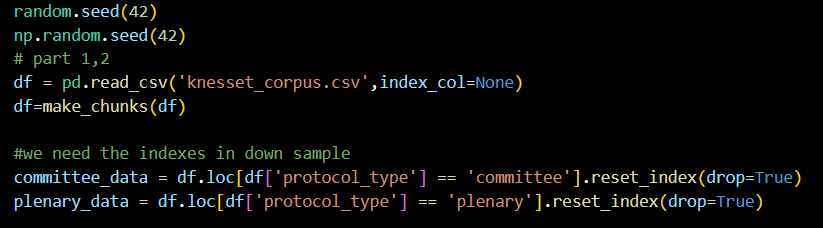
Natural Language processing – HW 3

Mias Ghantous – 213461692

Faisal Omari

Section 1,2.

As we see, we split the data according to his type after making the chunks



Make\_chunks:

1. we divide the data into groups by using the function which, takes the name of the columns as a parameter and then make a group for every possible compensation of values from those columns.

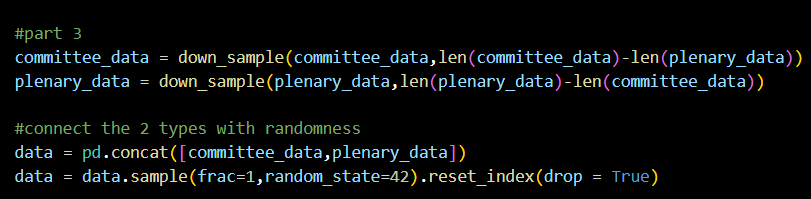
2. second we use which takes a function as a parameter and do this function for each group, we sent which extract the relevant data from the group and put them into a data frame.

3. apply concatenate the result by herself.

4. in the code we group by 'protocol\_type' and 'protocol\_name' because we want the keenest number in the classification

Section 3:

As we see, we call down sample for both of the data and then connect them and then randomize it:



For chunk size 5 we get that:

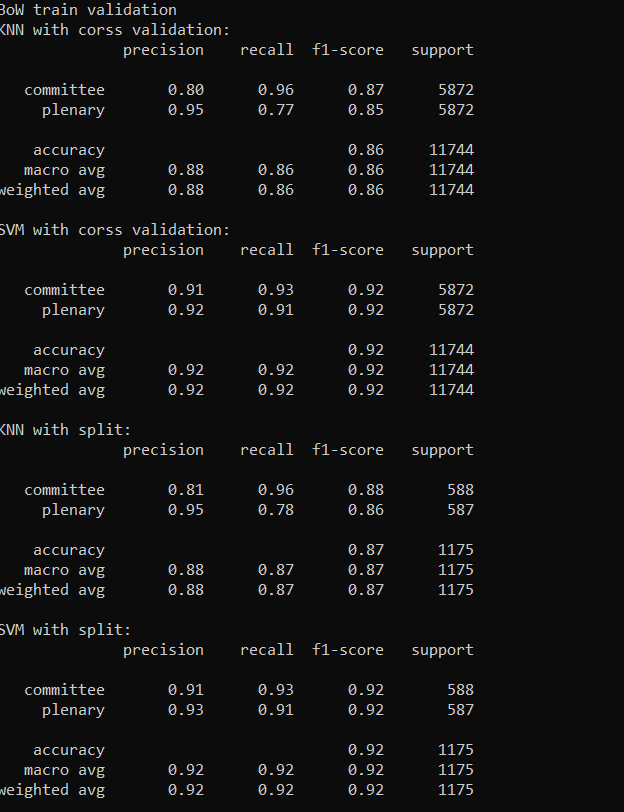
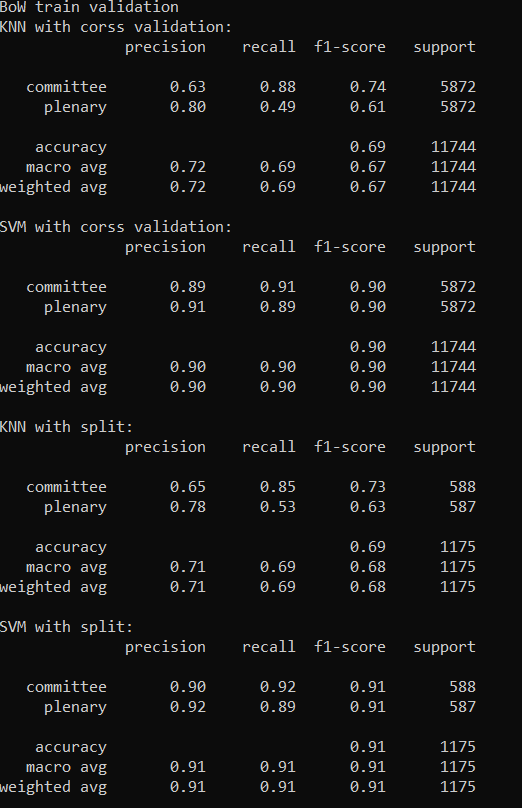
The size of the committee data before the down sample was 5872 and also after.

The size of the plenary data 14095 and after the down sample we have 5872

Section 4.

1. We used TFIDF vectorizer because he can give more accurate predictions:

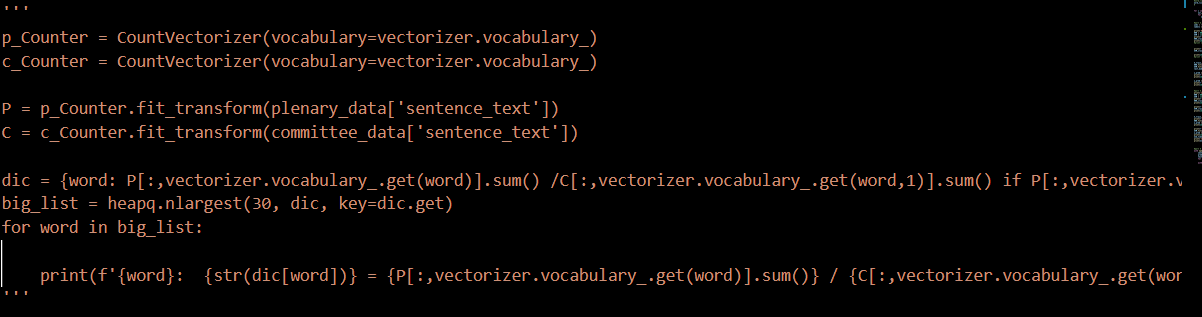
(first is IFIDF and the is the counter)

1. for our vector we used 8 words:

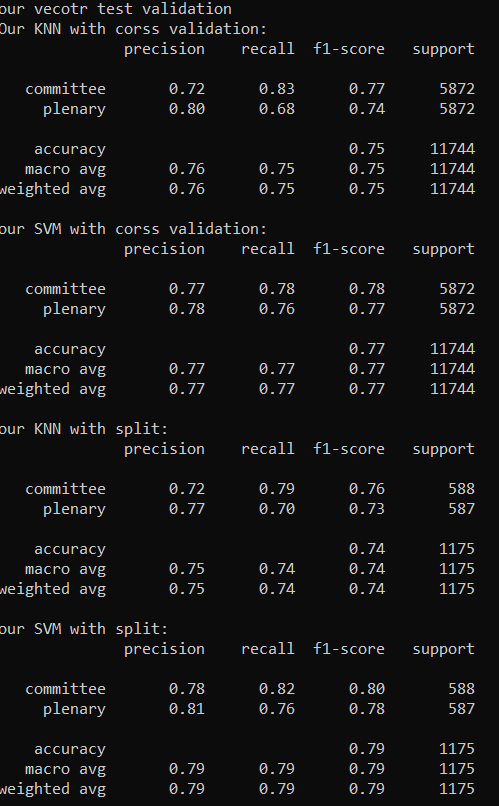
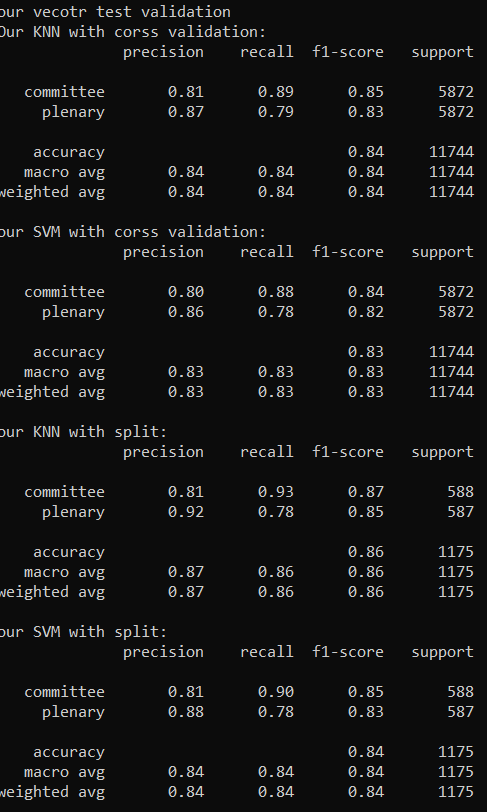


Because those words have a lot of occurrences in plenary documents compared to committee, how did I know? I wrote code that counts all the occurrences of all the word and took the words that has 3 times appearances in plenary and appear more than a 1000 times in all the plenary data: here is the code:

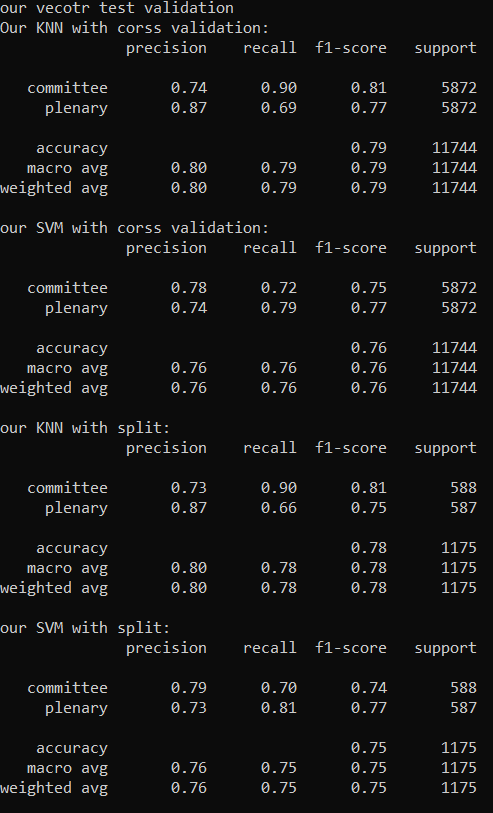


Also we used the because we realized that the numbers are not evenly distributed between the 2 type, helped the accuracy a lot:

(first is before and the second is after we use it):

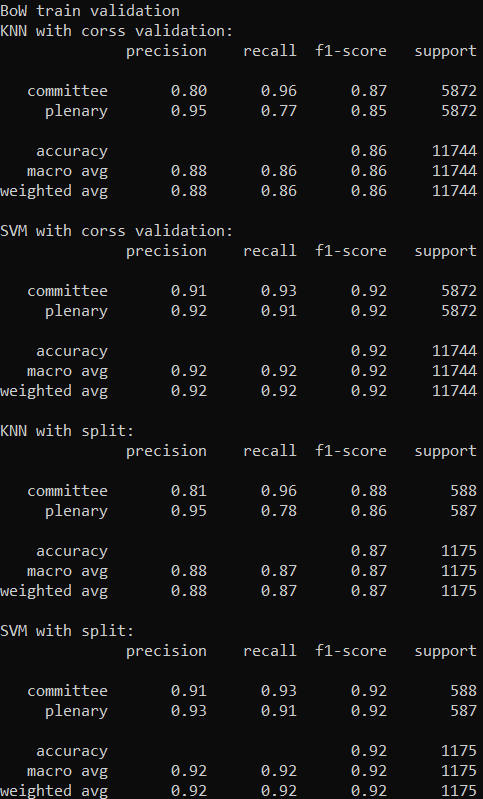
Also we tried to include the average length to the feature vector but it made the accuracy worst:



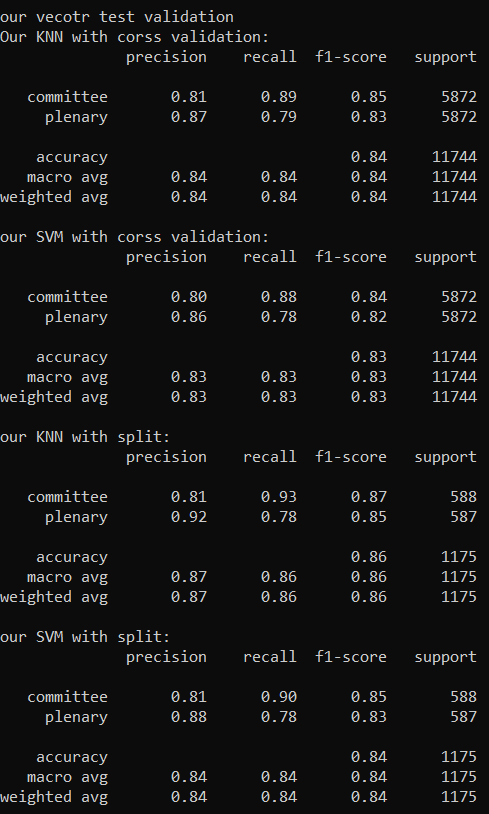
Section 5:

1. in knn also we used linear kernel for the SVM to classify BoW and rbf kernel for our feature vector.
2. And here are our results:

For BoW feature vector we get:



And for our feature vector we get:



Section 6:

We predicted using the SVM that belongs to the BoW because he has the best accuracy.

Section 7 (questions):

If the recall is smaller the false-negative is bigger, and if the precision is smaller the false-positive is bigger, we see that SVM on BoW has almost the same recall and precision, which means that he can identify them equally, but KNN of BoW (also for both classifiers of our feature vector) the committee precision is smaller which means he can less identify committee, and that's why also the recall is smaller for plenary (because if he does not identify it as committee then he gives plenary which will increase the false-negative).

We see that almost we have the same accuracy and confusion (recall and precision), and that is because we have the same number of test and train data with same distribution over the data.

SVM has better performance for BoW and worst performance for our feature vector, also he requires a lot more time to train comparing to KNN, but for prediction he requires less time.

KNN has worst performance for BoW and better performance for our feature vector comparing to SVM, he takes long time for predication, but less time training comparing to SVM.

If we look at the features above for both of the classifiers we see that features of SVM is better because we are going to train one time and then predict a lot more and he gave better accuracy for BoW than KNN gave for our feature vector.



Big chunk size: First the feature matrix would be smaller which means the run time would be less, and the accuracy get bigger (near perfect and even perfect for 10% test if big enough) and that is because of overfitting, why there is overfitting? Because we would look at a lot less vectors which means that we would have less variation and more close vectors which means we have less diverse data which automatically lead to overfitting.

Small chunk size: if we have small chunk size then we would have bigger feature matrix which is worst for run time, also each and every chunk would have features that doesn’t identify the types good enough which make the classification task harder.

1. Chunk size = 10 is the best size for our data

We try 5 and we have no over fitting, and then we tried 20 but have accuracy of 0.99 which is defiantly overfitting, and we try in the middle (13) and we got 0.97 (for split SVM) which feels like there is a bit of over fitting and, then we tried 10 which and got 0.96 (in split SVM) which sounds like the best accuracy without overfitting for this data.

Note: after we finished we returned the size to 5.