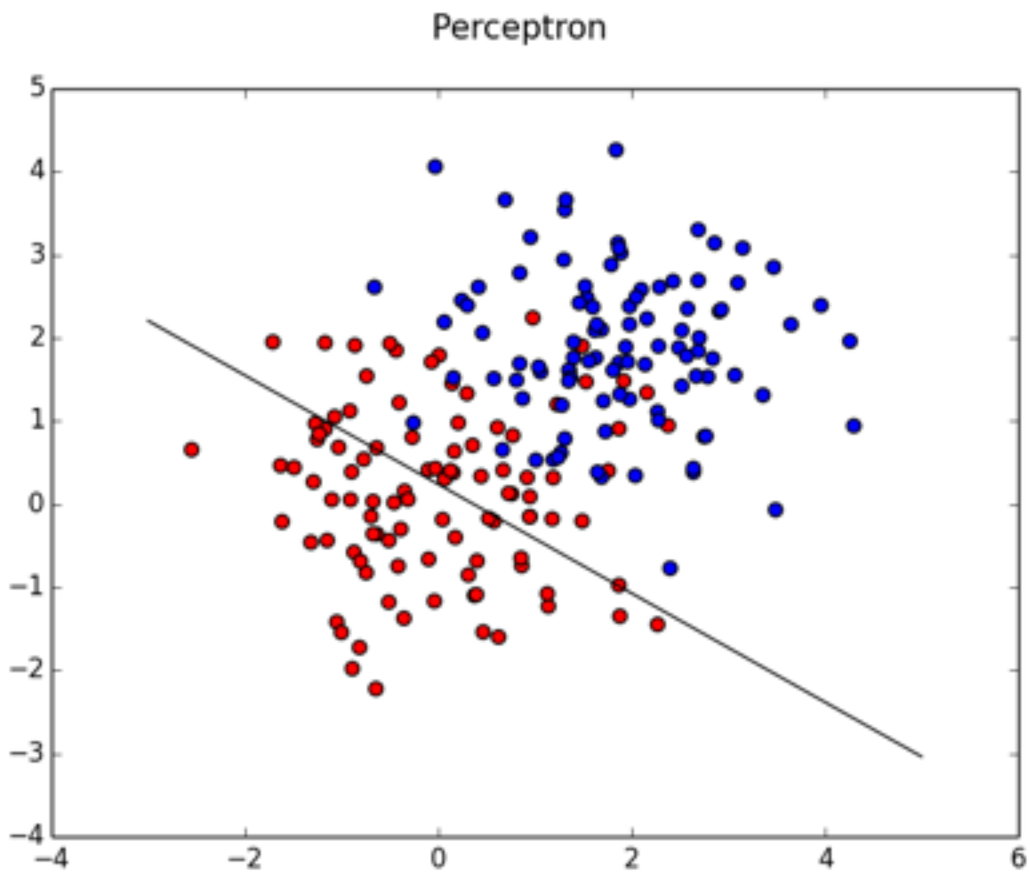
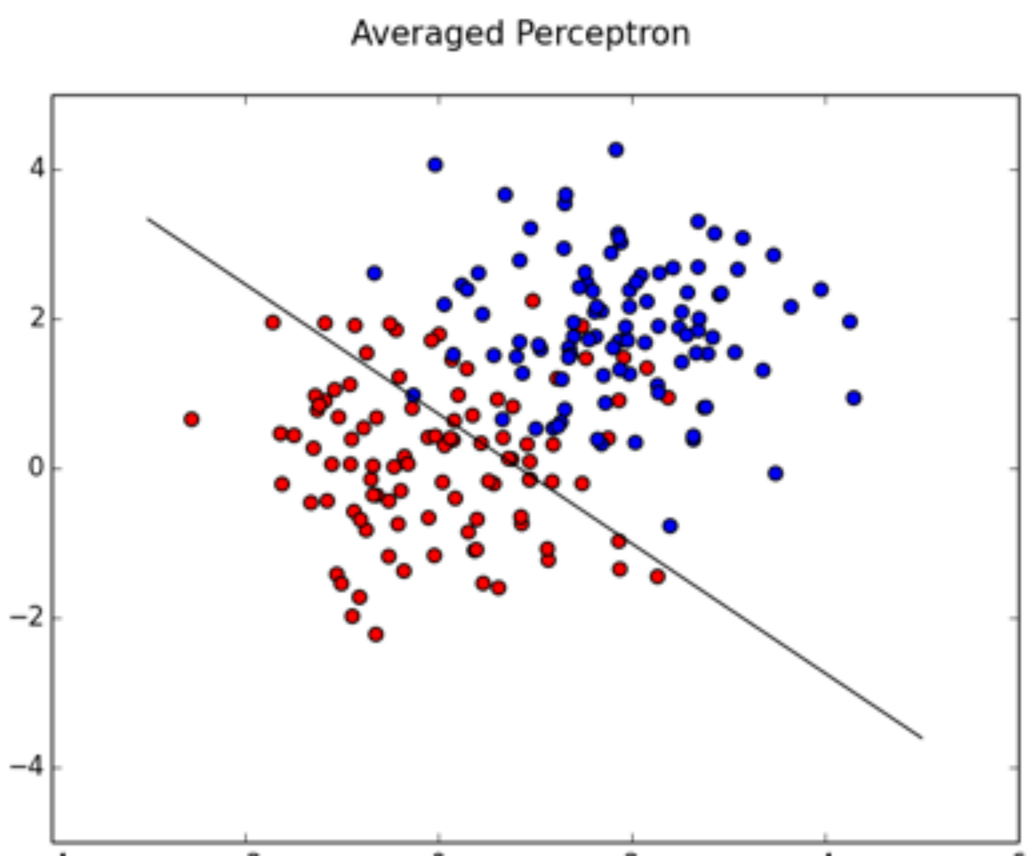
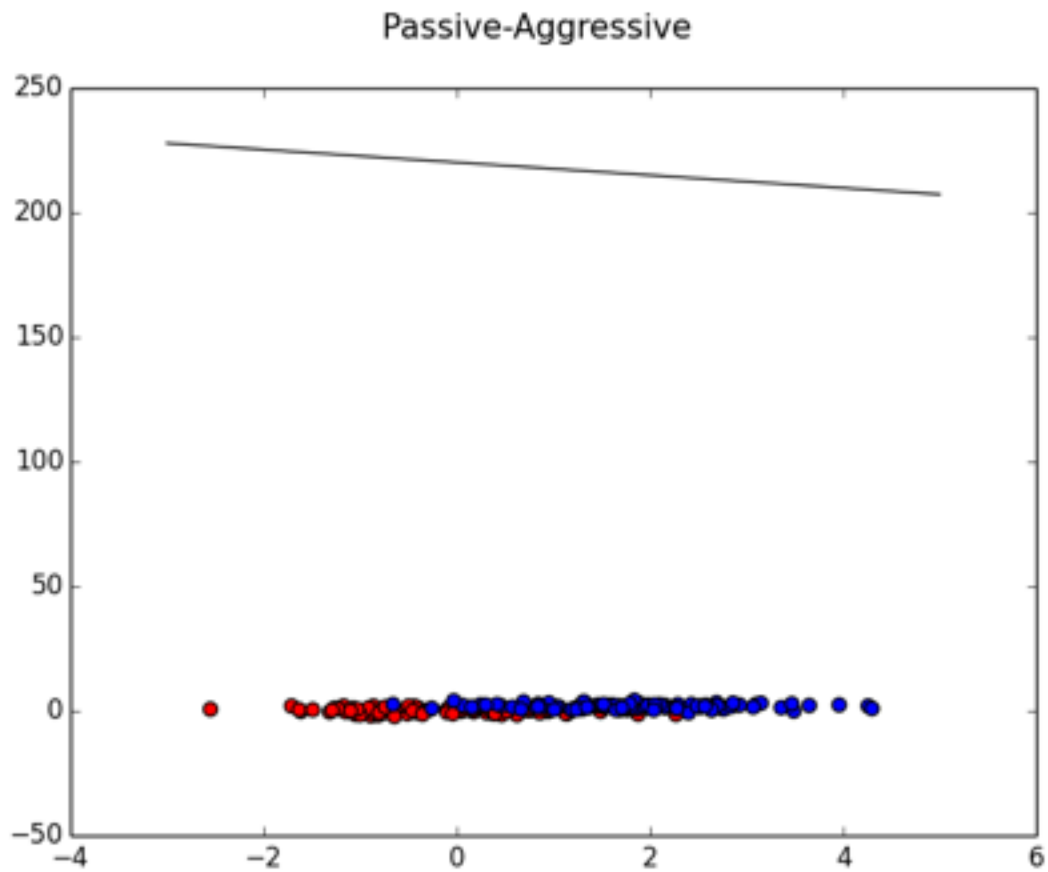
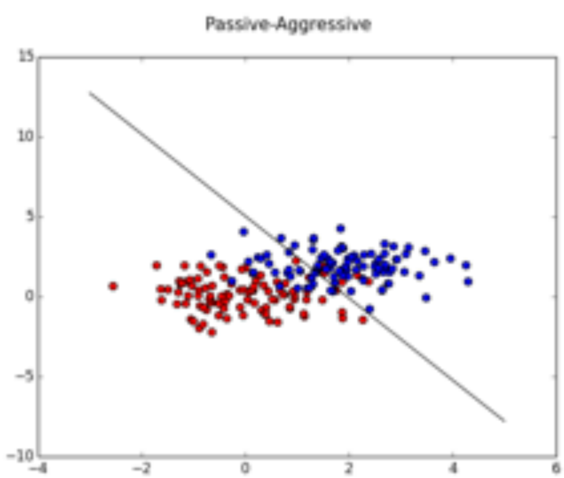
Hikari Senju

1. Implementing classifiers





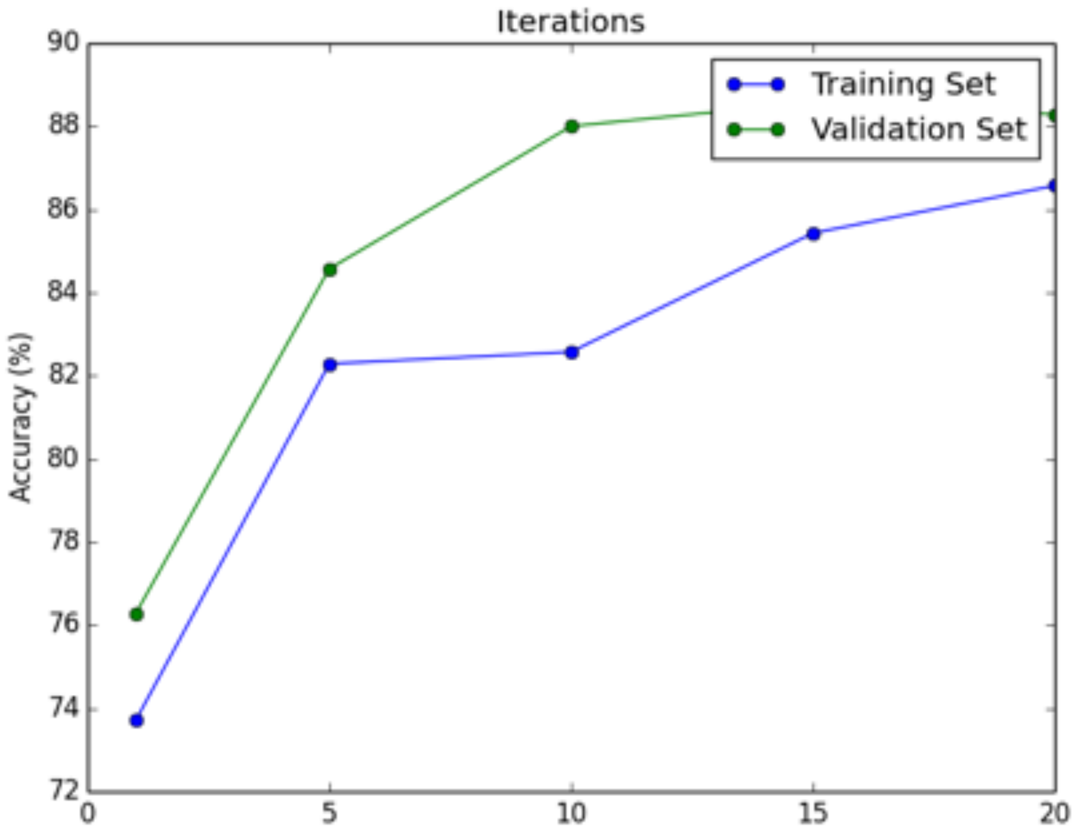


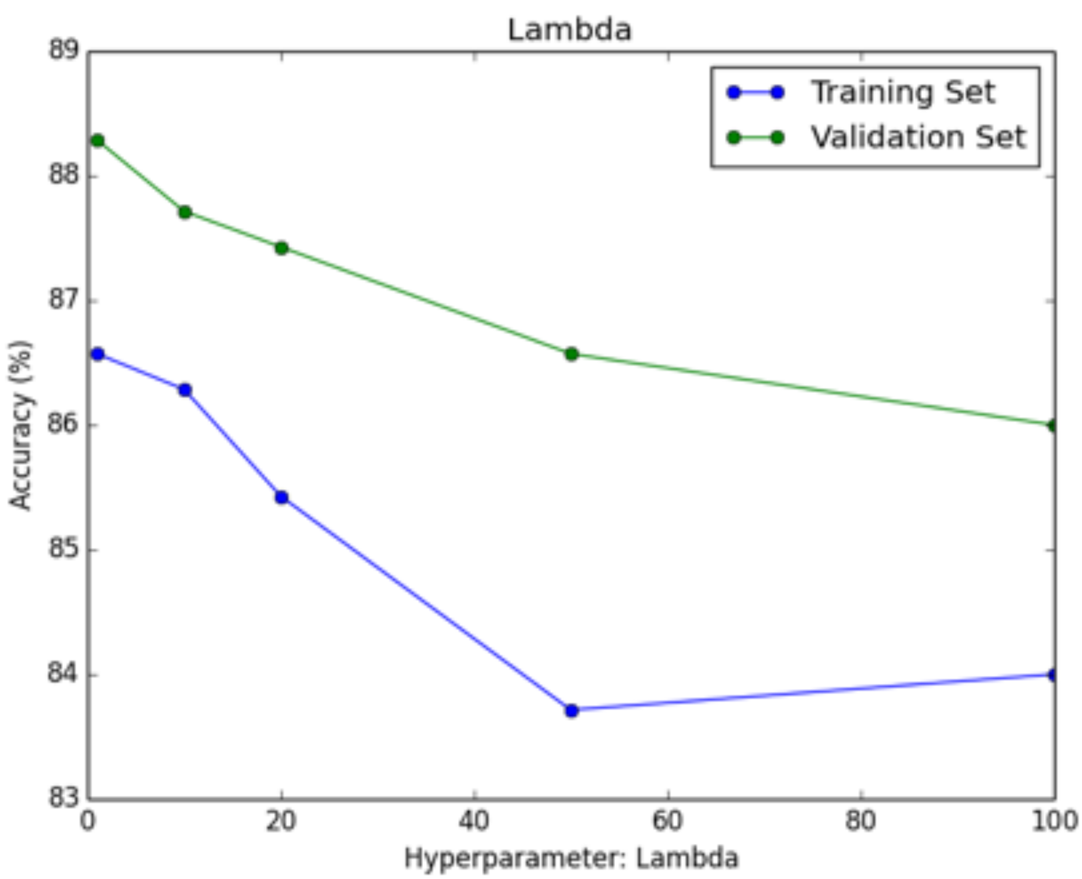


The best looking plot was definitely Averaged perceptron (because it performed better than regular perceptron as discussed in lecture, and because the passive aggressive performed poorly). The worst was passive aggressive. The passive aggressive algorithm, which is the same passive aggressive algorithm I used in part 2 (and actually got the highest result in part 2), for part 1 due to the eta added to theta_0 with a large lambda resulted in a highly skewed theta_0. Without the eta update to theta_0 the graph looked like the graph to the left, however this performed very poorly in part 2 since theta_0 wasn't being updated.

2.
perceptron: 72.0% correct (252 out of 350).

average perceptron: 82.0% correct (287 out of 350).
passive aggressive: 82.2857142857% correct (288 out of 350).



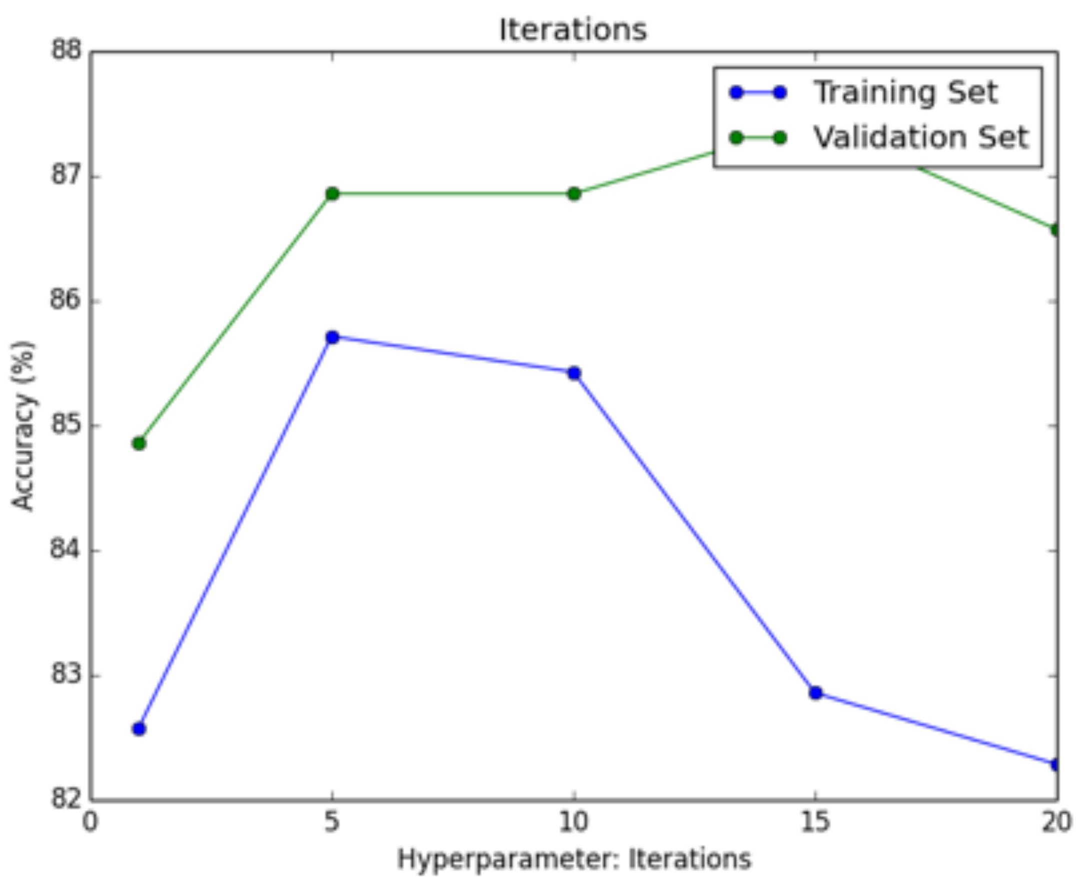


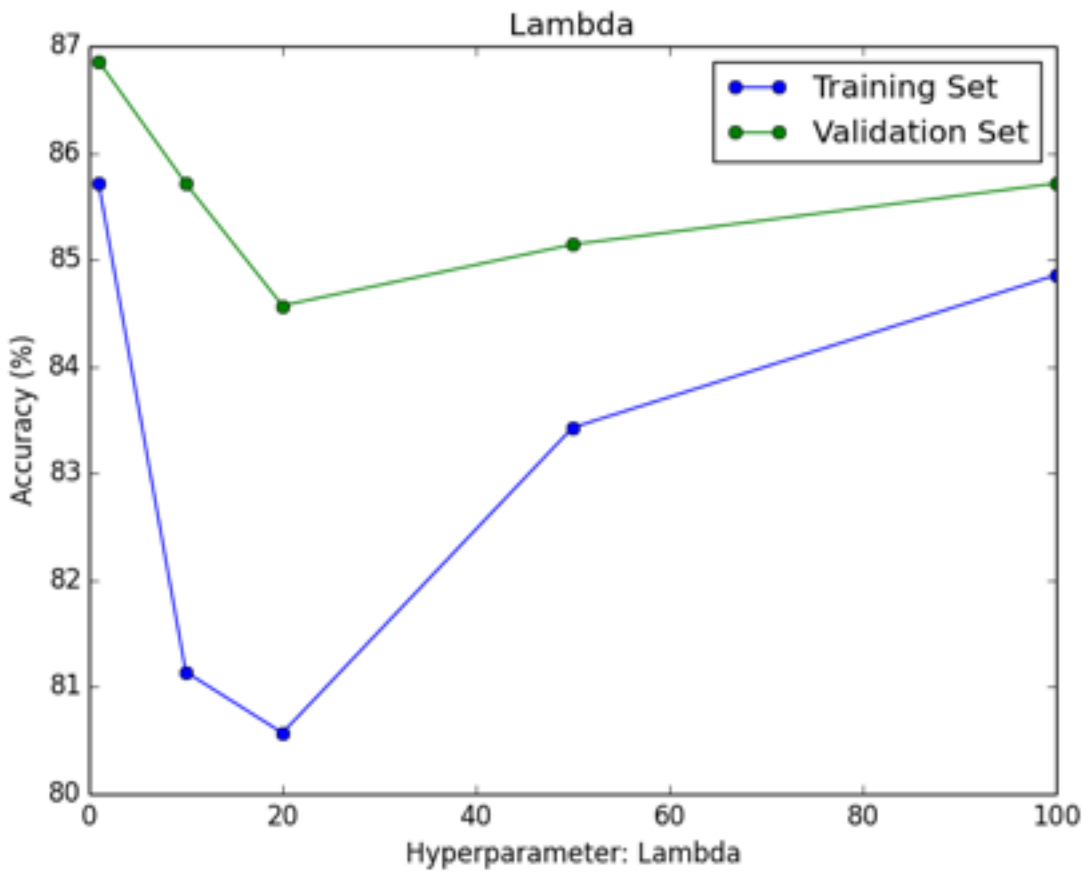
2a)
Training and validation sets behave relatively similarly, though the the validation set consistently performs at least +1 percent for lambda in comparison to the training set, and +2 percent for iterations. In general accuracy increased with increased iterations though with increased iterations, the benefit of each additional iteration decreased and there was an increased risk of overfitting, and hence towards the end (t=20), we see that the accuracy is slightly decreasing due to overfitting. It is also not that surprising to see that lambda appears to have a minimum in the middle of the range, since picking the best lambda is a balancing act between the loss value and the change in value of theta.

2b)
the optimal value for lambda = 1,
the optimal value for iterations = 20

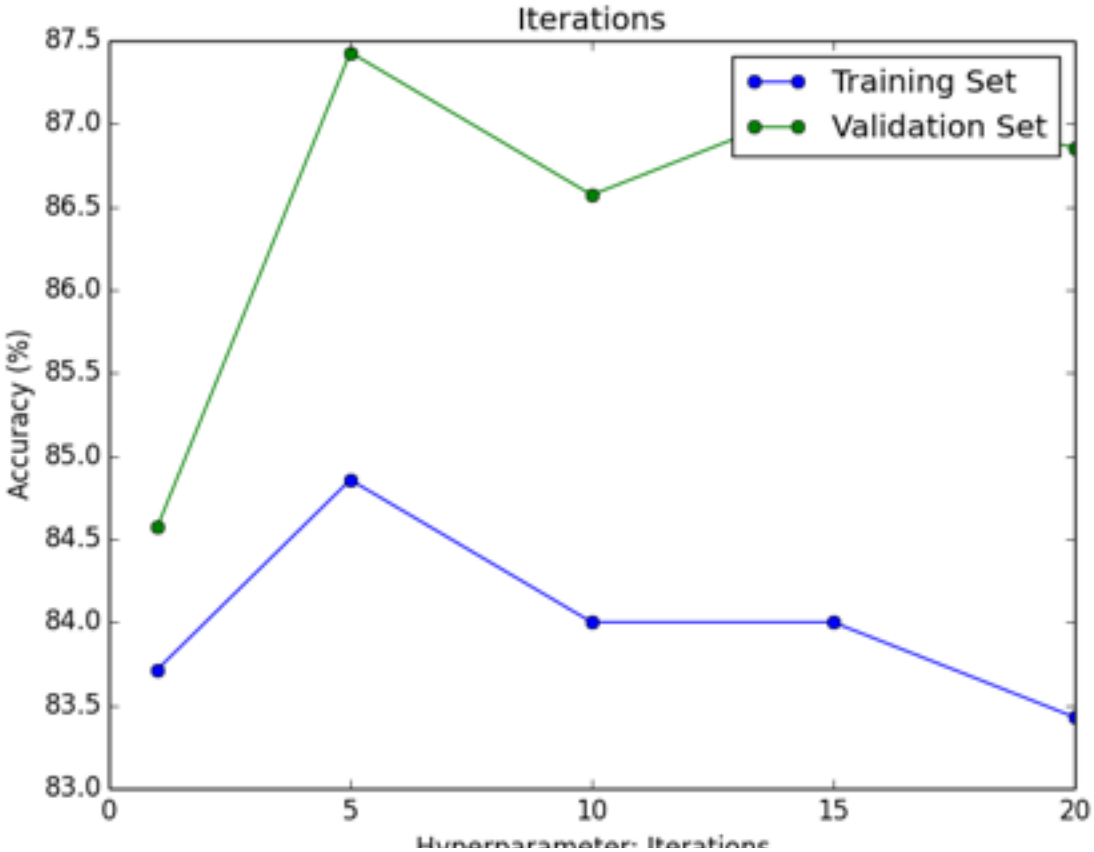
test = 86.5714285714% correct (303 out of 350).

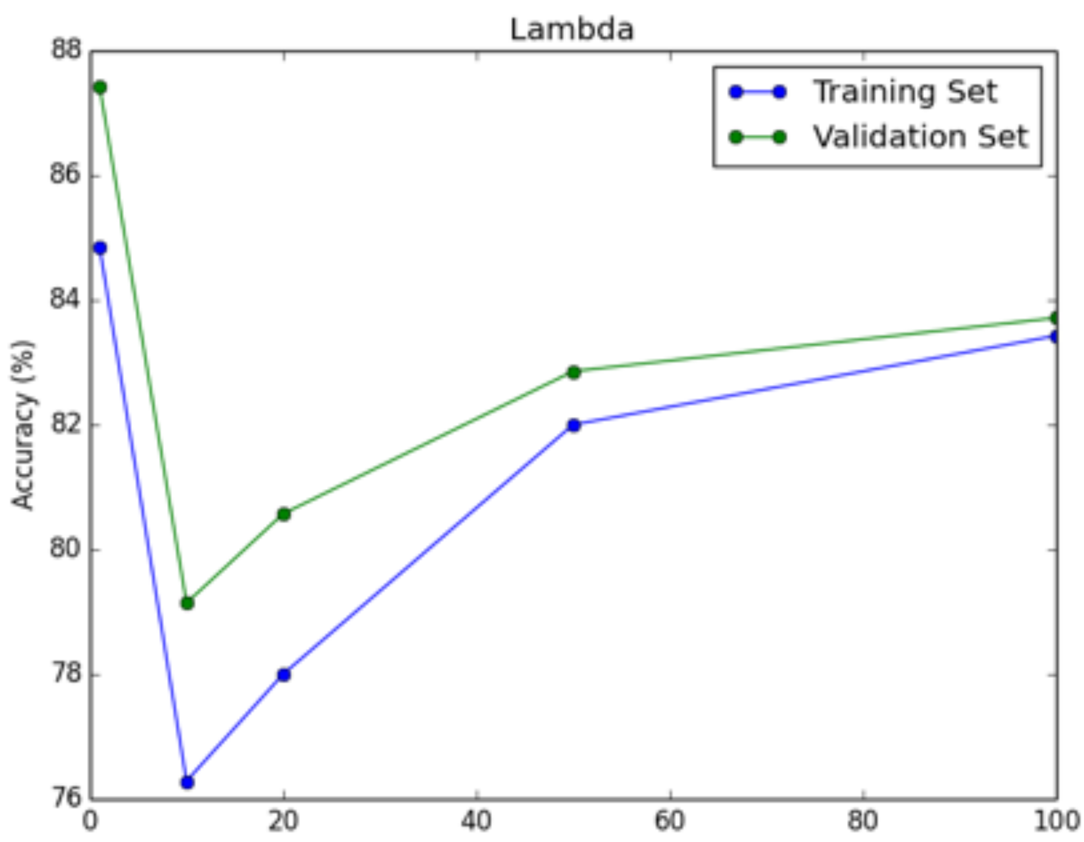
3)
Extra features usually just add extra noise which makes the algorithm less accurate. Therefore I first tried removing bigrams, Average word length, and Unique words. Here were my results:



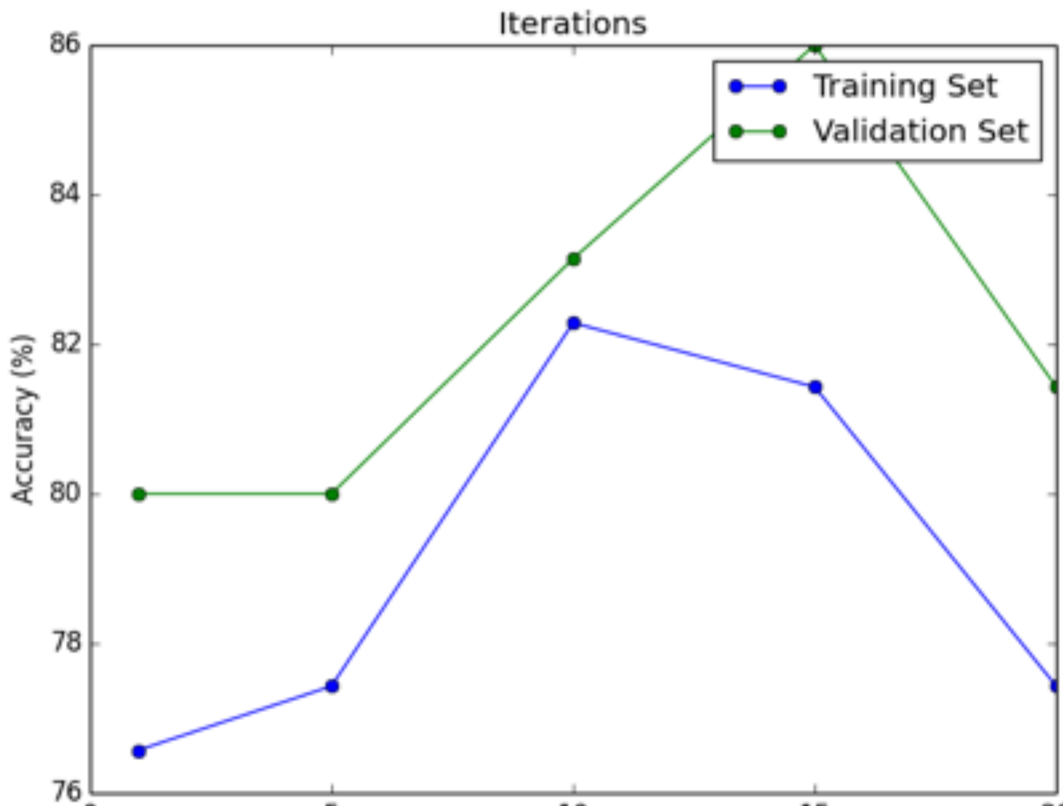

Lambda
Training Set
Validation Set
Accuracy (%)
Hyperparameter: Lambda
87
86
85
84
83
82
81
80
0
20
40
60
80
100

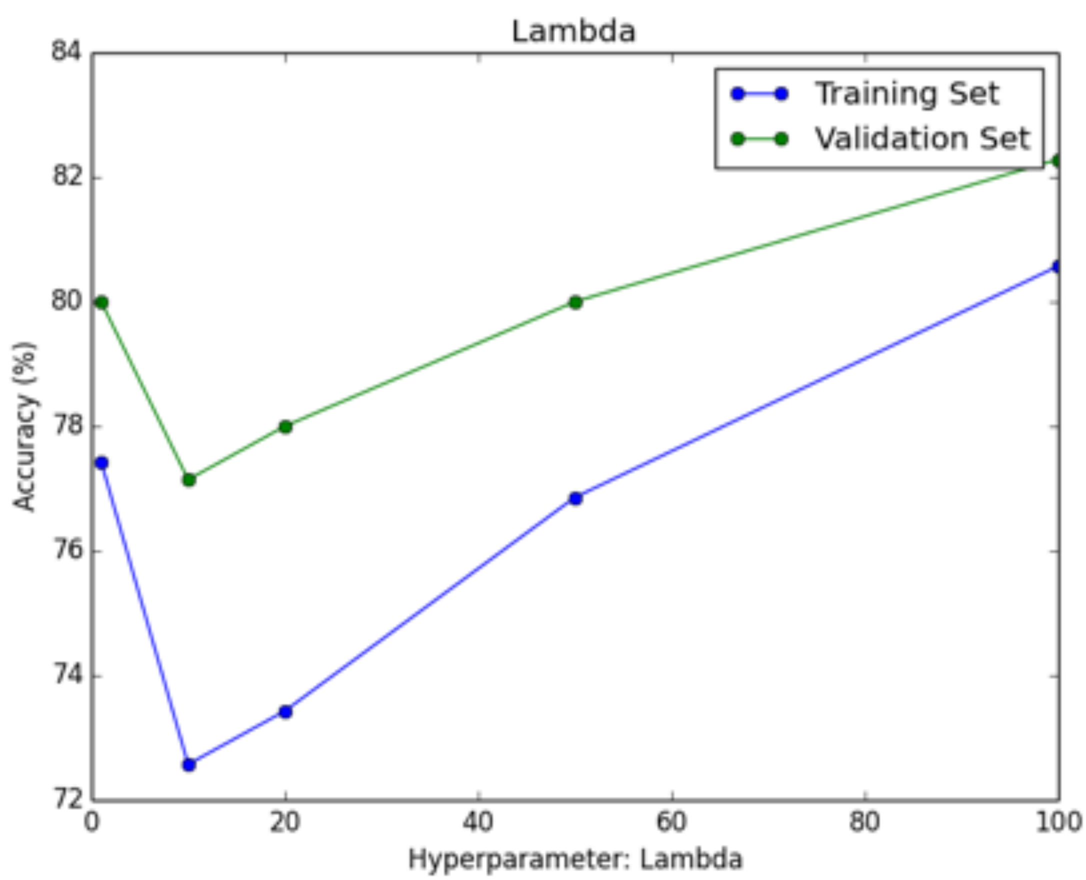
I then tried removing stop words from my unigrams (to only have a dictionary of words that actually “matter”). Below are the results for that:





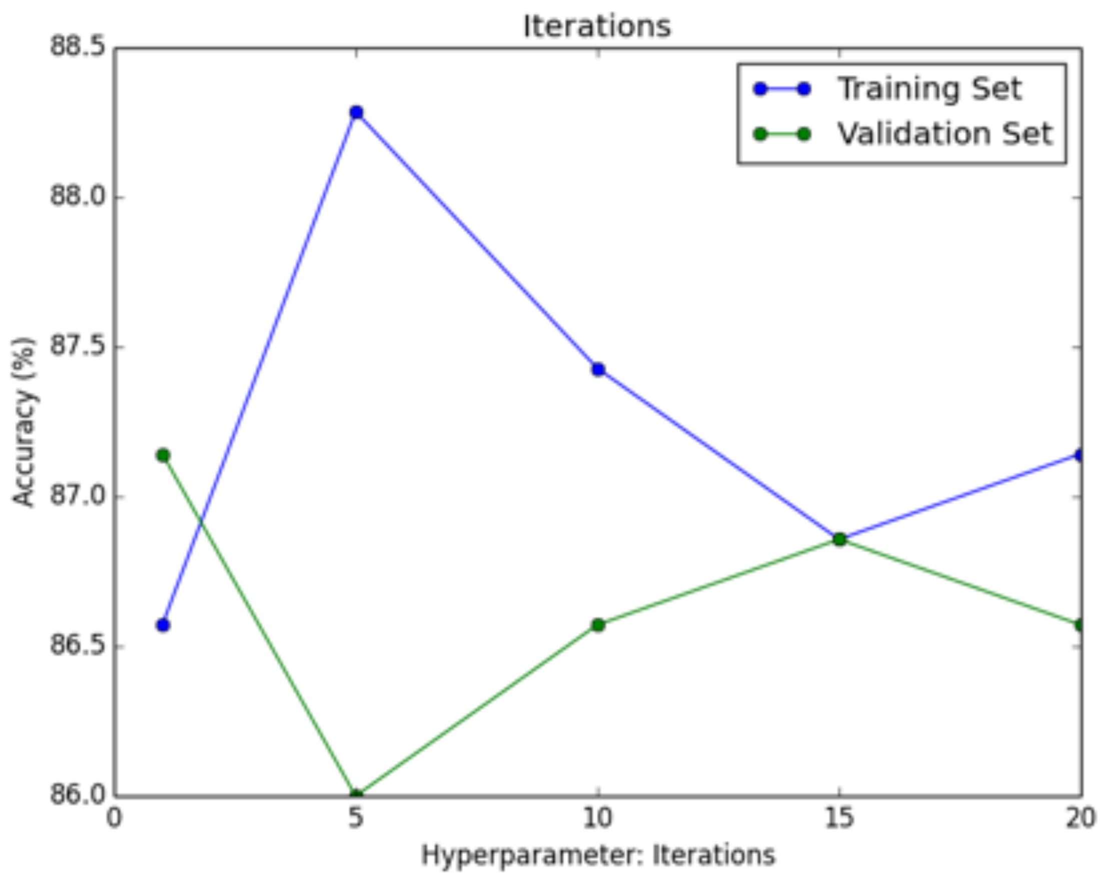
I then require that all unigrams be members of the dictionary of words (to remove the noise of plurals and words in variance tenses). Below are the results:

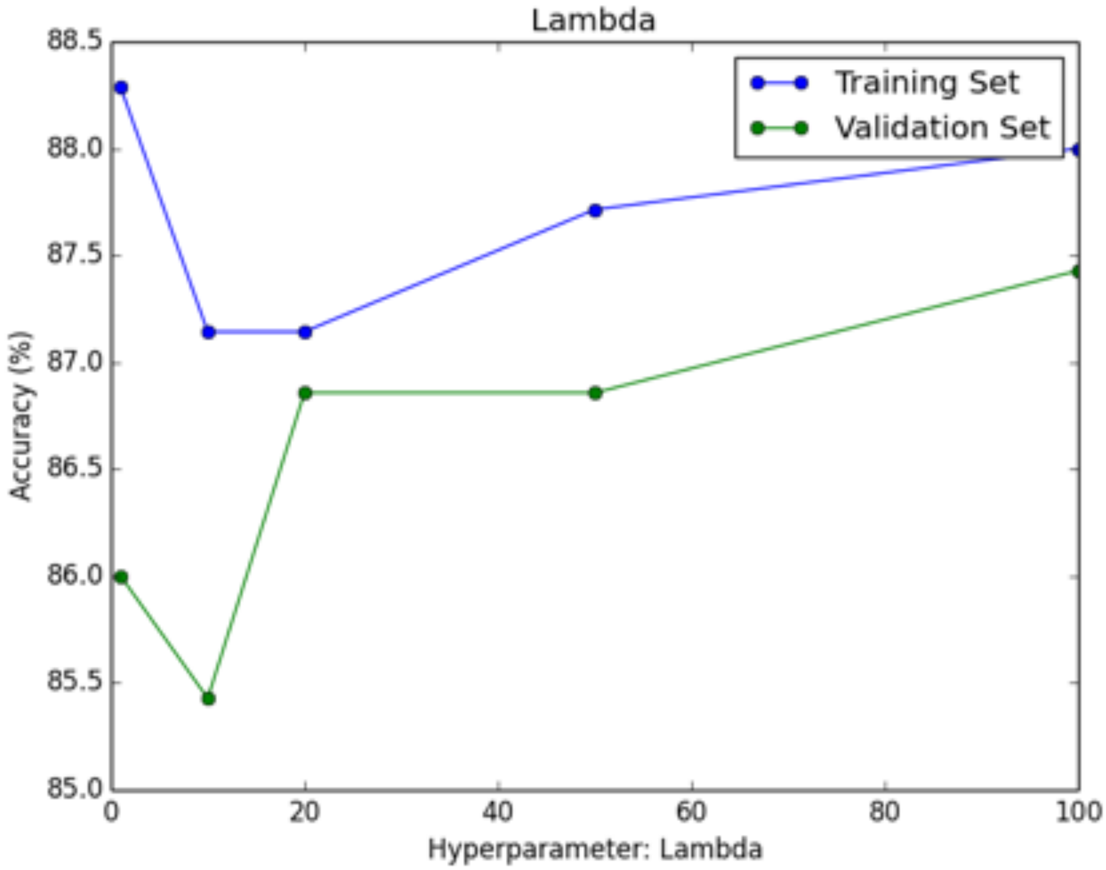




I also tried adding word lengths quantiles, number of SAT words quantiles, number of stop words quantiles, number of misspellings quantiles, and trigrams with mixed results.

I tried removing average word length feature, removing the unique word vector, and add the misspellings quantiles, along with the original bigrams and unigrams. This resulted in the method getting a 88.285% accuracy (309 out of 350), which was better than the 86.571 test case before optimization. However this high result was actually a result of overfitting, with the test performing better but validation performing worse.





However since my submission is based on my performance on test, I have decided to submit this version.