Text classification report:

1. I use two libraries: math for the avoiding underflow by using log and increase speed; string for delete all punctuations from the sentence.
2. First I define a function to help me tokenize the sentence, I lower all the letters and remove all the punctuations and use .split() to remove the id from the sentence.
3. Then I want to build a Naïve Bayes Classifier.
   From the book we know the formula:

$$P(c) = \frac{N_c}{N_{doc}}$$

$$P(w_i|c) = \frac{\text{count}(w_i,\ c) + 1}{(\sum_{w \in V} count(w, c)) + |V|}$$

$$c_{NB} = argmaxlogP(c) + \sum_{i \in positions} logP(w_i|c)$$

So, for the positive class and negative class:

We first calculate total word in each training file:

count_total_pos and count_total_neg

Then we create three dictionaries:

dictforvac: key is all vocabulary, value is 1.

dict_pos: key is positive word, value is times it occurs in training set

dict_neg: key is negtive word, value is times it occurs in training set

We then use formula:

$$P(c) = \frac{N_c}{N_{doc}}$$

To calculate:

$$P(pos) = \log(\frac{count\_total\_pos}{count\_total\_pos + count\_total\_neg})$$

$$P(neg) = \log(\frac{count\_total\_neg}{count\_total\_pos + count\_total\_neg})$$

Then we use formula to do the add-1 smoothing:

$$P(w_i|c) = \frac{\mathrm{count}(w_i,\ c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|}$$

The bottom are:

pos_bottom = log(sum(dict_pos.values) + len(dicforvac))
neg_bottom = log(sum(dict_neg.values) + len(dicforvac))

The top are:

dict_pos[word] + 1
dict_neg[word] + 1

And :

$$c_{NB} = argmaxlogP(c) + \sum_{i \in positions} logP(w_i|c)$$

Which we have P(c), and use a for loop to iterate though all test sentences, and for each sentence we keep adding new log values using a for loop of all words in sentence to calculate a positive probability and a negative probability, and compare them.

Then if positive probability is bigger then the sentence is positive
Or it is negative.