1. Looking at test data gives the machine learning algorithm an unfair advantage, as it has to classify data that is exactly the same as the data it learned from. Therefore it is a very easy task to just classify in the same category the data had during the training, and thus performing extremely well. The amount of unseen data on the other hand can be useful to experiment with however, as not always as much as 25% is needed of possible. If the dataset is too small for missing such a big part, sometimes a smaller percentage will suffice,
2. Cross validation is the process where the test- and training part of the data are shuffled after each learning and testing round, which makes for a very robust and well-informed dataset that achieves well on all unseen data, with a smaller chance of coming across new key features without recognizing them. Usually 5-10 times is sufficient for swapping all partitions of the data, as the unseen test-data usually is 10-20% of the total data. X-fold cross validation also makes for higher reliability, as the chance of a lucky shot with very familiar data is reduced.
3. For binary classification a most frequent class baseline would be appropriate, as this would already get an accuracy of 50%. The prior of positive is actually even 50,36. Of course this still is a very low baseline, as by picking one of two randomly every time you should get the same score in theory. That way you would even have a higher average, as the most frequent class baseline will neglect the other class completely, and so you will have a very poor average. Looking at the posteriors, mostly one of the options jumps out with about double the percentage of certainty of the other options (60-70 versus 30-35), which makes for a much higher score than the mentioned baseline.

Multiclass classification on the other hand is a lot harder, as there the priors are all in the range of 19,3 to 20,2. This makes for an extremely poor most frequent class baseline, which will never achieve better than 20%. For multiple classes the upper bound will be lower too on the other hand, thus making it a little easier to perform well. As it gets increasingly harder for humans to decide to which class something belongs if the amount of classes increases, the same is true for a trained machine. Two options can be very close together, making it almost impossible to choose the right one. Looking at the posteriors, in this case the trained machine is about 50% sure about one answer most of the time, with the other 50% shared over the rest of the options. This certainty explains the high performance on the unseen test data as well.

1. The features for the experiment have been mainly at the token level, as Naive Bayes normally calculates its probabilities from the bag of words that is created from the training-set. By also adding the TF-IDF feature, the bag of words gets a little more sophisticated, as the distribution between words puts some weight in the scale. Every word in all the texts was used as a feature, without filtering stop words from that list. By also excluding n-grams from multiple tokens, the system could only look at the data at the token level. Due to the source of the data, it could be an improvement to experiment with these other features in order to make for an even higher score.