IR Assignment 4: Naïve Bayes

- William Scott (MT18026)

Aim:

- Train and Test Naïve Bayes Classifier with different splits
- TF-IDF Feature selection in Naïve Bayes for 70:30 split

Tools Used:

- nltk
- pandas
- pickle
- matplotlib
- seaborn

Pre-processing Used:

- Convert to lowercase
- Remove stop words
- Remove Punctuations
- Convert Numbers to Words
- Lemmatization

Note:

- Corpus Generation Time: 30.5 s
- Run time mentioned is for both train and test together.
- Number of documents from each class are 1000
- The train-test split is done for class wise.
- Random seed: 41 (to replicate the results)

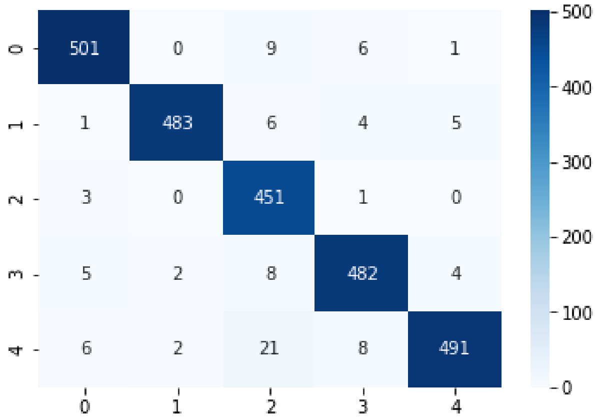
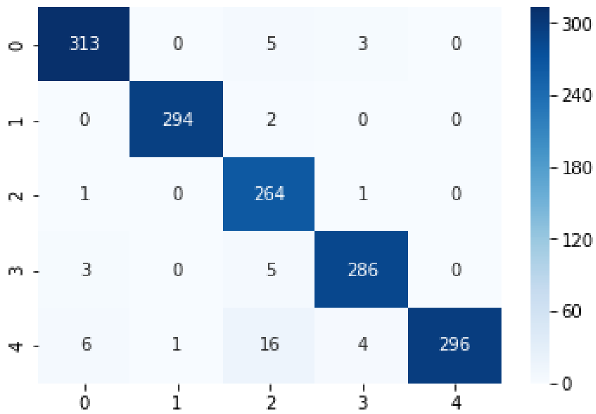
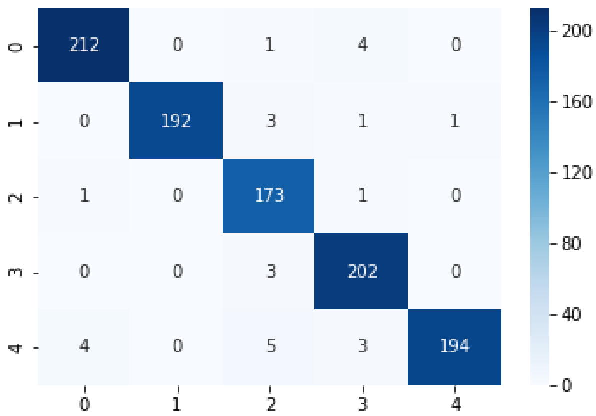
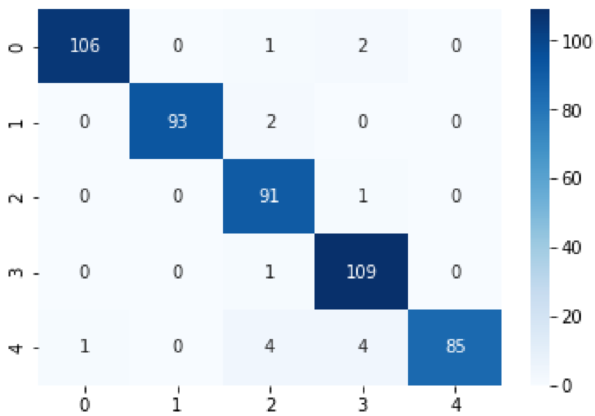
Question 1:

Methodology:

- Generate Train Test split
- In Training
    - Calculate p(x|c) for all the x in corpus. Consider a class to be the label
- In testing
    - Calculate p(x|c) for all tokens and for all classes.
    - Use log and add the values instead of multiplying the probabilities to save them from zeroing themselves.
    - Smooth the Naïve Bayes by adding 1 to the numerator and |v| to the denominator.
    - Take the max class with the maximum likelihood value.
- Labels Order: ['comp.graphics', 'rec.sport.hockey', 'sci.med', 'sci.space', 'talk.politics.misc']

Inferences:

- Run time increase with the increase in the train split.
- Corpus and the Unique words increase with increase in the train split.
- Accuracy will also tend to increase, but the increase in accuracy is not too certain. As we will be having all the noise variables also included in our corpus.

Split	Accuracy	Corpus	Unique Words	Run Time	Confusion Matrix – Heatmap
50:50	96.32%	554767	85789	8.86 sec	
70:30	96.86%	729357	106510	9.37 sec	
80:20	97.3%	826581	116427	11 sec	
90:10	96.8%	915524	125365	13.3 sec	

Question 2:

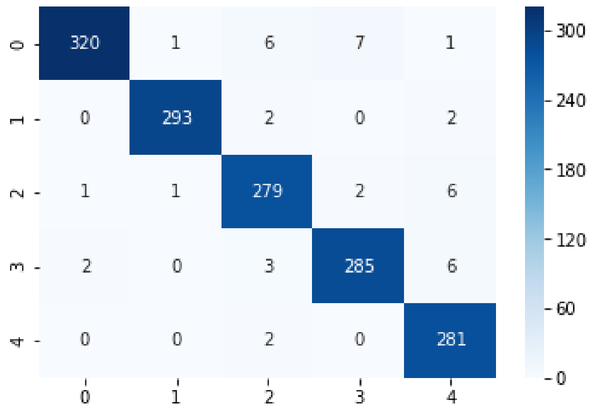
Methodology:

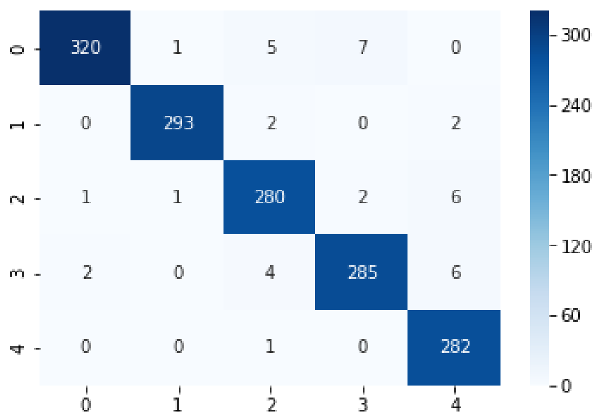
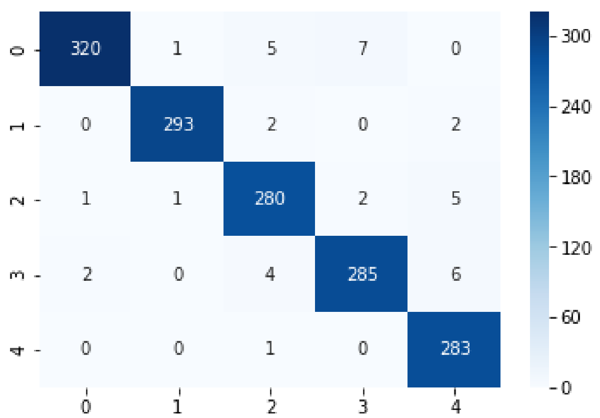
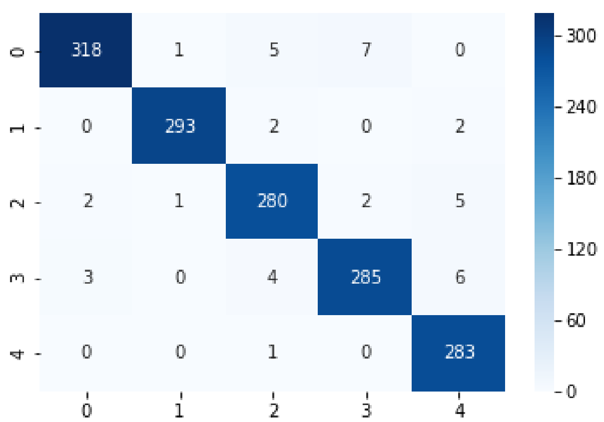
- Split dataset to 70:30 Train and Test respectively.
- Calculate TF-IDF
  - Take TF of by counting the word frequency in all the documents.
  - IDF also will be for all the documents.
  - Normalise the TF with the unique words (optional) and idf with +1 normalisation on numerator and denominator.
- Sort the tokens with the TF-IDF values.
- Split for the percentages on the dataset (e.g.: 50%).
- In Training
  - Calculate p(x|c) for all the refined corpus. Consider a class to be the label.
- In testing
  - Calculate p(x|c) for all refined tokens and for all classes.
  - Use log and add the values instead of multiplying the probabilities to save them from zeroing themselves.
  - Smooth the Naïve Bayes by adding 1 to the numerator and |v| to the denominator.
  - Take the max class with the maximum likelihood value.

Labels Order: ['comp.graphics', 'rec.sport.hockey', 'sci.med', 'sci.space', 'talk.politics.misc']

Inferences:

- With increase in the % of top TF-IDF values, the unique words increases
- Increase in unique words increase the processing time
- Performing the feature selection on the TF-IDF values is a good technique, as we will be taking the important features only in the whole corpus.
- For this reason, we got accuracy of 97.2 even with 50% corpus
- With increase in the corpus %, we might introduce little noise due to which maybe the 90% model gave little less accuracy compared to the others

Top	Accuracy	Corpus	Unique Words	Run Time	Confusion Matrix – Heatmap
50%	97.2%	729357	42604	11.1 sec	

60%	97.33%	729357	53255	11.6 sec	
70%	97.4%	729357	63906	12.7 sec	
80%	97.26	729357	74557	13.2 sec	

Question 1 vs Question 2:

- Less noise in Question 2, due to feature selection which gives only the important variables.
- Feature Selection is computationally efficient technique as we will be working on only the part of the corpus.
- Pre-processing doesn't play a huge role in both 1 and 2.
- Naïve Bayes does not need a lot of data for classification, it just needs the right amount of correct words which can link them to a particular class, and feature selection is a proof.
- Naïve Bayes works in either scenarios due to its independence assumption.

Additional Inferences

- Below is a table which shows the difference in accuracy for both the questions 1 and 2 with different pre-processing techniques.
- Stemming + Lemmatization doesn't seem to be necessary together, either would be enough
- As this is a Naïve Bayes algorithm, the pre-processing is not so much useful for accuracy. As we can observe in A, where we removed just the stop words and the accuracy almost remained the same.

Pre-processing Accuracy Changes:

Split	Top	A: stop words	B: A + num2word	C: A + lemmatization	D: A + Stemming	C + D	B + C + D
50-50		96.92	96.64	96.84	96.92	96.76	96
70-30		97.2	96.86	97.2	96.86	96.66	96
80-20		97.6	96.7	97.5	97.4	97.3	96.2
90-10		98	97	97.8	97.6	97.2	95.8
70-30	40%	97.33	97.2	97.33	97.13	97.06	96.86
70-30	50%	97.33	97.26	97.4	97.26	97.2	97.13
70-30	60%	97.33	97.2	97.4	97.26	97.2	97.13
70-30	70%	97.33	97.33	97.4	97.26	97.2	97.13