1 Problem: Naive Bayes

a) First thing to note is that since there are an even number of positive and negative reviews in the training data, we can say that the values of $P(+)$ and $P(-)$ are both $\frac{1}{2}$

Now based on the sentence S provided in the assignment PDF, we should sum up all the numbers in the training data based on their class (positive or negative). As well as the words in sentence S

	great	amazing	epic	boring	terrible	disaapointing
P(+)	6	8	5	2	1	1
P(-)	2	1	4	5	5	6
S	1	1	0	0	1	1

We can see that the sum of tokens in both the positive and negative class are 23. So, based on the word count of S we can calculate the probabilities as such.

$$\prod P(s|+) = \log\left(\frac{6}{23}\cdot\frac{8}{23}\cdot\frac{1}{23}\cdot\frac{1}{23}\right) = -3.77$$

$$\prod P(s|-) = \log\left(\frac{2}{23}\cdot\frac{1}{23}\cdot\frac{5}{23}\cdot\frac{6}{23}\right) = -3.67$$

Now with the following values we can use the naive bayes theorem as such.

$$P(+|s) = \frac{P(+)\prod P(s|+)}{P(+)\prod P(s|+) + P(-)\prod P(s|-)} = \frac{\frac{1}{2}\cdot -3.77}{(\frac{1}{2}\cdot -3.77) + (\frac{1}{2}\cdot -3.67)} = 0.506$$

$$P(-|s) = \frac{P(+)\prod P(s|+)}{P(+)\prod P(s|+) + P(-)\prod P(s|-)} = \frac{\frac{1}{2}\cdot -3.67}{(\frac{1}{2}\cdot -3.77) + (\frac{1}{2}\cdot -3.67)} = 0.493$$

Since $P(+|S) > P(-|s)$ Naive Bayes would classify this sentence as positive.

b) Now let us add *add1-smoothing*. We will add 1 to each word count. So now our table looks something like this. Now with our total word count for each class increased to 29, we

	great	amazing	epic	boring	terrible	disaapointing
P(+)	7	9	6	3	2	2
P(-)	3	2	5	6	6	7
S	1	1	0	0	1	1

can re-calculate the probabilities.

$$P(+|s) = \log\left(\frac{7}{29}\cdot\frac{9}{29}\cdot\frac{2}{29}\cdot\frac{2}{29}\right) = -3.45$$

$$P(-|s) = \log\left(\frac{3}{29}\cdot\frac{2}{29}\cdot\frac{6}{29}\cdot\frac{7}{29}\right) = -3.45$$

It seems that after smoothing the probabilities seem to be the same. Seeing as the probabilities are the same, there is no need to normalize the values. I am not sure how this model would label this sentence, and I believe it would depend on how you program your model.

c) I think the simplest feature that could be added to sentences like S, would be using a bi-gram model. This would the model we use to gain some small level of context. For instance, at the end of the sentence where it says "not disappointing" would have had a much higher probability in the positive class.

2 Hate Speech Detection

2.1 Naive Bayes

After implementing my model, these are the following accuracies that I got after testing with the three different datasets provided After looking at the dev samples and how my model

	train.csv	test.csv	dev.csv
Accuracy	93.28%	76%	72%

predicted some of the sentences, here are a few that it had classified wrongly.

"The soldiers of today may face this same treatment before they die if things keep going the way it has been . " Was predicted as hate speech when it was not. I believe it was due to words such as *die* or *face.*

Another examples would be *"Remember to not make it just about being anti-everyone else , but to also make it about them being proud of their own family heritage."*. I think the obvious word here would be *"anti-everyone"* being responsible for the sentence being labelled wrongly. I it obvious that Naive Bayes has a severe limitation due to its lack of context when calculating its probabilities.

To look at words with the highest and lowest probability of being considered as "hate-speech", we look at the ratio of $\frac{P(1|w)}{P(0|w)}$, here is the list of the top 10. (Side note: all of the following ratios found are after add1-smoothing was applied) On the left side are the highest 10 and on the right are the lowest.

	Word	Ratio
1	jews	13.24
2	mud	13.24
3	asian	10.75
4	leave	9.93
5	non	9.1
6	liberal	9.1
7	non-white	8.27
8	dumb	8.27
9	apes	8.27
10	running	8.27

	Word	Ratio
1	check	0.08
2	thanks	0.08
3	_	0.09
4	sf	0.1
5	spirit	0.12
6	father	0.12
7	sports	0.12
8	15	0.12
9	[	0.12
10	]	0.12

2.2 Logistic Regression

After implementing logistic regression, I had learning rate as 0.1 and ran stochastic descent for 40 epochs to get my weights. These are the following accuracies that I got. The results are

	train.csv	test.csv	dev.csv
Accuracy	99.8%	76.8%	76%

not too different from the Naive Bayes classifier, I was expecting a much larger improvement.

I then added L2 regularization with different α values, the table below shows the different accuracies I attained with the three provided data sets. As the weights got larger, the

Weights	train.csv	test.csv	dev.csv
0.0001	99.8%	77.2%	75.6%
0.001	95.3%	73.6%	75.2%
0.01	75.1%	65.2%	66.4%
0.1	64.7%	63.2%	66.8%
1	61.7%	58.4%	62%
10	45.5%	48.8%	46.4%

accuracies dropped as well. I assume this is because as the weights increased, the values that the weights were "penalized" became larger than needed. I am not sure that I did this section correctly as the formula stated that we had to square each of the weights, but every time I did so I would get a math overflow. So I skipped that part and the numbers above are what I got.