Task 1

Inside the split() function I read the xlsx file into a dataframe and then split the data into train and test based on the “Split” column, I then further split the data into review & sentiment and then return these lists. (Task 1.1, Lines 21→24) I also print the count of the positive and negative subsets for both the training and test data. (Task 1.2, Lines 26→29)

Task 2

Inside the extract() function I cleanse the data for each review in the training data subset (Task 2.1, Line 36), I set the data to be lowercase (Task 2.2, Line 37), and I then split the reviews into individual words. (Task 2.3, Line 38) Next, I horizontally stack the data and then count the unique values and put them in a DataFrame. (Task 2.4, Line 43) Finally, I drop the rows where the word was either less frequent or shorter than the minimum specified. (Task 2.5, Lines 44→45)

Task 3

Inside the countReviews() function I take a list of words from Task 2 and a list of reviews, I add each word from Task 2 to a dictionary and initialize it’s matching value to 0. I search for the words from Task 2 in the reviews and increment the value for that word in the dictionary if it’s found. I do this for both the positive and negative reviews. (Task 3, Lines 54→60)

Task 4

Inside the calculateLikelihoods() function I take the values generated so far and use them to calculate the priors and the likelihood that a given word is in a positive or negative review by calculating how many positive/negative reviews the word appears in, divided by the total count of positive/negative reviews. (Task 4.2, Lines 68→79) I also applied laplace smoothing so that in the event a word had a value of 0 the function would not multiply by 0. (Task 4.1, Lines 76 & 79)

Task 5

Inside the predict() function I take a review and both dictionaries generated in task 4 and I find the likelihood that the review is positive or negative given the words used in it using Naïve Bayes classification. (Task 5, Lines 90→100)

Task 6

Inside the evaluate() function I begin by splitting the training dataset into 5 folds inside a loop that runs 10 times, I then call the extract function from Task 2 passing in the new kfold subset with a minimum word length equal to the current iteration of the loop. Now that I have the extracted words for this subset of the the training data, I use the countReviews() function

from Task 3 to get the dictionaries of values for each word in the subset reviews, I do this for both the positive and negative reviews as in Task 3. Next, I use the calculateLikelihoods() function from Task 4 with the new subset to generate the priors and the likelihood that a given word in the subset's reviews are positive or negative by calculating how many positive/negative reviews the word appears in. Finally, I loop through each review in the subset and pass the review and the newly generated values into the predict() function from Task 5 to make a prediction whether or not this review in particular is likely to be positive or negative, I write the outcome for each review to a list so I have a list of the predicted results. (Task 6.1, Lines 108→123) Now that I have a list of predicted results, I measure the accuracy of this run (Task 6.2, Line 126) and get the average of the 5 fold's runs (Task 6.3, Line 128), I then add this average to a list that will hold each average run value for each of the 10 loops (Task 6.4, Line 132), I then get the max value of these average's and its position in the list to find which of the word lengths is the most optimal. (Task 6.5, Line 133) Now that I have found this value, I run Task's 2 through 4 again using the training data and a minimum word length of the value found. Next, I predict the sentiment of the testing data using the model generated from Tasks 2 through 4. I then apply a confusion matrix to determine the accuracy of my predictions (Task 6.6, Line 149), I print the percentages for true positive, false positive, true negative, and false negative predictions from this confusion matrix (Task's 6.7, 6.8, 6.9, 6.10, Lines 152→160), finally, I measure the accuracy of this prediction versus the real sentiment to find the classification accuracy score. (Task 6.11, Line 162)