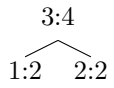
1. **Decision Trees and ID3**

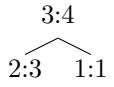
   (a) The result of splitting on A:

   The associated remainder (weighted entropy):

   $$-\left(\frac{1}{7}\ln\frac{1}{3}+\frac{2}{7}\ln\frac{2}{3}+\frac{2}{7}\ln\frac{2}{4}+\frac{2}{7}\ln\frac{2}{4}\right)\approx .669$$

   And for splitting on B:

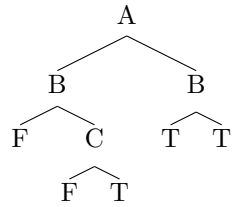
   $$-\left(\frac{2}{7}\ln\frac{2}{5}+\frac{3}{7}\ln\frac{3}{5}+\frac{1}{7}\ln\frac{1}{2}+\frac{1}{7}\ln\frac{1}{2}\right)\approx .679$$

   So splitting on A provides a result with a slightly lower remainder, and hence slightly higher information gain.

   Splitting on A may be preferable because the entropy, i.e. information required after the split, is lower. Intuitively, splitting on A might be more useful because it provides a more even separation of the data into the true and false branches, and hence a shorter tree; splitting on B might also be useful because if B is true, then that branch is completely decided.

   This shows that ID3 has inductive bias for shorter (more balanced) trees.

   (b) In the following tree, going down a left branch left indicates True, and right indicates False.

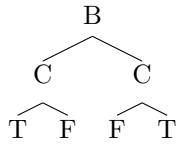
   In the first stage, splitting on A generates the lowest remainder. Then considering the other three attributes. Attributes B and C are tied for lowest remainder for both values of A; we split on B in both cases.

   If A is True and B is True, there is only one case, in which the classification is False. Otherwise, there are two cases that are exactly dependent on C.

   If A is False and B is True, there is one case, in which the classification is True. Otherwise, there are two cases that cannot be separated, in which case we arbitrarily pick True.

   6 of 7 are correctly identified by the tree.

(c) Consider the following tree, that also correctly identifies 6 out of 7. Again, going left indicates True, and going right indicates False.



We see that although the ID3 algorithm is designed to greedily try to get the shortest / simplest tree by maximizing information gain at every split, it does not always end up choosing the shortest or simplest tree, probably due to the myopic nature of the greedy algorithm.

2. **ID3 with Pruning**

(c) Since the pruned version does slightly better than the unpruned on 10-fold cross-validation, there is probably some overfitting: reducing complexity improves perfomrance. (Specifically, clean unpruned had success rate 0.87; clean pruned 0.89, noisy unpruned 0.78, noisy pruned 0.80.)

However, it is interesting to note this difference (by about the same amount) for both clean and noisy data, which indicates that generalization to noisier data is not affected by pruning, which indicates that ID3 is not overfitting to training data noise.

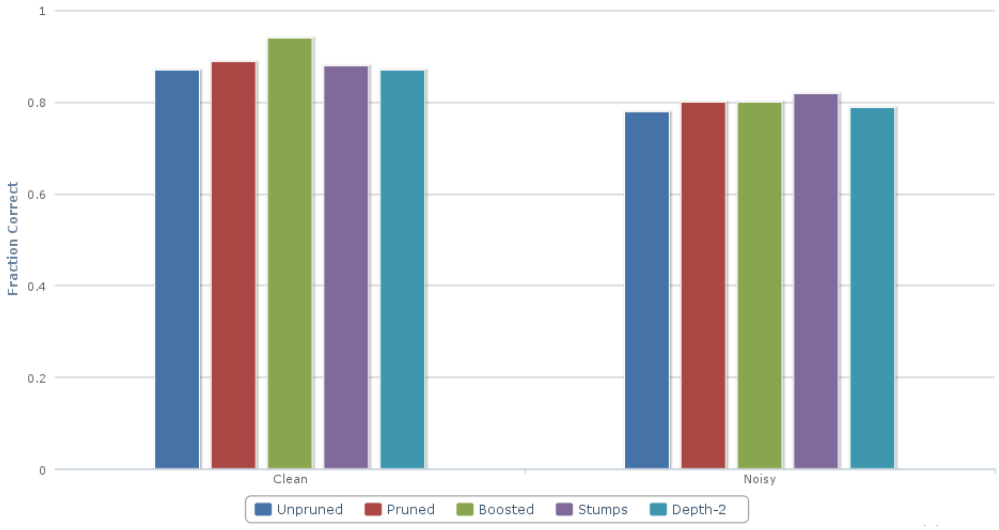
(d) i. Our implementation uses instance weight throughout. It is implemented already to calculate entropy of a split, find the majority label in a branch, evaluate correct / incorrect, and so on. Then it uses this information ***

If decisions were made only based on instance counts, there would be no change between rounds of boosting, so nothing new would happen and all the classifiers would be essentially the same. The performance of AdaBoost is based on the idea that later classifiers pay more attention to the instances that earlier ones have missed, so as to make up for their mistakes.

ii. In that case, the total weight of true and false instances are both .5, so the entropy is 1:

$$-\left(\frac{1}{2}\log_2\frac{1}{2}+\frac{1}{2}\log_2\frac{1}{2}\right)=1$$

i. Boosting seems to do noticeably better on the other methods for clean data, but not on noisy data (see figure).



The drop that boosting suffers due to noisy data is much larger than the drop that pruned trees suffer, indicating that boosting is badly particularly affected by noisy data. Perhaps this means that boosting is bad for inconsistent data; maybe the inconsistent elements, since some of them will always be wrong, tend to end up with very large weights, taking attention away from the majority of the data.

ii. Based on the figure above, we see that stumps do slightly better than depth-2 trees, which looks like overfitting.

iii. If we did not know that boosting produces a maximum-margin classifier, we would be surprised that boosting depth 1 on 10 rounds is improved upon by boosting depth 1 on 30 rounds on clean data. Perhaps we'd think that performance would taper off before that many rounds of boosting. However, the fact that it is margin-maximizing indicates that it should improve, so that it continues to approach the decision boundary. On the other hand, there is no improvement for the analogous tests on noisy data, which we can attempt to explain by the fact that with added noise, the decision boundary is less well-defined, so that further approaching the decision boundary is no longer fruitful. It is additionally surprising why this is not true for depth 2, which we can only attempt to explain as in part 2dii.

iv. More rounds of boosting training generally increases accuracy in cross-validated test sets. It appears that sometimes an extra iteration of training decreases accuracy a bit, but the trend is overall positive until the plateau, although the variation in accuracy can be pretty large (i.e., boosting 4 times is the same as boosting 10 times, though the peaks in between go up).

3. Consider the unpruned tree generated by ID3 from the BCW data:

```
attr = 4
    1 False
    2 False
    3 attr = 9
         1 False
```

```
     2 False
     3 True
     5 False
     8 True
4 attr = 6
     2 True
     3 True
     4 False
5 True
6 attr = 2
     5 True
     7 True
     8 False
7 True
8 True
9 True
10 True
```

`False` and `True` represent leaves with those labels; `attr = n` indicates an internal node that splits on attribute number n (according to the numbering in the data file); its children are listed immediately after it. The number before any of those indicates the attribute value of the parent node that leads to that node.

From this, it looks like attribute 4 (uniformity of cell shape) is the most important attribute in determining malignancy. In fact, it looks like malignancy is more likely with higher attribute values: the two lowest values branch straight to `False` leaves, while the four highest go to `True`. Meanwhile, the first child node has two `False` leaves and three `True` ones, while the other two Indeed, looking over the original data set gives the following statistics:

malignant?	attribute value	frequency
1	0	42
2	0	5
3	0	5
3	1	2
4	0	2
4	1	4
5	1	8
6	0	1
6	1	4
7	1	5
8	1	7
9	1	1
10	1	14

It can be clearly seen that there is quite a good correlation; a stump that split on this one attribute would achieve 95% accuracy.

The three children nodes split on the attributes normal nucleoli, single epithelial cell size, and clump thickness.