Task 2

This task was not as computationally heavy on the machine as the previous one, but I still decided to perform some vectorization to achieve better results. Firstly, I applied some basic algebra to the log probability function to simplify it:

$$ln(P(\boldsymbol{b}|Ck)) = ln\left(\prod_{i=1}^{D} P(b_i|C_k\right) = ln\left(\prod_{i=1}^{D} P(b_i = 0|C_k)^{1-b_i} P(b_i = 1|C_k)^{b_i}\right)$$

As $P(b_i = 0|C_k) = 1 - P(b_i = 1|C_k)$ and $ln(a^x b^y) = xln(a) + yln(b)$:

$$ln(P(\boldsymbol{b}|Ck)) = \sum_{i=1}^{D} (1 - b_i) ln(1 - P(b_i = 1|C_k)) + b_i\, ln\left(P(b_i = 1|C_k)\right)$$

Then, I vectorized the operation as a multiplication of the following two matrices:

$$\begin{matrix} 1 - b_{11} & b_{11} & \dots & 1 - b_{1D} & b_{1D} \\ \dots & \dots & \dots & \dots & \dots \\ 1 - b_{N1} & b_{N1} & \dots & 1 - b_{ND} & b_{ND} \end{matrix}$$

and

$$\begin{matrix} \ln(1 - P(b_1 = 1|C_1)) & \dots & \ln(1 - P(b_1 = 1|C_k)) \\ \ln(P(b_1 = 1|C_1)) & \dots & \ln(P(b_1 = 1|C_k)) \\ \dots & \dots & \dots \\ \ln(1 - P(b_D = 1|C_1)) & \dots & \ln(1 - P(b_D = 1|C_k)) \\ \ln(P(b_D = 1|C_1)) & \dots & \ln(P(b_D = 1|C_k)) \end{matrix}$$

The dimensions of the first matrix are $Nx2D$ and $2DxK$ of the second one, where K is the number of classes. As a result, after the matrix multiplication we get a matrix of size NxK, which stores the probability of each training vector to belong to each class. Vectorizing the data does require a bit of computation time but not nearly as much as it would take to use nested for loops instead of matrix multiplication. To avoid the problem of $ln(0)$ I used a value of $ln(1 * 10^{-10})$, as advised in the course webpage. While investigating the effects of the threshold on the accuracy of the classification, I found that the algorithm performs surprisingly well for even the most extreme values of the threshold, such as 0.05 or 0.95. Furthermore, I found that the best accuracy is achieved using 0.2 as a threshold. This gives the accuracy of 64.33%. It's not surprising that the algorithm is not very effective as a lot of information about the data is lost while performing the vectorization. The runtime was not a problem in this case, with a value of around 2.5 seconds. This proves what can be achieved by more clever programming, as the first draft of mine took almost 20 minutes to finish on my laptop.

UUN: s1661552

Threshold	**Time taken, s**	**Accuracy, %**	**N**	**N_{errs}**
0.05	2.49	64.06	7800	2803
0.1	2.35	64.23	7800	2790
0.2	2.39	64.33	7800	2782
0.5	2.34	63.85	7800	2820
0.8	2.30	62.64	7800	2914
0.95	2.37	58.58	7800	3230
1	2.31	3.84	7800	7500