The BBC News Dataset

Feature Extraction

1. **Naive Bayes**: we have extracted the following features to use them in the Naive Bayes algorithm. These are:
   - **Labels Prior Probability:** we first calculated the prior probability for each label from the data we obtained from the "**bbc.classes**" file.
   - **Terms Probability:** then we calculated the probability of each term in each class and stored the result in a dictionary for easy and efficient retrieval. We used the data from both "**bbc.classes**" and "**bbc.mtx**" files to compute the terms probability.
2. **Logistic Regression**: for the logistic regression we created a numpy array with the columns representing each term and the rows representing each article. Thus for each article we give we have almost 10,000 columns. Each column contains the frequency of the term in that article. If the term is not present in that article, we give it a value of 0.

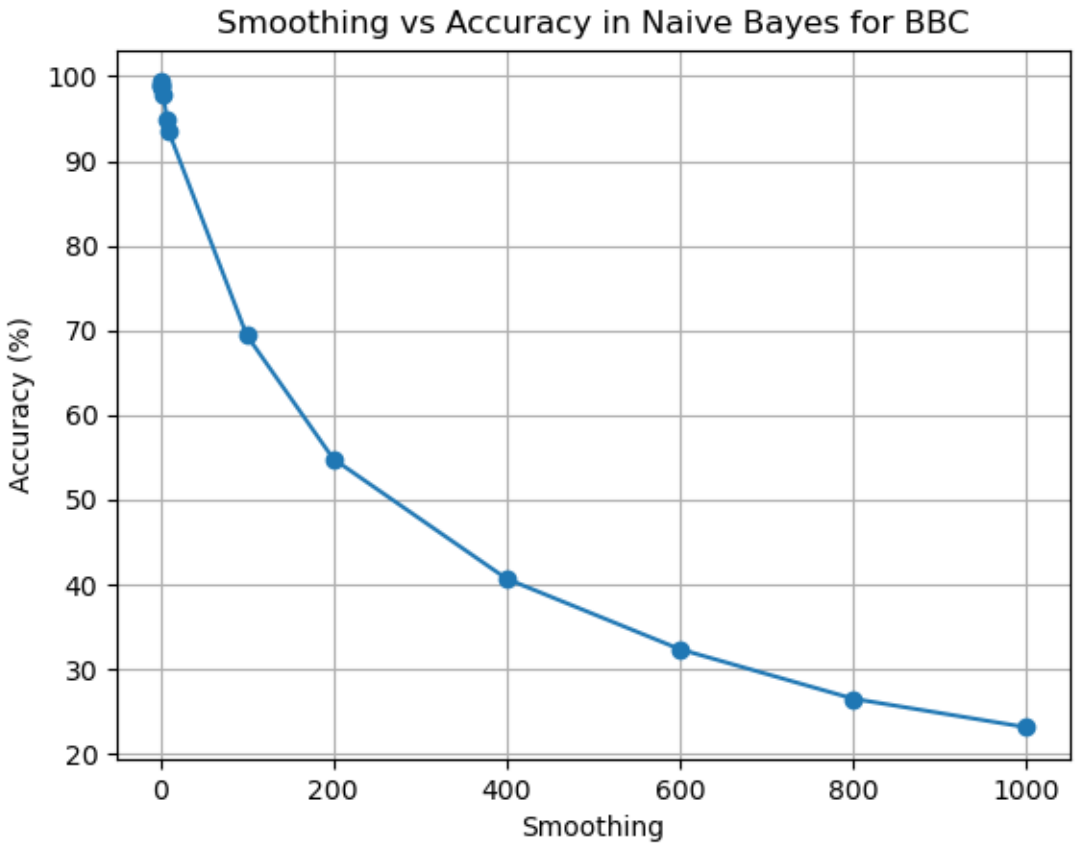
Implementation

1. **Naive Bayes**: the Naive Bayes works based on Bayes's theorem. We used the Decimal library to prevent python from rounding off very small values into zero. It iterates over each label and calculate P(label | article). It does so by converting P(label | article) into $\prod_{term=0}^{last_{term}} P(term \vee label)$ * P(label) for each term. If the term is present in that article we take its probability, otherwise we take laplace_smoothing/total_terms + laplace_smoothing.

2. **Logistic Regression:** we initialize our weights as zero first and then update them on each loop of the training based on the learning rate and gradient descent value. Then after finishing the training phase, it is assumed that we have good enough weights to predict labels of the test set.

Observations

1. **Naive Bayes**: we have been able to achieve an accuracy of **99%** for the BBC dataset. But that value is achieved when a carefully selected smoothing value is used. But generally, we have observed that, **as the smoothing value increases the accuracy of the prediction decreases dramatically.** This is illustrated in the following graph (run the function “try_naive” in the results.py file to see a similar graph).



2. Logistic Regression: we have achieved similar results with Logistic Regression too. We have tried the logistic regression with different learning rates and the same trend as above is observed. We got an accuracy of **97.8%** when running with **learning rate of 0.01** and the accuracy decreases as we move forward. But, it is noticeable that the decrease isn't as sharp the Naive Bayes one. The following graph illustrates this idea (to see the graph run the "try_logistic" inside the results.py page).