Jeanette Pranin (jrp338), Nishant Subramani (nso155), Jaiveer Kothari (jvk383)

	Bayes.py	**BayesBest.py**
Precision	0.8042	0.9845
Recall	0.9019	0.9801
F1-measure	0.8503	0.9823

Our Naive Bayes unigram model (in bayes.py) makes the assumption that given the class of document (5 star or 1 star), that word i's frequency is independent of word i+1's or i-1's frequency. This assumption is good but limited such that words occur in groups and in some sequential ordering. Furthermore, certain words may not be independent of each other given the class. We utilized add-one smoothing for this model.

In bayesbest.py, we chose to utilize orderings and groupings of words by utilizing bigrams. We stripped each string into two word intervals and used these bigram tokens as the features with which to classify from. We also considered a word's presence over frequency. Furthermore, we smoothed differently. Since bigrams are infrequent and many will never be seen, but those that are seen should be weighted much higher, we chose to do 0.005 smoothing. We gave a pseudo presence count of 0.005 rather than 1 for unseen bigram tokens. Using this, the performance wildly improved with an f1-score of 0.98.

Options for improvement include utilizing trigrams or quad-grams. Google has published research that show diminishing returns by increasing from bi to tri to quad to pent. Using various features beyond just presence such as types of words (nouns, adjectives, verbs), length of document, and using map estimates are also good to take into consideration.