# CS464: Introduction to Machine Learning Fall 2022

# Homework 1

**Duru Naz Han**

**21902864**

**30.10.2022**

## Question 1: The CS464 Case

1. From the question we have:

$$P(S_M|H) = 0.87$$
$$P(S_U|H) = 0.13$$
$$P(S_M|L) = 0.21$$
$$P(S_U|L) = 0.79$$
$$P(S_M|F) = 0.04$$
$$P(S_U|F) = 0.96$$

Using Bayes's Rule we have:

$$P(S_M) = P(S_M|H) * P(H) + P(S_M|L) * P(L) + P(S_M|F) * P(F)$$

where $P(H), P(L)\ and\ P(F)$ are given.

$$P(S_M) = 0.87 * 0.64 + 0.21 * 0.24 + 0.04 * 0.12 = 0.612$$

2. Again, using Bayes's Rule we have:

$$P(H|S_M) = \frac{P(S_M|H)*P(H)}{P(S_M)}$$
$$= \frac{0.87*0.64}{0.612}$$
$$= 0.9098$$

3. Again, using Bayes's Rule we have:

$$P(H|S_U) = \frac{P(S_U|H)*P(H)}{P(S_U)} = \frac{P(S_U|H)*P(H)}{P(S_U|H)*P(H) + P(S_U|L)*P(L) + P(S_U|F)*P(F)}$$
$$= \frac{0.13*0.64}{0.13*0.64 + 0.79*0.24 + 0.96*0.12}$$
$$= 0.2144$$

## Question 2.1: Sports News Classification

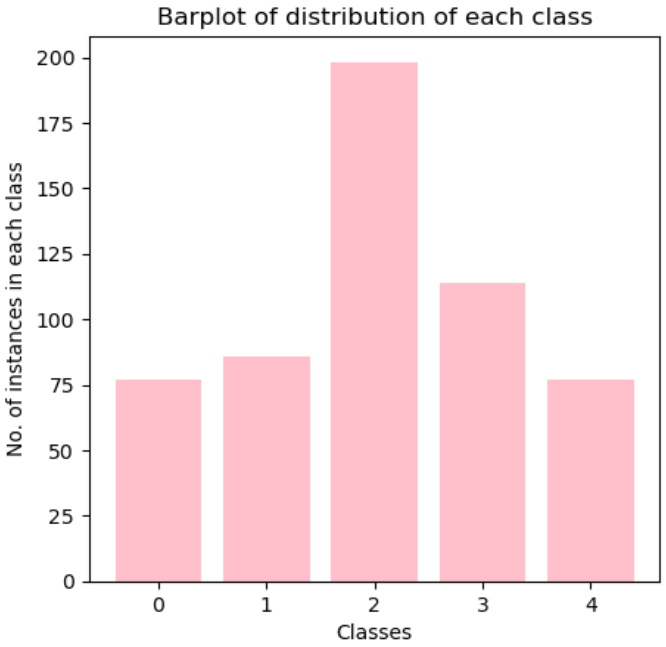


Fig. 1: Barplot of distribution classes of the training data

1. The distribution of the classes in the training data can be visualized from fig.1. Below table also shows the exact number of each class in the data set:

Table 1:Distribution classes of the training data

| Class Labels | Number of classes |
|---|---|
| 0 | 77 |
| 1 | 86 |
| 2 | 198 |
| 3 | 114 |
| 4 | 77 |

2. As it can be seen from the barplot in fig. 1 and the numbers in table 1, the training data set is skewed towards class "2" which indicates the category for football, with 198 instances. The rest of the classes have relatively close numbers of instances such as: athletics and tennis have 77, cricket has 86 and ruby has 114 instances. Having an imbalanced data set has a negative effect on the model. In the Naive Bayes model, we train our model by looking at the probability of a document belonging to a class $C_K$. The highest probability among all classes shows that the document belongs to that class. To do so, we use a prior probability which indicates the total number of samples

belonging to $C_K$ over the total number of samples. If $C_K$ has a lot more samples than others, the other classes will have imprecise estimates. This will affect the learning process as the posterior probabilities will be biased towards $C_K$. To solve this problem, we can balance the data set by increasing the number of samples for other classes too.

3. As the below fig. 2 indicates, the validation data set has a similar distribution to the training data set. Since class 2 has more samples than all the other classes, the accuracy of our model will be misleading. As it was mentioned in question 2.2, the probability of the total number of samples belonging to $C_K$ over the total number of samples would be misleading.

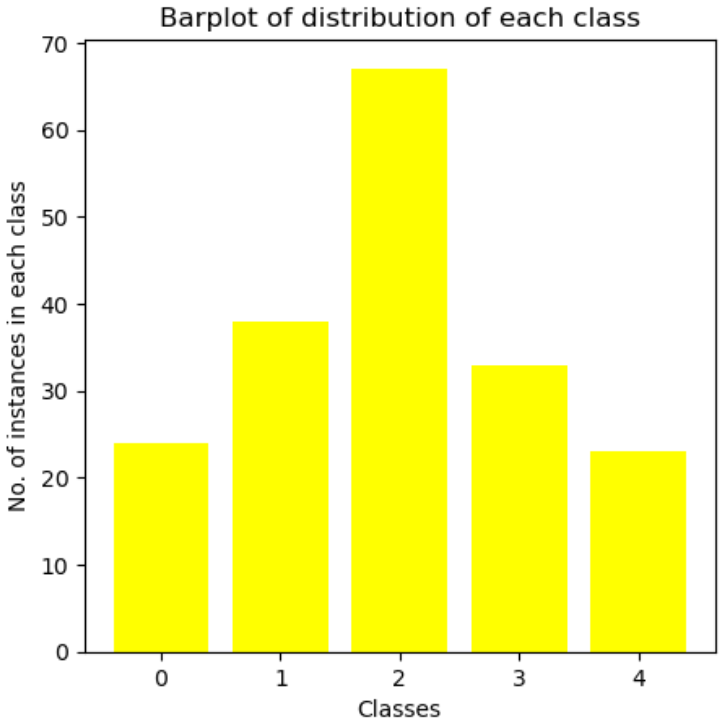


Fig. 2: Barplot of distribution classes of the validation data

4. Skewing towards one of the classes will lead to an increase in false positives. Because the trained model will be more likely to classify documents that do not belong to that class into that class. Therefore, the accuracy will be misleading.

## Question 2.2: Coding Part 1

- For this part, I assumed the log(0) as -inf by using np.nan_to_num(-np.inf).
- The accuracy of the model was reported as 0.31351351351351353. This is a very low accuracy due to the -inf theta probability of some words. When I put log(0) = log, which is an arbitrarily small number, the accuracy came out to be 0.97. However, we need to put the literal value of log(0) for a more correct model as it was specified in the homework manual too. As it can be seen from the following confusion matrix, due to this error, the model mostly predicts the labels as 0. Because whenever a word in a document has the probability of 0 for belonging to any class, the whole document's

probability automatically goes to -inf even when there are words with probabilities more than 0. Therefore the model places it to the class with the lowest label number, 0.

- The confusion matrix is as follows:
  [[24, 0, 0, 0, 0], [33, 5, 0, 0, 0], [45, 0, 22, 0, 0], [30, 0, 0, 3, 0], [19, 0, 0, 0, 4]].
- The model predicted 127 labels incorrectly and 58 labels correctly.

## Question 2.3: Coding Part 2

- For this part, I assumed the log(0) as -inf by using np.nan_to_num(-np.inf).
- The accuracy of the model was reported as 0.972972972972973.
- The confusion matrix is as follows:
  [[24, 0, 0, 0, 0], [0, 35, 1, 2, 0], [0, 0, 66, 1, 0], [0, 0, 0, 33, 0], [1, 0, 0, 0, 22]]
- The model predicted 5 labels incorrectly and 180 labels correctly.

## Question 2.4:

- The Dirichlet prior caused a significant increase in the accuracy of the model from 0.3135 to 0.9729. This prior is used for smoothing because there are some words with 0 probability of belonging to a given class. This probability can be misleading as it is “too strong” of an assumption, and can cause false negatives mostly. This abundance of false negatives alters the accuracy of the model therefore the first model, MLE estimator, has a lot less accuracy than the MAP estimator model.