CSCI 544: Assignment 1

Mounika Mukkamalla

January 25, 2024

Prerequisites:

```
!pip install requests
!pip install nltk
!pip install scikit-learn

import pandas as pd
import numpy as np
import sklearn
import requests
import io
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score, classification_report
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
nltk.download('wordnet')
nltk.download('stopwords')
```

1

Dataset Preparation:

Data Read:
1. Data is directly read from the url.
2. To get the content of the url, I used requests.get(url).content and since the data is returned in bytes, I read it using io.BytesIO(content).
3. Using pandas, I read the data using compression as gzip and separator as tab. Also deleted any bad lines and used only star_rating and review features.
4. Here, for review features, I took both review_body and review_headline for richer data.

Data Analysis:
1. star_rating is of type object, hence converted it to float.
2. 3 sample reviews are outputted in the jupyter notebook:
3. Statistics of each rating is also printed in the jupyter notebook. Printed number of rows with each rating.
4. Analysis of ratings after removing bad lines(in previous step):
4.a Number of positive reviews : 2001258
4.b Number of neutral reviews: 193694
4.c Number of negative reviews: 445383

Adding labels and sampling:
1. Divided data into 2 classes: ratings>3 to class 1, ratings<3 to label 0. Didn't consider ratings==

3 since they need to be discarded.
2. duplicate values of review_body and review_headline are removed, and null values of these 2 features are removed separately to get richer data.
3. Sampled 100000 values of positive and 100000 values of negative reviews and concatenated to one dataframe, reduced_labeled_data.

2

Data Cleaning:
1. Added a list of contractions - added a few, a few are taken online.
2. Combined review_body and review_headline to one column, review
3. Converted all characters to lower case using str.lower
4. Extra spaces are removed using regex expression which says to remove spaces greater than 2 and replace tabs with one space.
5. html characters are removed using the regex expression shown.
6. https, any other urls are removed by matching with http or www. as initial characters of a word.
7. contractions are expanded using contractions dict defined above.
8. non-alphabetical characters are removed now after contractions since character ' should be removed after performing contractions else we loose the data.
9. Review mean length before and after data processing: 358.48823, 341.911305

3

Removing stop words:
1. Downloaded all stop words from nltk. while importing stopwords from nltk.corpus.
2. since our text is in english, I used stop words in english language.
3. each review is split into tokens using space and if the token is stop word it is removed and finally joined tokens with space.
4. Review mean length before and after removing stop words: 341.911305, 219.24404

Lemmatization:
1. Imported WordNetLemmatizer from nltk.stem package. intialised the lemmatizer. wordnet is downloaded from nltk and is used to find lemmas of each word.
2. each review is split into tokens using space and each token is lemmatized and finally joined tokens with space.
3. Review mean length before and after lemmatization: 219.24404, 215.61809

4

Train Test Split:
1. Data is split into train and test datasets with 80% train and 20% test data set.
2. train and test split is done before tdidf because train featues need to be used for the test.

TdIDF Vectorizer:
1. Used TfidfVectorizer imported from sklearn.feature_extraction.text.
2. Tweaked parameters of TfidfVectorizer for better performance. Used max_features = 25000
3. Applied fit_transform on train_data to get train_features.
4. using training features, obtained test_features using tranform function.

5

Perceptron:
1. Perceptron is imported from sklearn.linear_model.
2. initialised perceptron with random state 42 and model is fit with train_features and train_labels.
3. Model is now predicted on test_features
4. Metrics are reported using classification report, and outputted in dict form by setting output_dict to True. Out of which macro avg values are reported here(as mentioned in piazza post).
5. Training Metrics:
6. Accuracy: 0.9214625
7. Precision: 0.9214625405181698
8. Recall: 0.9214624854646248
9. F1-score: 0.9214624955699939
10. Testing Metrics:
11. Accuracy: 0.887675
12. Precision: 0.887666609749872
13. Recall: 0.8877248742671354
14. F1-score: 0.8876696131260909

6

SVM:
1. LinearSVC is imported from sklearn.svm.
2. initialised svm and model is fit with train_features and train_labels.
3. Model is now predicted on test_features
4. Metrics are reported using classification report, and outputted in dict form by setting output_dict to True. Out of which macro avg values are reported here(as mentioned in piazza post).
5. Training Metrics:
6. Accuracy: 0.94736875
7. Precision: 0.9473682715049132
8. Recall: 0.9473719386271902
9. F1-score: 0.9473686099491045
10. Testing Metrics:
11. Accuracy: 0.92125
12. Precision: 0.9212486978095701
13. Recall: 0.92125154126387l3
14. F1-score: 0.9212496013261067

7

Logistic Regression:
1. LogisticRegression is imported from sklearn.linear_model.
2. initialised linear_regression with random state 41 and model is fit with train_features and train_labels.
3. Model is now predicted on test_features
4. Metrics are reported using classification report, and outputted in dict form by setting output_dict to True. Out of which macro avg values are reported here(as mentioned in piazza post).
5. Training Metrics:
6. Accuracy: 0.92858125
7. Precision: 0.9285803048003554
8. Recall: 0.9285925693132888
9. F1-score: 0.928580662238379
10. Testing Metrics:
11. Accuracy: 0.92235
12. Precision: 0.9223513252904819

13. Recall: 0.9223504805876201
14. F1-score: 0.9223499805874951

8

Naive Bayes:
1. MultinomialNB is imported from sklearn.naive_bayes.
2. initialised nb with alpha 0.8 and model is fit with train_features and train_labels.
3. Model is now predicted on test_features
4. Metrics are reported using classification report, and outputted in dict form by setting output_dict to True. Out of which macro avg values are reported here(as mentioned in piazza post).
5. Training Metrics:
6. Accuracy: 0.8903875
7. Precision: 0.8903876064607572
8. Recall: 0.8903875810855608
9. F1-score: 0.8903874997259688
10. Testing Metrics:
11. Accuracy: 0.882375
12. Precision: 0.8823695978315951
13. Recall: 0.8823953990632112
14. F1-score: 0.8823721468629047

In [1]:
```
!pip install requests
!pip install nltk
!pip install scikit-learn
```

```
Requirement already satisfied: requests in c:\users\mouni\appdata\local\programs\pyth
on\python312\lib\site-packages (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\mouni\appdata\loc
al\programs\python\python312\lib\site-packages (from requests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\mouni\appdata\local\programs
\python\python312\lib\site-packages (from requests) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\mouni\appdata\local\pro
grams\python\python312\lib\site-packages (from requests) (2.1.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\mouni\appdata\local\pro
grams\python\python312\lib\site-packages (from requests) (2023.11.17)

[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
Requirement already satisfied: nltk in c:\users\mouni\appdata\local\programs\python\p
ython312\lib\site-packages (3.8.1)
Requirement already satisfied: click in c:\users\mouni\appdata\local\programs\python
\python312\lib\site-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in c:\users\mouni\appdata\local\programs\python
\python312\lib\site-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in c:\users\mouni\appdata\local\progra
ms\python\python312\lib\site-packages (from nltk) (2023.12.25)
Requirement already satisfied: tqdm in c:\users\mouni\appdata\local\programs\python\p
ython312\lib\site-packages (from nltk) (4.66.1)
Requirement already satisfied: colorama in c:\users\mouni\appdata\local\programs\pyth
on\python312\lib\site-packages (from click->nltk) (0.4.6)

[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
Requirement already satisfied: scikit-learn in c:\users\mouni\appdata\local\programs
\python\python312\lib\site-packages (1.4.0)
Requirement already satisfied: numpy<2.0,>=1.19.5 in c:\users\mouni\appdata\local\pro
grams\python\python312\lib\site-packages (from scikit-learn) (1.26.3)
Requirement already satisfied: scipy>=1.6.0 in c:\users\mouni\appdata\local\programs
\python\python312\lib\site-packages (from scikit-learn) (1.12.0)
Requirement already satisfied: joblib>=1.2.0 in c:\users\mouni\appdata\local\programs
\python\python312\lib\site-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\mouni\appdata\local\p
rograms\python\python312\lib\site-packages (from scikit-learn) (3.2.0)

[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [2]:
```
import pandas as pd
import numpy as np
import sklearn
import requests
import io
import nltk
nltk.download('wordnet')
```

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

Out[2]:
```
True
```

Dataset Preparation

In [3]:
```
url = 'https://web.archive.org/web/20201127142707if_/https://s3.amazonaws.com/amazon-r
content = requests.get(url).content
df = pd.read_csv(io.BytesIO(content), compression='gzip', sep='\t', on_bad_lines='skip

print("Three sample review rows:")
print(df.sample(3))
```

```
C:\Users\mouni\AppData\Local\Temp\ipykernel_31844\210679158.py:3: DtypeWarning: Colum
ns (7) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv(io.BytesIO(content), compression='gzip', sep='\t', on_bad_lines='s
kip', usecols=['review_body', 'review_headline', 'star_rating'])
Three sample review rows:
        star_rating                   review_headline  \
1414143           5            Affordable and Durable
8058              1  Do not expect gulaity print outs
536042            4                        Four Stars

                                             review_body
1414143  Owned one of these for around a year now.  Gre...
8058     As other sites have noted, the cost per page i...
536042                                 Works OK so far.
```

In [67]:
```
#convert star_rating to numeric data
df = df[pd.to_numeric(df['star_rating'], errors='coerce').notnull()]
df['star_rating'] = df['star_rating'].astype(float)

print("Star_rating:")
print("1.0: " + str(df[df['star_rating']==1].shape[0]))
print("2.0: " + str(df[df['star_rating']==2].shape[0]))
print("3.0: " + str(df[df['star_rating']==3].shape[0]))
print("4.0: " + str(df[df['star_rating']==4].shape[0]))
print("5.0: " + str(df[df['star_rating']==5].shape[0]))

#positive reviews
positive_reviews = df[df['star_rating']>3].shape[0]

#neutral reviews
neutral_reviews = df[df['star_rating']==3].shape[0]

#negative reviews
negative_reviews = df[df['star_rating']<3].shape[0]

print("")
print("Positive(label 1), Neutral(label 0) and Negative reviews count:")
print("1 " + str(positive_reviews))
print("0 " + str(neutral_reviews))
print("Neutral " + str(negative_reviews))
```

```
Star_rating:
1.0: 306992
2.0: 138391
3.0: 193694
4.0: 418381
5.0: 1582877

Positive(label 1), Neutral(label 0) and Negative reviews count:
1 2001258
0 193694
Neutral 445383
```

In [5]:
```
# Adding labels
labeled_data = df
conditions = [(labeled_data['star_rating'] >3), (labeled_data['star_rating'] < 3)]
values = [1, 0]
labeled_data['labels'] = np.select(conditions, values)
print(labeled_data[['review_headline', 'review_body', 'star_rating', 'labels']].head()
```

```
                                     review_headline  \
0                                         Five Stars
1  Phfffffffft, 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

                                         review_body  star_rating  labels
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 &#34;new&#34; the...          1.0       0
4                      Gorgeous colors and easy to use          4.0       1
```

In [6]:
```
# Remove duplicate rows and drop null values for review_body or review_headline featur
num_of_rows_before = labeled_data.shape[0]
labeled_data = labeled_data.drop_duplicates(subset=['review_body', 'review_headline'],
num_of_rows_after = labeled_data.shape[0]

print("Number of rows before and after removing duplicates and null values: ")
print(str(num_of_rows_before) + "," + str(num_of_rows_after))
```

```
Number of rows before and after removing duplicates and null values:
2640335,2456663
```

In [7]:
```
# sample 100000 data
num_of_samples = 100000

positive_reviews = labeled_data[labeled_data.star_rating>3].sample(num_of_samples)
negative_reviews = labeled_data[labeled_data.star_rating<3].sample(num_of_samples)

reduced_labeled_data = pd.concat([positive_reviews, negative_reviews]).sample(frac=1)
print(reduced_labeled_data.shape)
```

```
(200000, 4)
```

Dataset Cleaning

In [8]:
```
contractions = {
    "'s": "is",
```

```
    "'S": "Is",
    "aren't": "are not",
    "arent": "are not",
    "can't": "can not",
    "cant": "can not",
    "can't've": "can not have",
    "'cause": "because",
    "cannot": "can not",
    "could've": "could have",
    "couldve": "could have",
    "couldn't": "could not",
    "couldnt": "could not",
    "couldn't've": "could not have",
    "couldntve": "could not have",
    "didn't": "did not",
    "didnt": "did not",
    "doesn't": "does not",
    "doesnt": "does not",
    "don't": "do not",
    "dont": "do not",
    "hadn't": "had not",
    "hadnt": "had not",
    "hadn't've": "had not have",
    "hasn't": "has not",
    "hasnt": "has not",
    "haven't": "have not",
    "havent": "have not",
    "he'd": "he would",
    "hed": "he would",
    "he'd've": "he would have",
    "hedve": "he would have",
    "he'll": "he will",
    "he'll've": "he will have",
    "he's": "he is",
    "hes": "he is",
    "how'd": "how did",
    "howd": "how did",
    "how'd'y": "how did you",
    "how'll": "how will",
    "howll": "how will",
    "how's": "how is",
    "hows": "how is",
    "i'd": "i would",
    "i'd've": "i would have",
    "i'll": "i will",
    "i'll've": "i will have",
    "i'm": "i am",
    "im": "i am",
    "i'ma": "i am going to",
    "i've": "i have",
    "isn't": "is not",
    "isnt": "is not",
    "it'd": "it would",
    "it'd've": "it would have",
    "it'll've": "it will have",
    "it'll": "it will",
    "itll": "it will",
    "it's": "it is",
    "let's": "let us",
    "lets": "let us",
```

```
    "ma'am": "madam",
    "mayn't": "may not",
    "mightn't": "it might not",
    "mightn't've": "might not have",
    "might've": "might have",
    "mustn't": "must not",
    "mustn't've": "must not have",
    "must've": "must have",
    "needn't": "need not",
    "needn't've": "need not have",
    "not've": "not have",
    "oughtn't": "ought not",
    "oughtn't've": "ought not to have",
    "so've": "so have",
    "so's": "so is",
    "shan't": "shall not",
    "sha'n't": "shall not",
    "shan't've": "shall not have",
    "she'd": "she would",
    "she'd've": "she would have",
    "she'll": "she will",
    "she'll've": "she will have",
    "she's": "she is",
    "should've": "should have",
    "shouldn't": "should not",
    "shouldn't've": "should not have",
    "that'd": "that would",
    "that'd've": "that would have",
    "that's": "that is",
    "thats": "that is",
    "there'd": "there would",
    "there'd've": "there would have",
    "there's": "there is",
    "they'd": "they would",
    "they'd've": "they would have",
    "they'll": "they will",
    "they'll've": "they will have",
    "they're": "they are",
    "they've": "they have",
    "to've": "to have",
    "wasn't": "was not",
    "we'd": "we would",
    "we'd've": "we would have",
    "we'll": "we will",
    "we'll've": "we will have",
    "we're": "we are",
    "we've": "we have",
    "weren't": "were not",
    "what'll": "what will",
    "what'll've": "what will have",
    "what're": "what are",
    "what's": "what has/is",
    "what've": "what have",
    "when's": "when is",
    "when've": "when have",
    "where'd": "where would",
    "where's": "where is",
    "where've": "where have",
    "who'd": "who would",
    "who'll": "who will",
```

```
    "who'll've": "who will have",
    "who're": "who are",
    "who's": "who is",
    "who've": "who have",
    "why've": "why have",
    "why'll": "why will",
    "why're": "why are",
    "why's": "why is",
    "will've": "will have",
    "won't": "will not",
    "wont": "will not",
    "won't've": "will not have",
    "would've": "would have",
    "wouldn't": "would not",
    "wouldn't've": "would not have",
    "y'all": "you all",
    "y'all'd": "you all would",
    "y'all'd've": "you all would have",
    "y'all're": "you all are",
    "y'all've": "you all have",
    "you'd": "you would",
    "you'd've": "you would have",
    "you'll": "you will",
    "you'll've": "you will have",
    "you're": "you are",
    "you've": "you have"
}
```

In [9]:

```
reduced_labeled_data['review'] = reduced_labeled_data['review_headline'] + ' ' + reduc

review_mean_before = reduced_labeled_data['review'].str.len().mean()

# to lower case
reduced_labeled_data['review'] = reduced_labeled_data['review'].astype(str).map(str.lc

# remove extra spaces
reduced_labeled_data['review'] = reduced_labeled_data['review'].str.replace(r' +|\t+',

# remove html characters
reduced_labeled_data['review'] = reduced_labeled_data['review'].str.replace('<[^<>]*>'

#remove urls, https
reduced_labeled_data['review'] = reduced_labeled_data['review'].str.replace(r'http\S+|

#expanding contractions
reduced_labeled_data['review'] = reduced_labeled_data['review'].map(lambda review: ' '

#remove non-alphabetical characters
reduced_labeled_data['review'] = reduced_labeled_data['review'].str.replace(r'[^a-zA-Z

review_mean_after = reduced_labeled_data['review'].str.len().mean()

print("Review mean length before and after data processing:")
print(str(review_mean_before) + "," + str(review_mean_after))
print("Review data after cleaning based on regex:")
print(reduced_labeled_data[['review', 'labels']].head())
```

```
Review mean length before and after data processing:
358.48823,341.911305
Review data after cleaning based on regex:
                                                    review  labels
2344666  color cartridge useless the black cartridge is...       0
1898085  avery tabs the tabs were delivered in a box th...       0
1837716  finally a decent sharpener as a teacher i am c...       1
1627807  love this tiny planner this is small enough to...       1
994495   the cover is not bends easily searching for mo...       0
```

Preprocessing

In [10]:

```
from nltk.corpus import stopwords

nltk.download('stopwords')
stop_words = set(stopwords.words('English'))

review_mean_before = reduced_labeled_data['review'].str.len().mean()
reduced_labeled_data['review']=reduced_labeled_data['review'].map(lambda review: ' '.j
review_mean_after = reduced_labeled_data['review'].str.len().mean()

print("Review mean length before and after removing stop words:")
print(str(review_mean_before) + "," + str(review_mean_after))
print("Review data after removing stop words:")
print(reduced_labeled_data[['review', 'labels']].head())
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\mouni\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Review mean length before and after removing stop words:
341.911305,219.24404
Review data after removing stop words:
                                                    review  labels
2344666  color cartridge useless black cartridge great ...       0
1898085  avery tabs tabs delivered box bent tabs unbend...       0
1837716  finally decent sharpener teacher constantly ne...       1
1627807  love tiny planner small enough tuck small pock...       1
994495   cover bends easily searching monthsweeks time ...       0
```

In [11]:

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
review_mean_before = reduced_labeled_data['review'].str.len().mean()
reduced_labeled_data['review'] = reduced_labeled_data['review'].map(lambda review: ' '
review_mean_after = reduced_labeled_data['review'].str.len().mean()

print("Review mean length before and after lemmatization:")
print(str(review_mean_before) + "," + str(review_mean_after))
print("Review data after lemmatization:")
print(reduced_labeled_data[['review', 'labels']].head())
```

```
Review mean length before and after lemmatization:
219.24404,215.61809
Review data after lemmatization:
                                                    review  labels
2344666  color cartridge useless black cartridge great ...       0
1898085  avery tab tab delivered box bent tab unbend on...       0
1837716  finally decent sharpener teacher constantly ne...       1
1627807  love tiny planner small enough tuck small pock...       1
994495   cover bend easily searching monthsweeks time c...       0
```

Feature Extraction

In [12]:

```python
from sklearn.model_selection import train_test_split

datasets = train_test_split(reduced_labeled_data['review'], reduced_labeled_data['labe
train_data, test_data, train_labels, test_labels = datasets

print("Number of train and test data sets:")
print(str(len(train_data)) + ", " + str(len(test_data)))
```

```
Number of train and test data sets:
160000, 40000
```

In [51]:

```python
from sklearn.feature_extraction.text import TfidfVectorizer

train_vectorizer = TfidfVectorizer(max_features = 25000)#, sublinear_tf=True, max_df=0
train_features = train_vectorizer.fit_transform(train_data)
test_features = train_vectorizer.transform(test_data)

print("Train features and test features printed below:")
print(train_features)
print("")
print(test_features)
```

```
Train features and test features printed below:
  (0, 14719)	0.1617671527569556
  (0, 607)	0.1854967796903625
  (0, 24644)	0.0829415096736689
  (0, 19846)	0.18364771178303446
  (0, 703)	0.12677120200163122
  (0, 4941)	0.11804295759303601
  (0, 24443)	0.1601157119328888
  (0, 18236)	0.2639949331109749
  (0, 22545)	0.12683602016095083
  (0, 4477)	0.11049237716603706
  (0, 22279)	0.5231792257398336
  (0, 2046)	0.3351584133156321
  (0, 3164)	0.37793635244839163
  (0, 12118)	0.43609698043928424
  (0, 333)	0.16162006372575302
  (1, 18942)	0.23329191098230276
  (1, 12524)	0.0992932504489623
  (1, 12322)	0.0992932504489623
  (1, 7224)	0.1732499454770853
  (1, 2264)	0.19756420093107313
  (1, 14484)	0.11506762941360901
  (1, 24934)	0.20119260656500018
  (1, 11944)	0.5601141270289941
  (1, 14951)	0.2943814267168828
  (1, 653)	0.13768858649609922
  :	:
  (159999, 10179)	0.12796048184155728
  (159999, 8475)	0.13368169771636454
  (159999, 10998)	0.2703000929058341
  (159999, 12469)	0.15649311016594838
  (159999, 10558)	0.13607500782550697
  (159999, 103)	0.17853018685756095
  (159999, 10591)	0.20736267935511934
  (159999, 10038)	0.12015087586957546
  (159999, 7427)	0.2269440876406883
  (159999, 11302)	0.14156983132261505
  (159999, 10091)	0.10467298249919706
  (159999, 12459)	0.08854441611678228
  (159999, 22642)	0.3155523575085092
  (159999, 4065)	0.10024329279324301
  (159999, 6833)	0.09608339352372998
  (159999, 20474)	0.26677499722963705
  (159999, 14787)	0.10661275615211258
  (159999, 7128)	0.09298731485445298
  (159999, 24113)	0.0700628188864133
  (159999, 24538)	0.08423918300634431
  (159999, 24553)	0.09384366281363611
  (159999, 22076)	0.06469392489015796
  (159999, 23983)	0.0950257015682831
  (159999, 24522)	0.05425606107267092
  (159999, 18236)	0.09418531903210747

  (0, 24829)	0.0684261383987795
  (0, 24522)	0.04711172441652683
  (0, 22076)	0.05617514984669285
  (0, 19854)	0.12235245317029361
  (0, 18943)	0.20853750912739108
  (0, 18600)	0.15566189828623833
  (0, 17827)	0.3900071444317543
```

```
  (0, 17585)	0.07414354034626747
  (0, 16161)	0.3583397621103081
  (0, 15722)	0.08840744354904827
  (0, 15413)	0.16018224576318704
  (0, 15328)	0.10297814140521475
  (0, 15293)	0.1369053348469291
  (0, 14416)	0.086512134817084
  (0, 13831)	0.06813855981189865
  (0, 12256)	0.07315813880641829
  (0, 10417)	0.2584446570251274
  (0, 9845)	0.09761020526272995
  (0, 7711)	0.13523548501949043
  (0, 7646)	0.2289658835591534
  (0, 7128)	0.08074291913281612
  (0, 7009)	0.18098750355752619
  (0, 6820)	0.15700900781252145
  (0, 6384)	0.0902812929742409
  (0, 4864)	0.11633758793592605
  :	:
  (39999, 24644)	0.05832033469746911
  (39999, 24079)	0.4606634024283574
  (39999, 22347)	0.18520334284470993
  (39999, 22076)	0.06375202062647427
  (39999, 21970)	0.0963460364969572
  (39999, 20762)	0.10667038421647033
  (39999, 20442)	0.16167052364912007
  (39999, 20207)	0.13773075105901011
  (39999, 15683)	0.15618715174049055
  (39999, 15452)	0.1746802177908336
  (39999, 15450)	0.3691220212248677
  (39999, 12167)	0.08322741674599839
  (39999, 12073)	0.06525614969227335
  (39999, 9424)	0.10240135626115843
  (39999, 9382)	0.1085084794454007
  (39999, 9133)	0.05590330823846196
  (39999, 7128)	0.09163347601288298
  (39999, 6541)	0.22490678038075776
  (39999, 5270)	0.10958450174330413
  (39999, 4816)	0.24479375759151778
  (39999, 3052)	0.5151575496093621
  (39999, 2619)	0.06749204384605022
  (39999, 2217)	0.07240882532006575
  (39999, 1931)	0.08631621831415157
  (39999, 1423)	0.1974271261591216
```

In [61]:
```python
def printMatrix(matrix):
    print("Accuracy: " + str(matrix['accuracy']))
    print("Precision: " + str(matrix['macro avg']['precision']))
    print("Recall: " + str(matrix['macro avg']['recall']))
    print("F1-score: " + str(matrix['macro avg']['f1-score']))
```

Perceptron

In [53]:
```python
#perceptron training

from sklearn.linear_model import Perceptron
```

```
p = Perceptron(random_state = 42)
p.fit(train_features, train_labels)
```

Out[53]:

```
Perceptron
Perceptron(random_state=42)
```

In [62]:

```
# Metrics
from sklearn.metrics import accuracy_score, classification_report

train_predictions = p.predict(train_features)
test_predictions = p.predict(test_features)

print("Perceptron Training Metrics:")
printMatrix(classification_report(train_predictions, train_labels, output_dict=True))
print("")
print("Perceptron Testing Metrics:")
printMatrix(classification_report(test_predictions, test_labels, output_dict=True))
```

```
Perceptron Training Metrics:
Accuracy: 0.9214625
Precision: 0.9214625405181698
Recall: 0.9214624854646248
F1-score: 0.9214624955699939

Perceptron Testing Metrics:
Accuracy: 0.887675
Precision: 0.887666609749872
Recall: 0.8877248742671354
F1-score: 0.8876696131260909
```

SVM

In [55]:

```
# SVM classifier

from sklearn.svm import LinearSVC

svm_classifier = LinearSVC()
svm_classifier.fit(train_features, train_labels)
```

```
C:\Users\mouni\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\svm
\_classes.py:31: FutureWarning: The default value of `dual` will change from `True` t
o `'auto'` in 1.5. Set the value of `dual` explicitly to suppress the warning.
  warnings.warn(
```

Out[55]:

```
LinearSVC
LinearSVC()
```

In [63]:

```
#SVM metrics
train_predictions = svm_classifier.predict(train_features)
test_predictions = svm_classifier.predict(test_features)

print("SVM Training Metrics:")
printMatrix(classification_report(train_predictions, train_labels, output_dict=True))
print("")
```

```
print("SVM Testing Metrics:")
printMatrix(classification_report(test_predictions, test_labels, output_dict=True))
```

```
SVM Training Metrics:
Accuracy: 0.94736875
Precision: 0.9473682715049132
Recall: 0.9473719386271902
F1-score: 0.9473686099491045

SVM Testing Metrics:
Accuracy: 0.92125
Precision: 0.9212486978095701
Recall: 0.9212515412638713
F1-score: 0.9212496013261067
```

Logistic Regression

In [57]:
```
# Logistic Regression
from sklearn.linear_model import LogisticRegression

LR = LogisticRegression(random_state = 41)
LR.fit(train_features, train_labels)
```

Out[57]:
```
▾ LogisticRegression
LogisticRegression(random_state=41)
```

In [64]:
```
#LR metrics

train_predictions = LR.predict(train_features)
test_predictions = LR.predict(test_features)

print("Logistic Regression Training Metrics:")
printMatrix(classification_report(train_predictions, train_labels, output_dict=True))
print("")
print("Logistic Regression Testing Metrics:")
printMatrix(classification_report(test_predictions, test_labels, output_dict=True))
```

```
Logistic Regression Training Metrics:
Accuracy: 0.92858125
Precision: 0.9285803048003554
Recall: 0.9285925693132888
F1-score: 0.928580662238379

Logistic Regression Testing Metrics:
Accuracy: 0.92235
Precision: 0.9223513252904819
Recall: 0.9223504805876201
F1-score: 0.9223499805874951
```

Multinomial Naive Bayes

In [59]:
```
# Naive Bayes
from sklearn.naive_bayes import MultinomialNB
```

```python
NB = MultinomialNB(alpha = 1)
NB.fit(train_features, train_labels)
```

Out[59]:

```
▾ MultinomialNB
MultinomialNB(alpha=1)
```

In [65]:

```python
# NB Metrics

train_predictions = NB.predict(train_features)
test_predictions = NB.predict(test_features)

print("Multinoimial Naive Bayes Training Metrics:")
printMatrix(classification_report(train_predictions, train_labels, output_dict=True))
print("")
print("Multinoimial Naive Bayes Testing Metrics:")
printMatrix(classification_report(test_predictions, test_labels, output_dict=True))
```

```
Multinoimial Naive Bayes Training Metrics:
Accuracy: 0.8903875
Precision: 0.8903876064607572
Recall: 0.8903875810855608
F1-score: 0.8903874997259688

Multinoimial Naive Bayes Testing Metrics:
Accuracy: 0.882375
Precision: 0.8823695978315951
Recall: 0.8823953990632112
F1-score: 0.8823721468629047
```