THIS PROJECT REQUIRES PYTHON 3.9 BECAUSE OF THE USE OF TYPE HINTING.
Dependencies: copy, numpy, matplotlib, nltk, typing, csv. All can be installed using pip or conda.
I have implemented a naive bayes classifier with 3 preprocessing steps (stemming, punctuation removal and lowercasing) and 1 feature selection technique (Zipf's law stoplist). Laplace smoothing is also implemented. The aforementioned can be clearly toggled by the booleans in the main method. A confusion matrix with macro_F1 and accuracy scores are created in an evaluation stage for development.
Laplace = L, Zipf = Z_k, k= num of words, Punctuation = P, Stemming = S, Lowercase = Lo. [accuracy%, f1macro%]
Baseline majority class classifier

	3c	5c
Baseline	43.3%, 0.2014	28.3%, 0.0882

Preprocessing data for 3 class (3c) and 5 class (5c)

3c	(none)	Lo	S	P	Lo,S	P,Lo	P,S	P,S,Lo
L	64.4%, 0.5167	**65.5%**, **0.5273**	64.6%, 0.5248	64.7%, 0.5128	64.6%, 0.5248	64.8%, 0.5153	64.5%, 0.5224	64.5%, 0.5224
(none)	46.4%, 0.3441	47.0%, 0.3503	48.9%, 0.3823	46.5%, 0.3440	48.9%, 0.3823	47.3%, 0.3527	48.9%, 0.3801	48.9%, 0.3801

5c	(none)	Lo	S	P	Lo,S	P,Lo	P,S	P,S,Lo
L	41.1%, 0.3198	**41.6**%, 0.3233	40.9%, 0.3273	40.8%, 0.3250	41.0%, 0.3279	41.0%, 0.3215	40.4%, 0.3337	40.5%, **0.3343**
(none)	19.6%, 0.1740	19.8%, 0.1808	22.3%, 0.2228	19.6%, 0.1721	22.3%, 0.2228	19.9%, 0.1808	22.6%, 0.2273	22.6%, 0.2273

Above is the data of the preprocessing techniques used and their performance. With no preprocessing applied, the naive bayes classifier performs well above the baseline in terms of f1_macro, suggesting that the classifier is working correctly. In terms of preprocessing, there is marginal performance increase over all configurations, with the most gain being seen in lowercasing for both accuracy and f1_macro. Laplace however, has a profound effect, giving ~+20% on both metrics across the board.

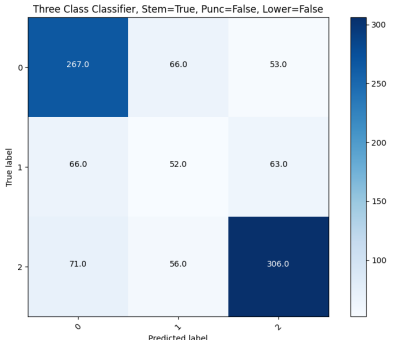
I implemented a stoplist based on the dataset. I took the features in the training data, ordered them in descending order of frequency, and used a parameter k to include the k most frequent words in the dataset. Below is a table with k=0-4 against the preprocessing techniques to show the effect the stoplist has on performance. Despite Lo performing the best, I decided to test against the other preprocessing steps to see if a stoplist had any effect on the results. Laplace is enabled as it is clear it has a significant positive effect on performance.

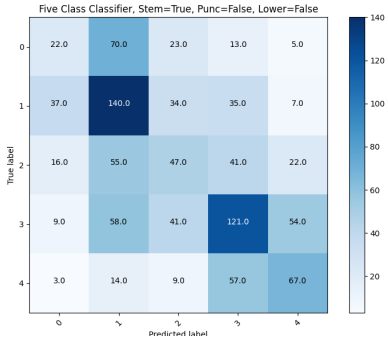
5c	Z_0,L	Z_1,L	Z_2,L	Z_3,L	Z_4,L
Lo	**41.6**%, 0.3233	40.5%, 0.3413	38.9%, 0.3443	37.7%, 0.3525	36.7%, 0.3542
S	40.9%, 0.3273	41.0%, 0.3613	39.7%, **0.3669**	38.1%, 0.3655	36.1%, 0.3536
P	40.8%, 0.3250	39.4%, 0.3231	39.0%, 0.3316	37.4%, 0.3295	36.5%, 0.3353
Lo,S,P	40.5%, 0.3343	40.1%, 0.3499	39.6%, 0.3606	38.2%, 0.3564	37.3%, 0.3569

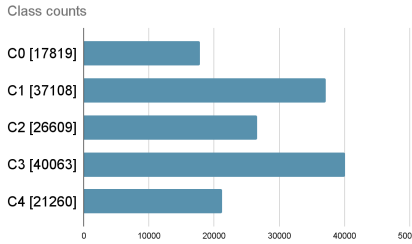
3c	Z_0,L	Z_1,L	Z_2,L	Z_3,L	Z_4,L
Lo	**65.5%**, 0.5273	64.5%,0.5352	62.8%, 0.5444	59.6%,0.5439	57.3%,0.5411
S	64.6%, 0.5248	64.2%, 0.5500	62.5%, **0.5615**	59.5%, 0.5561	55.8%, 0.5356
P	64.7%, 0.5128	63.1%, 0.5143	61.7%, 0.5150	60.1%, 0.5199	57.8%,0. 5186
Lo,S,P	64.5%, 0.5224	63.4%, 0.5344	62.0%, 0.5367	59.9%, 0.5425	56.5%, 0.5276

The stoplist results in a decrease in accuracy across the board. It does however maximise f1_macro at Z_2, with a maximum of 0.5615, an increase in 6.76% from Z_0. This is surprising as stemming did not have much effect on its own. The best config for f1_macro is Lo,Z_2, and for accuracy is Lo,Z_0. This config is consistent between 3 and 5 class configurations.

From the above results, I decided to maximise f1-macro. So the preprocessing steps applied are S, and the feature selection parameter set is Z_2. Laplace is also enabled.

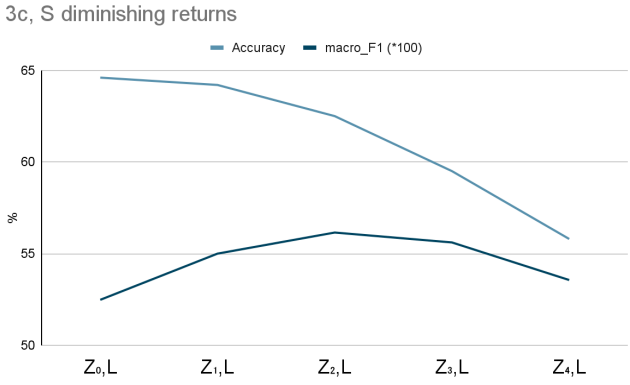






Above are the confusion matrices from the best performing configs. The predictions for these are in the predictions folder in the submission. The classifier is mostly classifying c1 and c3. This maps to c0 and c1 in the 3 class as (c0,c1) and (c3,c4) are grouped together. As shown in the bar graph, there are more occurrences of c1 and c3 in the training data. This has led to overfitting, as the classifier has more data on c1 and c3 sentiments, which is clearly showing in the results. C3 has this more severe as not only is the training data mostly c1 and c3, this classifier is also grouping c0 and c4 together in these classifications. There are only 52 classifications as c1 compared to 267, 306 for c0,c2 respectively. A solution to this would be to have a more evenly distributed training set. This would not fully fix the issue in 3 class however, as c1 would still have half as many samples as c0,c2.

In conclusion, it is clear that the naive bayes classifier is limited. Preprocessing and feature selection have a small effect on the overall performance in terms of macro_f1, and while macro_f1 increases, accuracy seems to take a hit. This could be a result of the size of the dataset. Only small files with a small number of features are being classified, which it could be argued that punctuation removal has less of an effect. The stoplist created based on Zipf's law saw marginal performance increase, similar to preprocessing. The efficacy of this could be hindered by the fact that adjectives would most likely be included in this list if grown too large, potentially explaining why it has diminishing returns from k=2.



Laplace however, massively improved performance, raising macro_f1 by 40.1% above the baseline for 3 class and 59.1% for 5m class. 87.8% and 113.5% above the baseline respectively. This was the only technique used that did not decrease performance of accuracy in hand for an increase in macro_f1.

There is a significant decrease in performance from 3 class to 5 class, despite being from the same dataset with classes being directly mapped to 3. This is because of the curse of dimensionality. An increase in classes results in more misclassifications and decreased accuracy across the board. For example, if in 3 class a document is classified as 0, it could be classified as either 0 or 1 in the 5 class, with only one of them being the correct classification. This can be seen in the confusion matrix as there is a faint "diagonal line" seen around the correct classifications.

Overall, it is clear that the most effective technique for improving the performance of a naive bayes classifier is to use laplace smoothing. Other preprocessing and feature selection techniques often have small increase, albeit with caveats such as decreased accuracy. Reducing dimensionality of the classifier also leads to significantly better results, with the 3 class outperforming the 5 class in accuracy even without laplace smoothing. It is noted however, that without laplace time performance is marginally better, similar results with stemming.