**ASSIGNMENT - 2**

**TEXT CLASSIFICATION**

The classification of emails as either spam or ham was implemented using two methods: Naïve Bayes and Logistic Regression. The text files containing both the emails were used, some for training and the rest as test data. This was implemented using Python 3.6.

**RESULTS:**

|  |  |
| --- | --- |
| Logistic Regression with stop words  (For eta =0.001 and lambda = 0.3)  Number of iterations =100 | Accuracy = 92.25941422594143 % |
| Logistic Regression without stop words  (For eta =0.001 and lambda = 0.3)  Number of iterations =100 | Accuracy = 95.18828451882845 % |
| Logistic Regression with stop words  (For eta =0. 001 and lambda = 3)  Number of iterations =100 | Accuracy = 92.46861924686192 % |
| Logistic Regression without stop words  (For eta =0. 001 and lambda = 5  Number of iterations =100 | Accuracy = 95.18828451882845 % |
| Logistic Regression with stop words  (For eta =0. 001 and lambda = 10)  Number of iterations =100 | Accuracy = 92.46861924686192 % |
| Logistic Regression without stop words  (For eta =0. 001 and lambda = 10)  Number of iterations =100 | Accuracy = 95.18828451882845 % |
| Naive Bayes with stop words | Accuracy: 94.9790794979 % |
| Naive Bayes without stop words | Accuracy: 94.35146443514645 % |

**Logistic Regression:**

In logistic Regression for text classification, the training data is used to calculate the weights and these weights are further used while testing a new data for calculating the probabilities. Overfitting occurs if a model is trained too much it will fit the training data too well but when applied on a new set of data the prediction accuracy is too less. To avoid overfitting, we need to use L2 regularization.

L2 weight regularization penalizes weight values by adding the sum of their squared values to the error term. In summary, large model weights can lead to overfitting, which leads to poor prediction accuracy. Regularization limits the magnitude of model weights by adding a penalty for weights to the model error function. L2 regularization uses the sum of the squared values of the weights.

**Regularization Weight**

There are several ways to find a good (but not necessarily optimal) regularization weight.  The major advantage of using regularization is that it often leads to a more accurate model. The major disadvantage is that it introduces an additional parameter value that must be determined, the regularization weight.  Increasing lambda results in reducing overfitting but in turn increases the bias.

* Run the code for different values of lambdas.
* For each lambda, calculate the corresponding weights.
* Choose the lambda values for which the weights fit the data properly.

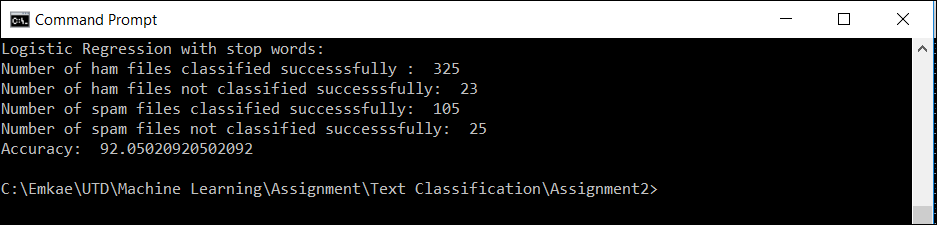
In this code, I have taken different values of lambda such as 0.001, 3, 10.

**Iterations:**

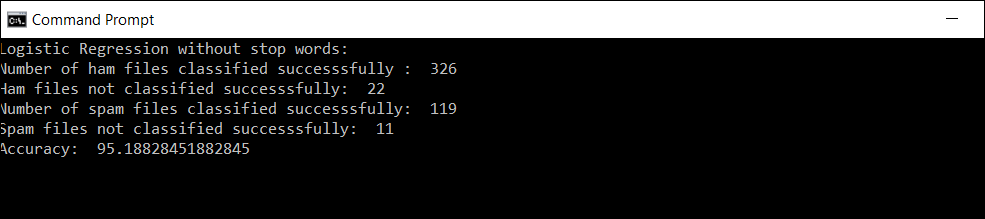
The number of iterations also affects the accuracy of the training data. If the number of iterations are too less, then the data will not even converge. If the number of iterations are too high, then the data will diverge too much. Hence the number of iterations must be kept moderate to get a good accuracy.

In this code, I have taken the number of iterations to be 100 which is not too high or too low.

**>>> python LogisticRegressionWithoutStopWords.py** **<traindata -path> <testdata-path>**



**>>> python LogisticRegressionWithStopWords.py** **<traindata -path> <testdata-path>**

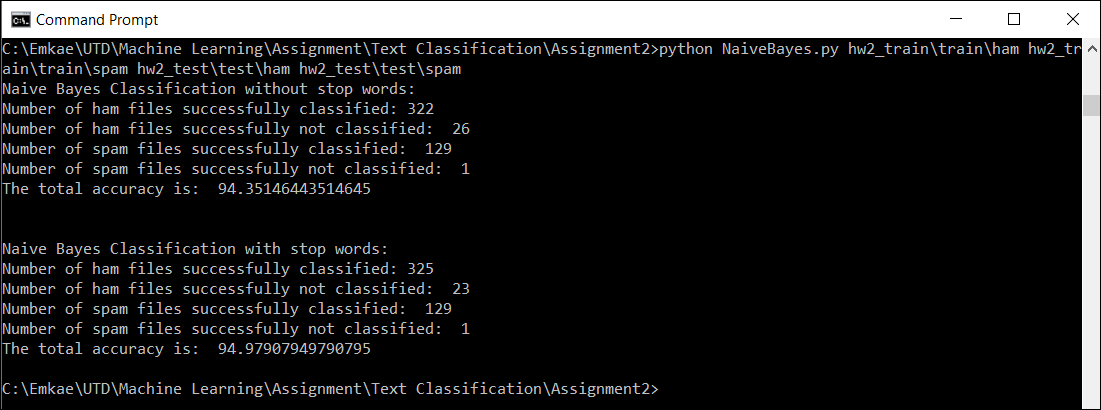


There is a 3 percent increase in the Logistic Regression accuracy when the stop words have been removed. Stop words are equal likely to be present in both ham and spam frequently and does not help in classifying one from other. Hence removing these weights, the dimensions of the problem decreases and hence increases the accuracy.

**Naïve Bayes**

Multinomial Naïve Bayes was implemented for text classification. Laplace Smoothing is done while implementing this, so that the zero that can be obtained while calculating the result, does not affect the probability of the words. An extra 1 is added to the numerator and the denominator to avoid ending up in a zero probability. It basically adds one extra count to each word count. This addition will not affect the result as it is negligible when compared to the actual word count.

**>>> python NaiveBayes.py <train-ham-path> <train-spam-path> <test-ham-path> <test-spam-path>**



There is not so significant difference in the accuracies of NB with and without stopwords. There can be two reasons for this:

1. Stop words were not collected properly and hence removal does not affect the accuracy.
2. Stop words are equal likely to be present in both ham and spam frequently and does not help in classifying one from other. Most of the stop words are present equal likely in both spam and ham, hence removing them do not have much impact.