Salva Karimisahari 240753803 NLP Assignment 1

Question 1 (coded in NLP_Assignment_1.ipynb):
Parse_data_line: Function will return the second and third elements which correspond to the label and the statement
Pre_process: Steps included in preprocessing at this stage are lowercasing the text, removing punctuation and splitting on whitespaces to create the tokens.

Question 2 (coded in NLP_Assignment_1.ipynb):
To_feature_vector: Binary feature values were used at this stage, so when a list of tokens is given as input, output is a dictionary where keys are tokens and values are 1 if the tokens appear in the text. The global_feature_dict gets updated so that if a token from the input doesn't already exist in it, it will be added and the value is the length of the dictionary so it can work as a unique ID.

Question 3 (coded in NLP_Assignment_1.ipynb):
Cross_validate: The function splits the data into k number of folds and we have k iterations of training and testing where in each iteration, a model is trained on data[:k]+ data [k+fold_size:] and tested on the data in the range of k and k+fold_size (data[k:k+fold_size]). The precision, recall, f1_score and accuracy for each fold is calculated in each iteration and added to the value in a dictionary. The dictionary's sum of metrics are divided by the number of folds to return an average of all metrics across all folds.

Question 4 (coded in NLP_Assignment_1.ipynb):
Error_analysis: The first fold is being used for error analysis and a classifier which is trained on 9 folds of the train data is used to make predictions on one of the folds and the confusion matrix heatmap is based on that. False positive and False Negative cases are saved in false_positive_error_analysis.txt and false_negative_error_analysis.txt. From the heatmap, it can be observed that from a total of 818 negative samples, 225 were wrongly classified as positive (FP) whereas 593 of them were correctly classified as negative (TN). From 1866 positive samples, 1676 were classified correctly (TP) whereas 190 were wrongly labelled as negative (FN). Since the amount of TP and TN is more compared to FP and FN, we can say that the model is performing quite well and is making more true predictions than false ones, especially in identifying the positive class correctly as seen by the numbers and verified by the report (positive precision is 0.88 and negative precision is 076).
The FP being high indicates that the model is prone to incorrectly labelling "negative" cases as "positive." The FN being high indicates that the model occasionally misses true "positive" cases, though to a lesser extent than the false positives. Further analysing FP and FN cases can give us this understanding:
False Positive:

- One thing that could enhance the model is a definition of positive/negative word dictionary so tokens indicating positivity or negativity (sentiment-rich lexicons) would have higher weights. In this sample, "angry" and "dead" are clear signs of the sentence being negative but the classifier labelled it as positive. Original Text: saving episode 12 for tomorrow because she made me angry and i just spent 13 hours playing naruto ultimate ninja 4 so my eyes are dead -- Actual Label: negative Extracted Features: {'saving': 1, 'episode': 1, '12': 1, 'for': 1, 'tomorrow': 1, 'because': 1, 'she': 1, 'made': 1, 'me': 1, 'angry': 1, 'and': 1, 'i': 1, 'just': 1, 'spent': 1, '13': 1, 'hours': 1, 'playing': 1, 'naruto': 1, 'ultimate': 1, 'ninja': 1, '4': 1, 'so': 1, 'my': 1, 'eyes': 1, 'are': 1, 'dead': 1}
- Using bigrams instead of unigrams to capture sentiment-laden phrases such as "not better" or "very good," providing more context than single words alone. In this sample, "not happy is an indicator of the sentiment being negative whereas the token 'better" might indicate positivity. False Positive: Original Text: @_2Shades_ maybe 3rd team bro, he's not better than trey Burke from Michigan -- Actual Label: negative
Extracted Features: {'_2shades_': 1, 'maybe': 1, '3rd': 1, 'team': 1, 'bro': 1, 'hes': 1, 'not': 1, 'better': 1, 'than': 1, 'trey': 1, 'burke': 1, 'from': 1, 'michigan': 1}
- Preprocessing can be enhanced by removing repetitive characters that make distinguishing words harder, since the context is casual and no lexical rules are followed. In this sample the word "buuuut" could be replaced with but since in english we don't have the repetition of the letter "U". False Positive: Original Text: @shelboxoxo I'm going to Cuba buuuttt I doubt I will be doing what y'all will #13.1 Sunday -- Actual Label: negative Extracted Features: {'shelboxoxo': 1, 'im': 1, 'going': 1, 'to': 1, 'cuba': 1, 'buuuttt': 1, 'i': 1, 'doubt': 1, 'will': 1, 'be': 1, 'doing': 1, 'what': 1, 'yall': 1, '131': 1, 'sunday': 1}

False Negative:

- Instances of wrong classification because of the binary weighting scheme can also be seen in the samples and using a weighting scheme that distinguishes sentiment-rich lexicons would be beneficial.
- Removing stop words like "the", "and", "is" that don't carry a significant meaning will reduce noise. In this sample, most of the tokens aren't contributing to the positive message. Also, removing links and mentions would result in more

distinguished features. False Negative: Original Text: Amazon may be cooking up a $50 tablet The Internet giant is reportedly working on new Fire tablets for the... http://t.co/yRmqFq3PxN -- Actual Label: positive Extracted Features: {'amazon': 1, 'may': 1, 'be': 1, 'cooking': 1, 'up': 1, 'a': 1, '50': 1, 'tablet': 1, 'the': 1, 'internet': 1, 'giant': 1, 'is': 1, 'reportedly': 1, 'working': 1, 'on': 1, 'new': 1, 'fire': 1, 'tablets': 1, 'for': 1, 'httptcoyrmqfq3pxn': 1}

Question 5 (coded in NLP_Assignment_1_Q5.ipynb):

To compare different outcomes, an input “method” was added to the preprocess function, the goal is to add different steps to preprocessing. Possible options are removing URLs, mentions and emails, removing non-alphanumeric characters, normalising elongated words, removing stopwords, lemmatizing and word stemming. Each of the methods have also been tested by being added separately and the results are available in the notebook.

For to_feature_vector different methods are also added such as bag of words, **positive/negative dictionary-based weighting** method assigns weights directly based on sentiment dictionaries and a TF-IDF approach assuming we have idf-values precomputed and using NLTK Linguistic features in NLP_Assignment1_Q5_feature_vec.ipynb

	Original version	**All preprocessing added**	**Bag of words feature vector**			
Precision	**0.84**	**0.93**	**0.99**			
Recall	**0.84**	**0.93**	**0.99**			
f1-score	**0.84**	**0.93**	**0.99**			
Accuracy	**0.84**	**0.93**	**0.99**			

Test results with all preprocessing and pos_neg_dict : `Precision: 0.704260`
`Recall: 0.671884`
`F Score:0.568799`