DEVYAN BISWAS REPORT

Some notes:

- Python version is 3.7.5

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
```

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_Jewelry_v1_00.tsv.gz
```

```
Requirement already satisfied: bs4 in /Users/devyanbiswas/Desktop/CSCI544/Homeworks/544/lib/python3.7/site-packages (0.0.1)
Requirement already satisfied: beautifulsoup4 in /Users/devyanbiswas/Desktop/CSCI544/Homeworks/544/lib/python3.7/site-packages (from bs4) (4.11.1)
Requirement already satisfied: soupsieve>1.2 in /Users/devyanbiswas/Desktop/CSCI544/Homeworks/544/lib/python3.7/site-packages (from beautifulsoup4->bs4) (2.3.2.post1)
WARNING: You are using pip version 19.2.3, however version 22.2.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Requirement already satisfied: contractions in /Users/devyanbiswas/Desktop/CSCI544/Homeworks/544/lib/python3.7/site-packages (0.1.72)
Requirement already satisfied: textsearch>=0.0.21 in /Users/devyanbiswas/Desktop/CSCI544/Homeworks/544/lib/python3.7/site-packages (from contractions) (0.0.21)
Requirement already satisfied: pyahocorasick in /Users/devyanbiswas/Desktop/CSCI544/Homeworks/544/lib/python3.7/site-packages (from textsearch>=0.0.21->contractions) (1.4.4)
Requirement already satisfied: anyascii in /Users/devyanbiswas/Desktop/CSCI544/Homeworks/544/lib/python3.7/site-packages (from textsearch>=0.0.21->contractions) (0.3.1)
WARNING: You are using pip version 19.2.3, however version 22.2.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

In [3]:
```
import pandas as pd
import numpy as np
import re
from bs4 import BeautifulSoup
import contractions
```

In [4]:
```
import nltk
import ssl

try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context
```

In [5]:
```
nltk.download('wordnet')
```

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

Out[5]: True

Read Data

In [6]:
```
df = pd.read_csv('./amazon_reviews_us_Jewelry_v1_00.tsv', sep='\t', usecols = ['star_rating','review_body'], he
```

In [7]:
```
df
```

Out[7]:

	star_rating	review_body
0	5	so beautiful even tho clearly not high end ......
1	5	Great product.. I got this set for my mother, ...
2	5	Exactly as pictured and my daughter's friend l...
3	5	Love it. Fits great. Super comfortable and nea...
4	5	Got this as a Mother's Day gift for my Mom and...
...	...	...
1767046	4	It is nice looking and everything (it is sterl...
1767047	4	my boyfriend bought me this last christmas, an...
1767048	4	This is a great way to quickly start learning ...
1767049	5	the 14kt gold earrings look remarkable...would...
1767050	5	It will be a gift to my special friend. We kno...

1767051 rows × 2 columns

In [8]:
```
df.dtypes
```

Out[8]:
```
star_rating    object
review_body    object
dtype: object
```

Keep Reviews and Ratings

- Done already in read

Data Cleaning

- NOTE: Regex expressions sourced from various online resoureces and documentations

In [9]:
```
df = df.loc[df['star_rating'].isin([5, 4, 3, 2, 1])]
df
```

Out[9]:

	star_rating	review_body
0	5	so beautiful even tho clearly not high end ......
1	5	Great product.. I got this set for my mother, ...
2	5	Exactly as pictured and my daughter's friend l...
3	5	Love it. Fits great. Super comfortable and nea...
4	5	Got this as a Mother's Day gift for my Mom and...
...	...	...
1767046	4	It is nice looking and everything (it is sterl...
1767047	4	my boyfriend bought me this last christmas, an...
1767048	4	This is a great way to quickly start learning ...
1767049	5	the 14kt gold earrings look remarkable...would...
1767050	5	It will be a gift to my special friend. We kno...

1701509 rows × 2 columns

In [10]:
```
# First, make sure all the datatypes are correct/consistent

# Convert ratings to int instead of double
df['star_rating'] = df['star_rating'].astype(int)

# Convert ratings to int instead of double
df['review_body'] = df['review_body'].astype(str)
```

In [11]:
```
df.dtypes
```

Out[11]:
```
star_rating     int64
review_body    object
dtype: object
```

In [12]:
```
# Getting the average character length of review body BEFORE data cleaning
before_dataproc = df['review_body'].str.len().mean()
```

In [13]:
```
# Lowercase the review bodies
df['review_body'] = df['review_body'].str.lower()
```

In [14]:
```
# Remove links, html tags
df['review_body'] = df['review_body'].str.replace(r'<[^<>]*>', '', regex=True)
df['review_body']  = df['review_body'].str.replace(r's*https?://S+(s+|$)', ' ').str.strip()
```

In [15]:
```
# Expand contractions
df['review_body'] = df['review_body'].astype(str)
df['review_body'] = df['review_body'].apply(lambda x: contractions.fix(x))
```

In [16]:
```
# Remove punctuation, non-alpha
# df['review_body'] = df['review_body'].str.replace(r'[^\w\s]+', ' ')
df['review_body'] = df.review_body.str.replace('[^a-zA-Z\s]', ' ')
```

In [17]:
```
# Remove extra spaces
df['review_body'] = df['review_body'].replace(r'\s+', ' ', regex=True)
```

In [18]:
```
# Remove Blank lines after all data cleaning is done
df['review_body'].replace('', np.nan, inplace=True)
df['review_body'].dropna(inplace=True)
```

In [19]:
```
df
```

Out[19]:

	star_rating	review_body
0	5	so beautiful even though clearly not high end ...
1	5	great product i got this set for my mother as ...
2	5	exactly as pictured and my daughter s friend l...
3	5	love it fits great super comfortable and neat ...
4	5	got this as a mother s day gift for my mom and...
...	...	...
1767046	4	it is nice looking and everything it is sterli...
1767047	4	my boyfriend bought me this last christmas and...
1767048	4	this is a great way to quickly start learning ...
1767049	5	the kt gold earrings look remarkable would def...
1767050	5	it will be a gift to my special friend we know...

1701509 rows × 2 columns

In [20]:
```
after_dataproc = df['review_body'].str.len().mean()
print("Before data proc: " + str(before_dataproc) + ",", "After data proc: " + str(after_dataproc))
```

```
Before data proc: 174.64626751900812, After data proc: 168.9937755251368
```

We select 20000 reviews randomly from each rating class.

In [21]:
```
# Figure out the different values for star rating column
cats = df['star_rating'].unique()
```

In [22]:
```
# Get the integer ones from the dataframe
# Not very pythonic but hey she gets the job done lol
star_5_df = df[df['star_rating'] == 5]
star_4_df = df[df['star_rating'] == 4]
star_3_df = df[df['star_rating'] == 3]
star_2_df = df[df['star_rating'] == 2]
star_1_df = df[df['star_rating'] == 1]
```

In [23]:
```
# CHOOSING 20k random entries from each
# Seeding them so that data is more consistent
df_20_5 = star_5_df.sample(n=20000, random_state=100)
df_20_4 = star_4_df.sample(n=20000, random_state=100)
df_20_3 = star_3_df.sample(n=20000, random_state=100)
df_20_2 = star_2_df.sample(n=20000, random_state=100)
df_20_1 = star_1_df.sample(n=20000, random_state=100)
```

In [24]:
```
# Splitting them 16k and 4k to make new datasets for training and testing
training_5 = df_20_5.iloc[:16000,:]
testing_5 = df_20_5.iloc[16000:,:]
training_4 = df_20_4.iloc[:16000,:]
testing_4 = df_20_4.iloc[16000:,:]
training_3 = df_20_3.iloc[:16000,:]
testing_3 = df_20_3.iloc[16000:,:]
training_2 = df_20_2.iloc[:16000,:]
testing_2 = df_20_2.iloc[16000:,:]
training_1 = df_20_1.iloc[:16000,:]
testing_1 = df_20_1.iloc[16000:,:]
```

In [25]:
```
# Merge all the ones above into one dataframe for training
# training_data = [training_5, training_4, training_3, training_2, training_1]
training_data = pd.concat([training_5, training_4])
training_data = pd.concat([training_data, training_3])
training_data = pd.concat([training_data, training_2])
training_data = pd.concat([training_data, training_1])
training_data=training_data.reset_index(drop=True)
```

In [26]:
```
# Merge all the remaining ones above into one dataframe for testing
testing_data = pd.concat([testing_5, testing_4])
testing_data = pd.concat([testing_data, testing_3])
testing_data = pd.concat([testing_data, testing_2])
testing_data = pd.concat([testing_data, testing_1])
testing_data=testing_data.reset_index(drop=True)
```

In [27]:
```
training_data
```

Out[27]:

	star_rating	review_body
0	5	i was looking for a pandora bracelet but i wan...
1	5	these earrings are cute and not at all gaudy j...
2	5	well made beautiful elegant mom loves them
3	5	love them since i now have so many for the sam...
4	5	my new favorite earrings
...	...	...
79995	1	not only did this thing not shine it was cheap...
79996	1	do not buy this product my boyfriend bought it...
79997	1	not even worth the sale price it is hollow in ...
79998	1	bought this set as an additional gift during t...
79999	1	terrible quality broke day i got it

80000 rows × 2 columns

In [28]:
```
testing_data
```

Out[28]:

	star_rating	review_body
0	5	these were the highlight of my wife s christma...
1	5	gave a gift very pretty
2	5	beautiful very stunning my daughter in law is ...
3	5	everything it is was really good thanks
4	5	so i was excited to see it come in the mail on...
...	...	...
19995	1	they are beautiful and sparkle like crazy but ...
19996	1	horrible no directions still cannot figuer out...
19997	1	i asked my boyfriend to buy this for me and wa...
19998	1	product came broken for the price it was not w...
19999	1	the clips are obvious on the smaller pair and ...

20000 rows × 2 columns

Pre-processing

In [29]:
```
# This is a bit convoluted, but I concat the training and testing data
# For the sake of getting a more accurate measure of the average length of
# the review body.
whole_dataset = pd.concat([training_data, testing_data])
```

In [30]:
```
whole_dataset
```

Out[30]:

	star_rating	review_body
0	5	i was looking for a pandora bracelet but i wan...
1	5	these earrings are cute and not at all gaudy j...
2	5	well made beautiful elegant mom loves them
3	5	love them since i now have so many for the sam...
4	5	my new favorite earrings
...	...	...
19995	1	they are beautiful and sparkle like crazy but ...
19996	1	horrible no directions still cannot figuer out...
19997	1	i asked my boyfriend to buy this for me and wa...
19998	1	product came broken for the price it was not w...
19999	1	the clips are obvious on the smaller pair and ...

100000 rows × 2 columns

remove the stop words

In [31]:
```
# Average character length before pre-processing
before_preproc = whole_dataset['review_body'].str.len().mean()
```

In [32]:
```
nltk.download('stopwords')
from nltk.corpus import stopwords
```

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

In [33]:
```
stop_words = stopwords.words('english')
whole_dataset['review_body'] = whole_dataset['review_body'].apply(lambda x : ' '.join([word for word in str(x).
```

In [34]:
```
whole_dataset
```

Out[34]:

	star_rating	review_body
0	5	looking pandora bracelet wanted sterling silve...
1	5	earrings cute gaudy enough detail design espec...
2	5	well made beautiful elegant mom loves
3	5	love since many price one stores fine losing o...
4	5	new favorite earrings
...	...	...
19995	1	beautiful sparkle like crazy stud broke second...
19996	1	horrible directions still cannot figuer open i...
19997	1	asked boyfriend buy really disappointed find u...
19998	1	product came broken price worth sending back r...
19999	1	clips obvious smaller pair bigger pair big pin...

100000 rows × 2 columns

perform lemmatization

In [35]:
```
from nltk.stem import WordNetLemmatizer
nltk.download('omw-1.4')
wnl = WordNetLemmatizer()
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     /Users/devyanbiswas/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

In [36]:
```
whole_dataset['review_body'] = whole_dataset['review_body'].apply(lambda x: ' '.join(wnl.lemmatize(word, pos="n
```

In [37]:
```
whole_dataset
```

Out[37]:

	star_rating	review_body
0	5	looking pandora bracelet wanted sterling silve...
1	5	earring cute gaudy enough detail design especi...
2	5	well made beautiful elegant mom love
3	5	love since many price one store fine losing on...
4	5	new favorite earring
...	...	...
19995	1	beautiful sparkle like crazy stud broke second...
19996	1	horrible direction still cannot figuer open in...
19997	1	asked boyfriend buy really disappointed find u...
19998	1	product came broken price worth sending back r...
19999	1	clip obvious smaller pair bigger pair big pinc...

100000 rows × 2 columns

In [38]:
```
# Average character length before pre-processing
after_preproc = whole_dataset['review_body'].str.len().mean()

# NOTE: Since this is being done on a new subset of the previous, the starting avg will be
# different, but the idea that this demonstrates is still useful
print("Before pre proc: " + str(before_preproc) + ",", "After pre proc: " + str(after_preproc))
```

```
Before pre proc: 181.06189, After pre proc: 106.77768
```

TF-IDF Feature Extraction

In [39]:
```
! pip install sklearn
```

```
Requirement already satisfied: sklearn in /Users/devyanbiswas/Desktop/CSCI544/Homeworks/544/lib/python3.7/site-packages (0.0)
Requirement already satisfied: scikit-learn in /Users/devyanbiswas/Desktop/CSCI544/Homeworks/544/lib/python3.7/site-packages (from sklearn) (1.0.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/devyanbiswas/Desktop/CSCI544/Homeworks/544/lib/python3.7/site-packages (from scikit-learn->sklearn) (3.1.0)
Requirement already satisfied: numpy>=1.14.6 in /Users/devyanbiswas/Desktop/CSCI544/Homeworks/544/lib/python3.7/site-packages (from scikit-learn->sklearn) (1.21.6)
Requirement already satisfied: scipy>=1.1.0 in /Users/devyanbiswas/Desktop/CSCI544/Homeworks/544/lib/python3.7/site-packages (from scikit-learn->sklearn) (1.7.3)
Requirement already satisfied: joblib>=0.11 in /Users/devyanbiswas/Desktop/CSCI544/Homeworks/544/lib/python3.7/site-packages (from scikit-learn->sklearn) (1.1.0)
WARNING: You are using pip version 19.2.3, however version 22.2.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

In [40]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
```

In [41]:
```
whole_dataset_review = whole_dataset['review_body']
```

In [42]:
```
x_whole_vectorized = vectorizer.fit_transform(whole_dataset_review)
```

In [43]:
```
# Now, we can finally re-split the data back into training and testing.
# NOTE: I know there's a built in sklearn funciton to do this, but
# I only learned about it later and I kinda wanna just stick with
# what works tbh
X_train = x_whole_vectorized[:80000,:]
X_test = x_whole_vectorized[80000:,:]

y_train = training_data['star_rating']
y_test = testing_data['star_rating']
```

Perceptron

In [44]:
```
# Import and training
from sklearn.linear_model import Perceptron
perc = Perceptron(max_iter=1)
perc.fit(X_train, y_train)
```

Out[44]: Perceptron(max_iter=1)

In [45]:
```
# Testing and score calcs
from sklearn.metrics import recall_score, precision_score, f1_score

perc_y_pred = perc.predict(X_test)

print("METRICS FOR PERCEPTRON")
print("=====================")

# Per class
recalls = recall_score(y_test, perc_y_pred, average=None)
precisions = precision_score(y_test, perc_y_pred, average=None)
f1s = f1_score(y_test, perc_y_pred, average=None)

for class_entry,value in enumerate(recalls):
    print(("Recall for class %s: " % str(class_entry+1)), value, end =", ")

print()

for class_entry,value in enumerate(precisions):
    print(("Precision for class %s: " % str(class_entry+1)), value, end =", ")

print()

for class_entry,value in enumerate(f1s):
    print(("F1 for class %s: " % str(class_entry+1)), value, end =", ")

print()

# Averages
recall_avg = recall_score(y_test, perc_y_pred, average='macro')
precision_avg = precision_score(y_test, perc_y_pred, average='macro')
f1_avg = f1_score(y_test, perc_y_pred, average='macro')

print("Recall Avg: ", recall_avg)
print("Precision Avg: ", precision_avg)
print("F1 Avg: ", f1_avg)
```

```
METRICS FOR PERCEPTRON
=====================
Recall for class 1:  0.5275, Recall for class 2:  0.3035, Recall for class 3:  0.275, Recall for class 4:  0.40825, Recall for class 5:  0.569,
Precision for class 1:  0.48921864131694875, Precision for class 2:  0.33452741802149355, Precision for class 3:  0.3402412619857717, Precision for class 4:  0.3458280389665396, Precision for class 5:  0.5547160614184743,

F1 for class 1:  0.5076386382773969, F1 for class 2:  0.31825927382356795, F1 for class 3:  0.3041614820959492,
F1 for class 4:  0.3744554001375831, F1 for class 5:  0.5617672466987534,
Recall Avg:  0.4166499999999999
Precision Avg:  0.4129062843418456
F1 Avg:  0.41325640820665005
```

SVM

In [46]:
```
from sklearn.svm import LinearSVC
lin_svc = LinearSVC(max_iter=2000)
lin_svc.fit(X_train, y_train)
```

Out[46]: LinearSVC(max_iter=2000)

In [47]:
```
svc_y_pred = lin_svc.predict(X_test)

print("METRICS FOR LINEAR SVC (SVM)")
print("=====================")

# Per class
recalls = recall_score(y_test, svc_y_pred, average=None)
precisions = precision_score(y_test, svc_y_pred, average=None)
f1s = f1_score(y_test, svc_y_pred, average=None)

for class_entry,value in enumerate(recalls):
    print(("Recall for class %s: " % str(class_entry+1)), value, end =", ")

print()

for class_entry,value in enumerate(precisions):
    print(("Precision for class %s: " % str(class_entry+1)), value, end =", ")

print()

for class_entry,value in enumerate(f1s):
    print(("F1 for class %s: " % str(class_entry+1)), value, end =", ")

print()

# Average
recall_avg = recall_score(y_test, svc_y_pred, average='macro')
precision_avg = precision_score(y_test, svc_y_pred, average='macro')
f1_avg = f1_score(y_test, svc_y_pred, average='macro')

print("Recall Avg: ", recall_avg)
print("Precision Avg: ", precision_avg)
print("F1 Avg: ", f1_avg)
```

```
METRICS FOR LINEAR SVC (SVM)
=====================
Recall for class 1:  0.652, Recall for class 2:  0.336, Recall for class 3:  0.3285, Recall for class 4:  0.39725, Recall for class 5:  0.71075,
Precision for class 1:  0.5505594257969179, Precision for class 2:  0.38866396761133604, Precision for class 3:  0.3929425837320574, Precision for class 4:  0.4274952919020716, Precision for class 5:  0.5992833052276559,
F1 for class 1:  0.5970012590133913, F1 for class 2:  0.3604183427192277, F1 for class 3:  0.3578431372549019,
F1 for class 4:  0.4118180640145134, F1 for class 5:  0.6502744739249771,
Recall Avg:  0.4849
Precision Avg:  0.47178891485400776
F1 Avg:  0.47547105538540224
```

Logistic Regression

In [48]:
```
from sklearn.linear_model import LogisticRegression
log_regr = LogisticRegression(max_iter=2000)
log_regr.fit(X_train, y_train)
```

Out[48]: LogisticRegression(max_iter=2000)

In [49]:
```
log_regr_pred = log_regr.predict(X_test)

print("METRICS FOR LOGISTIC REGRESSION")
print("=====================")

# Per class
recalls = recall_score(y_test, log_regr_pred, average=None)
precisions = precision_score(y_test, log_regr_pred, average=None)
f1s = f1_score(y_test, log_regr_pred, average=None)

for class_entry,value in enumerate(recalls):
    print(("Recall for class %s: " % str(class_entry+1)), value, end =", ")

print()

for class_entry,value in enumerate(precisions):
    print(("Precision for class %s: " % str(class_entry+1)), value, end =", ")

print()

for class_entry,value in enumerate(f1s):
    print(("F1 for class %s: " % str(class_entry+1)), value,  end =", ")

print()

# Average
recall_avg = recall_score(y_test, log_regr_pred, average='macro')
precision_avg = precision_score(y_test, log_regr_pred, average='macro')
f1_avg = f1_score(y_test, log_regr_pred, average='macro')

print("Recall Avg: ", recall_avg)
print("Precision Avg: ", precision_avg)
print("F1 Avg: ", f1_avg)
```

```
METRICS FOR LOGISTIC REGRESSION
=====================
Recall for class 1:  0.6485, Recall for class 2:  0.3855, Recall for class 3:  0.38675, Recall for class 4:  0.4355, Recall for class 5:  0.6935,
Precision for class 1:  0.5854208982171067, Precision for class 2:  0.4099973411326775, Precision for class 3:  0.4191276076943918, Precision for class 4:  0.4618239660657476, Precision for class 5:  0.6384349827387802,
F1 for class 1:  0.6153481200332107, F1 for class 2:  0.39737147274835716, F1 for class 3:  0.40228838902613445, F1 for class 4:  0.4482758620689655, F1 for class 5:  0.6648292390653087,
Recall Avg:  0.50995
Precision Avg:  0.5029609591697408
F1 Avg:  0.5056226165883954
```

Naive Bayes

In [50]:
```
from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB()
mnb.fit(X_train, y_train)
```

Out[50]: MultinomialNB()

In [51]:
```
mnb_pred = mnb.predict(X_test)

print("METRICS FOR MULTINOMIAL NAIVE BAYES")
print("=====================")

# Per class
recalls = recall_score(y_test, mnb_pred, average=None)
precisions = precision_score(y_test, mnb_pred, average=None)
f1s = f1_score(y_test, mnb_pred, average=None)

for class_entry,value in enumerate(recalls):
    print(("Recall for class %s: " % str(class_entry+1)), value,  end =", ")

print()

for class_entry,value in enumerate(precisions):
    print(("Precision for class %s: " % str(class_entry+1)), value,  end =", ")

print()

for class_entry,value in enumerate(f1s):
    print(("F1 for class %s: " % str(class_entry+1)), value,  end =", ")

print()

# Average
recall_avg = recall_score(y_test, mnb_pred, average='macro')
precision_avg = precision_score(y_test, mnb_pred, average='macro')
f1_avg = f1_score(y_test, mnb_pred, average='macro')

print("Recall Avg: ", recall_avg)
print("Precision Avg: ", precision_avg)
print("F1 Avg: ", f1_avg)
```

```
METRICS FOR MULTINOMIAL NAIVE BAYES
=====================
Recall for class 1:  0.6025, Recall for class 2:  0.382, Recall for class 3:  0.38575, Recall for class 4:  0.4255, Recall for class 5:  0.67025,
Precision for class 1:  0.594327990135635, Precision for class 2:  0.3960601347848626, Precision for class 3:  0.39422585590189063, Precision for class 4:  0.4300151591712986, Precision for class 5:  0.636061684460261,
F1 for class 1:  0.5983860955927994, F1 for class 2:  0.3889030287604989, F1 for class 3:  0.3899418751579479,
F1 for class 4:  0.4277456647398844, F1 for class 5:  0.6527084601339015,
Recall Avg:  0.4932
Precision Avg:  0.49013816489078954
F1 Avg:  0.4915370248770065
```