**1. Compare the performance of each classifier using the average macro precision and recall over the 5 folders, and write your analysis in a document.**

---------------- 1a ----------------

Test data set average precision:

[ 0.63034759 0.69768935 0.64402619 0.72222222 0.82921512]

Test data set average recall:

[ 0.60551948 0.671875 0.59821429 0.6875 0.76215278]

Test data set average fscore:

[ 0.5950938 0.66773535 0.57394257 0.6827262 0.76217054]

---------------- 1b ----------------

Test data set average precision:

[ 0.68188406 0.75 0.62066752 0.72222222 0.7434593 ]

Test data set average recall:

[ 0.63582251 0.72321429 0.60491071 0.66741071 0.69386574]

Test data set average fscore:

[ 0.62290256 0.72222222 0.5977849 0.65569314 0.68899225]

In this case, there is no much difference with/without lemmatization.

**2. Compare the performance with the classifier in Step 2(a) and write your conclusion in the document**

('tfidf\_\_stop\_words', ': ', None)

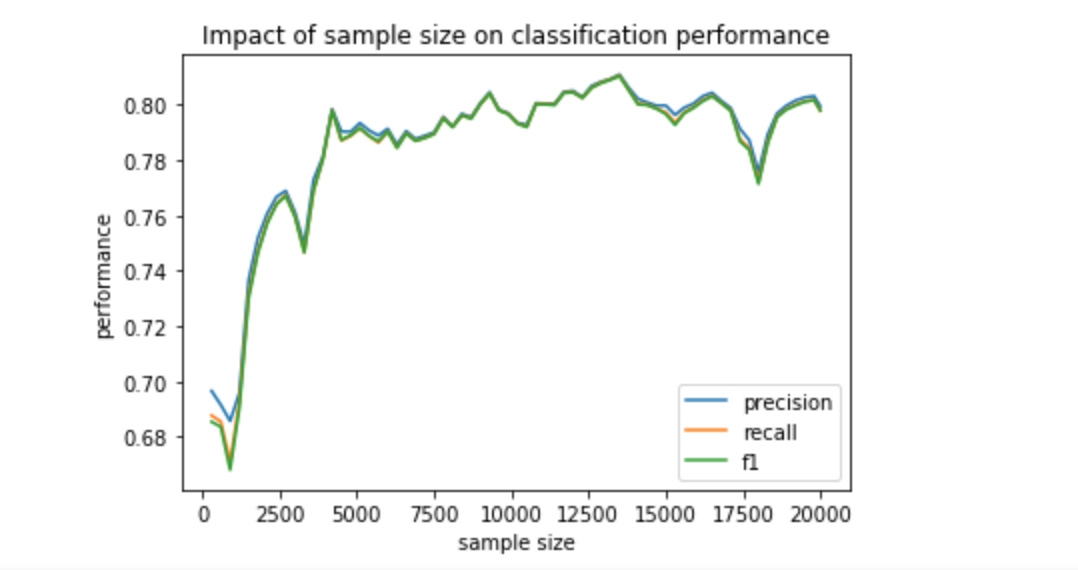
('tfidf\_\_min\_df', ': ', 3)

('clf\_\_alpha', ': ', 0.5)

('best f1 score:', 0.74038261618491519)

because best f score in 2 is larger than the f score in 1a, the performance in 2 is better than the one in 1a.

**3. Write your analysis on how sample size affects classification performance in the document**



As the diagram shows, at the beginning, the performance increase along with the sample size increase. And then, the trend is getting stable when the sample size increase to 5000 and performance arrive 0.80. The trend of the performance of precision, recall and fl respectively are nearly the same. I think it is because if we have more data to train, our model has more data to learn, we know more exactly the weight of the features. So we can classify data more precisely. But when we are getting more and more close to the real weight of features, the performance will be more and more stable. And there are two pits at near 2500 and 17500, which I think it is just because the data floating.