78bc4225 **Report for homework 2** CS1420 – Spring 2018 February 15, 2018

For chess:

average train error: 0.0

average test error: 0.042642140468227424

average train(pruned) error: 0.007

average test(pruned) error: 0.03762541806020067

average train error: 0.0

average test error: 0.013377926421404682

average train(pruned) error: 0.002

average test(pruned) error: 0.011705685618729096

average train error: 0.0

average test error: 0.014214046822742474

average train(pruned) error: 0.002

average test(pruned) error: 0.01254180602006689

For spam:

average train error: 0.014

average test error: 0.15301806997308728

average train(pruned) error: 0.028

average test(pruned) error: 0.14494425221068818

average train error: 0.014

average test error: 0.11841599384485198

average train(pruned) error: 0.02

average test(pruned) error: 0.11726259131103421

average train error: 0.014

average test error: 0.12495194156093810

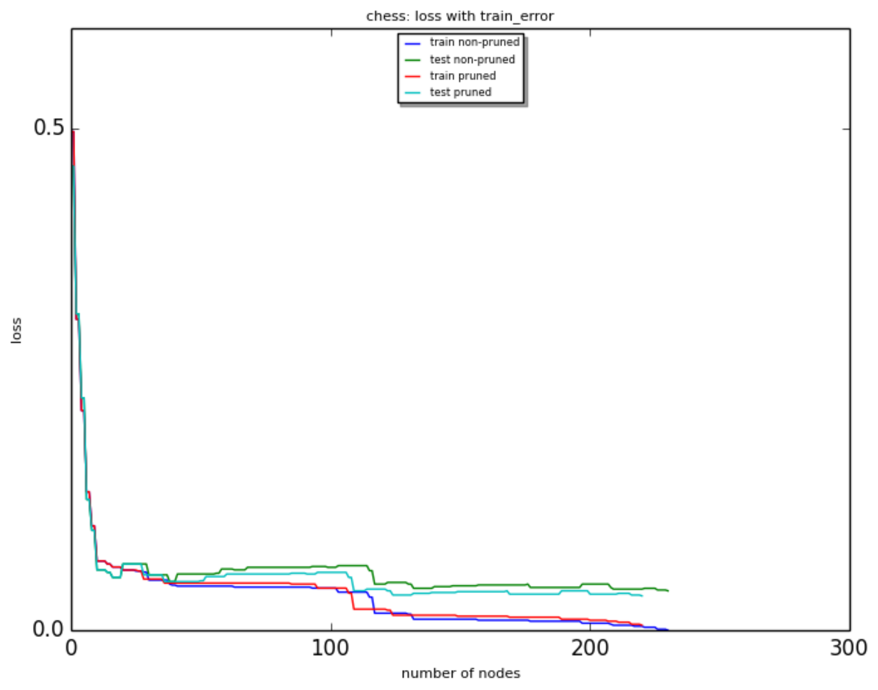
average train(pruned) error: 0.021

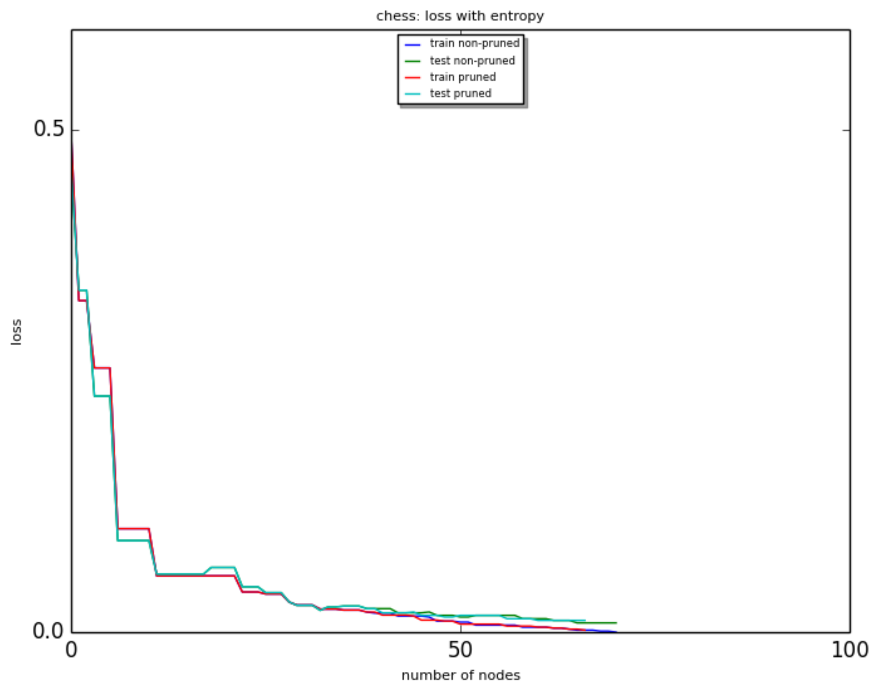
average test(pruned) error: 0.12379853902345252

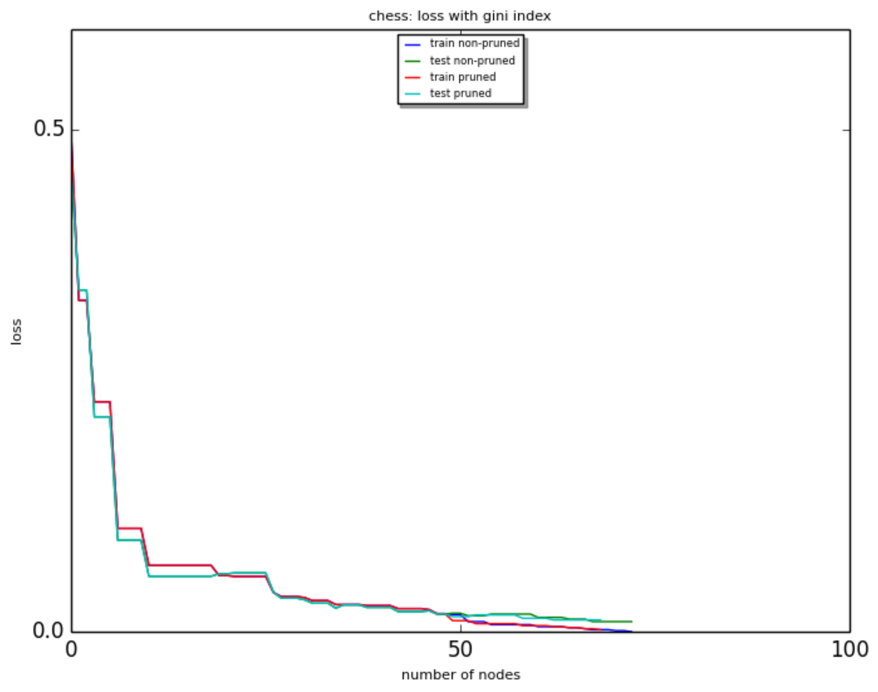
Overall, decision tree is very effective at correctly classifying data with large dimensions. However, we also noticed the overfitting problem with decision tree. The test error tends to be higher than the training error. Therefore, we used pruning to reduce the model complexity. We can see pruning helps to reduce the testing error marginally but also increases the training error. This is expected. Because the decision tree is trained on training data, it will try to reduce the training loss as much as possible. Pruning nodes from decision tree makes it less accurate to describe the training data. However, it helps to reduce the model complexity and thus can help to reduce the overfitting problem on testing data.

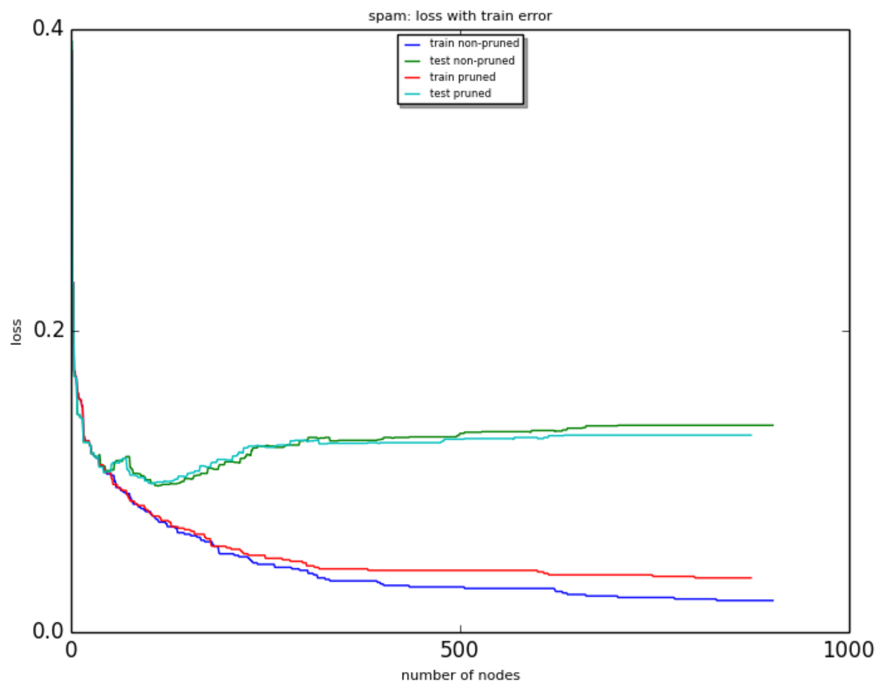
Overall, using entropy function gives the lowest loss for the two dataset. Pruning wasn't very effective when only removing the nodes with two children that are leaf node. This is because only few nodes are actually being removed. However, I also tried to remove any node that can reduce the validation loss, and this method helps to reduce more testing error.

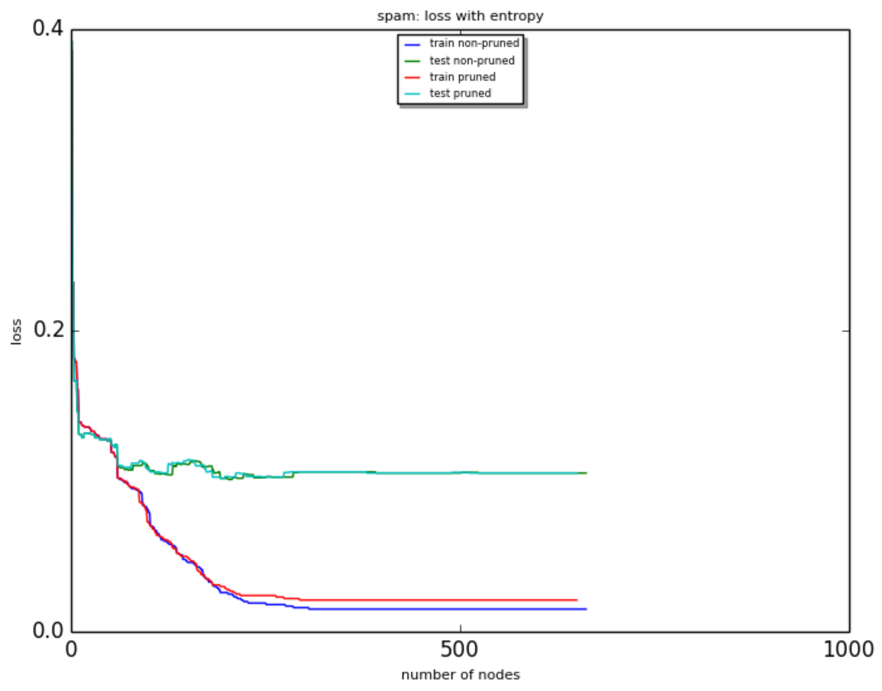
As the depth increasing, the training loss decreases as the last two figure show. Initially, all gain functions return the same loss. As the depth increases, gini index and entropy start to out-perform the train error function. This is because for train loss, there is a case when average child node loss is the same as the parent nodes loss and thus it generates 0 information gain and the split stops. However, given the bell shape of entropy and gini index function, the average child node entropy is not equal to the entropy of the parents and thus the information gain is larger than 0 and the nodes kept splitting until the child is pure.

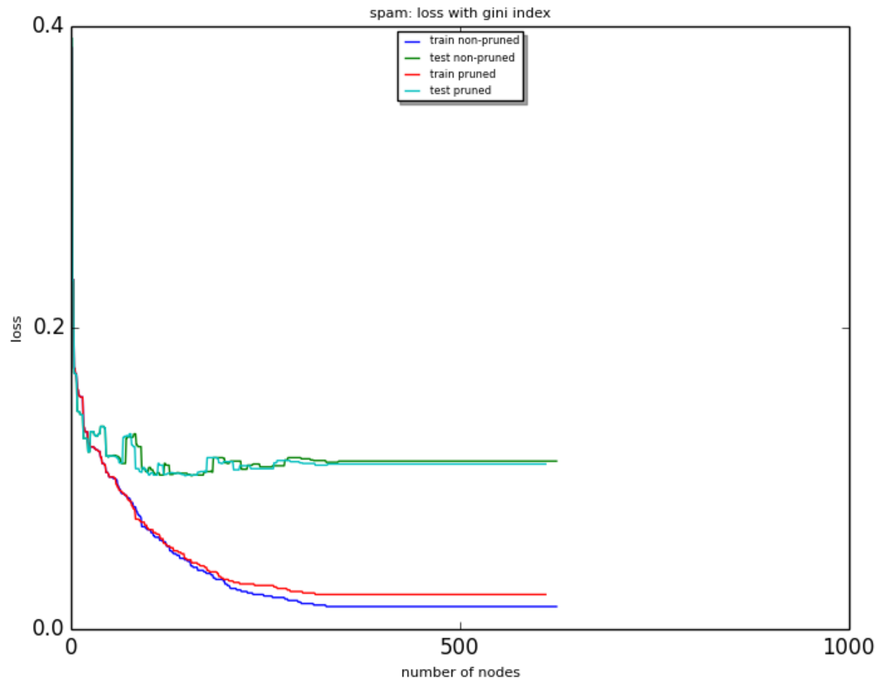


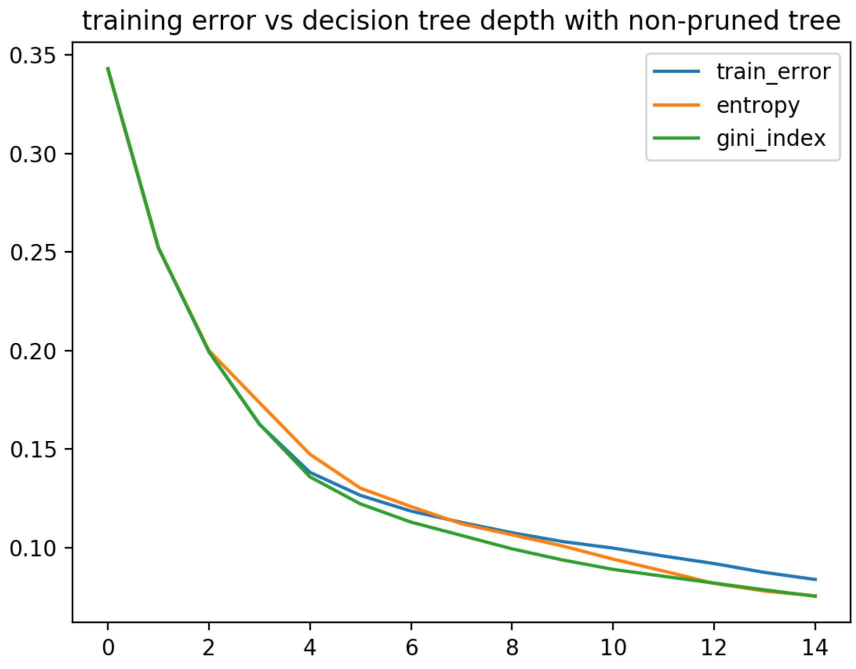

chess: loss with entropy
train non-pruned
test non-pruned
train pruned
test pruned
0.5
0.0
loss
0
50
100
number of nodes

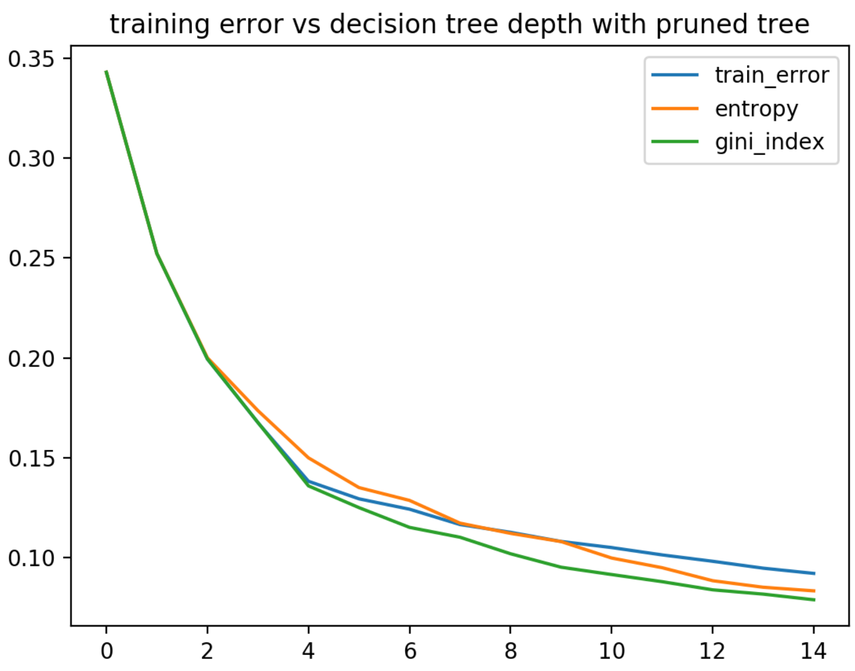

chess: loss with gini index
train non-pruned
test non-pruned
train pruned
test pruned
0.5
0.0
loss
0
50
100
number of nodes


spam: loss with train error
train non-pruned
test non-pruned
train pruned
test pruned
loss
0.0
0.2
0.4
0
500
1000
number of nodes


spam: loss with entropy
train non-pruned
test non-pruned
train pruned
test pruned
loss
0.0
0.2
0.4
0
500
1000
number of nodes


spam: loss with gini index
train non-pruned
test non-pruned
train pruned
test pruned
loss
number of nodes
0.4
0.2
0.0
0
500
1000


training error vs decision tree depth with non-pruned tree
train_error
entropy
gini_index
0.35
0.30
0.25
0.20
0.15
0.10
0
2
4
6
8
10
12
14


training error vs decision tree depth with pruned tree
train_error
entropy
gini_index
0.35
0.30
0.25
0.20
0.15
0.10
0
2
4
6
8
10
12
14