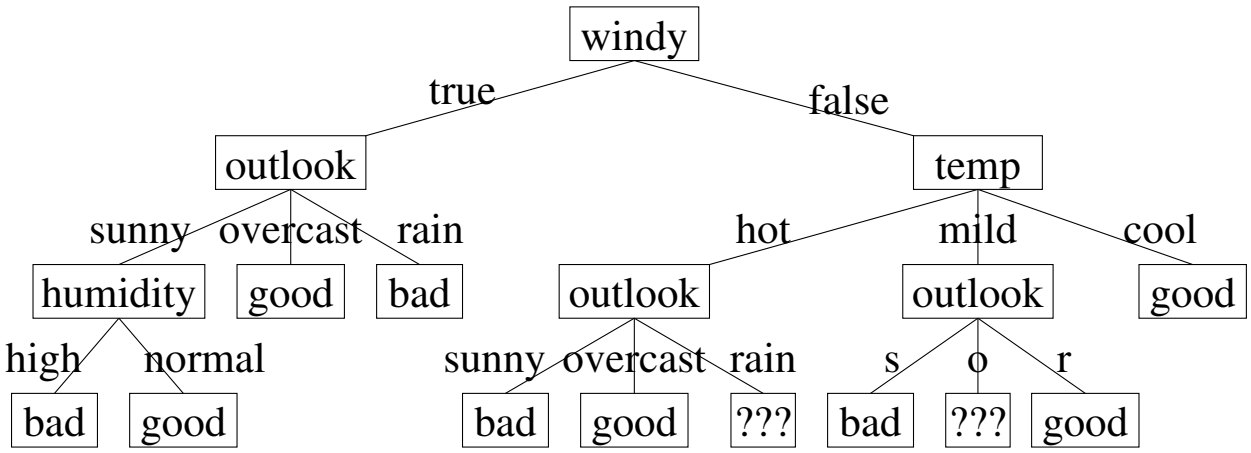
Decision Trees

Decision trees are a representation for classification.

- The root is labelled by an attribute.
- Edges are labeled by attribute values.
- Edges go to decision trees or leaves.
- Each leaf is labelled by a class.



TDIDT: Top-Down Induction of Decision Trees

Growth Phase: The tree is constructed top-down.

- Find the "best" attribute.
- Partition examples based on the attribute's values.
- Apply the method to each partition.

Pruning Phase: The tree is pruned bottom-up.

- For each node, keep subtree or change to leaf.
- Choose by comparing estimated error.

Algorithm for Growing Decision Trees

GROW-DT(*examples*)

1. $N \leftarrow$ a new node
2. $N.class \leftarrow$ most common class in *examples*
3. $N.test \leftarrow$ best attribute (or test)
4. **if** $N.test$ is not good enough
5. **then** mark N as a leaf and **return** N
6. **for** each value v_j of $N.test$
7. $examples_j \leftarrow$ *examples* with $N.test = v_j$
8. **if** $examples_j$ is empty
9. **then** $N.branch_j \leftarrow N.class$
10. **else** $N.branch_j \leftarrow$ GROW-DT($examples_j$)
11. **return** N

Measuring Information

Information gain is a popular way to select an attribute. Let $I(p, n)$ be the information in p positive examples and n negative examples.

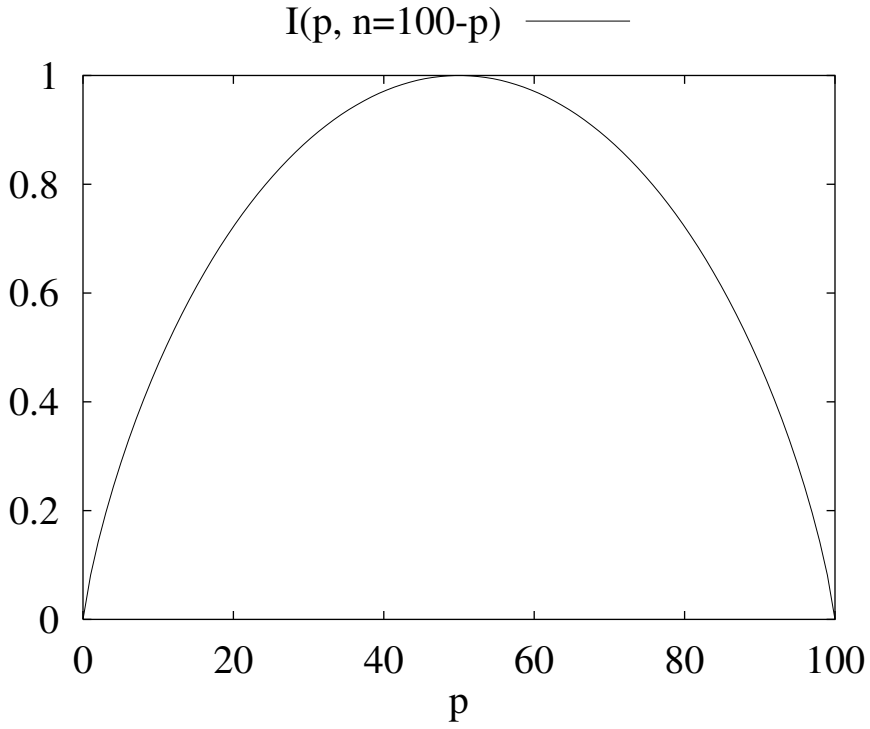
$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Suppose there are p_i positive and n_i negative examples for the ith value of an attribute. Then information gain $G(p_1, n_1, p_2, n_2)$ can be defined as:

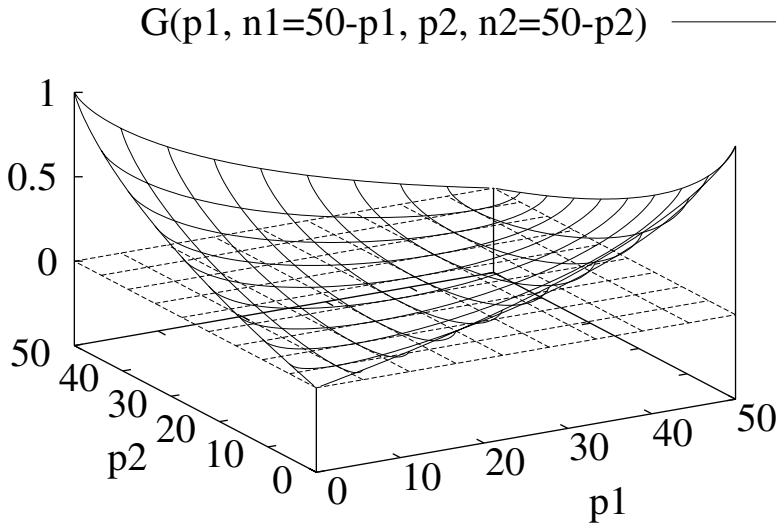
$$G(p_1,n_1,p_2,n_2) = I(p_1+p_2, n_1+n_2) =$$

$$-\frac{p_1+n_1}{p_1+n_1+p_2+n_2}I(p_1,n_1) - \frac{p_2+n_2}{p_1+n_1+p_2+n_2}I(p_2,n_2)$$

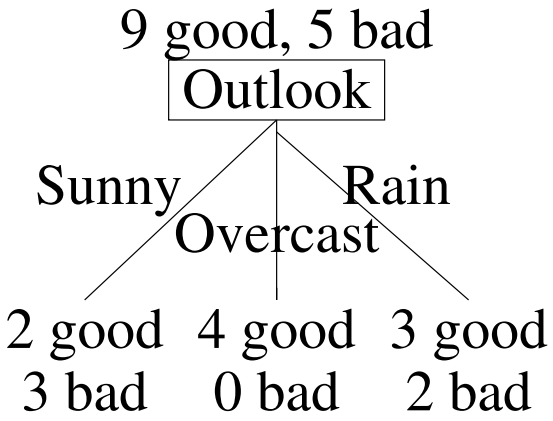
This graph shows $I(p, n)$ assuming $p + n = 100$.



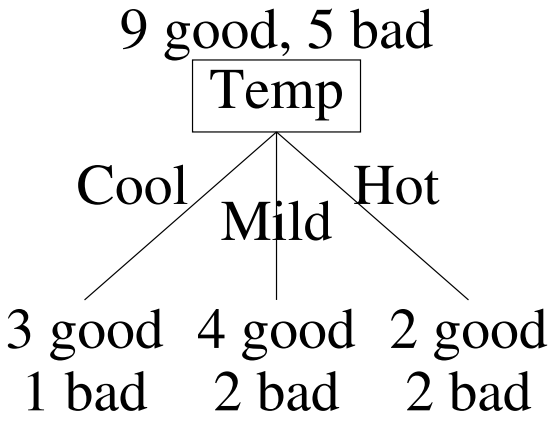
This graph shows $G(p_1, n_1, p_2, n_2)$ assuming $p_1 + n_1 = 50$ and $p_2 + n_2 = 50$.



Example of Attribute Selection

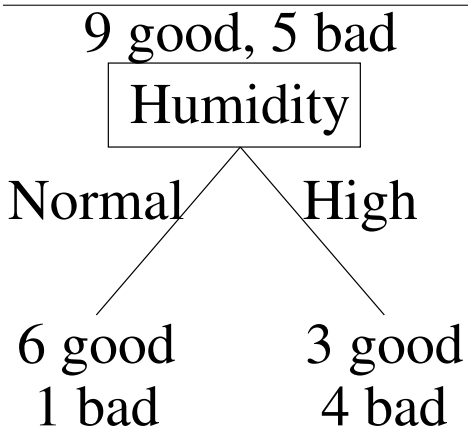


$G(\text{Outlook}) \approx 0.246$

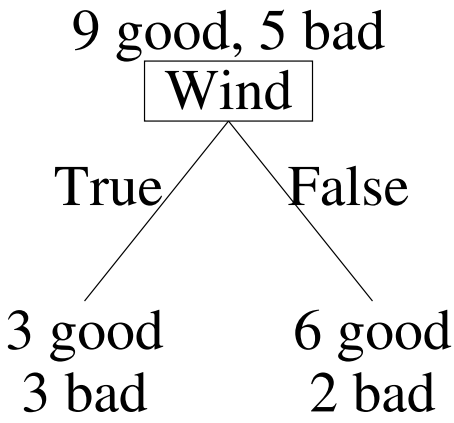


$G(\text{Temp}) \approx 0.029$

b



$G(\text{Humidity}) \approx 0.152$



$G(\text{Wind}) \approx 0.048$

Outlook has the highest gain.

Overcast branch is pure.

Need to construct DTs for Sunny and Rain branches.

Comments on Growing Decision Trees

Implicit preference for small trees.

Handling numeric attributes:
Find the test $\leq x$ that maximizes gain.

Handling missing values: Alternatives:
Treat missing values as separate values.
Weight example across branches.

Addressing "costs" of attributes:

Attribute might have different costs to obtain.

Include cost in attribute measure.

There are alternative attribute measures.

- Information Gain Ratio (for > 2 branches)
  $G(A)/I(p_1 + n_1, \ldots, p_v + n_v)$
- Gini Index (use this in place of I)
  $$Gini(p, n) = 1 - \left(\frac{p}{p+n}\right)^2 - \left(\frac{n}{p+n}\right)^2$$
- Chi-Squared Statistic
  $$\chi^2 = \sum_{j=1}^{v} \frac{(p_j - p\,s_j)^2}{p\,s_j} + \frac{(n_j - n\,s_j)^2}{n\,s_j}$$
  where $s_j = (p_j + n_j)/(p + n)$

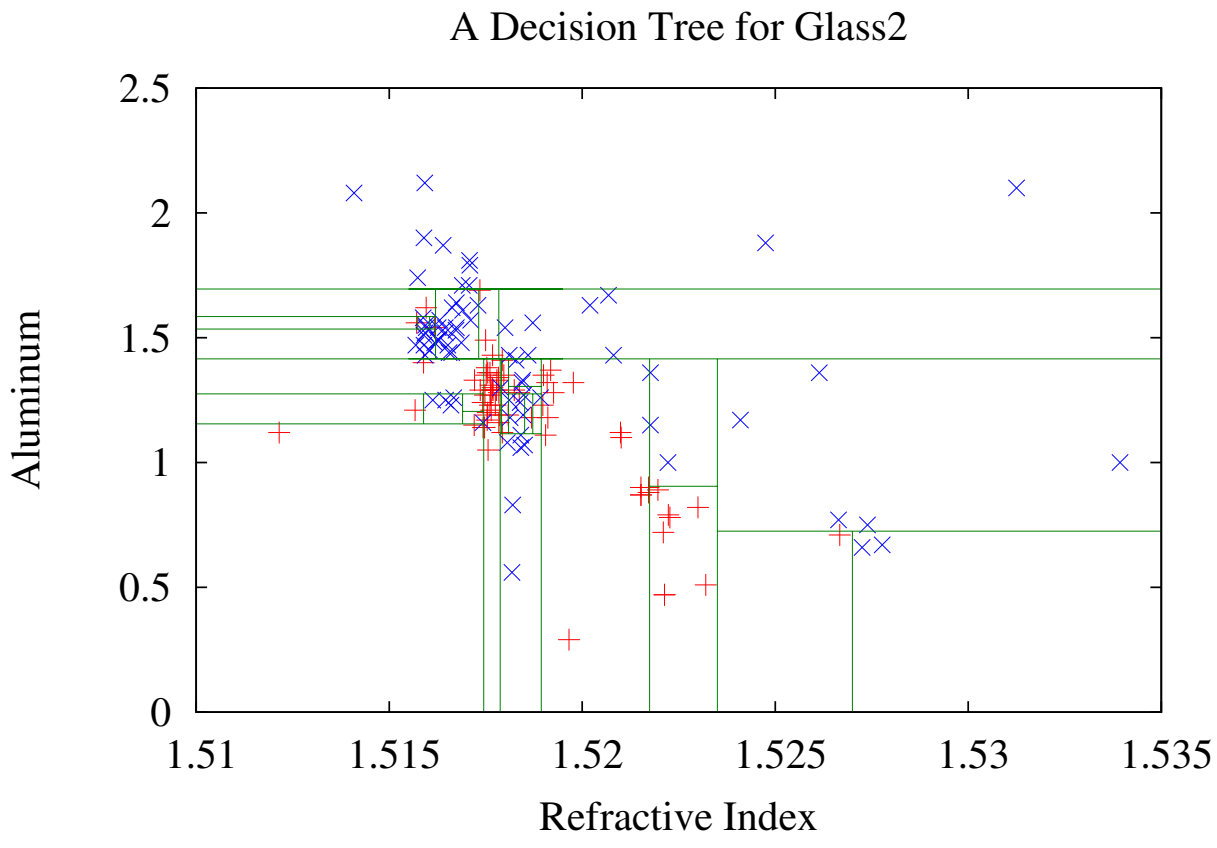
Overfitting

A hypothesis h overfits the training data if there is a hypothesis h' that is worse on the training data, but better over the whole distribution.

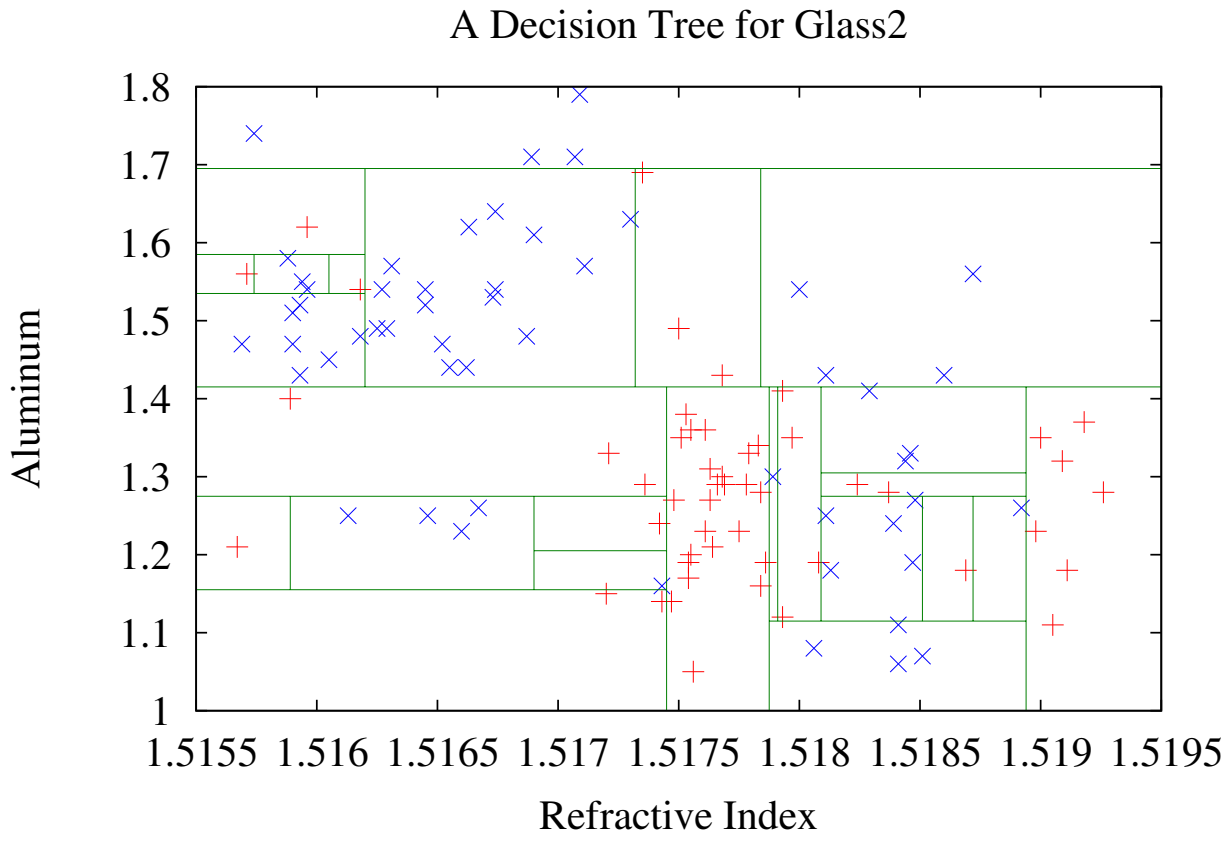
Reasons for overfitting:

- Noise
- Coincidence
- Lack of Data
- Boundary Approximation

This decision tree has no errors on the examples.



In this region, note the boxes with only one example.



Avoiding Overfitting by Pruning

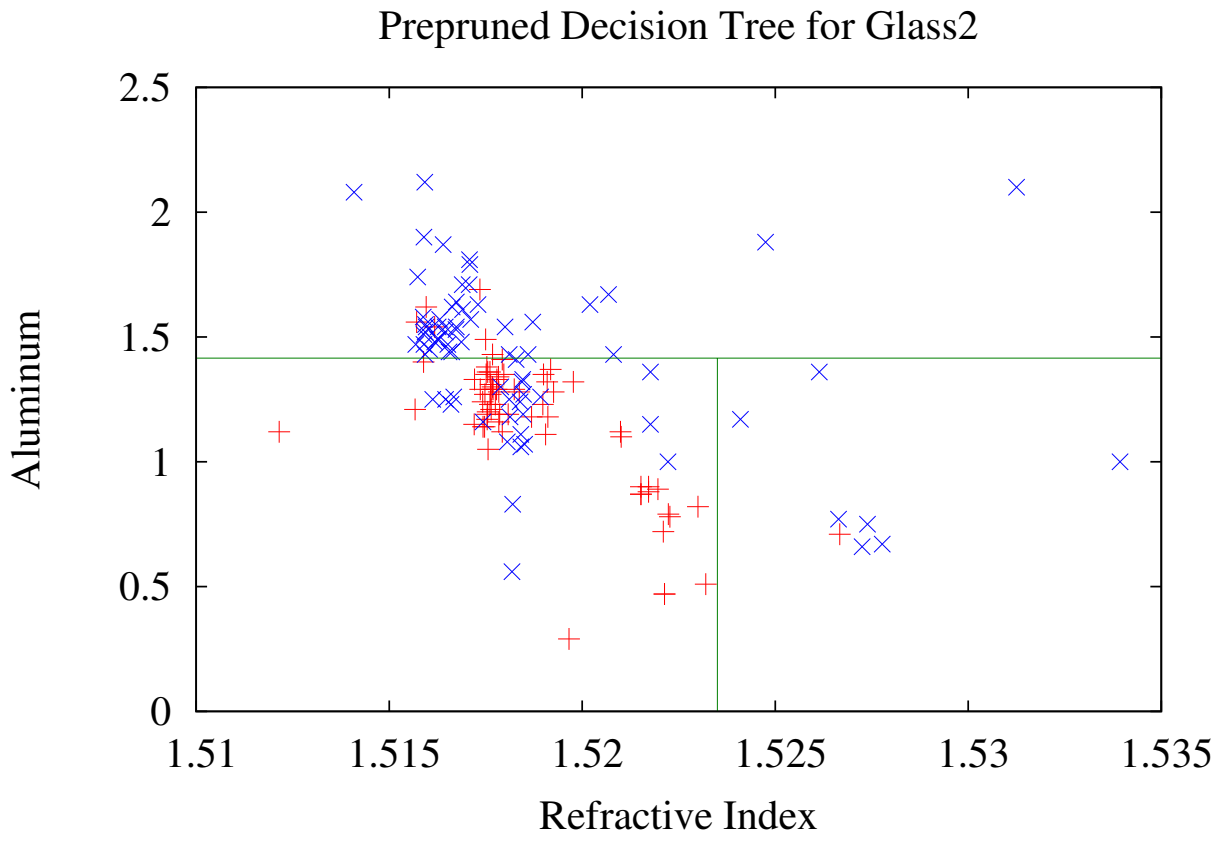
For decision trees, try to avoid overfitting by trading off smaller trees for small increases in training error.

Preferring smaller trees is justified by Occam's Razor.

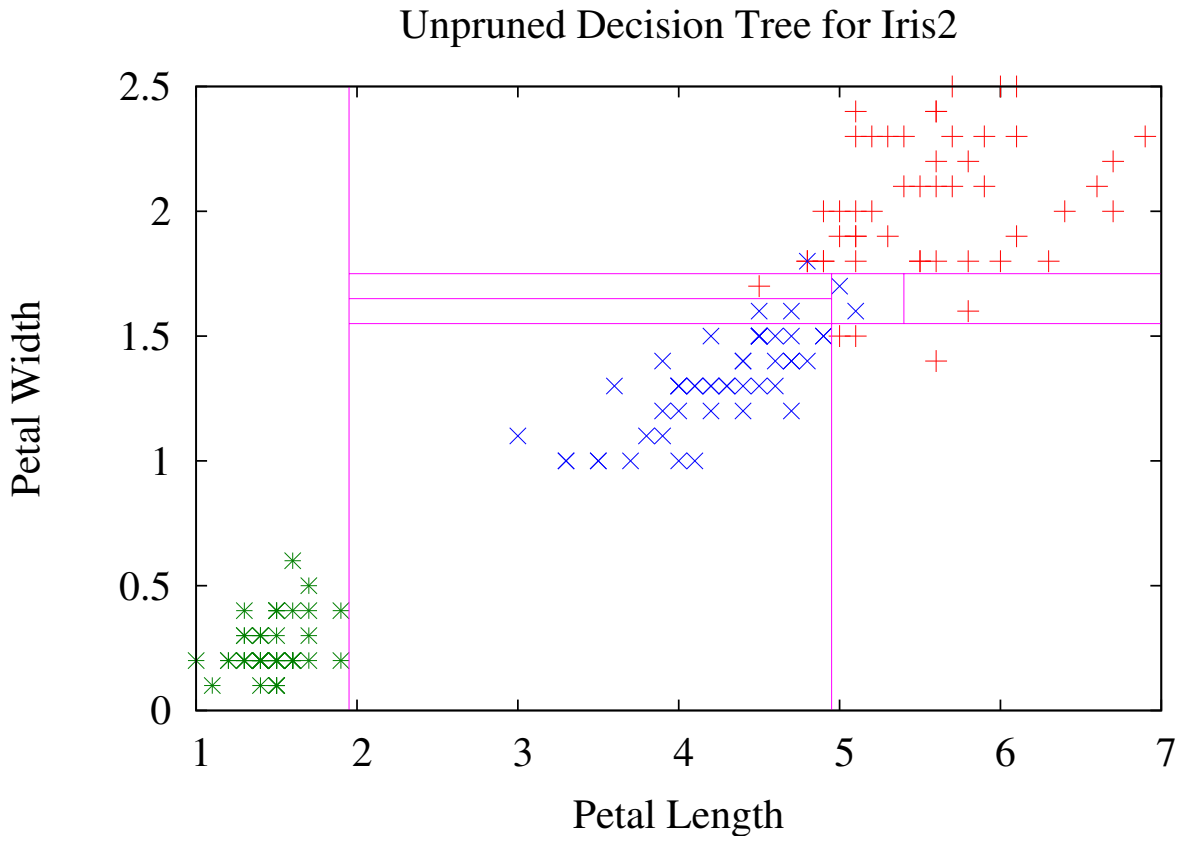
Modifying trees to make them smaller is called pruning.

- Prepruning: Avoid creation of subtrees based on number of examples or attribute relevance.
- Postpruning: Create overfitting DT and substitute subtrees with leaves if estimated error is reduced.

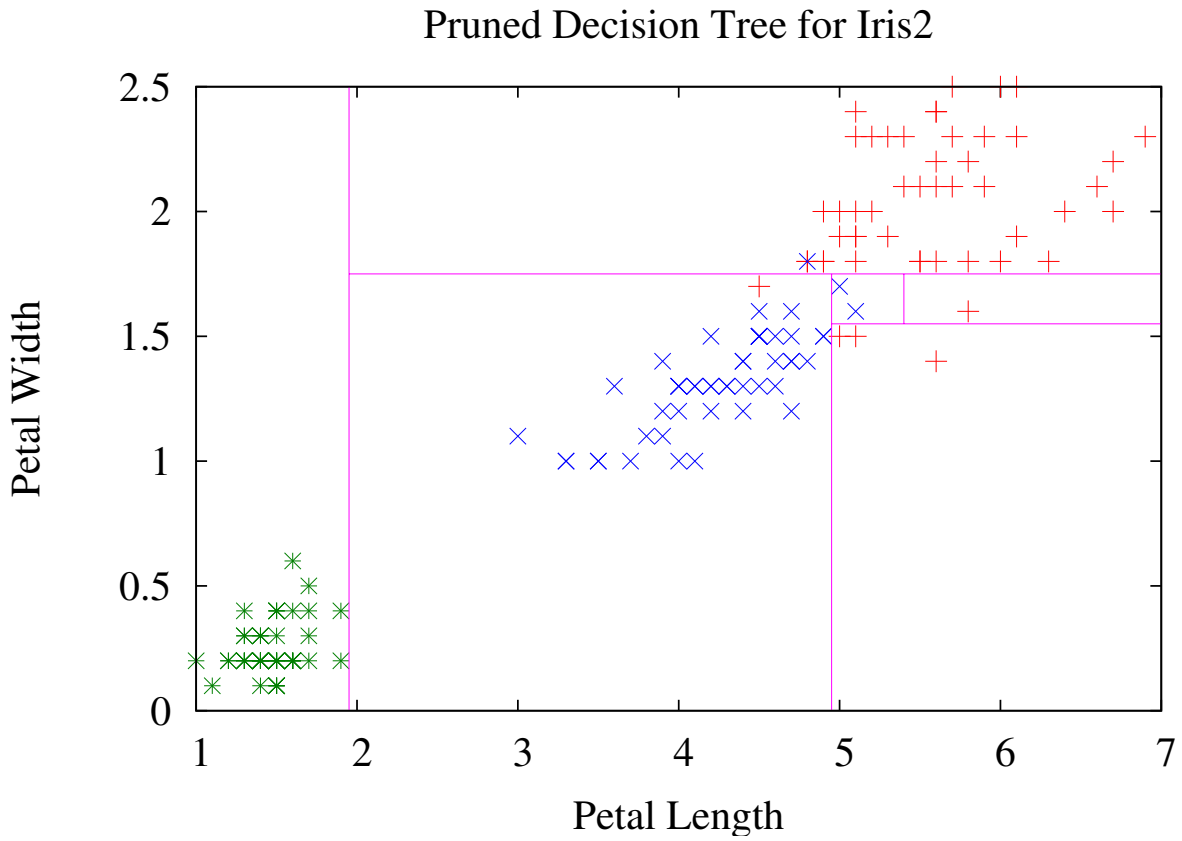
Prepruning Example (c4.5)



Unpruned Example (c4.5 -m 1)



Pruned Example (c4.5 -m 1)



Postpruning Algorithm

PRUNE-DT(N: node, *examples*)

1. $leaferr \leftarrow$ number of $examples \neq N.class$
2. revise $leaferr$ upward if *examples* were training set
3. **if** N is a leaf **then return** $leaferr$
4. $treeerr \leftarrow 0$
5. **for** each value v_j of $N.test$
6. $\quad examples_j \leftarrow examples$ with $N.test = v_j$
7. $\quad suberr \leftarrow$ PRUNE-DT($N.branch_j$, $examples_j$)
8. $\quad treeerr \leftarrow treeerr + suberr$
9. **if** $leaferr \leq treeerr$
10. $\quad$ **then** make N a leaf and **return** $leaferr$
11. $\quad$ **else return** $treeerr$

Comments on Pruning

The training and validation set approach is:
Remove "validation" exs. from training exs.
Grow decision tree using training exs.
Prune decision tree using validation set.

Subtree raising is replacing a tree with one of its subtrees.

Rule post-pruning as described in the book is performed by the C4.5rules program, not C4.5.