Anthony Tummillo

HW Report #10

Problem 1.

Part a.)

Base:

train_error_base:
0
test_error_base:
0.12857

Bag:

avg_train_error_bag for T: 2
0.088077
avg_test_error_bag for T: 2
0.15357

avg_train_error_bag for T: 3
0.051923
avg_test_error_bag for T: 3
0.175

avg_train_error_bag for T: 4
0.059231
avg_test_error_bag for T: 4
0.14071

avg_train_error_bag for T: 5
0.043462
avg_test_error_bag for T: 5
0.15357

avg_train_error_bag for T: 6
0.033077
avg_test_error_bag for T: 6
0.13929

avg_train_error_bag for T: 7
0.029231
avg_test_error_bag for T: 7
0.15429

avg_train_error_bag for T: 8
0.028846
avg_test_error_bag for T: 8
0.14214

avg_train_error_bag for T: 9
0.023077
avg_test_error_bag for T: 9
0.145

avg_train_error_bag for T: 10
0.020385
avg_test_error_bag for T: 10
0.13214

Boost:
avg_train_error_boost for T: 2
0.11038
avg_test_error_boost for T: 2
0.14857

avg_train_error_boost for T: 3
0.011154
avg_test_error_boost for T: 3
0.17857

avg_train_error_boost for T: 4
0.013462
avg_test_error_boost for T: 4
0.13929

avg_train_error_boost for T: 5
0.00038462
avg_test_error_boost for T: 5
0.15429

avg_train_error_boost for T: 6
0.00038462
avg_test_error_boost for T: 6
0.15429

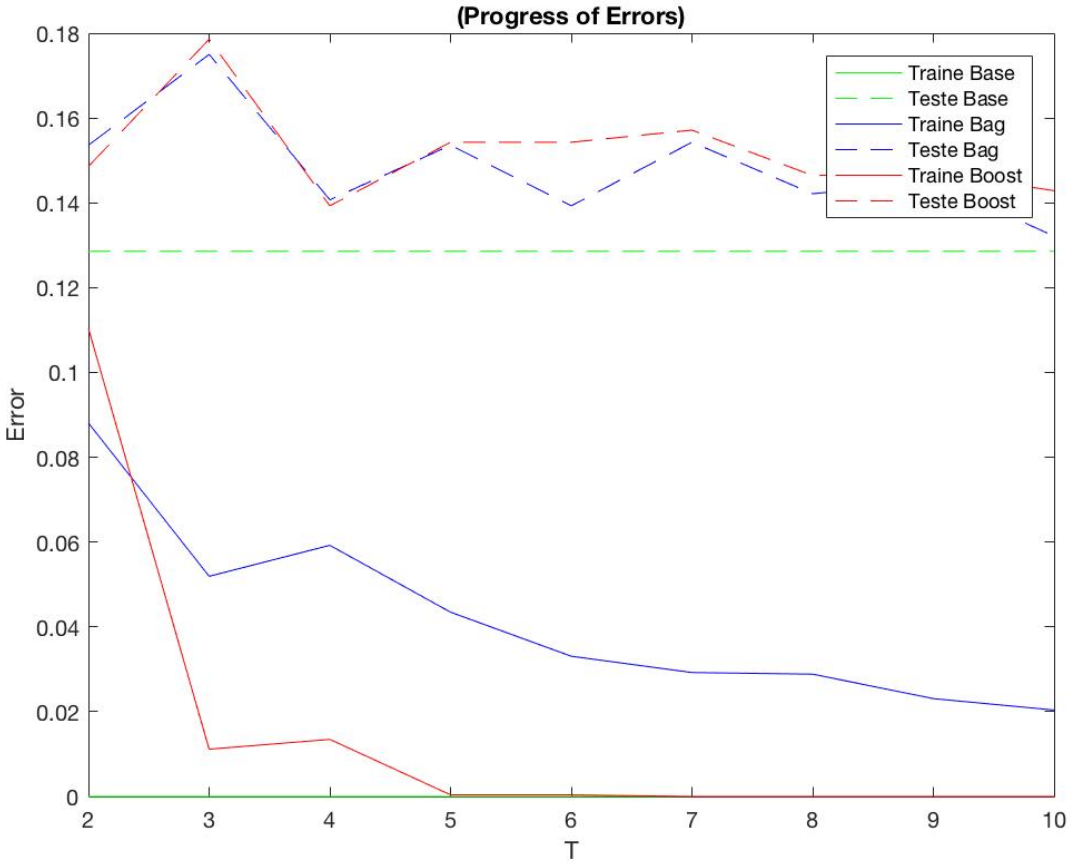
avg_train_error_boost for T: 7
0
avg_test_error_boost for T: 7
0.15714

avg_train_error_boost for T: 8
0
avg_test_error_boost for T: 8
0.14643

avg_train_error_boost for T: 9
0
avg_test_error_boost for T: 9
0.14714

avg_train_error_boost for T: 10
0
avg_test_error_boost for T: 10
0.14286

Analysis:



Here I believe our model does a good job of predicting the class overall. This is because neither bagging nor boosting offered any significant improvement on the performance realized from just the base svm model. This strikes me as odd, but for this particular data, the base SVM seems to be the best approach.

Part b.)

Base:

train_error_base:
0.030769
test_error_base:
0.25714

Bag:

avg_train_error_bag for T: 2
0.13885
avg_test_error_bag for T: 2
0.17857

avg_train_error_bag for T: 3
0.075385
avg_test_error_bag for T: 3
0.21214

avg_train_error_bag for T: 4
0.080769
avg_test_error_bag for T: 4
0.16929

avg_train_error_bag for T: 5
0.046538
avg_test_error_bag for T: 5
0.18429

avg_train_error_bag for T: 6
0.055769
avg_test_error_bag for T: 6
0.15

avg_train_error_bag for T: 7
0.029231
avg_test_error_bag for T: 7
0.17786

avg_train_error_bag for T: 8
0.032308
avg_test_error_bag for T: 8
0.14

avg_train_error_bag for T: 9
0.024615
avg_test_error_bag for T: 9
0.15214

avg_train_error_bag for T: 10
0.028846
avg_test_error_bag for T: 10
0.13929

Boost:
avg_train_error_boost for T: 2
0.14577
avg_test_error_boost for T: 2

0.185

avg_train_error_boost for T: 3
0.027692
avg_test_error_boost for T: 3
0.21429

avg_train_error_boost for T: 4
0.041538
avg_test_error_boost for T: 4
0.175

avg_train_error_boost for T: 5
0.0042308
avg_test_error_boost for T: 5
0.19071

avg_train_error_boost for T: 6
0.0026923
avg_test_error_boost for T: 6
0.15714

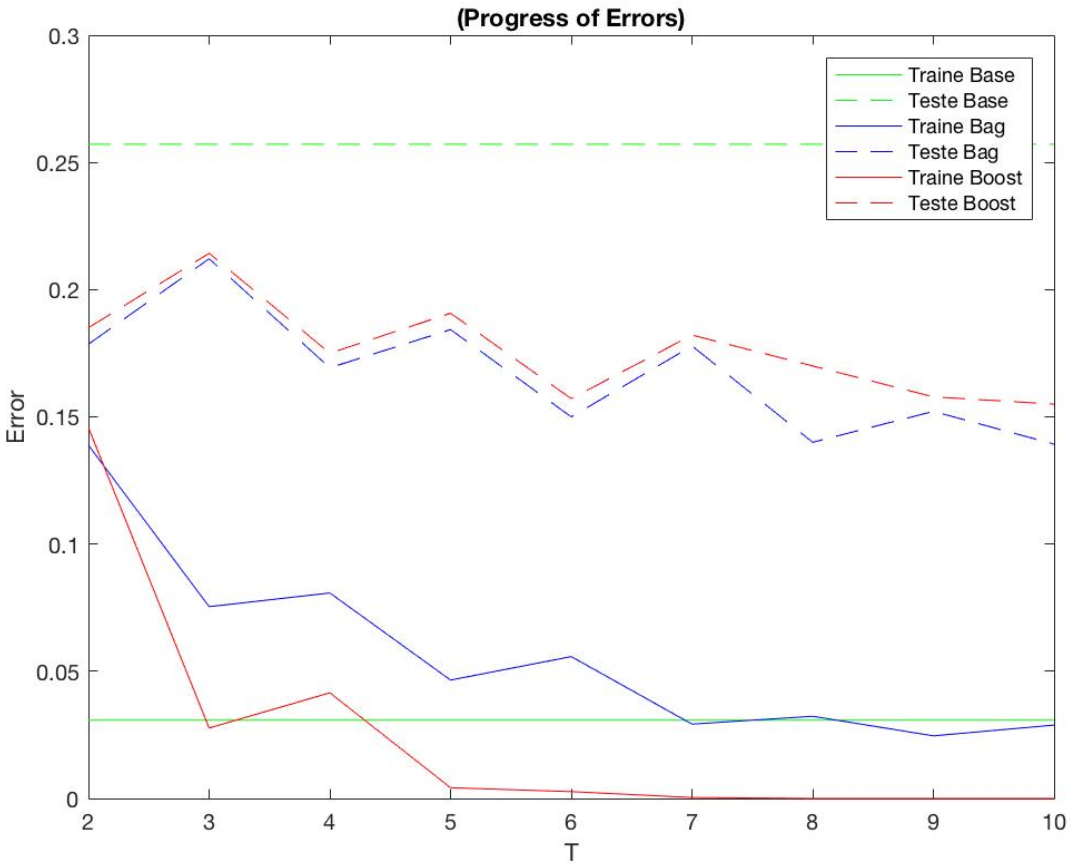
avg_train_error_boost for T: 7
0.00038462
avg_test_error_boost for T: 7
0.18214

avg_train_error_boost for T: 8
0
avg_test_error_boost for T: 8
0.17

avg_train_error_boost for T: 9
0
avg_test_error_boost for T: 9
0.15786

avg_train_error_boost for T: 10
0
avg_test_error_boost for T: 10
0.155

Analysis:



In these results it's clear that the base fully grown tree does a really good job predicting on the training data, but performs fairly poorly on the testing data. To me this seems to indicate a large amount of overfitting. The variance of the data is too high for the base model to do a good job on the test data.

Bagging and boosting are both known to perform well on data where the variance is high so I predicted that they would both improve on the test error when compared to the base model. This hypothesis is confirmed by the graph above in which the test error for both bagging and boosting decreases below that of the base model.

Part c.)

Base:
train_error_base:
0.21538
test_error_base:
0.14286

Bag:
avg_train_error_bag for T: 2

0.24808
avg_test_error_bag for T: 2
0.19214

avg_train_error_bag for T: 3
0.22462
avg_test_error_bag for T: 3
0.18929

avg_train_error_bag for T: 4
0.22731
avg_test_error_bag for T: 4
0.15929

avg_train_error_bag for T: 5
0.20192
avg_test_error_bag for T: 5
0.17357

avg_train_error_bag for T: 6
0.21385
avg_test_error_bag for T: 6
0.15143

avg_train_error_bag for T: 7
0.20654
avg_test_error_bag for T: 7
0.17286

avg_train_error_bag for T: 8
0.22308
avg_test_error_bag for T: 8
0.16143

avg_train_error_bag for T: 9
0.19692
avg_test_error_bag for T: 9
0.15286

avg_train_error_bag for T: 10
0.21154
avg_test_error_bag for T: 10
0.13714

Boost:
avg_train_error_boost for T: 2
0.24385
avg_test_error_boost for T: 2
0.17

avg_train_error_boost for T: 3
0.25115
avg_test_error_boost for T: 3
0.16643

avg_train_error_boost for T: 4
0.20615
avg_test_error_boost for T: 4
0.16714

avg_train_error_boost for T: 5
0.19692
avg_test_error_boost for T: 5
0.15643

avg_train_error_boost for T: 6
0.17385
avg_test_error_boost for T: 6
0.16643

avg_train_error_boost for T: 7
0.16654
avg_test_error_boost for T: 7
0.15786

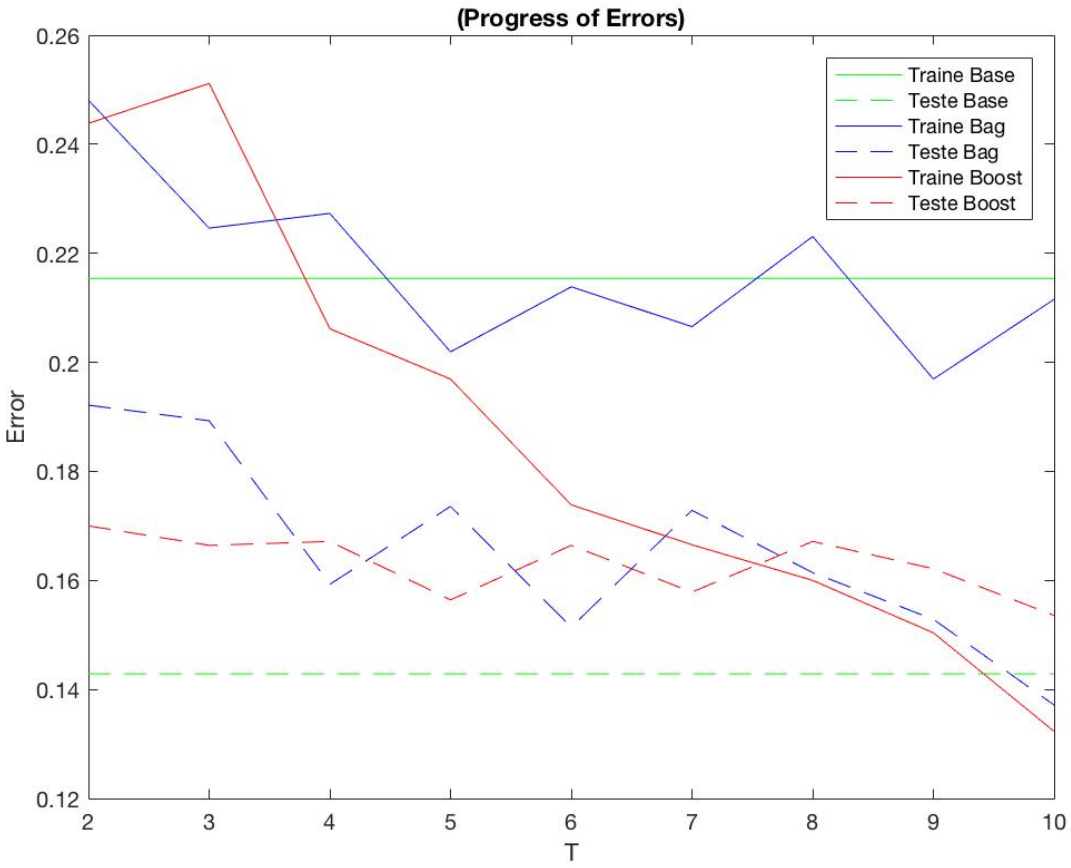
avg_train_error_boost for T: 8
0.16
avg_test_error_boost for T: 8
0.16714

avg_train_error_boost for T: 9
0.15038
avg_test_error_boost for T: 9
0.16214

avg_train_error_boost for T: 10
0.13231
avg_test_error_boost for T: 10

0.15357

Analysis:



In these results it can be observed that the base simple tree does not perform very well on the training data, but does on the testing data. I believe this is just purely by random chance that it performs so well on the test data.

I would hypothesize that there is a large amount of bias present in this model due to how simple our tree is and that bagging would not offer a significant amount of improvement over the base since bagging doesn't improve on high bias well, where as boosting should offer some more significant improvement.

Oddly bagging performed better than boosting after higher Ts, but clearly with lower Ts boosting appears to be performing better. I believe that it was by random chance that bagging ended up with the lowest test error in the end.