***Implementation of a spam filter through a Bayesian classifier***

*Statistics and Mathematical Analysis*

*Name: Valèria Ezquerra Rodriguez*

*Name: Eugènia Pacheco Ferrando*

*Name: Laia Casas Irure*

**Question 1 *(2p)***

How many spam and ham messages make up the original “SMSSpamCollection” dataset? Write a short Matlab code to calculate it:

| *Since there’s a table called all that has all the messages from the dataset, together with the information of it being ham or spam, we simply do a for loop that iterates through all the rows and count spam or ham depending on the word stored. To check it, we sum both counters and we get the total number of rows contained in the all matrix (5574) being count\_ham 4827 and 747.*  count\_ham = 0;  count\_spam = 0;  for i = 1: size(all, 1)  string\_message = all(i,1);  if strcmp(string\_message, 'ham')  count\_ham = count\_ham + 1;  else  count\_spam = count\_spam + 1;  end  end |
| --- |

**Question 2 *(2p)***

Explain how you calculated the probability of each class, that is, P(ham) and P(spam) within the training set; if necessary, include some lines of Matlab code or pseudocode to help you in the explanation.



**Question 3 *(2p)***

Explain how you calculated the probability of occurrence of each word of each class in the training set, i.e. P(w\_i |C), where class C can be either spam or ham; if necessary, include some lines of Matlab code or pseudocode to help you in the explanation.



**Question 4 *(2p)***

When classifying a text message belonging to the test set in one of the two categories, please explain:

a) what happens if a word from the message does not appear in the training set

b) what have you done in the face of this situation

| *If a word is not found in hamBag keys or spamBag, we may lose the accumulated probabilities of the message when multiplying this result by zero. To solve it, we have set it to 0.0001 in order to keep it as small as possible without reaching zero.* |
| --- |

**Question 5 *(2p)***

The division of the original data set into training and test subsets is random. Therefore, each time you run the code you may get a slightly different accuracy rate.

Run your code 5 times and calculate the accuracy at each run, and also calculate the average accuracy of the 5 runs. Do the process twice: the first time commenting out the call to the crossDeleteWords function, and the second time, uncommenting it.

With the results obtained, complete the following table:

|  | **Run 1** | **Run 2** | **Run 3** | **Run 4** | **Run 5** | **Average** |
| --- | --- | --- | --- | --- | --- | --- |
| With crossDeleteWords | 0.8260 | 0.8350 | 0.8269 | 0.8278 | 0.8296 | 0.829 |
| Without crossDeleteWords | 0.8341 | 0.8305 | 0.8323 | 0.8414 | 0.8395 | 0.8356 |

Next, discuss the results obtained in both cases in terms of accuracy and execution time. Do you notice differences in the results with and without calling the function? What can they be due to?

| *Overall, we can conclude that the accuracy is higher without the function since deleting some common words may not consider them when analyzing a message. In our case, whenever we look for a word in a Spam or Ham bag and it doesn’t exist, we multiply the accumulative probability by zero. Also, in terms of execution time, compiling without the function is way faster than doing with it.* |
| --- |