CS 7616
Pattern Recognition
HW 2
Date: February 28, 2016

I. Random Forests

i. Wine Data

1. Data split into train (75%) and test (25%) matrices. Each row of the matrices is a set of features. The first feature is the true class of that sample.
2. We have two types of Random Forests, one in which each tree is of full length, and another which is restricted to depth 5. We have executed 500 random forests, each with number of trees increasing by 2. Thus, first random forest has 2 trees, second has 4, third has 6, and so on. As generating 500 random forests is time taking, we generate one random forest every 25 steps, and use interpolation to estimate the others. These values have been chosen as here we can see a smooth decreasing error percentage curve. For higher number of random forests, or higher number trees per forest the curve converges rapidly.
3. Confusion Matrix:
   The table given below describes the confusion matrices for both Full tree, and limited tree(Depth 5) for Train set, Test set, and out of bag set. Also provided is the final accuracy score for each case. We see that for both the trees, the simulation converges to the same accuracy score. This is because for large number of iterations, their average gives a good estimate of the true class, even with weaker classifiers.

```
Train Set

Confusion Matrix using Full tree
[[47  0  0]
 [ 0 51  0]
 [ 0  0 35]]
Final Accuracy Score =  1.0

Confusion Matrix using Depth 5 Decision Tree
[[47  0  0]
 [ 0 51  0]
 [ 0  0 35]]
Final Accuracy Score =  1.0
--------------------

Test Set
```

Confusion Matrix using Full tree
[[12 0 0]
 [0 20 0]
 [0 0 13]]
Final Accuracy Score = 1.0

Test Set Confusion Matrix using Depth 5 Decision Tree
[[12 0 0]
 [0 20 0]
 [0 0 13]]
Final Accuracy Score = 1.0

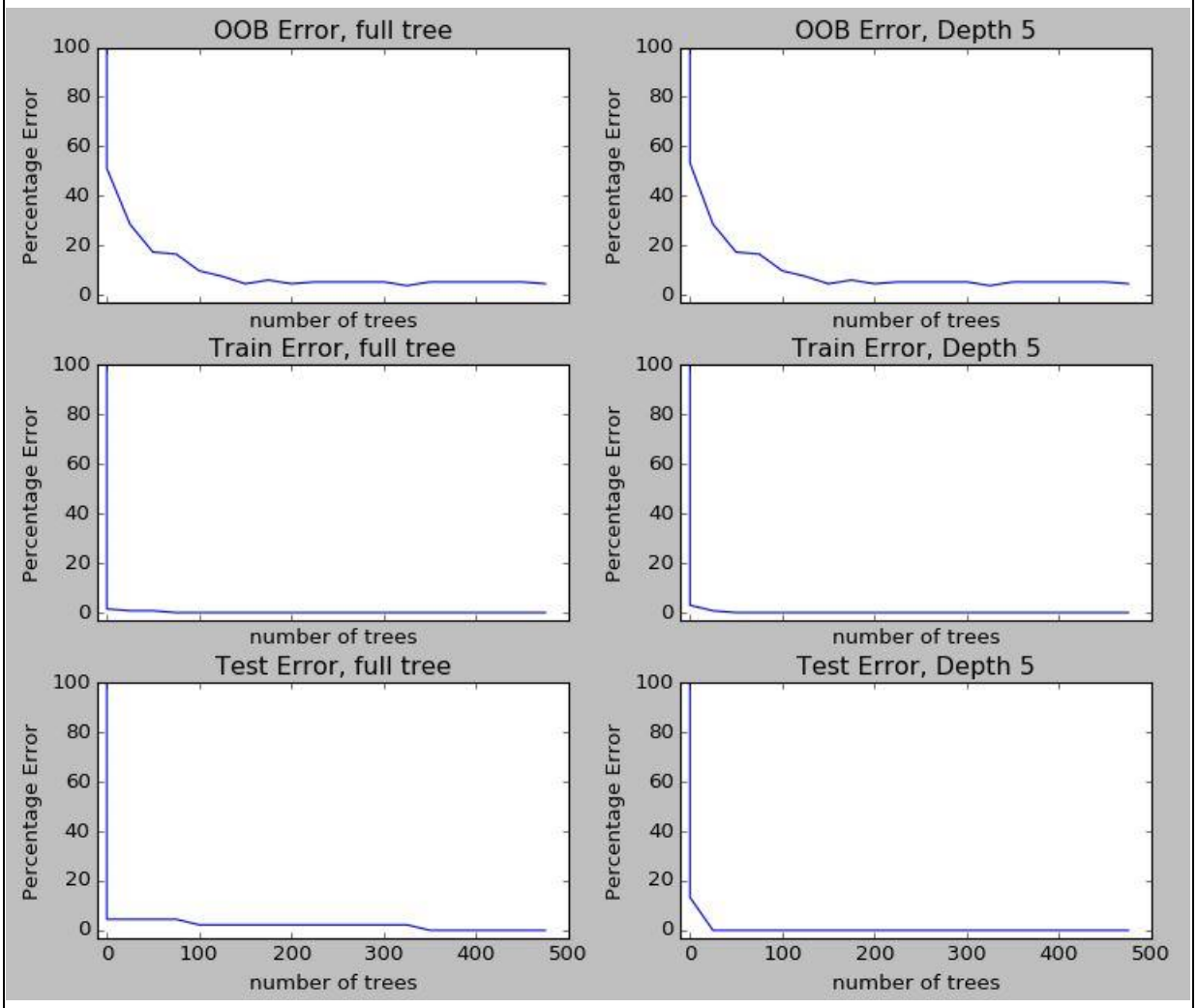
OOB Set

Confusion Matrix using Full tree
[[47 0 0]
 [1 47 3]
 [0 0 35]]
Final Accuracy Score = 0.96992481203

Confusion Matrix using Depth 5 Decision Tree
[[47 0 0]
 [1 47 3]
 [0 0 35]]
Final Accuracy Score = 0.96992481203

4. Plot of Error Percentage with respect to number of trees:

(fig. in next page)



Plot of Error Percentage with respect to number of trees

5. Analysis and Discussion:
   a. Train Error: In this execution, we observe that the train error converges to 0 almost instantly. On looking closely, we can see error of first RF with trees of Depth 5 has slightly higher error (~ 2% higher). This is because, in the full tree, we are using a classifier that fits perfectly to the training set thus giving perfect results with just a few trees. It may even be overfitting the training set. On the other hand, a tree of depth 5 fits less perfectly but converges to 0% error as we increase the number of tree.
   b. Out of Bag Error: We can see that the out of bag error has the lowest rate of convergence. This can be explained as in the first few forests with low number of trees, we are testing those samples that have not been trained ever, or enough number of times. Thus as we increase number of trees, there is more probability of every sample showing up in at least one of our bootstrap samples. Thus the OOB error gradually reduces with increasing number of trees.

c. Test Error: For Test error, we can see that the full depth tree gives a higher error that Depth 5 tree. As discussed above, this is due to overfitting of classifier to training data. Thus, although a full tree performs better in training data classification, it is not able to perform as well for data it has never seen before. It eventually converges and plateaus out as we generate around 350 trees. The RF with Depth 5 trees on the other hand converges much faster (within 50 trees). Thus we see that overfitting data can cost us higher number of computations.

ii. MNIST Dataset

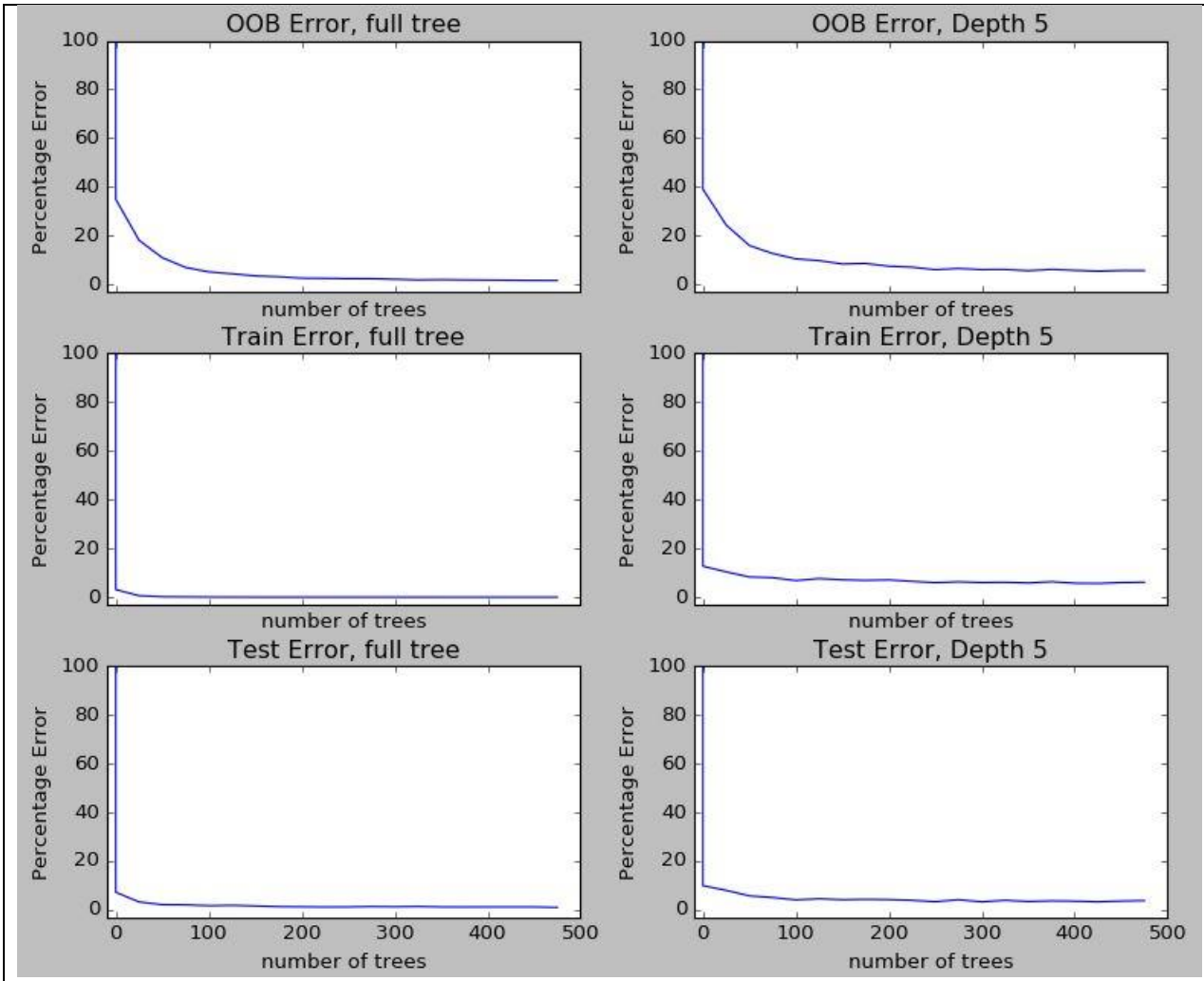
1. The data has been provided in the form of train and test sets. Thus, no further splitting is done.
2. We have two types of Random Forests, one in which each tree as stump (depth 1), and another which is restricted to depth 10. We have executed 500 random forests, each with number of trees increasing by 10. Thus, first random forest has 10 trees, second has 20, third has 30, and so on. As generating 500 random forests is time taking, we generate one random forest every 25 steps, and use interpolation to estimate the others. These values have been chosen as here we can see a smooth decreasing error percentage curve. For higher number of random forests, or higher number trees per forest the curve converges rapidly.
3. Confusion Matrix, and Accuracy:

```
Train Set

Confusion Matrix using Full tree
[[5923    0    0    0]
 [   0 6742    0    0]
 [   0    0 6131    0]
 [   0    0    0 5421]]
Final Accuracy Score =  1.0

Confusion Matrix using Depth 5 Decision Tree
[[5792    2   83   46]
 [   0 6582   75   85]
 [  22  129 5716  264]
 [  86   86  243 5006]]
Final Accuracy Score =  0.913710203576
----------

Test Set

Confusion Matrix using Full tree
[[ 977    0    1    2]
 [   0 1129    5    1]
 [   1    0  999   10]
 [   7    3   10  872]]
```

```
Final Accuracy Score =  0.990540204132

Test Set Confusion Matrix using Depth 5 Decision Tree
[[ 965    0    4   11]
 [   0 1117   13    5]
 [   8    9  954   39]
 [  16   12   45  819]]
Final Accuracy Score =  0.959671396565
--------------------

OOB Set

Confusion Matrix using Full tree
[[1585  104 1373 2861]
 [   0 6572   85   85]
 [   0  146 5673  312]
 [   0   89  395 4937]]
Final Accuracy Score =  0.925259115497

Confusion Matrix using Depth 5 Decision Tree
[[1585  104 1373 2861]
 [   0 6463   185   94]
 [   0  146 5626  364]
 [   0   99  408 4914]]
Final Accuracy Score =  0.890490977413
```

4. Plot of Error Percentage with respect to number of trees:

(fig. in next page)



Plot of Error Percentage with respect to number of trees

5. Analysis and Discussion:
   a. Train Error: In this execution, we observe that the train error converges to 0 almost instantly for the full tree. On looking closely, we can see error of first RF with trees of Depth 5 has slightly higher error (~ 2% higher). This is because, in the full tree, we are using a classifier that fits perfectly to the training set thus giving perfect results with just a few trees. On the other hand, a tree of depth 5 fits less perfectly. As the number of features is much higher, the depth limited tree is unable to perform as well as a tree of larger depth.
   b. Out of Bag Error: We can see that the out of bag error has the lowest rate of convergence. This can be explained as in the first few forests with low number of trees, we are testing those samples that have not been trained ever, or enough number of times. Thus as we increase number of trees, there is more probability of every sample showing up in at least one of our bootstrap samples. Thus the OOB error gradually reduces with increasing number of trees.

c. Test Error: For Test error, we can see that the full depth tree gives a lower error that Depth 5 tree. As discussed above, this is due to higher dimensionality of training data. The Depth 5 tree It eventually converges and plateaus out as we generate around 100 trees. The RF with full trees on the other hand converges much faster (within 50 trees) at to a lower error rate. As the number of features is much higher, the depth limited tree is unable to perform as well as a tree of larger depth. In the Wine dataset case, the number of features were lesser, thus there was the case of overfitting. But here, by the graph, we see that the larger dimentioned data fits better with larger tree.

II. Boosting

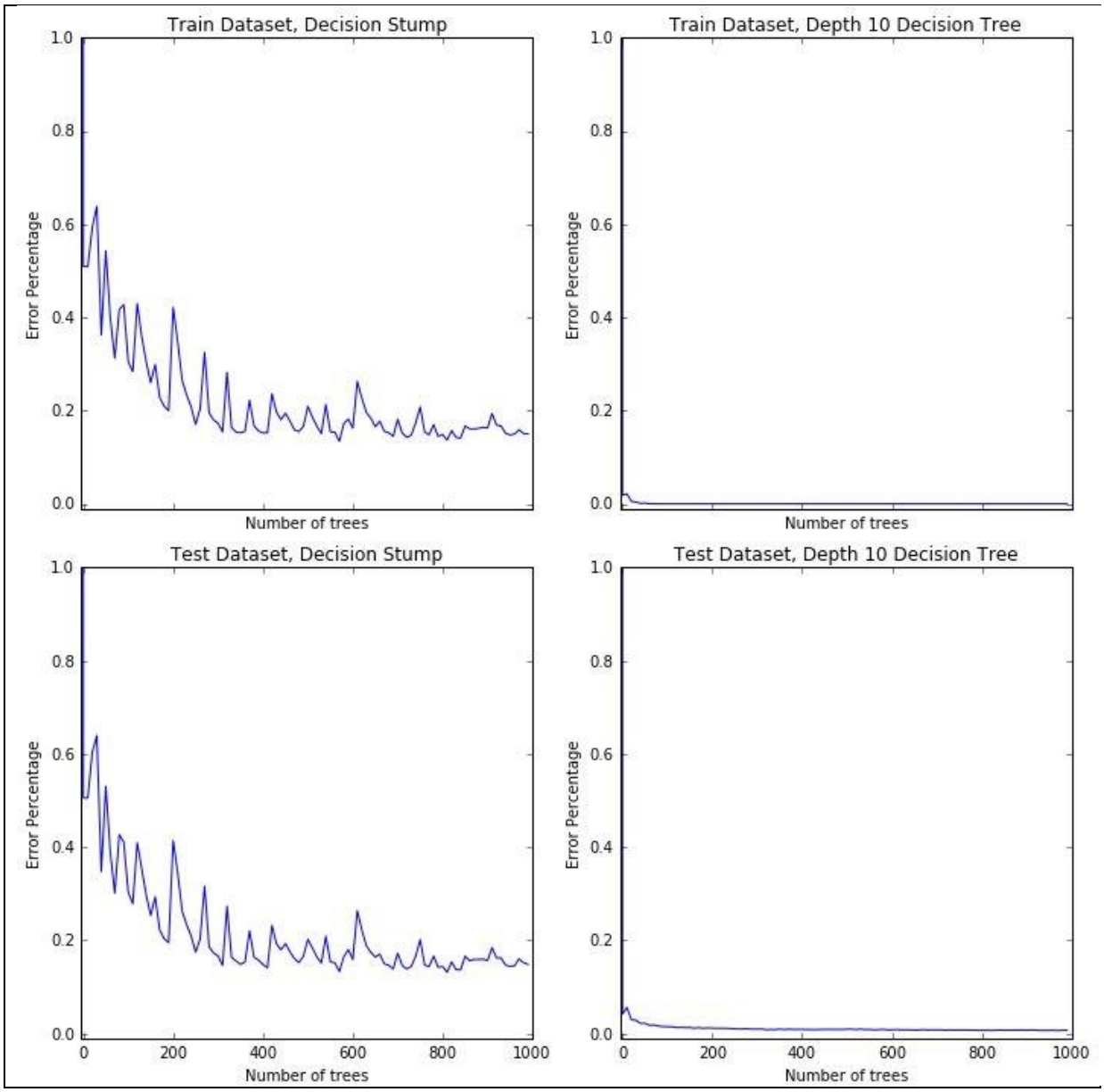
i. MNIST Data

1. The data has been provided in the form of train and test sets. Thus, no further splitting is done.
2. We have two types of AdaBoosts, one in which each tree as stump (depth 1), and another which is restricted to depth 10. We have executed 1000 iterations, with number of boosts(trees) increasing at every step. As generating 1000 iterations is time taking, we generate one classifier every 25 steps, and use interpolation to estimate the others. These values have been chosen as here we can see a decreasing error percentage curve which plateaus. We have implemented the SAMME AdaBoost algorithm here, and used in inbuilt library Sklearn's DecisionTreeClassifier for tree.
3. Confusion Matrix, and Accuracy:

```
Test Set Confusion Matrix using Decision Stump
[[5458    2   52  411]
 [   2 6185  453  102]
 [ 181   60 5526  364]
 [ 324  196 1495 3406]]
Final Accuracy Score =  0.849609778255

Test Set Confusion Matrix using Depth 10 Decision Tree
[[5923    0    0    0]
 [   0 6742    0    0]
 [   0    0 6131    0]
 [   0    0    0 5421]]
Final Accuracy Score =  1.0
```

4. Plot of Error Percentage with respect to number of trees:

(fig. in next page)

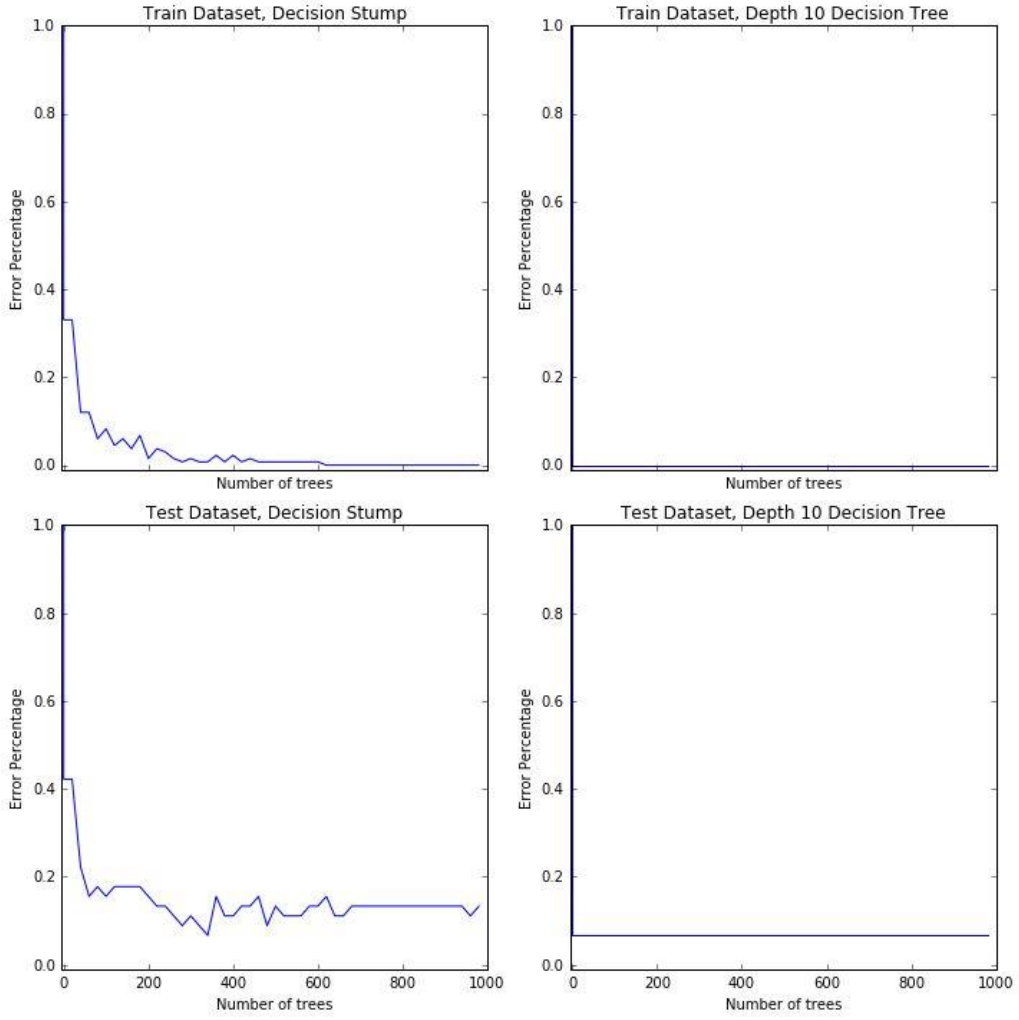


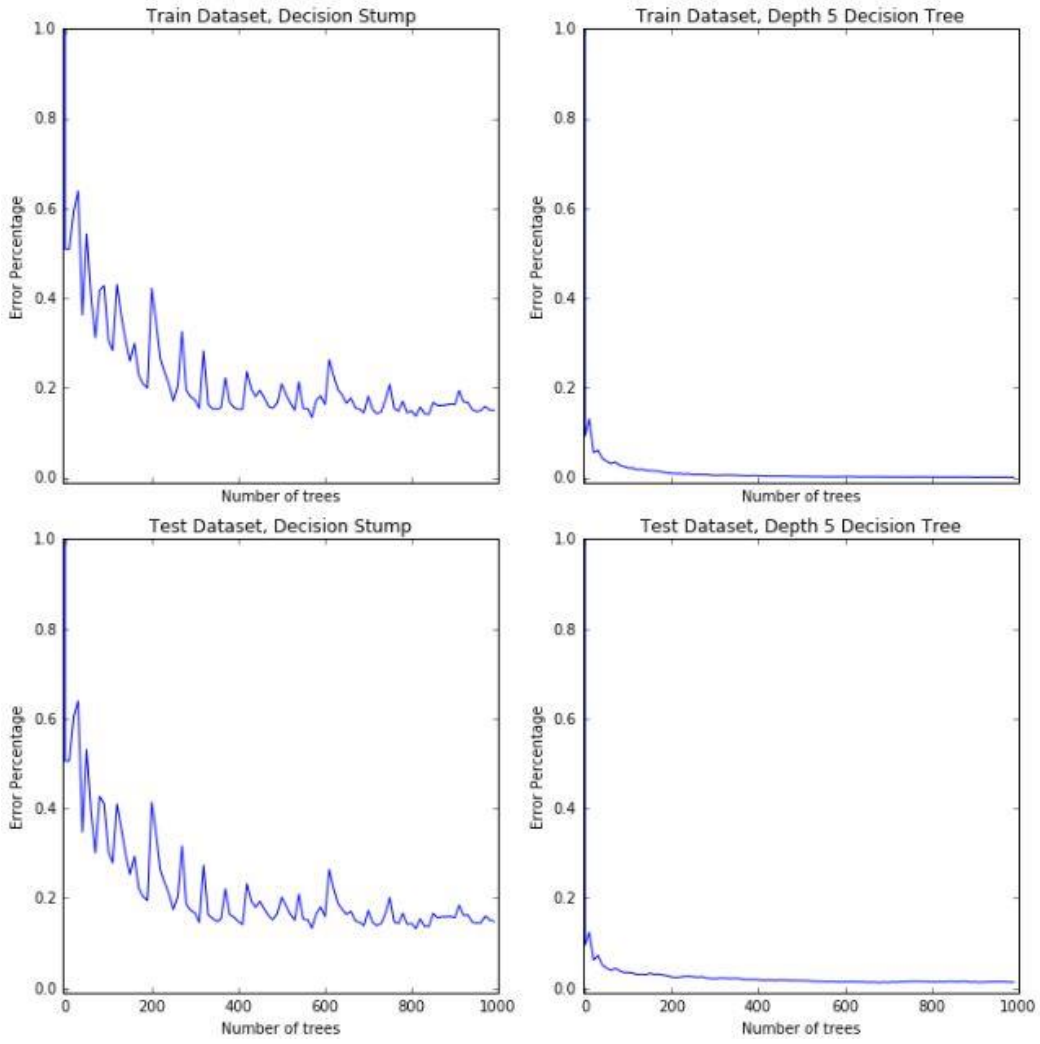
5. Analysis and Discussion:
   a. Train Error: In this execution, the most striking is the major difference between the plots for decision stump, and depth 10 decision tree. We observe that the train error for depth 10 converges to 0 almost instantly, whereas it starts to settle down around 800 iterations at a plateau. This high inconsistency of decision stump is mainly due to the high dimensionality of the dataset. (For confirmation, I ran the same code, with same number of iterations for the wine data set, and got much less skewed results). The reasoning behind this can be that the deeper tree fits much better than a sump for a given number of iterations. The stump classifies based on just one feature, whereas, a tree of depth 10 has 9 additional classifying points.

b. Test Error: For Test error, the the classifier behaves the same way as train error, ie. The weak classifier gives a highly inconsistent plot, and the depth 10 tree converges to near zero almost instantly. The only small difference we see here is that the test error is slightly (~1%) higher than train error. This may be due to overfitting of the train data. If we ran for a full tree, we can guess that the test error would be higher. In between the weak classifier and over fitted data, we may get a good fit for a classifier of depth 5 maybe.

6. Additional Graphs:

a) AdaBoost for WineDataset: Here we see that for the same number of iterations, and step, we see a much less skewed plot. Thus out fluctuations in MNIST dataset graph for decision stump is due to curse of dimensionality.





b) MNIST Dataset: Comparison between Stump and tree of depth 5.

ii. Office Data

1. The data has been provided in the form of train and test sets. Thus, no further splitting is done.
2. We have two types of AdaBoost classifiers, one in which each tree as stump (depth 1), and another which is restricted to depth 10. We have executed 1000 iterations, with number of boosts(trees) increasing at every step. As generating 1000 iterations is time taking, we generate one classifier every 25 steps, and use interpolation to estimate the others. These values have been chosen as here we can see a decreasing error percentage curve that plateaus. We have implemented the SAMME AdaBoost algorithm here, and used inbuilt library Sklearn's DecisionTreeClassifier for tree.

3. Confusion Matrix, and Accuracy:

Office Data Adaboost
Test Set Confusion Matrix using Decision Stump

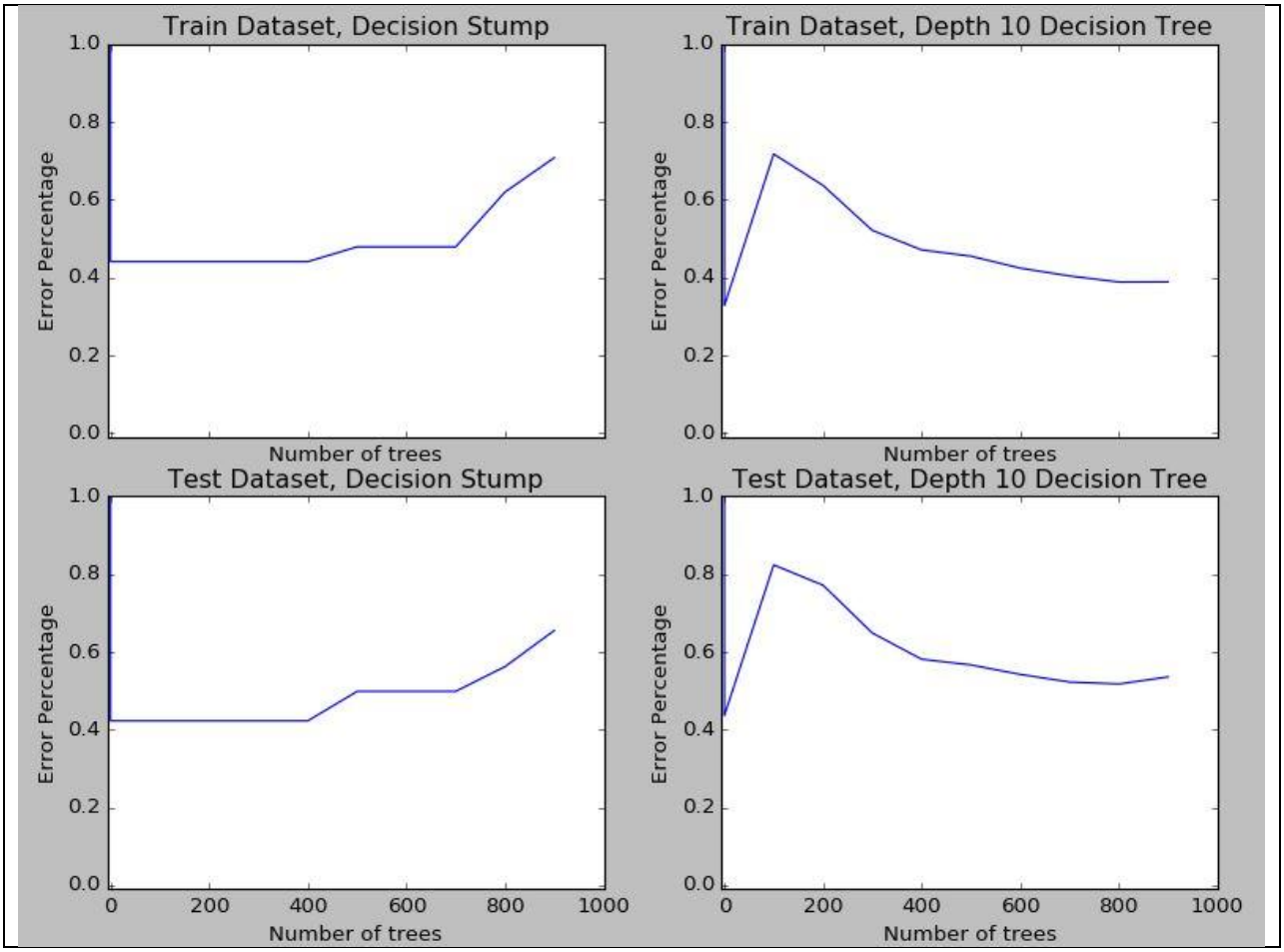
```
[[10987    0    0 7154    0    0    0    0    0 10543    0    0    0    0    0 7944    0 3468    0]
 [ 195    0    0   54    0    0    0    0    0   53    0    0    0    0    0  142    0   17    0]
 [1421    0    0  245    0    0    0    0    0 1116    0    0    0    0    0  739    0  274    0]
 [ 163    0    0 4430    0    0    0    0    0   30    0    0    0    0    0   51    0  287    0]
 [ 539    0    0    1    0    0    0    0    0 1974    0    0    0    0    0  795    0   12    0]
 [ 445    0    0    4    0    0    0    0    0  324    0    0    0    0    0   30    0   45    0]
 [ 165    0    0    0    0    0    0    0    0   24    0    0    0    0    0  154    0    7    0]
 [ 427    0    0  136    0    0    0    0    0   62    0    0    0    0    0  112    0   85    0]
 [  67    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    2    0]
 [  58    0    0    0    0    0    0    0    0 2401    0    0    0    0    0  334    0    2    0]
 [   4    0    0    0    0    0    0    0    0    1    0    0    0    0    0   96    0    0    0]
 [1363    0    0   53    0    0    0    0    0  104    0    0    0    0    0  126    0  139    0]
 [  13    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1    0    5    0]
 [ 327    0    0  150    0    0    0    0    0    0    0    0    0    0    0    4    0  232    0]
 [  11    0    0   15    0    0    0    0    0    0    0    0    0    0    0    0    0   51    0]
 [ 310    0    0    4    0    0    0    0    0 1136    0    0    0    0    0 1613    0   12    0]
 [ 119    0    0    0    0    0    0    0    0  109    0    0    0    0    0  112    0    0    0]
 [2596    0    0 1957    0    0    0    0    0  281    0    0    0    0    0  333    0 1495    0]
 [ 622    0    0  439    0    0    0    0    0  140    0    0    0    0    0  100    0  128    0]]
Final Accuracy Score =  0.291777632148
```

Office Data Adaboost
Test Set Confusion Matrix using Depth 10 Decision Tree

```
[[30387   27 1343 1797 1141   84   18  187    4  687    0  454    0  204    8  683   24 2962   86]
 [ 258   94   23    7   10    0    0    2    0    4    0    7    0    4    0   34    0   18    0]
 [1866    3 1398   14   98   11    1   19    0   94    0   50    0    3    0   69    0  163    6]
 [1187    1   39 3210    3    0    0   11    0    0    0    4    0   25    1    0    0  444   36]
 [1909    3  125    5 1047    3    0    4    0   89    0   34    0    1    0   47    1   52    1]
 [ 407    0   44    1   17  225    2    2    0    7    0  117    0    0    0    5    0   20    1]
 [ 232    0    8    3    4    0   64    2    0    1    0    1    0    2    0   17    0   16    0]
 [ 359    0   50   21   11    0    0  258    1    9    0    4    0    2    0    6    0   98    3]
 [  39    1    0    0    0    0    0    1   17    0    0    1    0    3    0    0    0    7    0]
 [1380    0   99    0  139    4    0    5    0 1115    0    1    0    0    0   32    2   16    2]
 [  61    0    5    0   12    0    0    0    0    0   17    1    0    0    0    4    0    1    0]
 [ 645    3   72   11   10   65    0    6    0    8    0  855    0    4    0   13    0   93    0]
 [   6    0    3    0    1    0    0    0    0    0    0    3    1    2    0    1    0    2    0]
 [ 288    1   13   17    0    0    0    4    1    0    0    2    0  274    0    0    0  112    1]
 [  59    0    0    1    0    0    0    0    0    0    0    0    0    2    6    0    0    8    1]
 [1570    9   37    1   42    2    3    1    0   56    0   11    0    3    0 1277    2   60    1]
 [ 226    0   10    0   10    5    0    2    0   11    0    6    0    3    0    4   40   23    0]
 [2896    3  180  368   49    5    2   60    1   19    0   49    0   95    0   37    0 2857   41]
```

```
[ 456   0  20 102   8   0   1   4   0   7   0   7   0   4   0   7   2 124 687]]
Final Accuracy Score =  0.611121181277
```

4. P lot of Error Percentage with respect to number of trees:



5. Analysis and Discussion:

a. Train Error: In this execution, the most striking is the higher error percentage. Also, the major difference between the plots for decision stump, and depth 10 decision tree is that the former gives an increasing error with more number if trees (iterations), and the latter gives a plots that starts to decrease after a while. The increase in error percentage after certain time can be accounted to pollution or overcrowding of data. Beyond a certain number of trees, the classifier has more opportunity to misclassify, or rather give higher weights to the misclassified points due to the background. Thus, along with higher chances of misclassification, we also see a higher penalty for each misclassified data point. Thus beyond 400 trees, the data starts to show greater error.

b. Test Error: For Test error, the the classifier behaves the same way as train error. The higher error in graph may be due to large background presence; hence it is misclassifying many things to background. Due to large training set, the execution time in quite long. To get better results, we can implement PCA and LDA to make the data more classifiable, and easier to handle.