PYTHON 3.12

CSCI544: Homework Assignment №1

ADITYA DUTTA

Goal is to do sentiment analysis of Amazon kitchen product reviews, employing various natural language processing (NLP) techniques and machine learning algorithms. The goal is to classify reviews into positive or negative sentiments, which is a valuable tool for understanding customer feedback.

1. Dataset Preparation

- The dataset, sourced from Amazon, consists of product reviews. Using Pandas, we chose 'review_body' for text and 'star_rating' for sentiment.

-Used Pandas' read_csv function, which is good for reading large datasets like the Amazon reviews. Specified on_bad_lines='skip'

-Filtered the dataset to include only 'review_body' and 'star_rating' using data[['review_body', 'star_rating']]. This simplification focuses the analysis on essential data.

-Converted the 5-point star ratings into binary labels (positive and negative). Ratings above 3 were marked as positive (1), and 2 or below as negative (0). Excluded neutral ratings (3) to create a clear distinction between sentiments

-Filtered and sampled an equal number of positive and negative reviews. Used data.sample(n=100000) for both positive and negative sentiments

-Utilized train_test_split from Scikit-learn to divide the dataset into training (80%) and testing (20%) sets, a common split ratio in machine learning tasks

2. Data Cleaning

- Cleaning included normalizing text to lowercase(Uniform case reduces complexity, as text processing typically considers different cases (e.g., 'Word' vs. 'word') as different tokens), stripping HTML tags and URLs, removing non-alphabetical characters, and eliminating extra spaces. This standardization is crucial for effective NLP.

-Lowercasing: Applied str.lower() to standardize the text, as text data often contains a mix of uppercase and lowercase, and for most NLP tasks, the case is irrelevant.

- Used regex it to remove HTML tags (re.sub(r'<.*?>', '', text)), URLs (re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)), and non-alphabetical characters (re.sub(r'[^a-zA-Z\s]', '', text)). The goal was to clean the text data from irrelevant characters and contents.

-Whitespace Removal: Extra spaces can be problematic in text processing and can arise especially after removing characters and words. Used Regex to replace multiple spaces with a single space (re.sub(r'\s+', ' ', text).strip()).

3. Preprocessing

-Tokenization: The split() function breaks the text into individual words, for both stop word removal and lemmatization.

-Stop Words Removal: The list comprehension [word for word in words if word not in stop_words] filters out any word present in the NLTK's stop words list.

-Lemmatization: In the same list comprehension, lemmatizer.lemmatize(word) is applied to each word. This function transforms the word to its base form.

-After processing, the words are joined back into a single string using ' '.join(words). This step is crucial to revert the tokenized text back into its standard form.

-Application to Dataset: The preprocess_text function is applied to the 'cleaned_review' column of the dataset using apply(). This Pandas method efficiently applies the function to each row in the column.

4. Feature Extraction

- The TfidfVectorizer from Scikit-learn is initialized, which prepares the vectorizer for processing the text data.

-The vectorizer is then fitted to the 'review_body' data, which means it learns the vocabulary and idf (inverse document frequency) from the text data. The output is a sparse matrix where each column represents a word in the overall corpus, and each row represents each document (review).

- Feature Space: The shape of the TF-IDF features (tfidf_features.shape) gives an idea of the number of features (unique words) extracted from the text data.

5. Model Training

- Models: Training of four models: Perceptron, SVM (Support Vector Machine), Logistic Regression, and Multinomial Naive Bayes, each offering different strengths and perspectives on the data.

-Used Scikit-learn for its comprehensive set of tools for machine learning.Used SVC (Support Vector Classifier) from Scikit-learn for SVM. The default RBF (Radial Basis Function) kernel was chosen as it can handle non-linear data well.

-Each model was trained using .fit() on the training set.The fit method trains the Perceptron on the training dataset (X_train, y_train).For Logistic Regression, max_iter is set higher to ensure convergence.

- Training: Explained the process of fitting these models on the training dataset.

- Evaluation Metrics: Evaluated model performance using accuracy, precision, recall, and F1-score on both training and testing datasets, providing a comprehensive view of each model's effectiveness.

- SVM took the most time compared to other models

6. Conclusion

- Summarized the outcomes and insights from the sentiment analysis, highlighting key findings.

- Model Comparison: Discussed which models performed best in terms of different metrics and why, providing insights into the suitability of each model for sentiment analysis.

In [3]:

```python
# 1. DATASET PREPARATION

#Loading Data
import pandas as pd
```

In [4]:

```python
import pandas as pd
from sklearn.model_selection import train_test_split


import pandas as pd

file_path = r"C:\Users\adity\Desktop\NLP\amazon_reviews_us_Office_Products_v1_00.tsv"
data = pd.read_csv(file_path, sep='\t', on_bad_lines="skip")
print (data)
```

```
C:\Users\adity\AppData\Local\Temp\ipykernel_3176\1579639910.py:8: DtypeWarning: Columns (7) have mixed types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv(file_path, sep='\t', on_bad_lines="skip")
        marketplace  customer_id       review_id  product_id  product_parent  \
0                US     43081963  R18RVCKGH1SSI9  B001BM2MAC       307809868
1                US     10951564  R3L4L6LW1PUOFY  B00DZYEXPQ        75004341
2                US     21143145  R2J8AWXWTDX2TF  B00RTMUHDW       529689027
3                US     52782374  R1PR37BR7G3M6A  B00D7H8XB6       868449945
4                US     24045652  R3BDDDZMZBZDPU  B001XCWP34        33521401
...             ...          ...             ...         ...             ...
2640249          US     53005790   RLI7EI10S7SN0  B00000DM9M       223408988
2640250          US     52188548  R1F3SRK9MHE6A3  B00000DM9M       223408988
2640251          US     52090046  R23V0C4NRJL8EM  0807865001       307284585
2640252          US     52503173  R13ZAE1ATEUC1T  1572313188       870359649
2640253          US     52585611   RE8J5O2GY04NN  1572313188       870359649

                                             product_title product_category  \
0           Scotch Cushion Wrap 7961, 12 Inches x 100 Feet  Office Products
1                  Dust-Off Compressed Gas Duster, Pack of 4  Office Products
2        Amram Tagger Standard Tag Attaching Tagging Gu...  Office Products
3        AmazonBasics 12-Sheet High-Security Micro-Cut ...  Office Products
4        Derwent Colored Pencils, Inktense Ink Pencils,...  Office Products
...                                                    ...              ...
2640249                  PalmOne III Leather Belt Clip Case  Office Products
2640250                  PalmOne III Leather Belt Clip Case  Office Products
2640251                   Gods and Heroes of Ancient Greece  Office Products
2640252  Microsoft EXCEL 97/ Visual Basic Step-by-Step ...  Office Products
2640253  Microsoft EXCEL 97/ Visual Basic Step-by-Step ...  Office Products

        star_rating  helpful_votes  total_votes vine verified_purchase  \
0                 5            0.0          0.0    N                 Y
1                 5            0.0          1.0    N                 Y
2                 5            0.0          0.0    N                 Y
3                 1            2.0          3.0    N                 Y
4                 4            0.0          0.0    N                 Y
...             ...            ...          ...  ...               ...
2640249           4           26.0         26.0    N                 N
2640250           4           18.0         18.0    N                 N
2640251           4            9.0         16.0    N                 N
2640252           5            0.0          0.0    N                 N
2640253           5            0.0          0.0    N                 N

                                           review_headline  \
0                                               Five Stars
1        Phfffffft, Phffffffft. 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
...                                                    ...
2640249  Great value! A must if you hate to carry thing...
2640250          Attaches the Palm Pilot like an appendage
2640251  Excellent information, pictures and stories, I...
2640252                                         class text
2640253                                 Microsoft's Finest

                                               review_body review_date
0                                           Great product.  2015-08-31
1        What's to say about this commodity item except...  2015-08-31
2          Haven't used yet, but I am sure I will like it.  2015-08-31
3        Although this was labeled as &#34;new&#34; the...  2015-08-31
4                           Gorgeous colors and easy to use  2015-08-31
...                                                    ...         ...
2640249  I can't live anymore whithout my Palm III. But...  1998-12-07
2640250  Although the Palm Pilot is thin and compact it...  1998-11-30
2640251  This book had a lot of great content without b...  1998-10-15
2640252  I am teaching a course in Excel and am using t...  1998-08-22
2640253  A very comprehensive layout of exactly how Vis...  1998-07-15

[2640254 rows x 15 columns]
```

In [5]:

```python
data = data[['review_body', 'star_rating']]
```

In [7]:

```python
data = data[data['star_rating'] !=3]
```

In [8]:

```python
#Removing rows with non-numeric 'star_rating' values:
data = data[pd.to_numeric(data['star_rating'], errors='coerce').notnull()]

#Converting 'star_rating' to integers:
data['star_rating'] = data['star_rating'].astype(int)

#Function:
data['Sentiment'] = data['star_rating'].apply(lambda rating : +1 if rating > 3 else 0)
```

In [9]:

```python
print (data)
```

```
                                               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 &#34;new&#34; the...            1
4                           Gorgeous colors and easy to use            4
...                                                    ...          ...
2640249  I can't live anymore whithout my Palm III. But...            4
2640250  Although the Palm Pilot is thin and compact it...            4
2640251  This book had a lot of great content without b...            4
2640252  I am teaching a course in Excel and am using t...            5
2640253  A very comprehensive layout of exactly how Vis...            5

         Sentiment
0                1
1                1
2                1
3                0
4                1
...            ...
2640249          1
2640250          1
2640251          1
2640252          1
2640253          1

[2460366 rows x 3 columns]
```

In [10]:

```python
#Printing no. of reviews for each sentiment:
print('Number of positive reviews:', len(data[data['Sentiment'] == 1]))
print('Number of negative reviews:', len(data[data['Sentiment'] == 0]))
```

```
Number of positive reviews: 2001183
Number of negative reviews: 459183
```

In [11]:

```python
# Selecting 100,000 positive reviews and 100,000 negative reviews:
data_pos = data[data['Sentiment'] == 1].sample(n=100000)
data_neg = data[data['Sentiment'] == 0].sample(n=100000)
```

In [12]:

```python
# Concatenating the reviews:
data2 = pd.concat([data_pos, data_neg])
print(data2)
```

```
                                               review_body  star_rating  \
1750243  My dad used to have this calculator, just in a...            5
221107                                          Great item            5
145543   They worked for the Pinterest Mugs :) We did 1...            5
1887530  I used it to stick my pingpong rubber on the b...            5
1025803  I have been using the shredder for 3 weeks now...            5
...                                                    ...          ...
2428708  Unless you have some special psychic abilities...            1
2366247  I purchased a new Kodak ESP7 about 6 months ag...            1
1167057  Not pleased! The majority of the books were le...            1
2075335  Printer does not always recognize the cartridg...            1
1089230              Does not work at all . Waste my money            1

         Sentiment
1750243          1
221107           1
145543           1
1887530          1
1025803          1
...            ...
2428708          0
2366247          0
1167057          0
2075335          0
1089230          0

[200000 rows x 3 columns]
```

In [13]:

```python
train_data, test_data = train_test_split(data2, test_size=0.2, random_state=42)
```

In [14]:

```python
#Splitting into training and testing sets:
print('Number of reviews in the training set:', len(train_data))
print('Number of reviews in the testing set:', len(test_data))
```

```
Number of reviews in the training set: 160000
Number of reviews in the testing set: 40000
```

In [17]:

```python
# 2. Data Cleaning

import re
def clean_text(text):

    text = str(text)

     # Convert text to lower case:
    text = text.lower()

     # Remove HTML tags:
    text = re.sub(r'<.*?>', '', text)

    # Remove URLs:
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)

     # Remove non-alphabetical characters:
    text = re.sub(r'[^a-zA-Z\s]', '', text)

     # Remove extra spaces:
    text = re.sub(r'\s+', ' ', text).strip()


    return text

#Apply Cleaning:
data2['cleaned_review'] = data2['review_body'].apply(clean_text)
```

In [18]:

```python
#  Average review length before cleaning

data2['review_length_before'] = data2['review_body'].astype(str).apply(len)
avg_length_before = data2['review_length_before'].mean()

# Apply the cleaning function
data2['cleaned_review'] = data2['review_body'].apply(clean_text)

#  Average review length after cleaning
data2['review_length_after'] = data2['cleaned_review'].apply(len)
avg_length_after = data2['review_length_after'].mean()

print("Average length before cleaning:", avg_length_before)
print("Average length after cleaning:", avg_length_after)
```

```
Average length before cleaning: 316.88439
Average length after cleaning: 299.121885
```

In [19]:

```python
# 3. PRE-PROCESSING

import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

nltk.download('stopwords')
nltk.download('wordnet')

#Defining Stop words and Lemmatizer:
lemmatizer = WordNetLemmatizer()

stop_words = set(stopwords.words('english'))

def preprocess_reviews(reviews):

    #Tokenization:
    reviews = reviews.str.split()

   #Removing stop words and perform lemmatization :
    reviews = reviews.apply(lambda x: [lemmatizer.lemmatize(word) for word in x if word not in stop_words])

   #Joining back into single string:
    reviews = reviews.str.join(' ')

    return reviews

# Print 3 sample reviews before preprocessing
print('Before preprocessing:')
print(data2['review_body'].head(3))

data2['review_body'] = data2['review_body'].astype(str)

#  Average length of the reviews before preprocessing
average_length_before = data2['review_body'].apply(len).mean()
print('Average length before preprocessing:', average_length_before)

data2['review_body'] = preprocess_reviews(data2['review_body'])

# Print 3 sample reviews after preprocessing
print('After preprocessing:')
print(data2['review_body'].head(3))

# Average length of the reviews after preprocessing
average_length_after = data2['review_body'].apply(len).mean()
print('Average length after preprocessing:', average_length_after)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\adity\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\adity\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Before preprocessing:
1750243    My dad used to have this calculator, just in a...
221107                                            Great item
145543     They worked for the Pinterest Mugs :) We did 1...
Name: review_body, dtype: object
Average length before preprocessing: 316.88439
After preprocessing:
1750243    My dad used calculator, another colour scheme ...
221107                                            Great item
145543     They worked Pinterest Mugs :) We 12 mug gotten...
Name: review_body, dtype: object
Average length after preprocessing: 220.907925
```

In [20]:

```python
# 4. TF-IDF FEATURE EXTRACTION

from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize
vectorizer = TfidfVectorizer()

#Transform the reviews into TF-IDF features:
tfidf_features = vectorizer.fit_transform(data2['review_body'])

# Print the shape of the TF-IDF features:
print('Shape of TF-IDF features:', tfidf_features.shape)
```

```
Shape of TF-IDF features: (200000, 73378)
```

In [21]:

```python
# 5. PERCEPTRON MODEL

from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split

# Split the dataset into training and testing sets:
X_train, X_test, y_train, y_test = train_test_split(tfidf_features, data2['Sentiment'], test_size=0.2, random_state=42)

perceptron = Perceptron()

# Training the model:
perceptron.fit(X_train, y_train)

# Using on training and testing data:
y_train_pred = perceptron.predict(X_train)
y_test_pred = perceptron.predict(X_test)


train_accuracy = accuracy_score(y_train, y_train_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)
train_precision = precision_score(y_train, y_train_pred)
test_precision = precision_score(y_test, y_test_pred)
train_recall = recall_score(y_train, y_train_pred)
test_recall = recall_score(y_test, y_test_pred)
train_f1 = f1_score(y_train, y_train_pred)
test_f1 = f1_score(y_test, y_test_pred)

# Print:
print("Perceptron Training Metrics:")
print("Accuracy:", train_accuracy)
print("Precision:", train_precision)
print("Recall:", train_recall)
print("F1 Score:", train_f1)

print("\nPerceptron Testing Metrics:")
print("Accuracy:", test_accuracy)
print("Precision:", test_precision)
print("Recall:", test_recall)
print("F1 Score:", test_f1)
```

```
Perceptron Training Metrics:
Accuracy: 0.90673125
Precision: 0.9037168448998834
Recall: 0.9104828327521342
F1 Score: 0.9070872220804045

Perceptron Testing Metrics:
Accuracy: 0.852125
Precision: 0.8495729042510926
Recall: 0.855649477317061
F1 Score: 0.852600363826659
```

In [22]:

```python
# 6. lOGISTIC REGRESSION AND MULTINOMIAL NAIVE BAYES

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB


log_reg_model = LogisticRegression(max_iter=1000)
nb_model = MultinomialNB()


log_reg_model.fit(X_train, y_train)
nb_model.fit(X_train, y_train)


models = {'Logistic Regression': log_reg_model, 'Naive Bayes': nb_model}

for name, model in models.items():
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)

    print(f"{name} Training Metrics:")
    print("Accuracy:", accuracy_score(y_train, y_train_pred))
    print("Precision:", precision_score(y_train, y_train_pred))
    print("Recall:", recall_score(y_train, y_train_pred))
    print("F1 Score:", f1_score(y_train, y_train_pred))

    print(f"\n{name} Testing Metrics:")
    print("Accuracy:", accuracy_score(y_test, y_test_pred))
    print("Precision:", precision_score(y_test, y_test_pred))
    print("Recall:", recall_score(y_test, y_test_pred))
    print("F1 Score:", f1_score(y_test, y_test_pred))
    print("\n")
```

```
Logistic Regression Training Metrics:
Accuracy: 0.911375
Precision: 0.9136056900863315
Recall: 0.9086954891446999
F1 Score: 0.9111439743332665

Logistic Regression Testing Metrics:
Accuracy: 0.899425
Precision: 0.9049188640973631
Recall: 0.8925623968388936
F1 Score: 0.8986981592929267


Naive Bayes Training Metrics:
Accuracy: 0.877025
Precision: 0.8916630961191394
Recall: 0.8583623932905871
F1 Score: 0.8746959102314267

Naive Bayes Testing Metrics:
Accuracy: 0.864025
Precision: 0.8809548738352005
Recall: 0.8416945931075877
F1 Score: 0.8608773500447628
```

In [23]:

```python
# 7. SVM MODEL

from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Initialize the SVM model
model = SVC()

# Fit the model to the training data
model.fit(X_train, y_train)

# Predict the labels for the training and testing data
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)

# Calculate and print the metrics for the training data
print('Training data:')
print('Accuracy:', accuracy_score(y_train, y_train_pred))
print('Precision:', precision_score(y_train, y_train_pred))
print('Recall:', recall_score(y_train, y_train_pred))
print('F1 score:', f1_score(y_train, y_train_pred))

# Calculate and print the metrics for the testing data
print('Testing data:')
print('Accuracy:', accuracy_score(y_test, y_test_pred))
print('Precision:', precision_score(y_test, y_test_pred))
print('Recall:', recall_score(y_test, y_test_pred))
print('F1 score:', f1_score(y_test, y_test_pred))
```

```
Training data:
Accuracy: 0.97333125
Precision: 0.9730202348238821
Recall: 0.9736648043296211
F1 score: 0.973342412864613
Testing data:
Accuracy: 0.907
Precision: 0.9110796746324458
Recall: 0.9019656879907968
F1 score: 0.9064997737897753
```

In [ ]: