Nicholas Sam - 190148430
Mehan Sritharan - 190741420
Nihal Medak - 190827670

Contributions:

Nicholas Sam
- Question 1

Mehan Sritharan
- Question 2

Nihal Medak
- Question 2

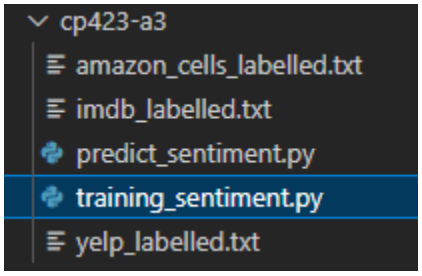
Task Summary:

Q1:

USAGE: `python training_sentiment.py [options]`
Options:
`--imdb, --amazon, --yelp, --naive, --knn [k], --svm, --decisiontree`

EXAMPLE 1: `python training_sentiment.py`
Uses the IMDB dataset and Naive Classifier if no options are selected
EXAMPLE 2: `python training_sentiment.py --amazon --yelp --svm`
Will train an SVM model on the Amazon and Yelp datasets.

Make sure the datasets are in the same folder as the .py file, like so:



`training_sentiment.py` uses SKLearn libraries to train different classifiers on different datasets. Model options are mutually exclusive. If multiple are picked, the model is picked in the priority: Naive Bayes > KNN > SVM > Decision Tree.
Once the model has been trained, it prints the cross-validation accuracy, recall, precision and f1-score. Each number in the dictionary of scores represents the respective score of a certain slice of the data. For example: `'test_accuracy': array([0.745, 0.685, 0.63 , 0.66 , 0.735])` means that when testing the model on the first 20% of the chosen datasets, it achieved an accuracy of 74.5%.

Once it finishes printing the scores, it dumps the TFIDF Vectorizer, Count Vectorizer and the Classifier to use in `predict_sentiment.py`

USAGE: `python predict_sentiment.py "text to predict"`

EXAMPLE 1: `python predict_sentiment.py "This product was great!"`
EXAMPLE 2: `python predict_sentiment.py "I hate this product! It broke and was very inconvenient to use!"`

`predict_sentiment.py` uses SKLearn libraries to predict the sentiment of given text, based on the model that was trained in `training_sentiment.py`. If no model. TFIDF vectorizer, or count vectorizer is present, it gracefully crashes while informing the user to re-run training_sentiment.py

Otherwise, it prints one of two outputs: "The text `"text to predict"` is positive" or "The text `"text to predict"` is negative"

Q2:

USAGE: `python cluster_news.py [options]`
Options:
`--ncluster, --kmeans, --whc, --ac, --dbscan`

EXAMPLE 1: `python cluster_news.py --ncluster --whc 3 7 8`
EXAMPLE 2: `python cluster_news.py --kmeans`

The way that cluster_news.py works is it downloads the dataset of news articles onto using the load_data() method. Then once it gathers these articles it preprocesses the text in the method preprocess_text(), with the use of regular expressions (re library in Python) in order to remove irrelevant information such as titles, paths, extended whitespaces, date, subheadings, etc., further, the program uses the NLTK library in order to remove stopwords from the text. Once the texts are preprocessed, their contents and labels are each returned in an array called file_contents and file_labels, respectively, along with the other articles in the dataset.
After this, the cluster_documents() method uses the scikit-learn machine learning library (sklearn in python) in order to vectorize each set of text based on its TF-IDF (Term Frequency - Inverse Document Frequency). For k-means clustering, agglomerative clustering, and DBscan, the Kmeans, AgglomerativeClustering, and DBSCAN imports from the sklearn library are used to process the vectors and cluster them accordingly using the methods KMeans(amount of clusters, random state), AgglomerativeClustering(amount of clusters), DBSCAN(distance, minimum samples), respectively. For ward hierarchical clustering, it uses the agglomerative clustering method, and converts the vectors for the text into a matrix, and combining vectors and clusters. Finally, the cluster_documents() provides the adjusted mutual info score, adjusted

rand score, completeness score of each clustering operation using the metrics module of the sklearn library and prints them for the user.