Assignment Report

Naïve Bayes Algorithm:

Algorithm consists of the following components:

1) **Def: initialize_training_data -**
   This module in Python does the following:
   a) Parses in the file directory containing ham and spam training folders
   b) Sends the parsed data to training module.

2) **Def: training-** This module does the following:
   a) Tokenizes the data in to words.
   b) Excludes the stop words from the words if the label used is "tSSwords"
   c) Add the category to categories set if that is already not present otherwise increase its count.
   d) Add the occurrence of this word mapped to the given category.

3) **Def: writeToFile-** This module writes the training results to following files:
   a) bagofwords.pkl , categories.pkl,word_counts.pkl (without stop words)
   b) bagofwordswithss.pkl , categorieswithss.pkl,word_countswithss.pkl (without stop words)

4) **Def: ReadDSFromFile:** After the training is done, the method is used to retrieve back the data results of training in classification of test data.

5) **NBClassification_test**:
   This module in Python does the following:
   c) Parses in the file directory containing ham and spam training folders
   d) Sends the parsed data to classify_test module.

6) **Classify_posterior:**
   a) Tokenizes the data in to words.
   b) Excludes the stop words from the words if the label used is "tSSwords"
   c) Calculates the probabilities as follows:
      - Checks if the word belongs to the word count of that Add the category to category (looping on categories).
      - Equation: Log(p(word in cat |total words in cat) : Using laplace smoothening:
        If word is present:
        Log((wordcount_cat + 1)/(total_words + length0fWords)
        Else:
        Log(1/(total_words + length0fWords)
   d) Calculate the prior and then calculate the posterior for each category
      Posterior = likelihood (prob) * prior

7) **classify_SentbySent**: This module does the following:
   a) Predict the class based on the category with maximum posterior value.

8) **Accuracy_sentence_score:**
   Once the class is predicted , check if the predicted class is equal to the given label. If it matches increase the true parameter of Performance_params (Performance_params is the class defined for calculating accuracy)

Logistic Regression Algorithm:

Algorithm consists of the following components:

1） **Def: Initialize_Data**
   This module in Python does the following:
   a) Parses in the file directory containing ham and spam training folders
   b) Tokenizes the words from files of both Ham and spam folders.
   c) Create a set of unique words from both files using union. Assign it to Bag of words.

9) **Def: Training_Data -** This module does the following:
   a) Initialize the weights (0.5), theta , learning rate (as per given in arg).
   b) Send the file data for calculating cost regression.

10) **Def: writeToFile-** This module writes the training results to following files:
   a) bagofwords.pkl (without stop words)
   b) bagofwordswithss (with stop words)

11) **Def: ReadDSFromFile:** After the training is done, the method is used to retrieve back the data results of training in classification of test data.

12) **costReg:** For each word in the file, do the following**:**
   a) If the word is present in the bag of words, multiply the weight for that word (unique feature).
   b) Calculate the sum by adding all the terms calculated in a)
   c) Putting the term = sum*theta as X .
   d) Calculate the sigmoid function with X.
   e) Calculate the delta value by subtraction the sigmoid output from given y (0 or 1)
   f) delta = LR.sigmoid(sum*theta) – y
      grad = (1.0 / m) * delta + ((learningRate / 2*m)*(theta*theta))
   g) Add the gradient to all the weights.

13） **Predict :**
   The module predicts the value based on cost regression values . The same accuracy parameter of Performance _ params is used to calculate the accuracy.

Results:

Naïve Bayes:

1) When used without filtering stop words:
   Accuracy : 76.56903765690377

   When stop words are filtered :
   Accuracy: 78.24267782426779

Reason:
The accuracy has increased because the stop words are irrelevant to the classification of text as they appear almost with equal probabilities in all the classes/ categories.
Thus the stop words add noise to the prediction. Filtering them out will thus increase the accuracy.

Logistic Regression:
Results:

Initial Theta = 0.001
Initial weights = 0.5
W0 = 4

Learning rates:

1) When used 1:
   accuracy for the Logistic Regression with theta=0.001 and learning rate =1 with Stop words is:
   72.80334728033473
   *** Test Set ***
   *** Test Set ***
   accuracy for the Logistic Regression with theta=0.001 and learning rate =1 without Stop words is:
   72.80334728033473

2) When used 5:
   accuracy for the Logistic Regression with theta=0.001 and learning rate =5 with Stop words is:
   72.80334728033473
   *** Test Set ***
   *** Test Set ***
   accuracy for the Logistic Regression with theta=0.001 and learning rate =5 without Stop words is:
   72.80334728033473