(1.3 Questions)

Question 1)

*Note: Code for results are at testingStuff.py

	Tennis Data		Sentiment Data	
	Train	Test	Train	Test
AlwaysPredictOne	0.642857	0.5	0.504167	0.5025
AlwaysPredictMostFrequent	0.642857	0.5	0.504167	0.5025
FirstFeatureClassifier	0.714286	0.666667	0.504167	0.5025

Question 2)

None, AlwaysPredictMostFrequent and AlwaysPredictOne has same performance for both Datasets

Question 3)

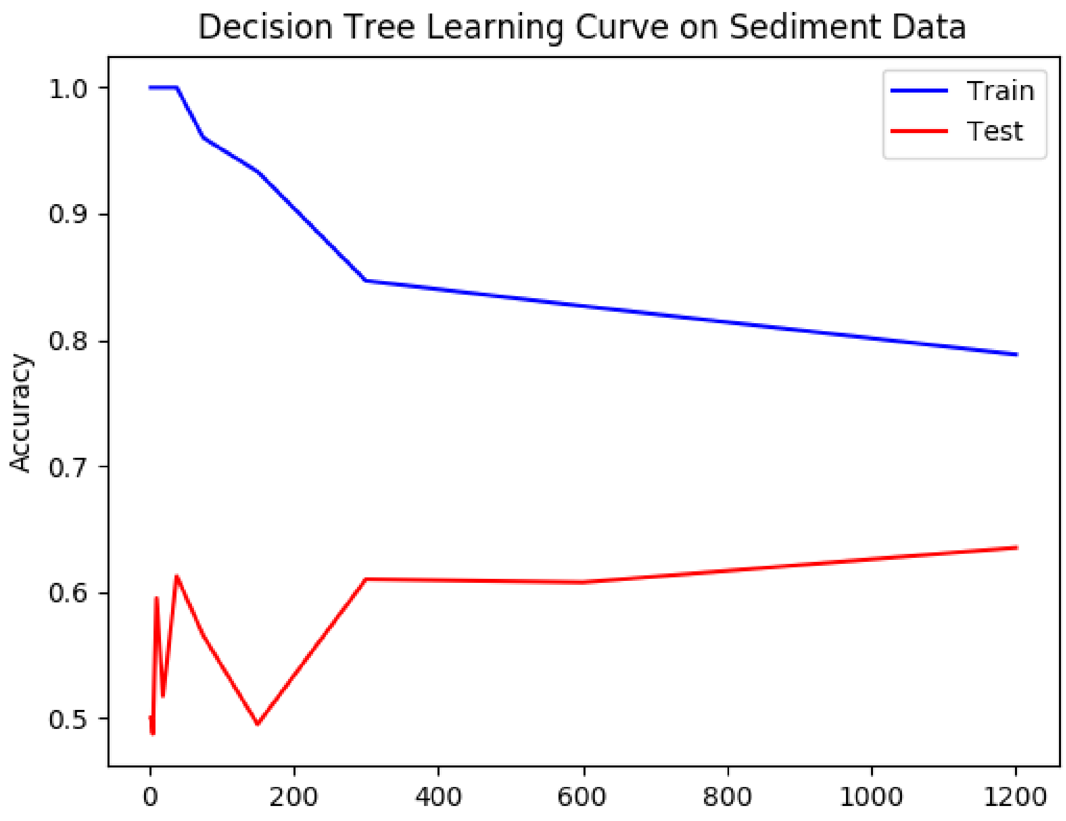
Tennis Data, is where FirstFeatureClassifier outperforms AlwaysPredictOne

Question 4)

The second line computes the training accuracy or average of the values of that are Y > 0 and X > 0;
because it gets the average of the points where the labels y are greater than 0 AND prediction of X is
greater then 0; then averages

Question 2 Decision Tree)

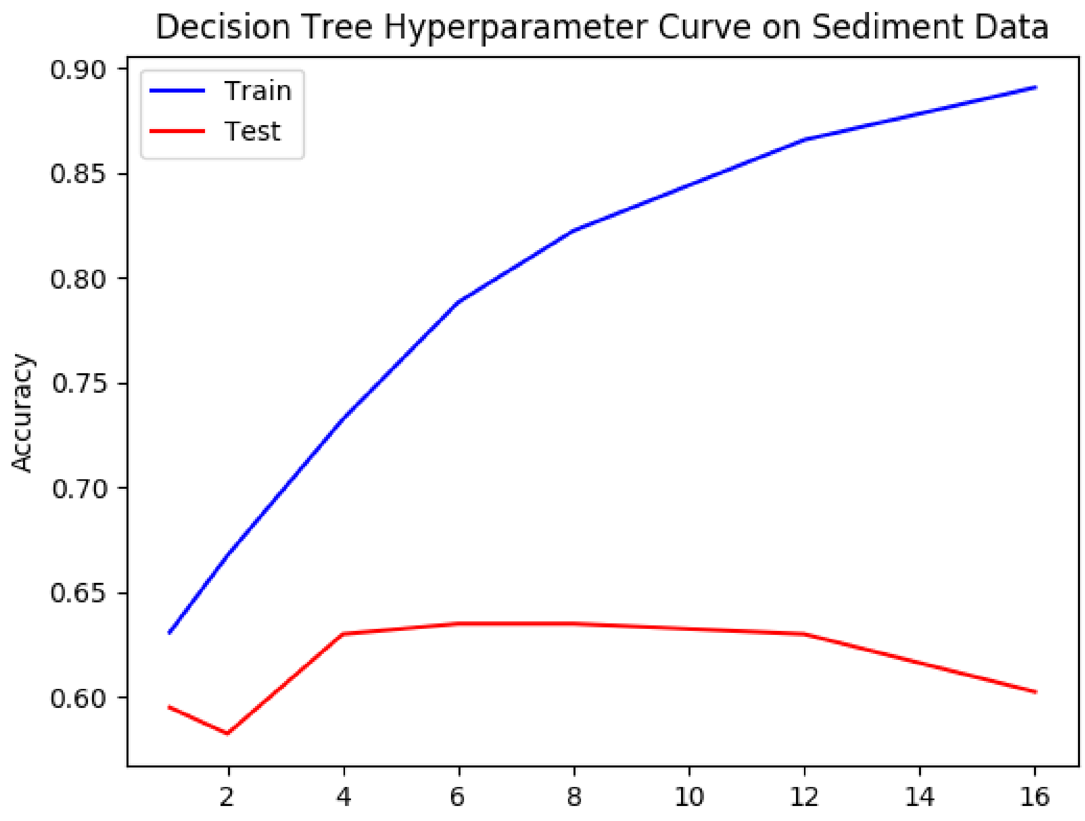
5)



*Note: DT tested on testDecisonTree.py

6) There reason why the training data goes down is because we are limited to a certain depths. With lower data do test on, we can see that the predictions have a high accuracy, but as we increase the amount of data, it starts to decrease, and at one point, it starts to decrease at a steady rate. The test data starts to increase as we get as we get more data. It is not monotonically because there are times when the accuracy decreases and then decreases.

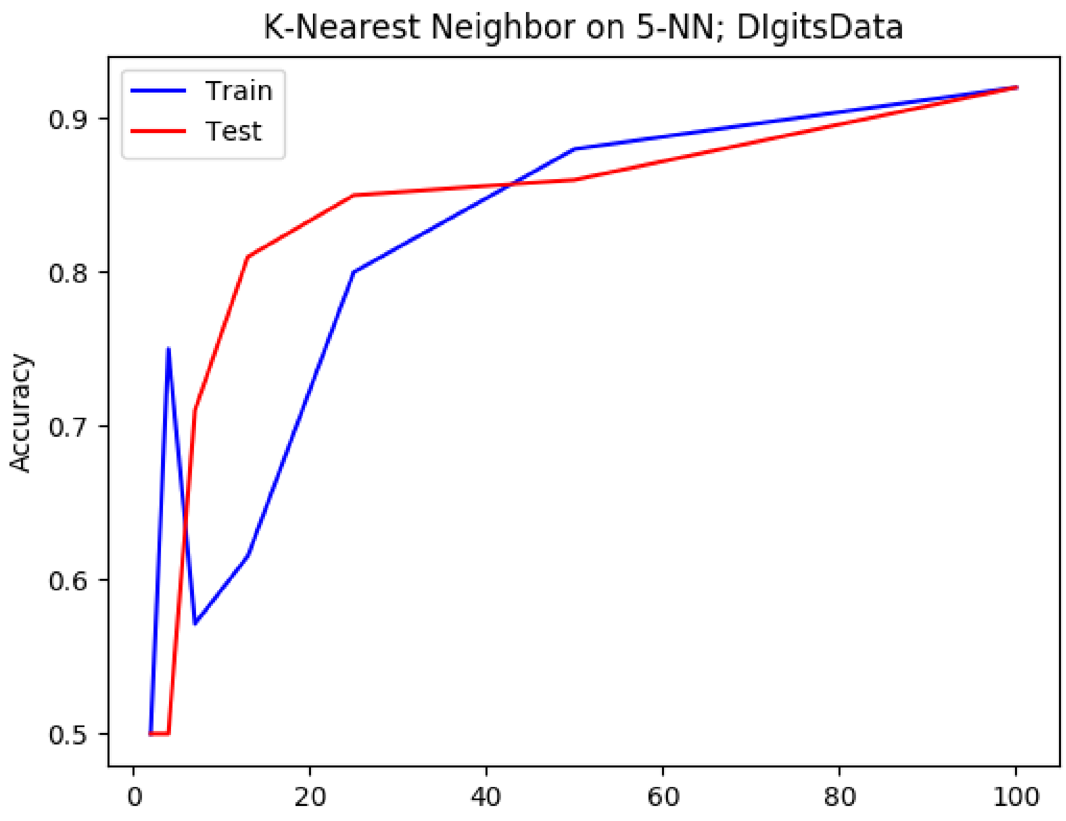
7)



8) There reason why the Training set is monotonically increasing is because we have more depth to train our data, and give higher accuracy. The reason we it is monotonically increase is because we start to overfit the training set. We can see that around depth 12 is when we start starts to show how overfit the data is because as training accuracy increases, the test data accuracy starts decreasing and is no longer able to generalize.

3 Nearest Neighbor)

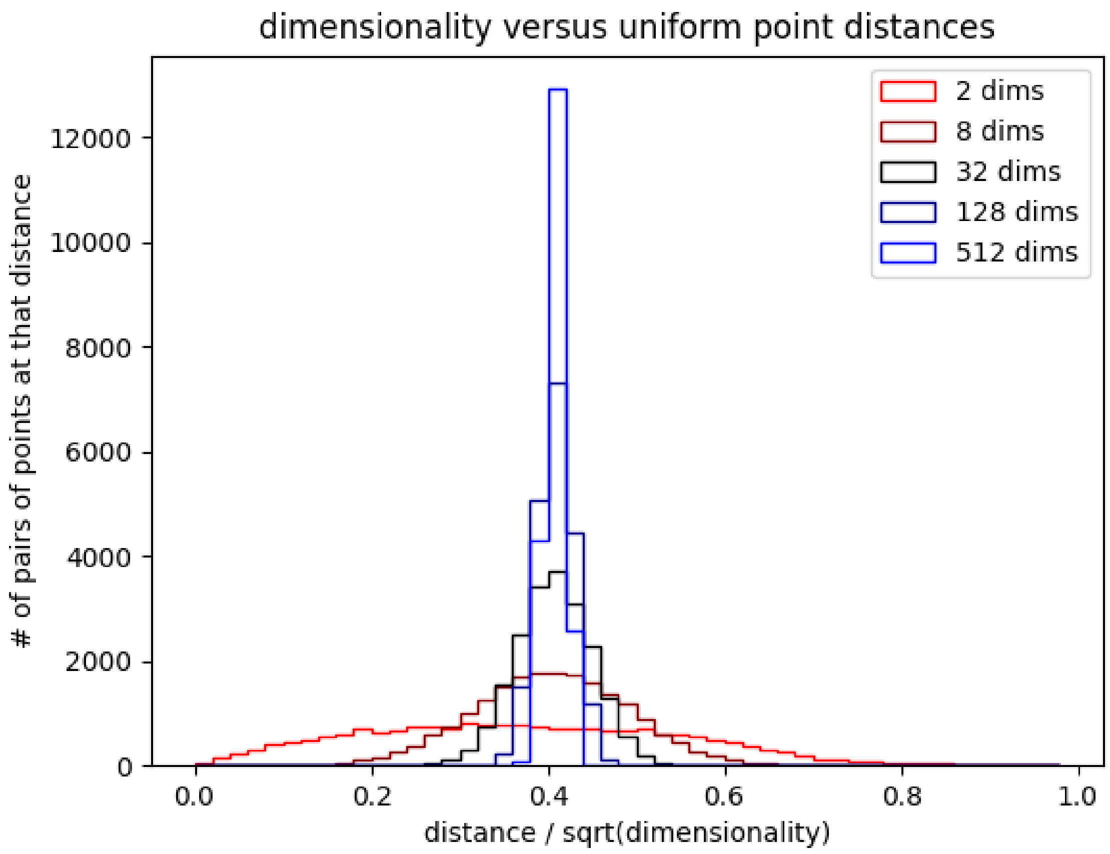
*Note: code test on testKNN



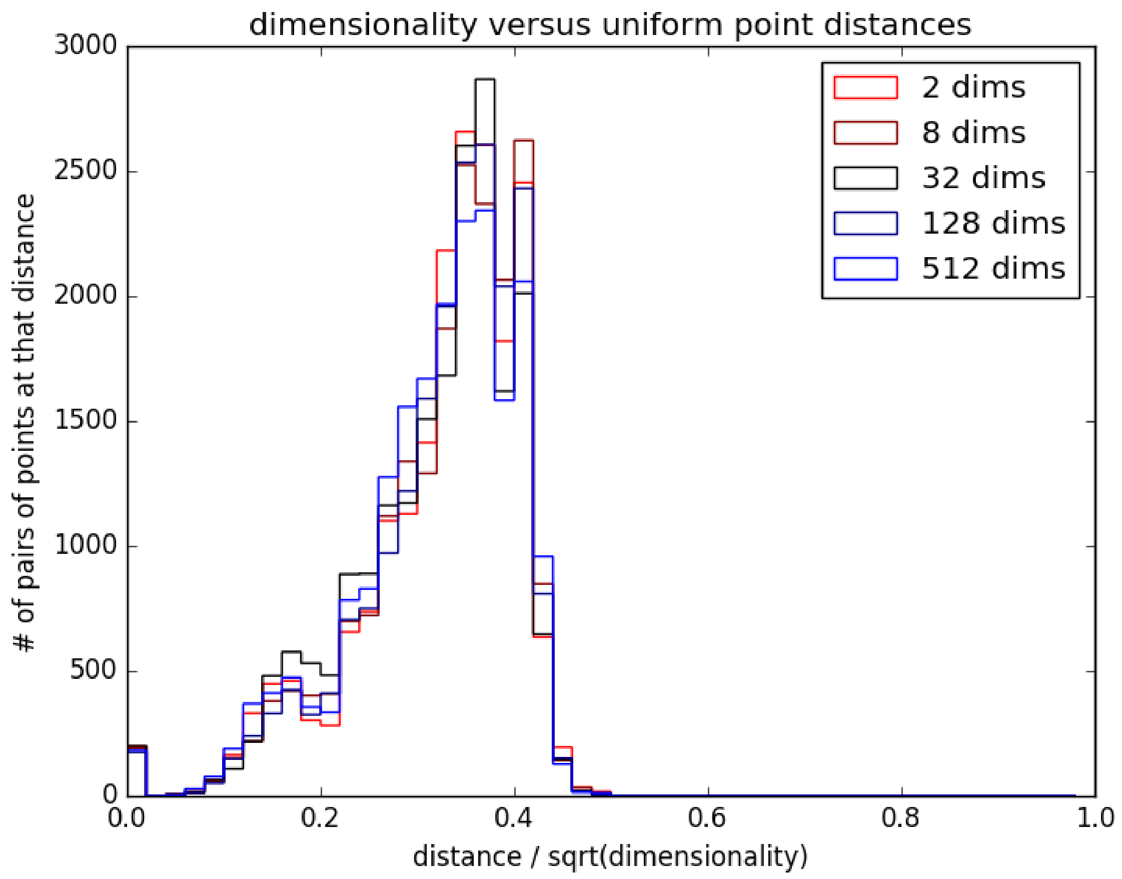
9) There is a spike in the training accuracy on the left because we are overfitting. We have 5-NN on approximately 2 – 5 data points; which leads to high accuracy in training; it is overfitting. The trend that we notice on the test accuracy is that its starts to increase in the rate of growth as more data points are tested on. It starts to become more accurate because of more data point inputs.

10)

(a)


dimensionality versus uniform point distances
2 dims
8 dims
32 dims
128 dims
512 dims
# of pairs of points at that distance
distance / sqrt(dimensionality)
0
2000
4000
6000
8000
10000
12000
0.0
0.2
0.4
0.6
0.8
1.0

(b)

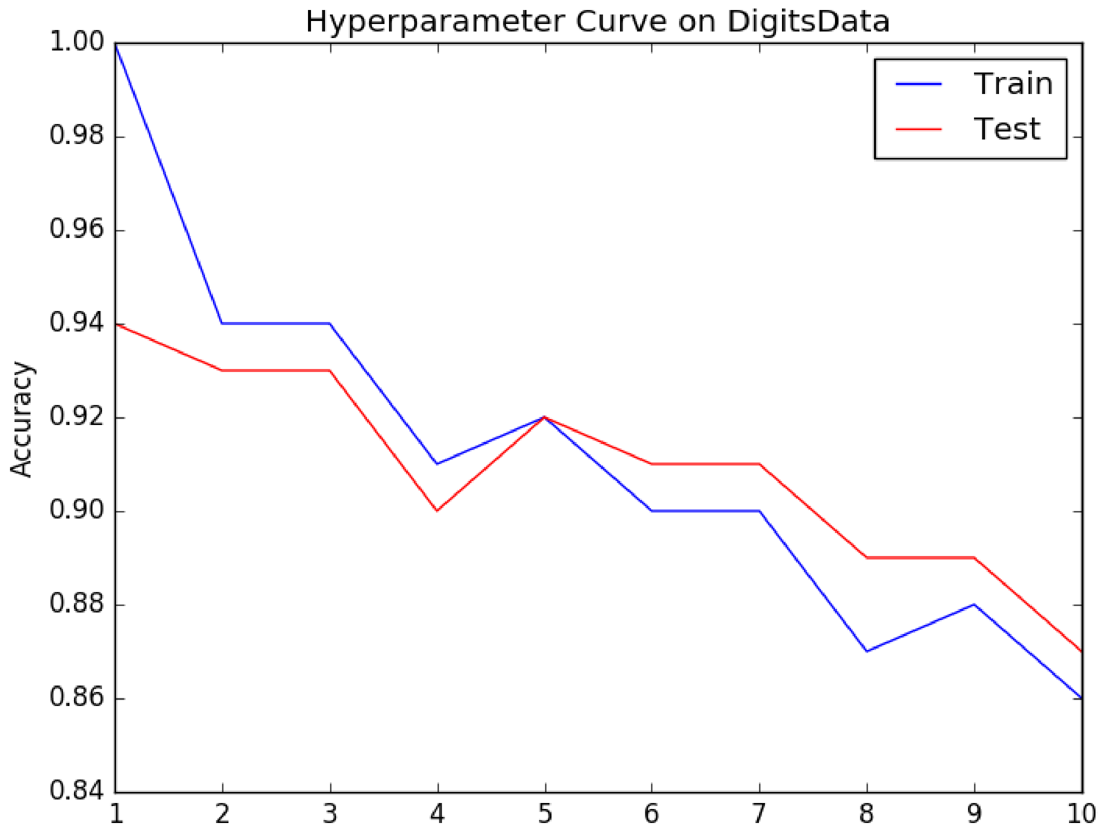


(c)The general difference of HighD and Datadigits is that the points are skewed to the left.

(d) Based on my plots, a good choice for epsilon would be dis/sqrt(dim) = .35 ~ 9.8

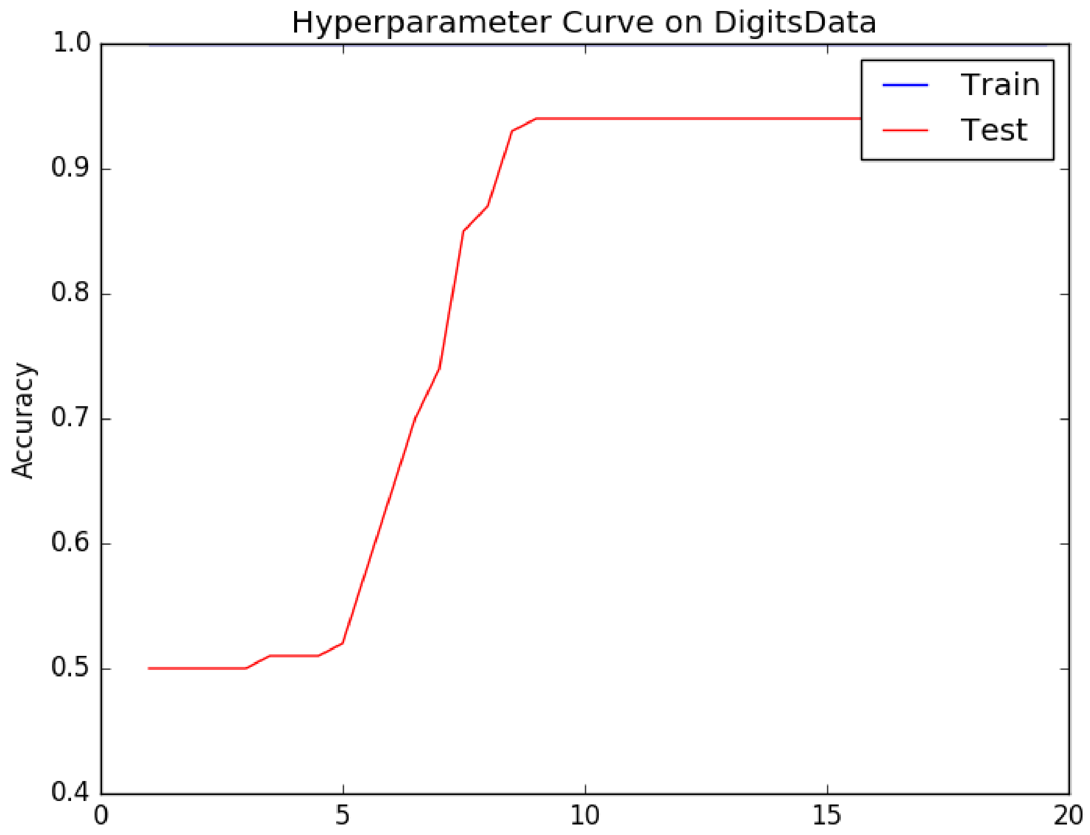
11)

Based on my results, the best value for K would be 5

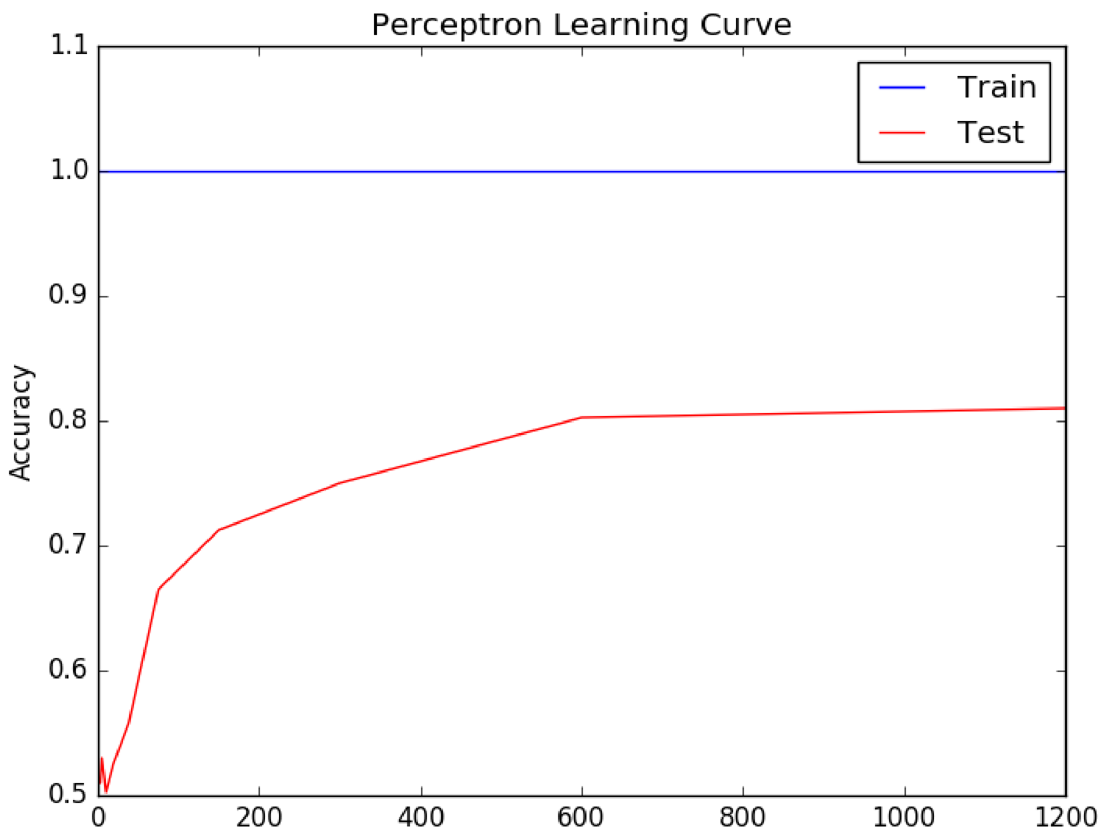


12)

Based on my guess on 10(d), where I guessed that 9.8 would be a good choice of epsilon, it shows that the accuracy is about 90% after 9.8

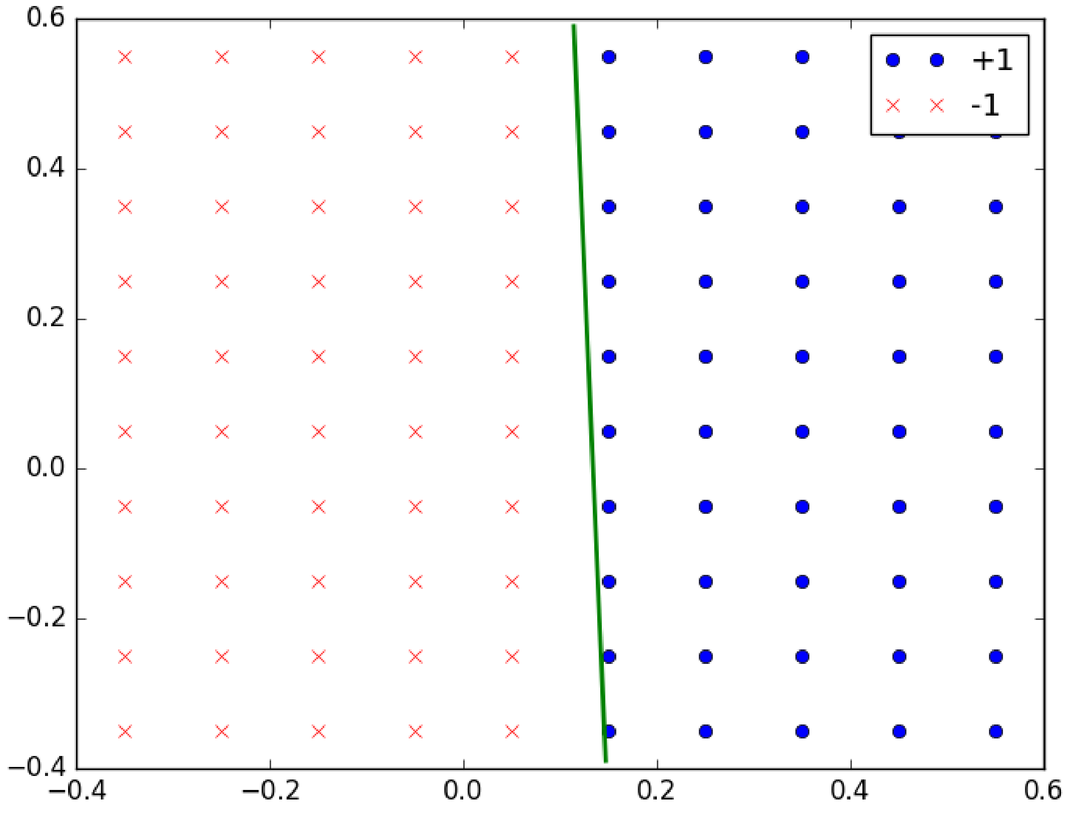

Hyperparameter Curve on DigitsData
Accuracy
Train
Test
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20

13)

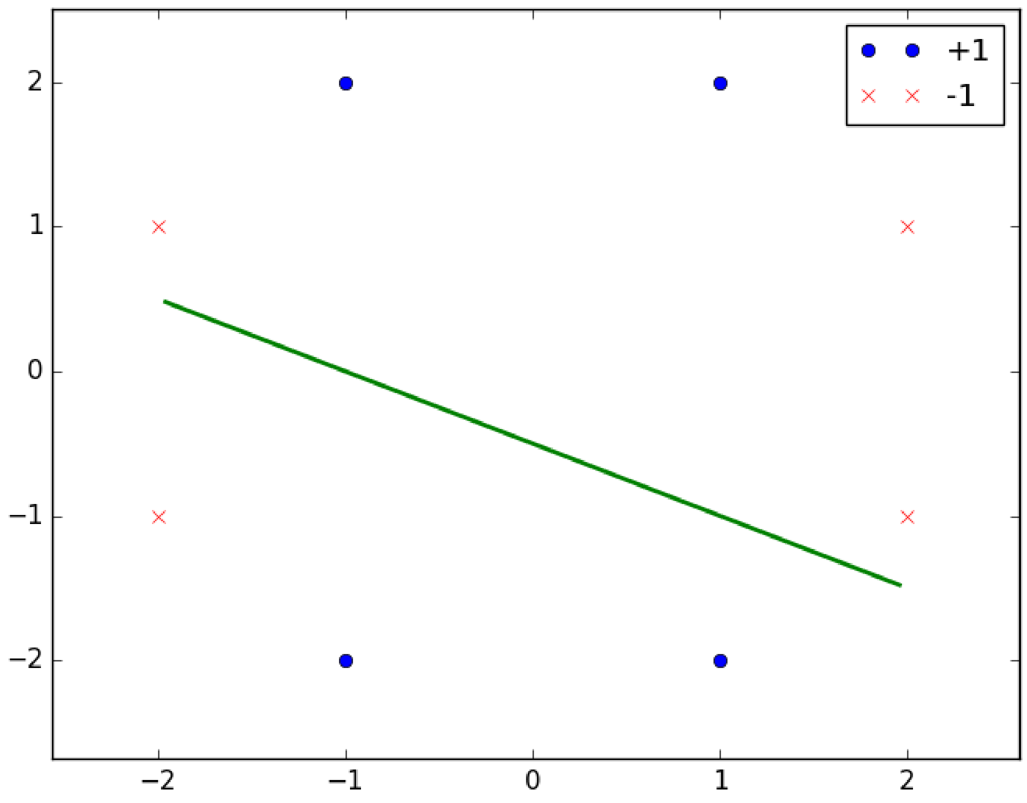


The data is not overfitting because as we start to test more data, we still a pretty high accuracy; does not degrade. Thus, We can tell that the points are linearly separable.

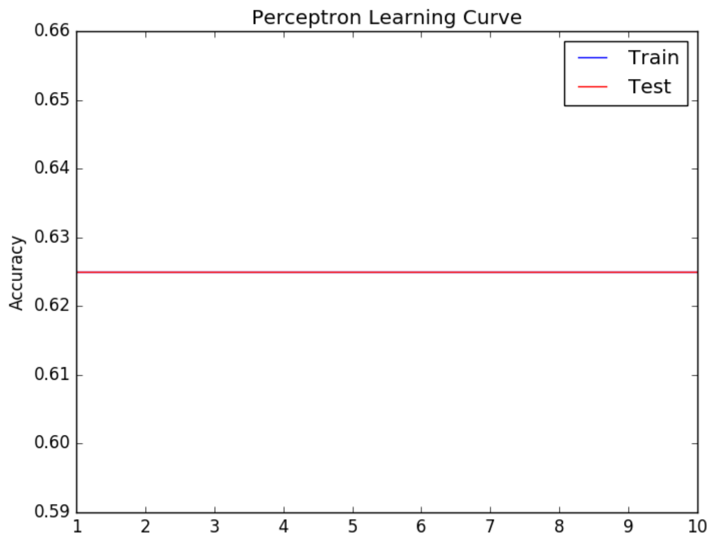
14)

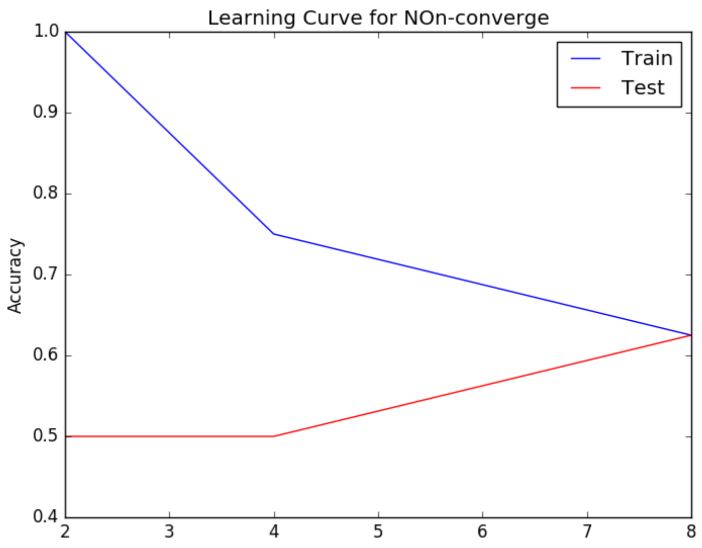


15)



The Perceptron algorithm will not converge because the points of positive and negatives are symmetrical to each other respectively. We will never be able to find a hyper parameter that will perfectly separate the positives and negatives.





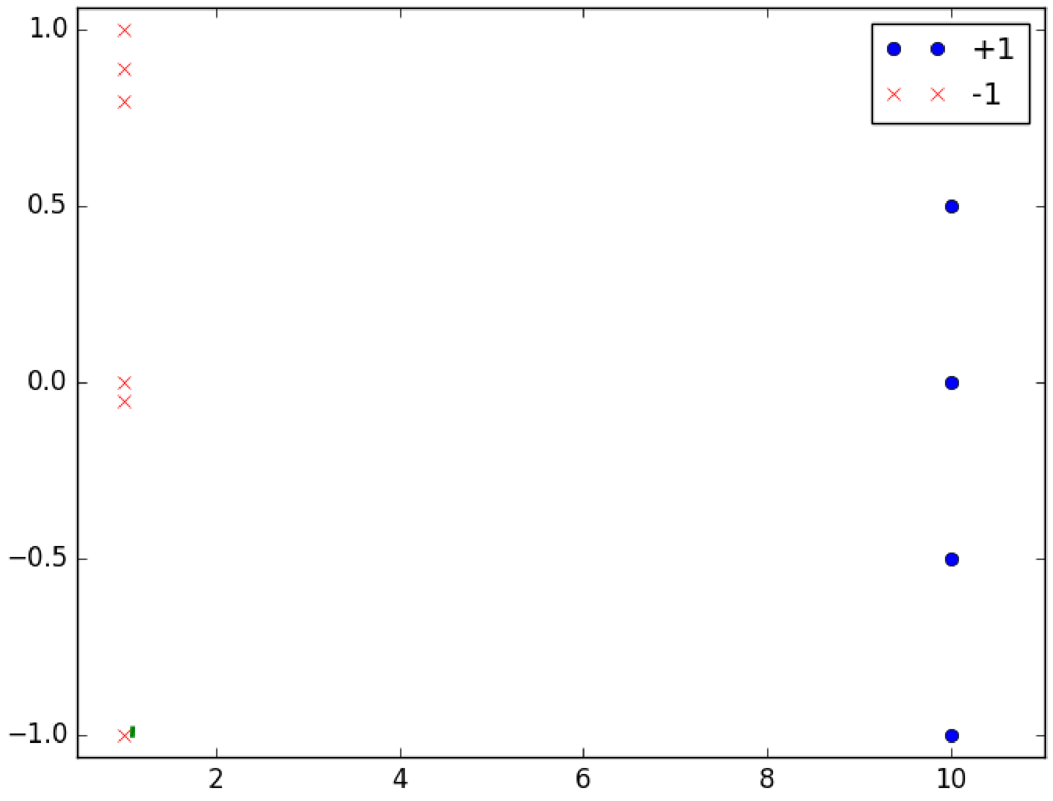
As we can see, the hyper parameter accuracy does not change. It cannot improve based on our data

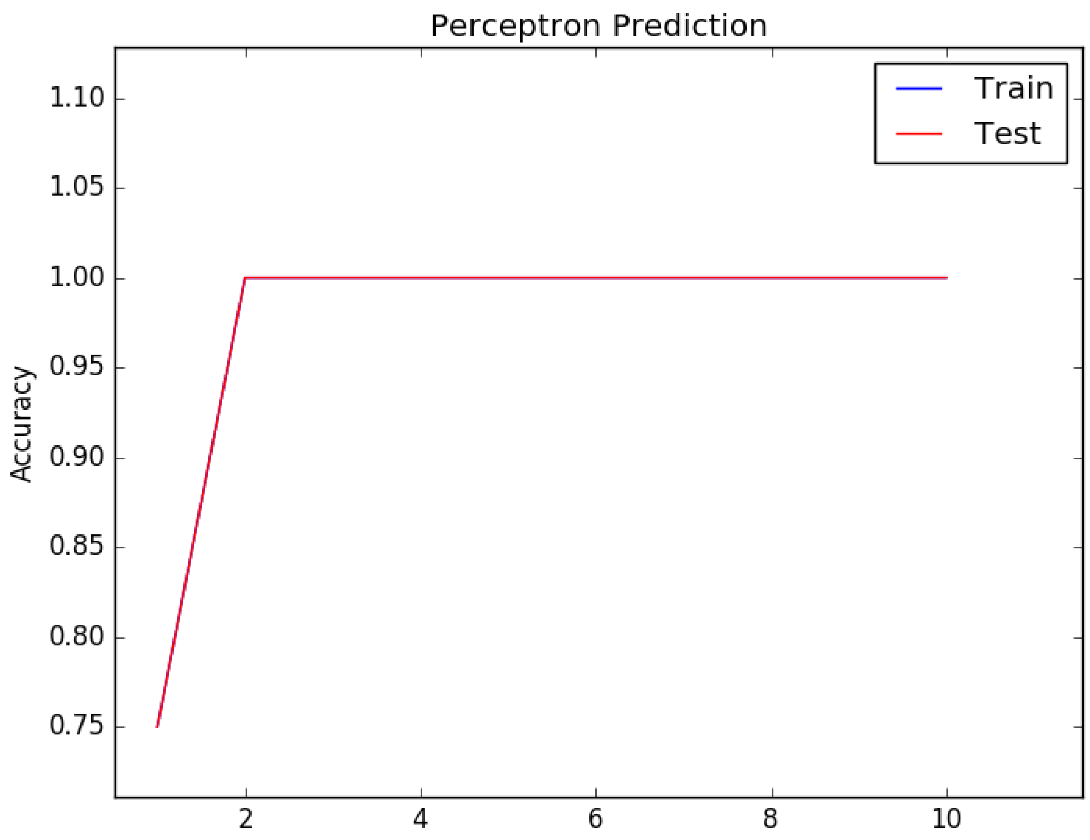
It does enter a repeating cycle, it is not necessary because once it finds the best hyperplane, it can no longer improve on it because the data does not converge

16)

(a) Yes, I believe that the points will converge faster when trying to find the points, it will have a huge difference when computing the dot product

(b)





As we can see on our data, after having an initial accuracy of 75%,
we can run only epoch = 2, we will get about 100%. The reason we would want to randomize the points before training is because, on average, we will get more runs on our data , and it will become more normalized

(c) It seems that my prediction was true because after running the data more than once, my accuracy was extremely high, after having 2 epochs. Which means that it was extremely fast to to converge the data that have big and small X values.