score: 0.9266706985182944

8. Multinomial Naïve Bayes
Used MultinomialNB() in sklearn with all default args.

The statistics of the model performance:
Train accuracy: 0.97436875
Train precision: 0.9905254313966264
Train recall: 0.9579
Train f1 score: 0.9739395672481175
Test accuracy: 0.910625
Test precision: 0.9422693736873283
Test recall: 0.87485
Test f1 score: 0.9073089787134745

These models all achieved good performances even with default settings after careful data cleaning and preprocessing.
The Python code used the Python version of 3.11.4. It should be executed with 'amazon_reviews_us_Office_Products_v1_00.tsv' file prepared in the same directory.

In [48]:
```
import pandas as pd
import numpy as np
import nltk
nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('punkt')
import re
from bs4 import BeautifulSoup
import contractions
import warnings
```

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

In [49]:
```
! pip install bs4 # in case you don't have it installed
! pip install contractions # in case you don't have it installed
! pip install scikit-learn # in case you don't have it installed

# Dataset: https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us
_Beauty_v1_00.tsv.gz
```

```
Requirement already satisfied: bs4 in /Users/xiaoyuhang/opt/anaconda3/envs/u
sc/lib/python3.11/site-packages (0.0.2)
Requirement already satisfied: beautifulsoup4 in /Users/xiaoyuhang/opt/anaco
nda3/envs/usc/lib/python3.11/site-packages (from bs4) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /Users/xiaoyuhang/opt/anacon
da3/envs/usc/lib/python3.11/site-packages (from beautifulsoup4->bs4) (2.4)
Requirement already satisfied: contractions in /Users/xiaoyuhang/opt/anacond
a3/envs/usc/lib/python3.11/site-packages (0.1.73)
Requirement already satisfied: textsearch>=0.0.21 in /Users/xiaoyuhang/opt/a
naconda3/envs/usc/lib/python3.11/site-packages (from contractions) (0.0.24)
Requirement already satisfied: anyascii in /Users/xiaoyuhang/opt/anaconda3/e
nvs/usc/lib/python3.11/site-packages (from textsearch>=0.0.21->contractions)
(0.3.2)
Requirement already satisfied: pyahocorasick in /Users/xiaoyuhang/opt/anacon
da3/envs/usc/lib/python3.11/site-packages (from textsearch>=0.0.21->contract
ions) (2.0.0)
Requirement already satisfied: scikit-learn in /Users/xiaoyuhang/opt/anacond
a3/envs/usc/lib/python3.11/site-packages (1.3.2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in /Users/xiaoyuhang/opt/a
naconda3/envs/usc/lib/python3.11/site-packages (from scikit-learn) (1.26.2)
Requirement already satisfied: scipy>=1.5.0 in /Users/xiaoyuhang/opt/anacond
a3/envs/usc/lib/python3.11/site-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /Users/xiaoyuhang/opt/anacon
da3/envs/usc/lib/python3.11/site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/xiaoyuhang/op
t/anaconda3/envs/usc/lib/python3.11/site-packages (from scikit-learn) (3.2.
0)
```

Read Data

In [50]:

```
dtype={7: object}
url = 'https://web.archive.org/web/20201127142707if_/https://s3.amazonaws.co
m/amazon-reviews-pds/tsv/amazon_reviews_us_Office_Products_v1_00.tsv.gz'
data = pd.read_csv(url, sep='\t', on_bad_lines='skip', dtype=dtype)
# path = 'amazon_reviews_us_Office_Products_v1_00.tsv.gz'
# data = pd.read_csv(path, sep='\t', on_bad_lines='skip', dtype=dtype)
# unzipped_path = 'amazon_reviews_us_Office_Products_v1_00.tsv'
# data = pd.read_csv(unzipped_path, sep='\t', on_bad_lines='skip', dtype=dty
pe)
print("Rows: ", data.shape[0])
data.head()

# def cannot_convert_to_float(x):
#     try:
#         float(x)  # Try converting to float
#         return False
#     except ValueError:  # If conversion fails, return False
#         return True

# column_data = data.iloc[:, 7]
# unique_types = set(map(type, column_data))
# print(unique_types)
# filtered_data = column_data.apply(lambda x: x if isinstance(x, (float, st
r)) else None)
# filtered_data = column_data.apply(lambda x: x if isinstance(x, float) and
x % 1 != 0 else None)
# filtered_data = column_data.apply(lambda x: x if isinstance(x, str) and (c
annot_convert_to_float(x) or float(x) % 1 != 0) else None)
# print(filtered_data.dropna())
```

```
Rows:  2640254
```

Out[50]:

	marketplace	customer_id	review_id	product_id	product_parent	product_title	product_
0	US	43081963	R18RVCKGH1SSI9	B001BM2MAC	307809868	Scotch Cushion Wrap 7961, 12 Inches x 100 Feet	Office
1	US	10951564	R3L4L6LW1PUOFY	B00DZYEXPQ	75004341	Dust-Off Compressed Gas Duster, Pack of 4	Office
2	US	21143145	R2J8AWXWTDX2TF	B00RTMUHDW	529689027	Amram Tagger Standard Tag Attaching Tagging Gu...	Office
3	US	52782374	R1PR37BR7G3M6A	B00D7H8XB6	868449945	AmazonBasics 12-Sheet High-Security Micro-Cut ...	Office
4	US	24045652	R3BDDDZMZBZDPU	B001XCWP34	33521401	Derwent Colored Pencils, Inktense Ink Pencils,...	Office

Keep Reviews and Ratings

In [51]:
```
data = data[['star_rating', 'review_headline', 'review_body']]
data.dropna(inplace=True)
print("Rows: ", data.shape[0])
print("Three sample reviews:")
data.sample(3, random_state=1)
```

```
Rows:  2640037
Three sample reviews:
```

Out[51]:

	star_rating	review_headline	review_body
246582	5	Fast Shipping!	Fast Shipping. Works Perfectly!
2639050	5	it works!	5 stars based on transmission clarity. compar...
697475	5	Five Stars	Love it, looks great on my wall!

data statistics

In [52]:
```
data['star_rating'] = pd.to_numeric(data['star_rating'], errors='coerce')
print("All reviews: ", data.shape[0])
print(f"1 rating reviews: {data[data['star_rating'] == 1].shape[0]}, {100 * data[data['star_rating'] == 1].shape[0]/data.shape[0]}%")
print(f"2 rating reviews: {data[data['star_rating'] == 2].shape[0]}, {100 * data[data['star_rating'] == 2].shape[0]/data.shape[0]}%")
print(f"3 rating reviews: {data[data['star_rating'] == 3].shape[0]}, {100 * data[data['star_rating'] == 3].shape[0]/data.shape[0]}%")
print(f"4 rating reviews: {data[data['star_rating'] == 4].shape[0]}, {100 * data[data['star_rating'] == 4].shape[0]/data.shape[0]}%")
print(f"5 rating reviews: {data[data['star_rating'] == 5].shape[0]}, {100 * data[data['star_rating'] == 5].shape[0]/data.shape[0]}%")
```

```
All reviews:  2640037
1 rating reviews: 306962, 11.627185528081615%
2 rating reviews: 138380, 5.241593204943719%
3 rating reviews: 193674, 7.336033548014668%
4 rating reviews: 418339, 15.845952159003833%
5 rating reviews: 1582682, 59.949235559956165%
```

We form three classes and select 100000 reviews randomly from positive and negtive class.

In [53]:
```python
pos_reviews = data[data['star_rating'] > 3]
neg_reviews = data[data['star_rating'] <= 2]
neu_reviews = data[data['star_rating'] == 3]
print("All reviews: ", data.shape[0])
print("Positive reviews: ", pos_reviews.shape[0])
print("Negative reviews: ", neg_reviews.shape[0])
print("Neutral reviews: ", neu_reviews.shape[0])
print("All reviews == Positive + Negative + Neutral: ", data.shape[0] == pos
_reviews.shape[0] + neg_reviews.shape[0] + neu_reviews.shape[0])
```

```
All reviews:  2640037
Positive reviews:  2001021
Negative reviews:  445342
Neutral reviews:  193674
All reviews == Positive + Negative + Neutral:  True
```

In [54]:
```python
pos_samples = pos_reviews.sample(n=100000, random_state=0)
neg_samples = neg_reviews.sample(n=100000, random_state=0)
pos_samples['star_rating'] = 1
neg_samples['star_rating'] = 0
pos_samples.rename(columns={'star_rating': 'label'}, inplace=True)
neg_samples.rename(columns={'star_rating': 'label'}, inplace=True)
dataset = pd.concat([pos_samples, neg_samples])
dataset['review'] = dataset[['review_headline', 'review_body']].agg(' '.joi
n, axis=1)
dataset.drop(columns=['review_headline', 'review_body'], inplace=True)
print("Rows: ", dataset.shape[0])
print("Three sample reviews:")
dataset.sample(3, random_state=1)
```

```
Rows:  200000
Three sample reviews:
```

Out[54]:

	label	review
1448408	1	Works GREAT! So many great uses! Initially wa...
2070049	0	not good ink. Toner ink is not clear and look ...
2471912	1	An almost perfect product! The Bic Wite-Out Co...

Data Cleaning

In [55]:
```
print("Average length of reviews before cleaning: ", dataset['review'].str.len().mean())
print("Three sample reviews:")
dataset.sample(3,random_state=1)
```

```
Average length of reviews before cleaning:  343.482135
Three sample reviews:
```

Out[55]:

	label	review
1448408	1	Works GREAT! So many great uses! Initially wa...
2070049	0	not good ink. Toner ink is not clear and look ...
2471912	1	An almost perfect product! The Bic Wite-Out Co...

In [56]:
```
from bs4 import MarkupResemblesLocatorWarning
warnings.filterwarnings("ignore", category=MarkupResemblesLocatorWarning)
dataset['review'] = dataset['review'].apply(str.lower)
dataset['review'] = dataset['review'].apply(lambda x: re.sub(r'https?://\S+|www\.\S+', '', x))
dataset['review'] = dataset['review'].apply(lambda x: BeautifulSoup(x, "html.parser").text)
dataset['review'] = dataset['review'].apply(lambda x: re.sub(r'[^a-zA-Z]+', ' ', x))
dataset['review'] = dataset['review'].apply(lambda x: re.sub(r'\s+', ' ', x).strip())
dataset['review'] = dataset['review'].apply(lambda x: ' '.join([contractions.fix(word) for word in x.split()]))

print("Average length of reviews after cleaning: ", dataset['review'].str.len().mean())
print("Three sample reviews:")
dataset.sample(3,random_state=1)
```

```
Average length of reviews after cleaning:  326.426925
Three sample reviews:
```

Out[56]:

	label	review
1448408	1	works great so many great uses initially was b...
2070049	0	not good ink toner ink is not clear and look l...
2471912	1	an almost perfect product the bic wite out cor...

Pre-processing

In [57]:
```
print("Average length of reviews before preprocessing: ", dataset['review'].str.len().mean())
print("Three sample reviews:")
dataset.sample(3,random_state=1)
```

```
Average length of reviews before preprocessing:  326.426925
Three sample reviews:
```

Out[57]:

	label	review
1448408	1	works great so many great uses initially was b...
2070049	0	not good ink toner ink is not clear and look l...
2471912	1	an almost perfect product the bic wite out cor...

remove the stop words

In [58]:
```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

stopwords = set(stopwords.words('english'))
dataset['review'] = dataset['review'].apply(lambda x: ' '.join([word for word in word_tokenize(x) if word not in stopwords]))
print("Three sample reviews:")
dataset.sample(3,random_state=1)
```

```
Three sample reviews:
```

Out[58]:

	label	review
1448408	1	works great many great uses initially bought b...
2070049	0	good ink toner ink clear look like dry ink pri...
2471912	1	almost perfect product bic wite correction tap...

perform lemmatization

In [59]:
```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
dataset['review'] = dataset['review'].apply(lambda x: ' '.join([lemmatizer.lemmatize(word) for word in word_tokenize(x)]))
print("Average length of reviews after preprocessing: ", dataset['review'].str.len().mean())
print("Three sample reviews:")
dataset.sample(3,random_state=1)
```

```
Average length of reviews after preprocessing:  204.287135
Three sample reviews:
```

Out[59]:

	label	review
1448408	1	work great many great us initially bought barc...
2070049	0	good ink toner ink clear look like dry ink pri...
2471912	1	almost perfect product bic wite correction tap...

TF-IDF Feature Extraction

In [60]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(ngram_range=(1, 3))
vectors = vectorizer.fit_transform(dataset['review'])
print(vectors.shape)
```

```
(200000, 6740105)
```

Train-Test Split

In [61]:
```
from sklearn.model_selection import train_test_split
labels = dataset['label']
x_train, x_test, y_train, y_test = train_test_split(vectors, labels, test_si
ze=0.2, random_state=42, stratify=labels)
print("Training set size: ", x_train.shape, y_test.shape)
print("Test set size: ", x_test.shape, y_test.shape)
```

```
Training set size:  (160000, 6740105) (40000,)
Test set size:  (40000, 6740105) (40000,)
```

Statistics Printing

In [62]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f
1_score
def summary(y_train, pred_train, y_test, pred_test):
    print("Train accuracy: ", accuracy_score(y_train, pred_train))
    print("Train precision: ", precision_score(y_train, pred_train))
    print("Train recall: ", recall_score(y_train, pred_train))
    print("Train f1 score: ", f1_score(y_train, pred_train))
    print("Test accuracy: ", accuracy_score(y_test, pred_test))
    print("Test precision: ", precision_score(y_test, pred_test))
    print("Test recall: ", recall_score(y_test, pred_test))
    print("Test f1 score: ", f1_score(y_test, pred_test))
```

Perceptron

In [63]:
```
from sklearn.linear_model import Perceptron
perceptron = Perceptron()
perceptron.fit(x_train, y_train)
```

Out[63]:
```
▾ Perceptron
Perceptron()
```

In [64]:
```
pred_train = perceptron.predict(x_train)
pred_test = perceptron.predict(x_test)
summary(y_train, pred_train, y_test, pred_test)
```

```
Train accuracy:  0.999125
Train precision:  0.9985765657776446
Train recall:  0.999675
Train f1 score:  0.9991254809854581
Test accuracy:  0.931675
Test precision:  0.9258199753390876
Test recall:  0.93855
Test f1 score:  0.9321415270018621
```

SVM

In [65]:
```
from sklearn.svm import LinearSVC
svm = LinearSVC(dual='auto', random_state=0)
svm.fit(x_train, y_train)
```

Out[65]:
```
▾ LinearSVC
LinearSVC(dual='auto', random_state=0)
```

In [66]:
```
pred_train = svm.predict(x_train)
pred_test = svm.predict(x_test)
summary(y_train, pred_train, y_test, pred_test)
```

```
Train accuracy:  0.99944375
Train precision:  0.999225338914225
Train recall:  0.9996625
Train f1 score:  0.9994438716530759
Test accuracy:  0.937675
Test precision:  0.9408713170486024
Test recall:  0.93405
Test f1 score:  0.9374482499059089
```

Logistic Regression

In [67]:
```
from sklearn.linear_model import LogisticRegression
logit = LogisticRegression(solver='saga', random_state=0)
logit.fit(x_train, y_train)
```

Out[67]:
```
▾ LogisticRegression
LogisticRegression(random_state=0, solver='saga')
```

In [68]:
```
pred_train = logit.predict(x_train)
pred_test = logit.predict(x_test)
summary(y_train, pred_train, y_test, pred_test)
```

```
Train accuracy:  0.9618125
Train precision:  0.9673742536180548
Train recall:  0.9558625
Train f1 score:  0.9615839243498818
Test accuracy:  0.92725
Test precision:  0.9341089209510263
Test recall:  0.91935
Test f1 score:  0.9266706985182944
```

Naive Bayes

In [69]:
```
from sklearn.naive_bayes import MultinomialNB
naive_bayes = MultinomialNB()
naive_bayes.fit(x_train, y_train)
```

Out[69]:
```
▾ MultinomialNB
MultinomialNB()
```

In [70]:
```
pred_train = naive_bayes.predict(x_train)
pred_test = naive_bayes.predict(x_test)
summary(y_train, pred_train, y_test, pred_test)
```

```
Train accuracy:  0.97436875
Train precision:  0.9905254313966264
Train recall:  0.9579
Train f1 score:  0.9739395672481175
Test accuracy:  0.910625
Test precision:  0.9422693736873283
Test recall:  0.87485
Test f1 score:  0.9073089787134745
```