Assignment Report - CSCI544

Shreenidhi Hegde

Read Data:

The data is read from "https://s3.amazonaws.com/amazon-reviewspds/tsv/amazon_reviews_us_Kitchen_v1_00.tsv.gz" file as pandas data frame. The "\t' is used to separate columns by a tab and "error_bad_lines = False" to drop bad lines from the data frame.

Keep Reviews and Ratings:

First, “dropna” is used to look for missing values in the columns which has no review body and drop the corresponding rows. Only the reviews and ratings of the initial data is kept for the further processing. Sample review is captured below along with the rating frequency.

```
 -x--x- Three sample reviews with corresponding ratings -x--x-

         star_rating                                        review_body
2264076          5.0  Do you know what's better than a Seattle Seaha...
2973343          4.0  the soup bowls are little bit smaller for nood...
2090625          3.0   They stain easy & handles get really hot to grab

 -x--x- Statistics of the ratings -x--x-

5.0    3124595
4.0     731701
1.0     426870
3.0     349539
2.0     241939
```

Labelling Reviews:

The reviews with the ratings 4,5 are labelled as 1 (positive reviews) and the reviews with the ratings 1,2 are labelled as 0 (negative reviews). The remaining ratings with 3 are labelled as -1(Neutral) and will be discarded. Below are the statistics. From the sample, 100,000 positive and 100,000 negative reviews are selected randomly.

```
 -x--x- Counts of all review labels before removing the reviews with ra
ting 3 -x--x-

 1    3856296
 0     668809
-1     349539
```

```
Name: label, dtype: int64

 -x--x- Counts of all review labels after removing the reviews with rat
ing 3 -x--x-

1    3856296
0     668809
```

Here is the average length of the reviews in terms of character length before cleaning the data

```
 -x--x- The average length of the reviews in terms of character length
in the dataset before cleaning is -x--x-
322.12

 -x--x- The sample reviews before cleaning -x--x-

4471643    Ordered these in order to help organize my swe...
227922     Expensive plastic ware. If they had been porce...
1984131    they were not gold at all... they were actuall...
```

Data Cleaning:

The corpus is been converted to lower case using inbuilt python function. "***BeautifulSoup***" is used to remove the HTML tags. URLs, non-alphabetical characters are removed by making use of regex expressions. All the extra spaces between the words are been removed. Then contractions is performed using the python's "contractions" library : Eg. After contraction, won't will be converted to will not.

Here is the average length of the reviews in terms of character length after cleaning the data.

```
-x--x- The average length of the reviews in terms of character length i
n the dataset after cleaning is -x--x-
 308.89177

 -x--x- The sample reviews after cleaning -x--x-

4471643    ordered these in order to help organize my swe...
227922     expensive plastic ware if they had been porcel...
1984131    they were not gold at all they were actually b...
```

Pre-Processing:

Stop words are being removed from the dataset using the NLTK library and Lemmatization is performed on the data set.

Here is the average length of the reviews in terms of character length after pre-processing the data.

```
-x--x- The average length of the reviews in terms of character length i
n the dataset after pre-processing is -x--x-
 188.983915

 -x--x- The sample reviews after pre-processing -x--x-

4471643    ordered order help organize sweater shirt fold...
227922                expensive plastic ware porcelain maybe
1984131            gold actually brown color happy told gold
```

TF-IDF Feature Extraction:

The data is been split into training (80%) and testing (20%) datasets. We use sklearn's "***train_test_split***" to do the same. We give the ***test_size*** = 0.2 , which will take train_size as 0.8(1-0.2) and splits the data into test (20%) and train (80%).
"***random_state***" - Guarantees that the output of every run will be the same. The number won't matter.
The data is being standardized using ***StandardScaler*** from sklearn and scaler is being fit to the training feature set only. Testing set is being scaled or transformed using the scaler that was fitted to training data.

Training and testing the data set:

Perceptron:

A perceptron object is created with the parameters: 40 iterations (epochs) over the data, and a learning rate of 0.1. Then the model is being fit to the standardized data. The performance will be measured using the "accuracy_score,f1_score,precision_score and recall_score.

```
 -x--x- Accuracy, Precision, Recall, and f1-score on test data -x--x-

Testing Accuracy:0.8279
Testing f1_Score:0.8274
Testing Precision:0.8267
Testing recall_score:0.8280

 -x--x- Accuracy, Precision, Recall, and f1-score on train data -x--x-

Training Accuracy:0.9262
Training f1_Score:0.9263
```

```
Training Precision:0.9266
Training recall_score:0.9259
```

SVM:

A SVM model is trained on the training dataset using the sklearn built-in implementation. A linear classifier is created using ***linearSVC()*** to process the data efficiently. The performance will be measured using the "accuracy_score,f1_score,precision_score and recall_score".

```
-x--x- Accuracy, Precision, Recall, and f1-score on test data -x--x-

Testing Accuracy:0.8977
Testing f1_Score:0.8970
Testing Precision:0.8992
Testing recall_score:0.8949

 -x--x- Accuracy, Precision, Recall, and f1-score on train data -x--x-

Training Accuracy:0.9335
Training f1_Score:0.9334
Training Precision:0.9353
Training recall_score:0.9315
```

Logistic Regression:

A Logistic regression model is trained on training dataset with default parameters. sklearn's built-in implementation is used for the same. After training, the trained model is applied on the X data to make predicts for the Y test data. The performance will be measured using the "accuracy_score,f1_score,precision_score and recall_score".

```
-x--x- Accuracy, Precision, Recall, and f1-score on test data --x--x-

Testing Accuracy:0.8982
Testing f1_Score:0.8973
Testing Precision:0.9021
Testing recall_score:0.8925

-x--x- Accuracy, Precision, Recall, and f1-score on train data -x--x-

Training Accuracy:0.9136
Training f1_Score:0.9133
Training Precision:0.9174
Training recall_score:0.9092
```

Naive Bayes:

A Naïve Bayes model is trained on the training dataset with Gaussian Classifier parameters. Sklearn's built-in implementation is used for the same. After training , the trained Naïve Bayes model is applied on the data to make predicts for the y test data. The performance will be measured using the "accuracy_score,f1_score,precision_score and recall_score".

```
-x--x- Accuracy, Precision, Recall, and f1-score on test data -x--x-

Testing Accuracy:0.8695
Testing f1_Score:0.8681
Testing Precision:0.8740
Testing recall_score:0.8622

 -x--x- Accuracy, Precision, Recall, and f1-score on train data -x--x-

Training Accuracy:0.8866
Training f1_Score:0.8864
Training Precision:0.8891
Training recall_score:0.8836
```