Table

Description automatically generated

According to the product above, the Naïve Bayes model would assign this sentence to the negative class.

2a: Implemented in NB.py

2b: Parameters hard coded into testnb.py. The parameters are stored as dictionaries.

2c: ﻿Action

I predict that the test set is a part of the action class. The P(action) is 0.00028577960676726106 and the P(comedy) is 0.00018310546875.

2d: Files preprocessed and pickled into vectors and dictionaries. You can easily generate this set by running preprocess.py and having the following directory structure in your current working directory:

﻿#positive class of training data

path\_to\_pos\_train = r'movie-review-HW2/aclImdb/train/pos'

#negative class of training data

path\_to\_neg\_train = r'movie-review-HW2/aclImdb/train/neg'

#positive class of testing data

path\_to\_pos\_test = r'movie-review-HW2/aclImdb/test/pos'

#negative class of testing data

path\_to\_neg\_test = r'movie-review-HW2/aclImdb/test/neg'

Result analysis:

The model successfully predicted about 5500 of the 25000 test cases accurately. The reason for such a high error is because of precision errors. There were approximately 12500 cases where python couldn’t compare likelihoods of classes. I tried this in both regular non-log math and with log math and either way, the precision eventually didn’t work with whatever values I choose for the log base or however big/small the original probabilities were.

Instead of discluding those results, I randomly chose neg or pos for those classes and the result was that ﻿12416 of the results were predicted correctly or about 50%.

I noticed that when the reviews are smaller, it is more likely that the prediction is correct and as the reviews get larger, the accuracy decreases. I predict that this is because of the naïve approach of multiplying probabilities.

Happy holidays!