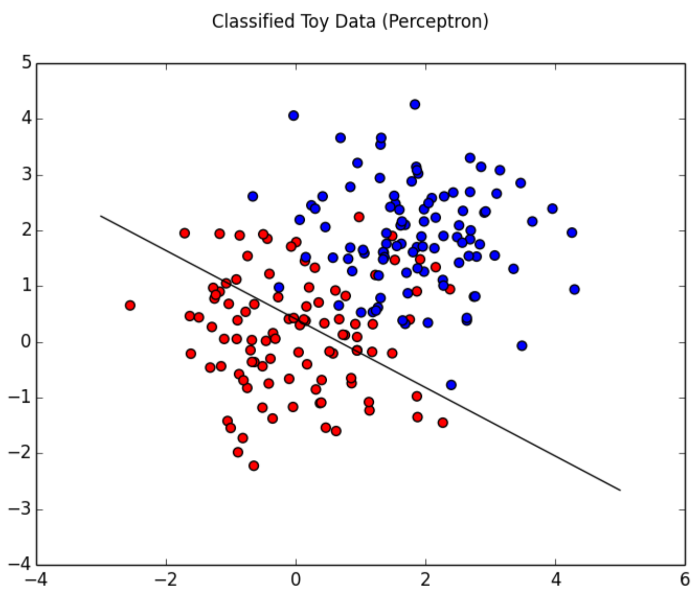
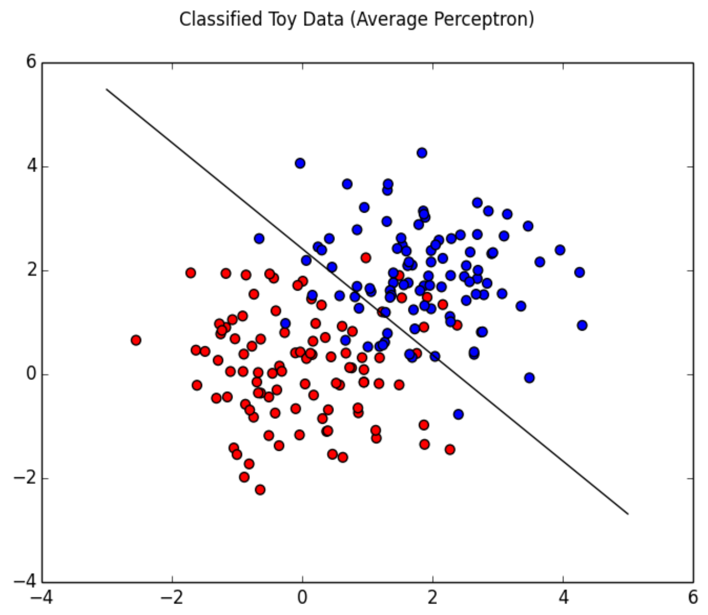
Abishkar Chhetri
6.036 Writeup

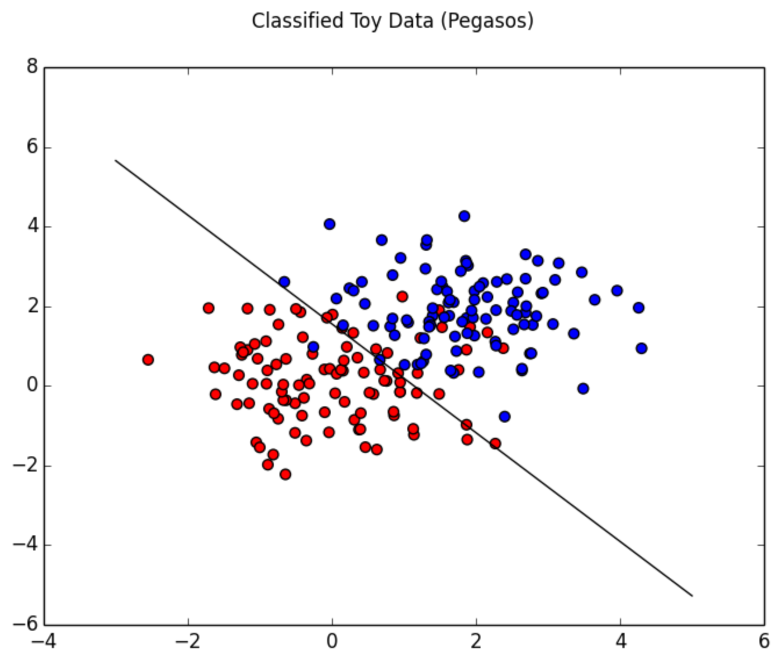
Why do these algorithms provide different decision boundaries? Hand in the plots obtained for each algorithm along with a short paragraph answering the question in your pdf write-up. [5 Points]



Our data is not well separated so the perceptron doesn't do the best job in classifying the values since it doesn't have a well defined loss function.

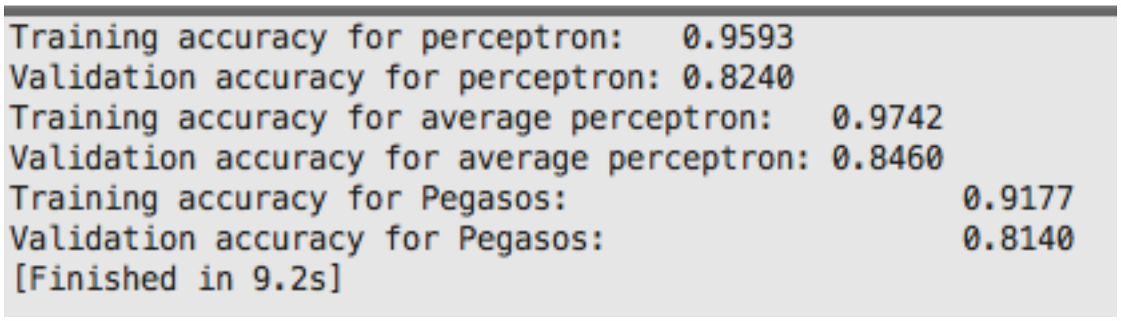


The average perceptron algorithm does a better job in classifying the data as it takes a more balanced approach in determining the classifier as it takes the average of all the theta updates to determine the final classifier.



The Pegasos by far does the best job of classifying the values. It makes it so that the updates are slow and gradual. It does so using a regularization parameter Lambda. It also directly tries to minimize the hinge loss function on every update.

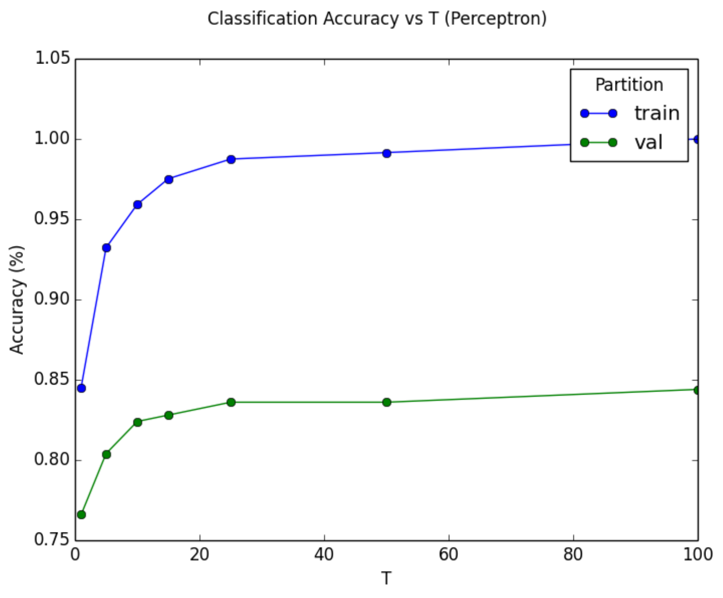
Part 2: 9ab)

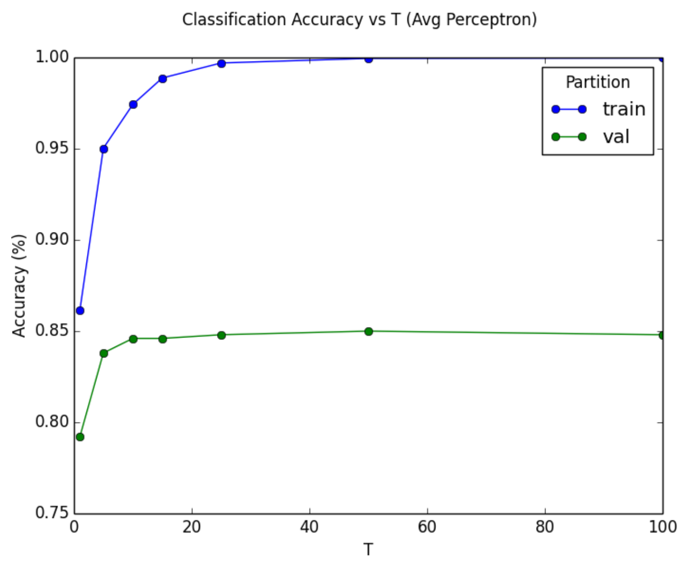


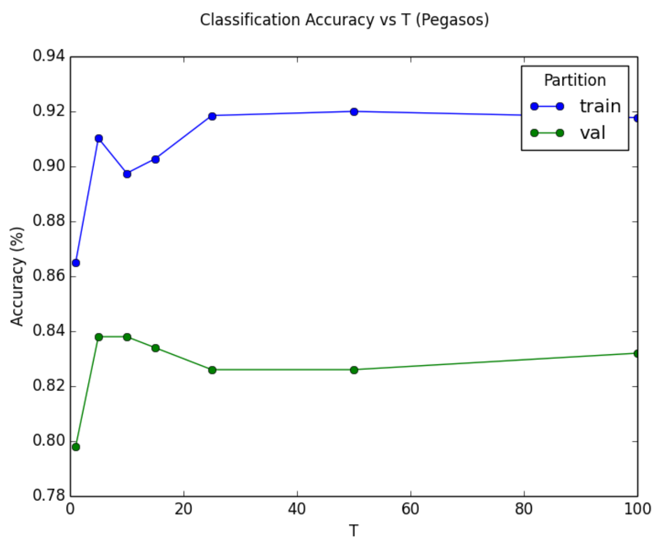
```
Training accuracy for perceptron:   0.9593
Validation accuracy for perceptron: 0.8240
Training accuracy for average perceptron:   0.9742
Validation accuracy for average perceptron: 0.8460
Training accuracy for Pegasos:                   0.9177
Validation accuracy for Pegasos:                 0.8140
[Finished in 9.2s]
```

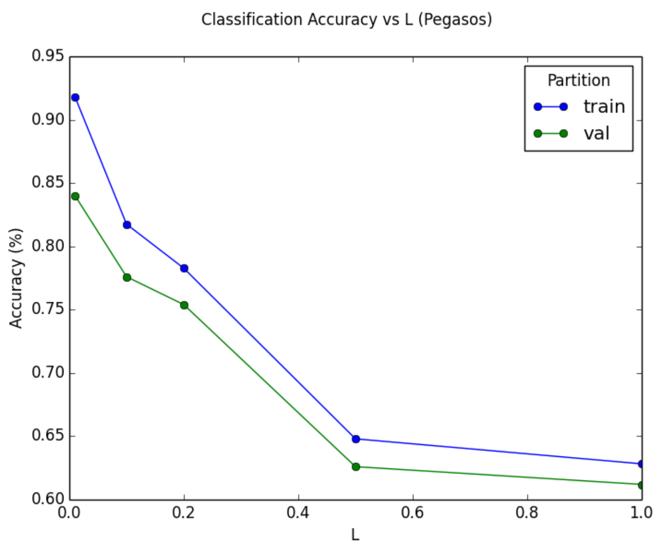
The average perceptron was the best algorithm followed by Pegasos. This dramatic increase in average perceptron accuracy was caused by the availability of more significant feature vectors when we stripped the stop words from the word list.

For L = 0.01 and Lambda = 15









10abc)

(a) Do the training and validation accuracies behave similarly as a function of λ and T? Why or why not?

Yes, the training and validation accuracies behave similarly as a function of λ and T because the training and validation data are similar in nature and the learning algorithm classification is implemented in such a way that it generalized well to the validation data.

(b) Which algorithm performed the best of the three?

The average perceptron performs the best out of the three algorithm and it also reaches a high accuracy the quickest.

(c) What are the optimal values of T for your perceptron? For average perceptron? What are the optimal values of T and λ for your Pegasos algorithm?

According to the graph, the following were the best parameters for each of the algorithms:

Best T for Perceptron: 25 -> 83%
Best T for Avg Perceptron: 10 -> 84.5%
Best T for Pegasos: 25 -> 83%
Best L for Pegasos: .01 -> 84%

11a) Call the appropriate accuracy function to find the accuracy on the test set. Report this value in your writeup.

The testing accuracy that was returned was **.8**

11b. Report top ten most explanatory word features in your writeup.

['delicious', 'amazing', 'wonderful', 'glad', 'pleased', 'plan', 'canned', 'helped', 'excellent', 'run']

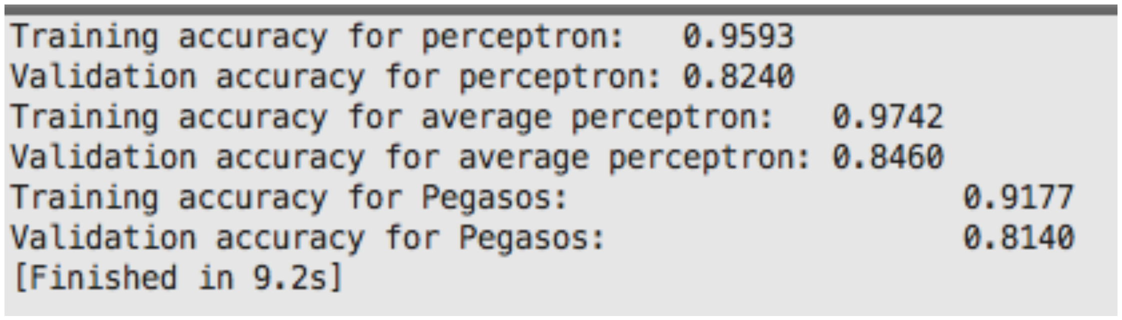
12) A clear explanation of how you changed the feature set, and why you think the features you chose might be useful.

A description of the experiment you conducted to compare it to the original feature-computation method, and the results of that experiment. Include plots if applicable.

I was considering Normalizing the Feature Vectors, but ran into multiple bugs, therefore I made several small implementations. I made the following changes:

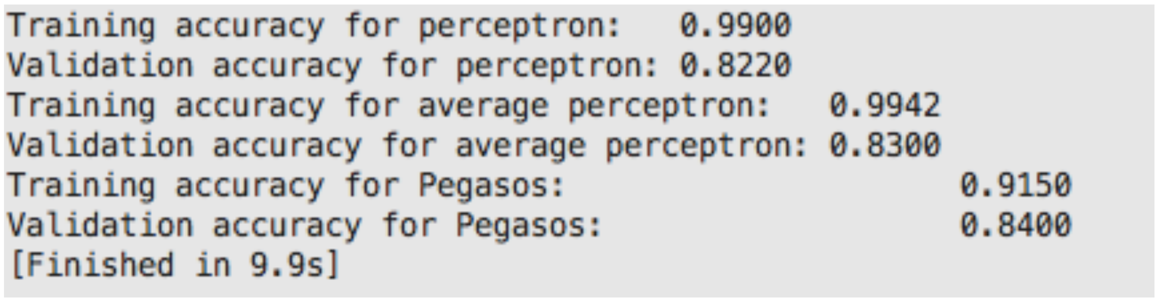
- Extracted/removed stop words from the dictionary
- Added a threshold for a number (2) of times a word had to appear before it was included in the dictionary.
- Changed the factor of eta (X1.5)

I removed the stop words because I believed removing the unnecessary features from the feature vector will allow theta to adapt to features that have a bigger weight on the classification of the data point. After I added the stop words, I got the following results:



```
Training accuracy for perceptron:   0.9593
Validation accuracy for perceptron: 0.8240
Training accuracy for average perceptron:   0.9742
Validation accuracy for average perceptron: 0.8460
Training accuracy for Pegasos:                     0.9177
Validation accuracy for Pegasos:                   0.8140
[Finished in 9.2s]
```

Without stop words removed

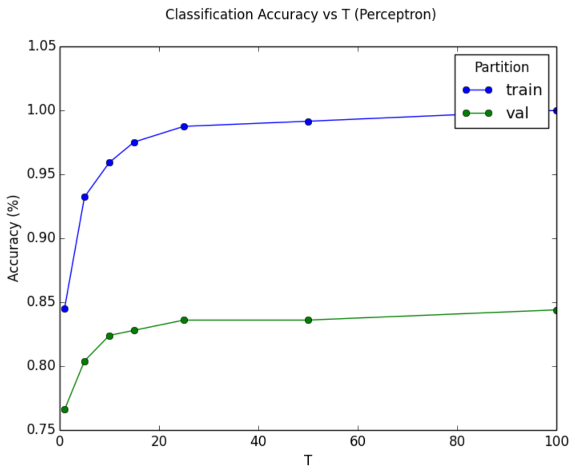


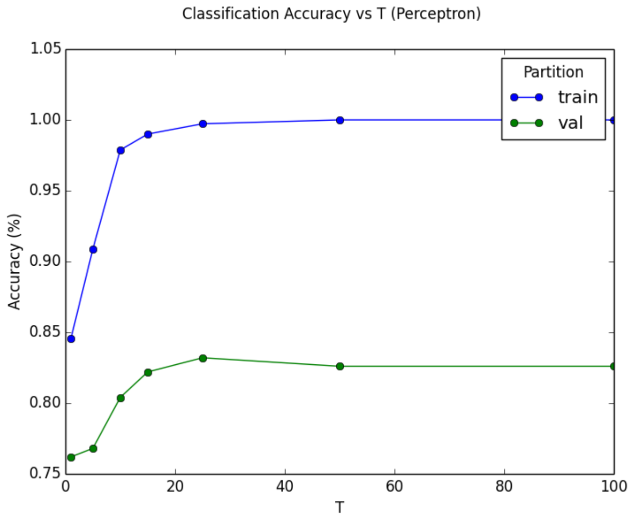
```
Training accuracy for perceptron:   0.9900
Validation accuracy for perceptron: 0.8220
Training accuracy for average perceptron:   0.9942
Validation accuracy for average perceptron: 0.8300
Training accuracy for Pegasos:                     0.9150
Validation accuracy for Pegasos:                   0.8400
[Finished in 9.9s]
```

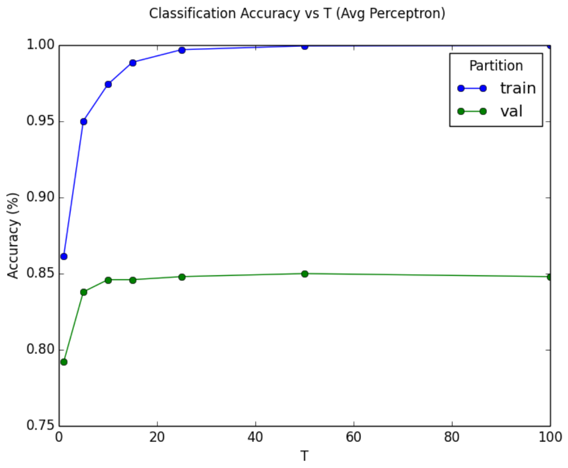
After stop words removed

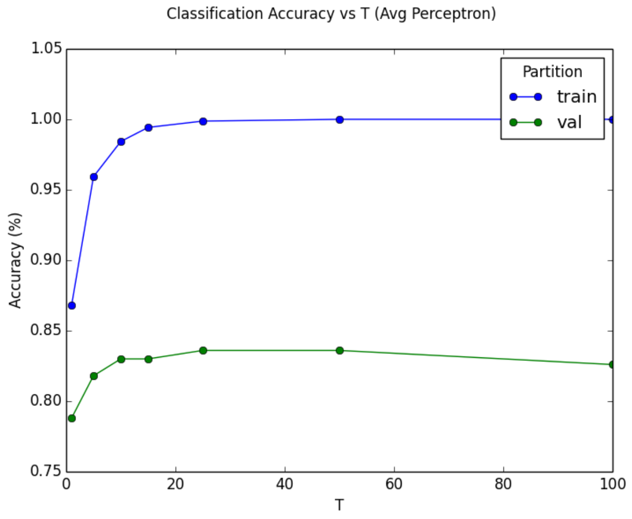
Before stop words removed

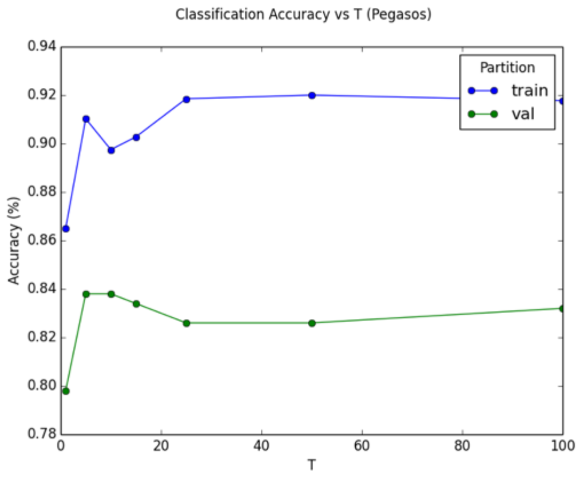
After stop words removed

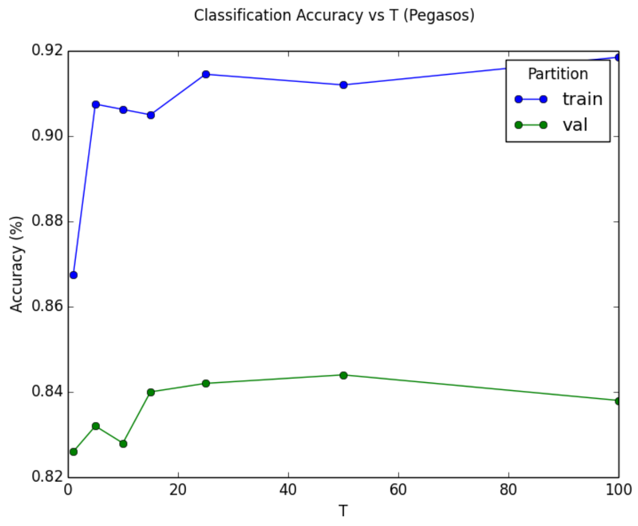




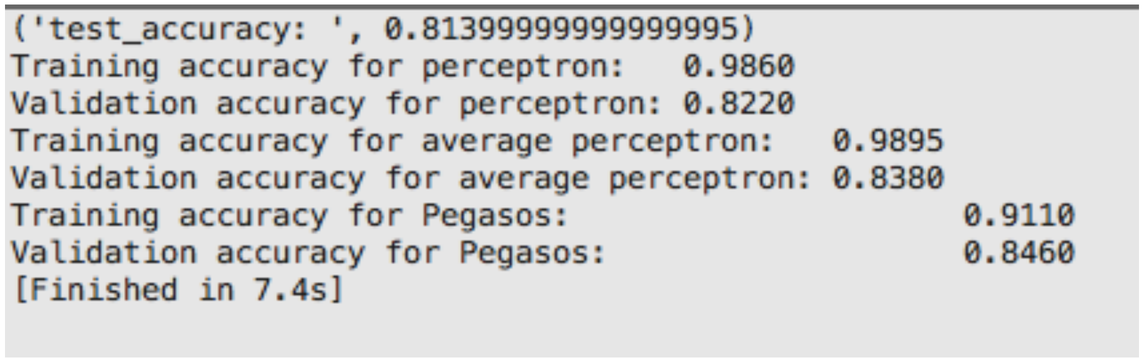








- After I removed the stopped words, adding a threshold for a number (2) of times a word had to appear before it was included in the dictionary and an additional constant for eta had a very little change. It improved the accuracy of the pegasos and perceptron algorithm but did nothing to the average perceptron. It also increased the test accuracy from .8 to .814.



```
('test_accuracy: ', 0.8139999999999995)
Training accuracy for perceptron:   0.9860
Validation accuracy for perceptron: 0.8220
Training accuracy for average perceptron:   0.9895
Validation accuracy for average perceptron: 0.8380
Training accuracy for Pegasos:                     0.9110
Validation accuracy for Pegasos:                   0.8460
[Finished in 7.4s]
```

Ultimately, after the changes I made, Pegasos turned out to be the most accurate out of all with an accuracy of .8460 on the Validation and .814 on the test data. Average perceptron accuracy actually decreased from .8460 to .8380. I think this was because of