Design Document

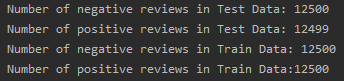
**Question 1**

* Data represents the reviews themselves
* Label represents the sentiment or prediction label

The data is split according to the indicator in the csv file which indicates which cells belong to the training data and which cells belong to the test data

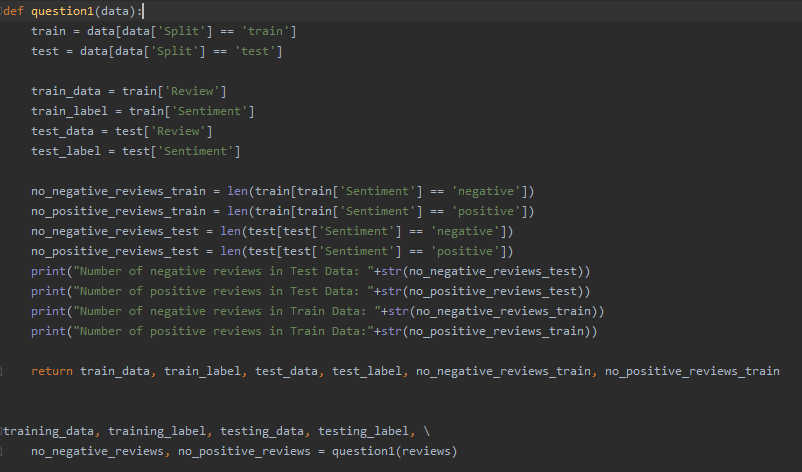
Question 1 extracts the data and labels from the csv file and returns the training data, training label, test data and test labels.

It also counts the number of negative and positive reviews in the test and training data



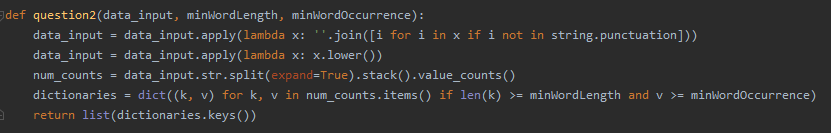
This function will return the following variables that will be used in later functions

* the training data,
* training label,
* test data,
* test labels,
* number of negative reviews in the training data,
* number of positive reviews in the training data



**Question 2**

Only extract the words found in the labels according to the parameters



The parameters include

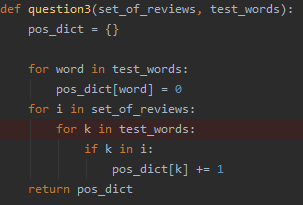
* Data\_input 🡪 this represents the pandas dataframe in which the word is extracted from
* minWordLength 🡪 represents the minimum word length that the function extracts
* minWordOccurence 🡪 represents the threshold that indicates the minimum number of times the word occurred in a review

the function will return a list containing the words that satisfy these requirements

1. all of the words will be converted to lower case and any punctuation is removed
2. a dictionary Is created which will input all of the datasets
3. num\_counts 🡪 this returns a list of words along with the number of times it has occurred in the datasets including the reviews
4. dictionaries 🡪 this returns a list of words that are len than or equal to the minimum Word length and occurs more times than the minWordOccurence
5. it will return a list of keys of the dictionary representing the words extracted

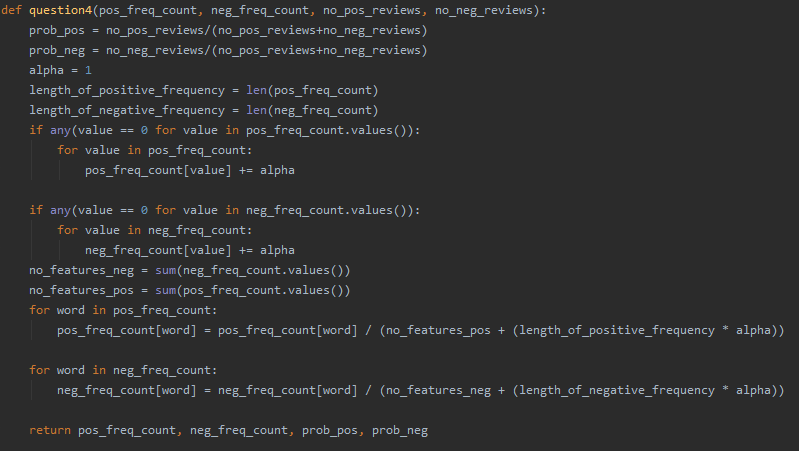
**Question 3**

Count the number of times a word has occurred in the data



1. it creates a dictionary which initially maps the values to 0
2. for every review the word has occurred, the value for that word will increment to 1
3. the function returns a dictionary with the words and the amount of reviews containing that word (**pos\_dict**)

**Question 4**



where the likelihoods and priors are calculated. It takes in the following parameters

* pos\_freq\_count 🡪 the dictionary containing the words in the positive reviews along with the number of reviews the word occurred in
* neg\_freq\_count 🡪 the dictionary containing the words in the negative reviews along with the number of reviews the word occurred in
* no\_pos\_reviews 🡪 the number of positive reviews
* no\_neg\_reviews 🡪 the number of negative reviews

the priors are simply calculated by:

the number of reviews/total length of the training data

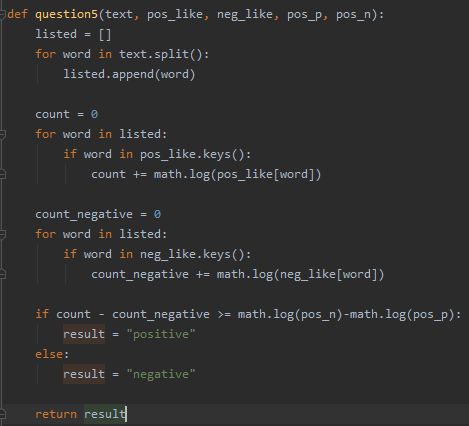
laplace smoothing is applied by adding 1 to every frequency count of each word whenever any of the frequency count = 0

then the likelihood is calculated by:

* number of reviews containing the word/(total sum of the frequency counts of each word + (length of the frequency count dictionary \* alpha)

function returns the positive\_frequency, negative\_frequency, positive\_prior, negative prior

**Question 5**



The function takes in a string representing the review itself and the positive\_likelihood for each words in the training data, negative\_likelihood for each words in the training data, positive\_prior, negative prior previously calculated in question4()

First, it splits up the string into separate words and adds them into an array called listed

Count and count\_negative takes the total sum of the log value of each word’s likelihood in the text for the positive likelihoods and the negative likelihoods respectively.

If the (total sum of the log value of the positive likelihood – total sum of the log value of the negative likelihood) is greater than or equal to (log of prior positive(pos\_p) – log of prior negative(pos\_n))

result = positive

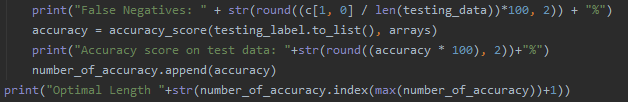
else:

result = negative

the function will return either the string “positive” or “negative” depending on its predictions

**Question 6**





First, the function creates a kfold cross validation of KFold(n\_splits=5, shuffle=False, random\_state=None)

Then the function loops from 1 to 10 to test for each of the word lengths.

It creates two arrays,

* arrays 🡪 this will contain all of the labels predicted on each of the reviews on the test data
* kfold\_accuracies 🡪 this will contain all the accuracy scores calculated from the predictions on the validation set and the validation labels themselves

during the kfolds:

* kfold\_arrays 🡪 this will hold all of the predicted values on the validation set
* question3() function takes in the training data indicated by the train index represented by (k) where the label is positive i.e. training\_data.iloc[k][training\_label.iloc[k] == ‘positive] and the list of words extracted in question2() function according to the parameter.
* The same is done for the negative reviews.
* Then question4() function calculates the likelihoods and priors for positive and negative values in the dictionaries created in question3()
* For each item (pika) in the validation set (train\_data.iloc[kl]:
  + Predict if the review is positive or negative and append to the array called kfold\_arrays
* The mean score of the KFold Cross validation is calculated as the mean of all the elements in the kfold\_accuracies array (np.mean(kfold\_accuracies)

The function then loops through the testing data and appends the labels predicted by the function **question5()** on each of the reviews in the testing data to an array called **arrays** which will then be predicted by the testing\_label converted to a list by the method to\_list()

**c** represents the Confusion matrix which is represented as:

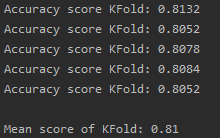
[[True positives|False Negatives]

[False positives| True Negatives]]

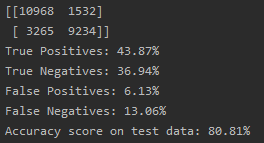
Thus:

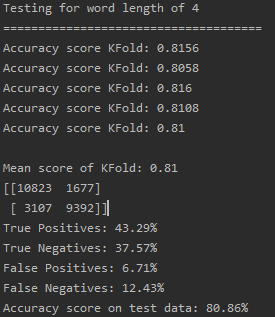
* True positive percentage = c[0,0]\length of test data
* True negative percentage = c[1,1]\length of test data
* False positive percentage = c[0, 1]\length of test data
* False negative percentage = c[1,0]\length of test data

Accuracy scores for the KFolds for length 1



Total accuracy for the classifier prediction on the test\_data



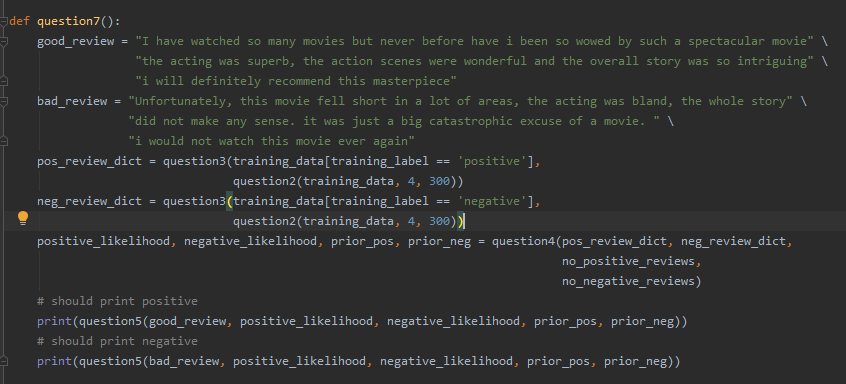




The optimal word length is a minimum of 4 letters in the word because this is the parameter that will produce the highest frequency with a frequency of 80.86%



**Question 7(optional)**



This basically tests two reviews, one positive, one negative

They run questions 2 🡪 question 5 by training the classifier on the training data, then predicting on the reviews if it is positive or negative