Algorithms: Decision Trees

A small dataset: Miles Per Gallon

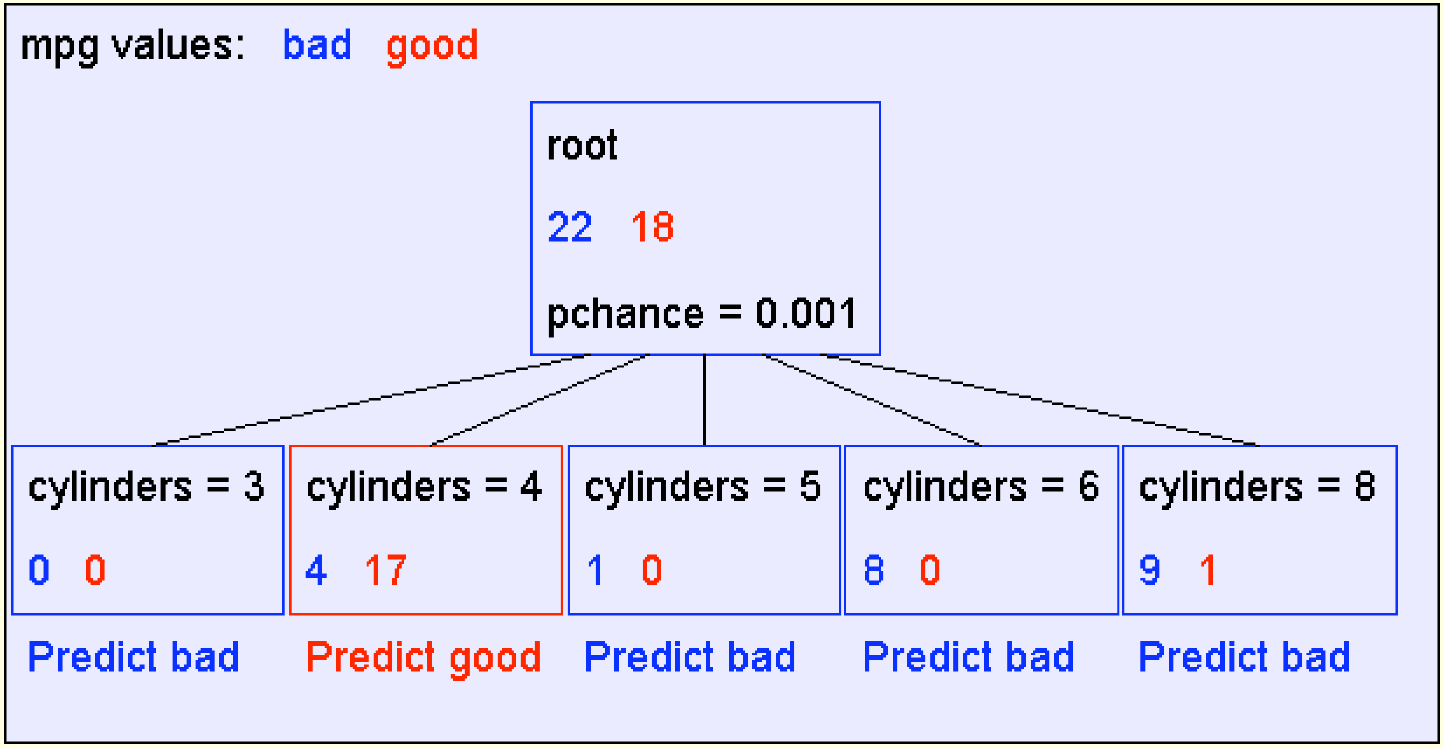
- Suppose we want to predict MPG

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe

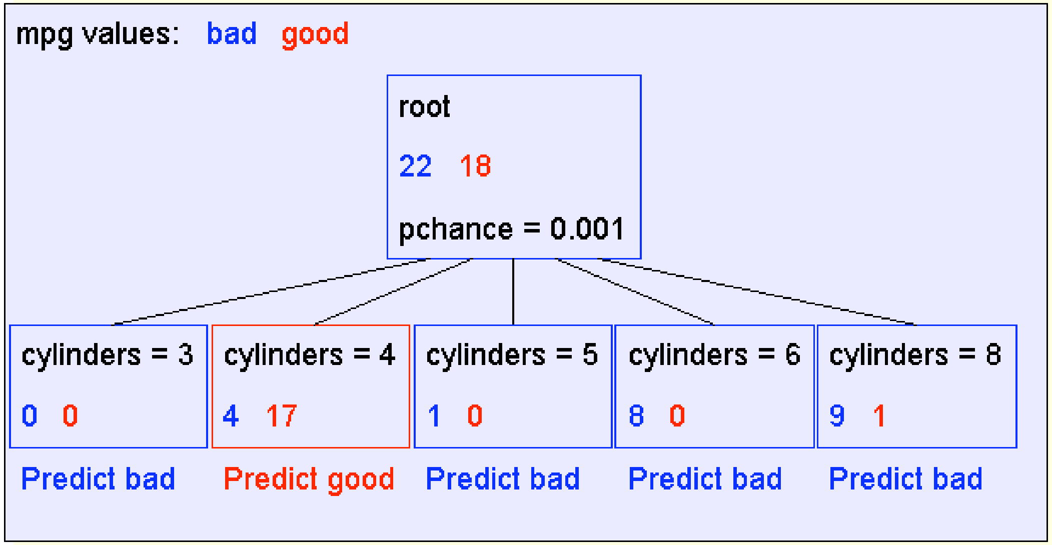
40 Records

- From the UCI repository

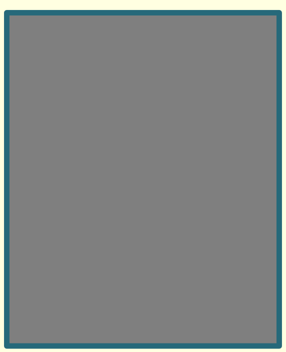
A Decision Stump



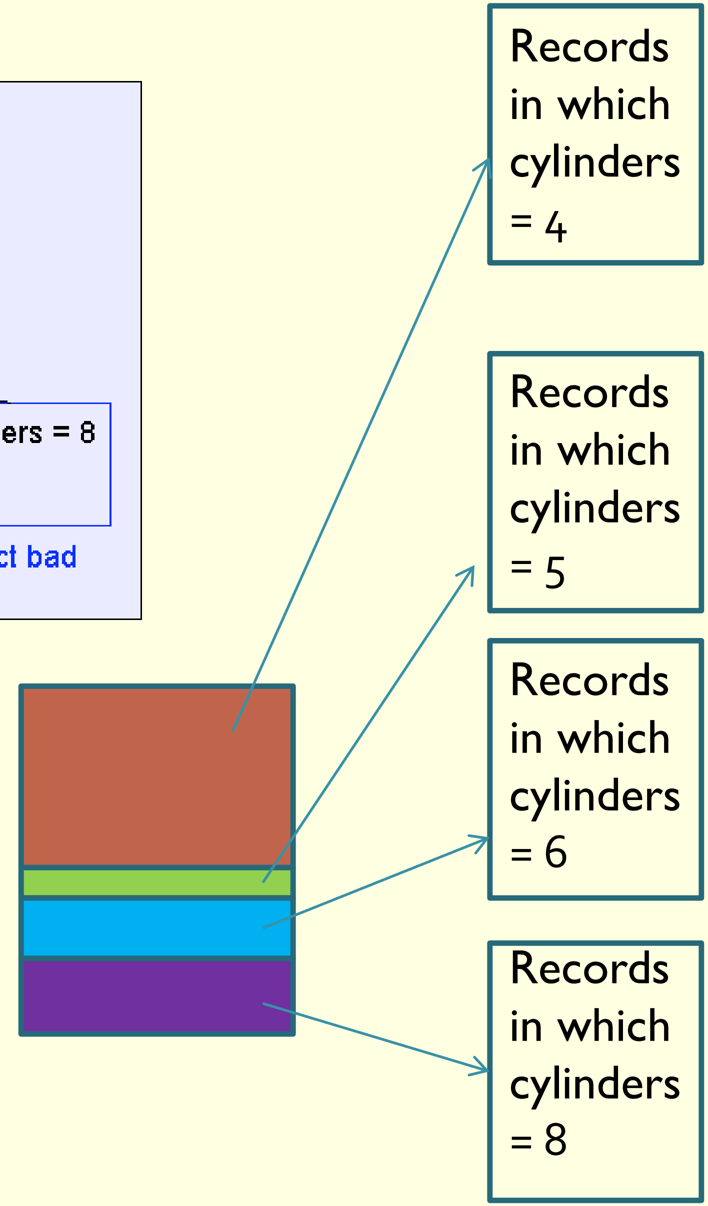
Recursion Step



Take the Original Dataset..

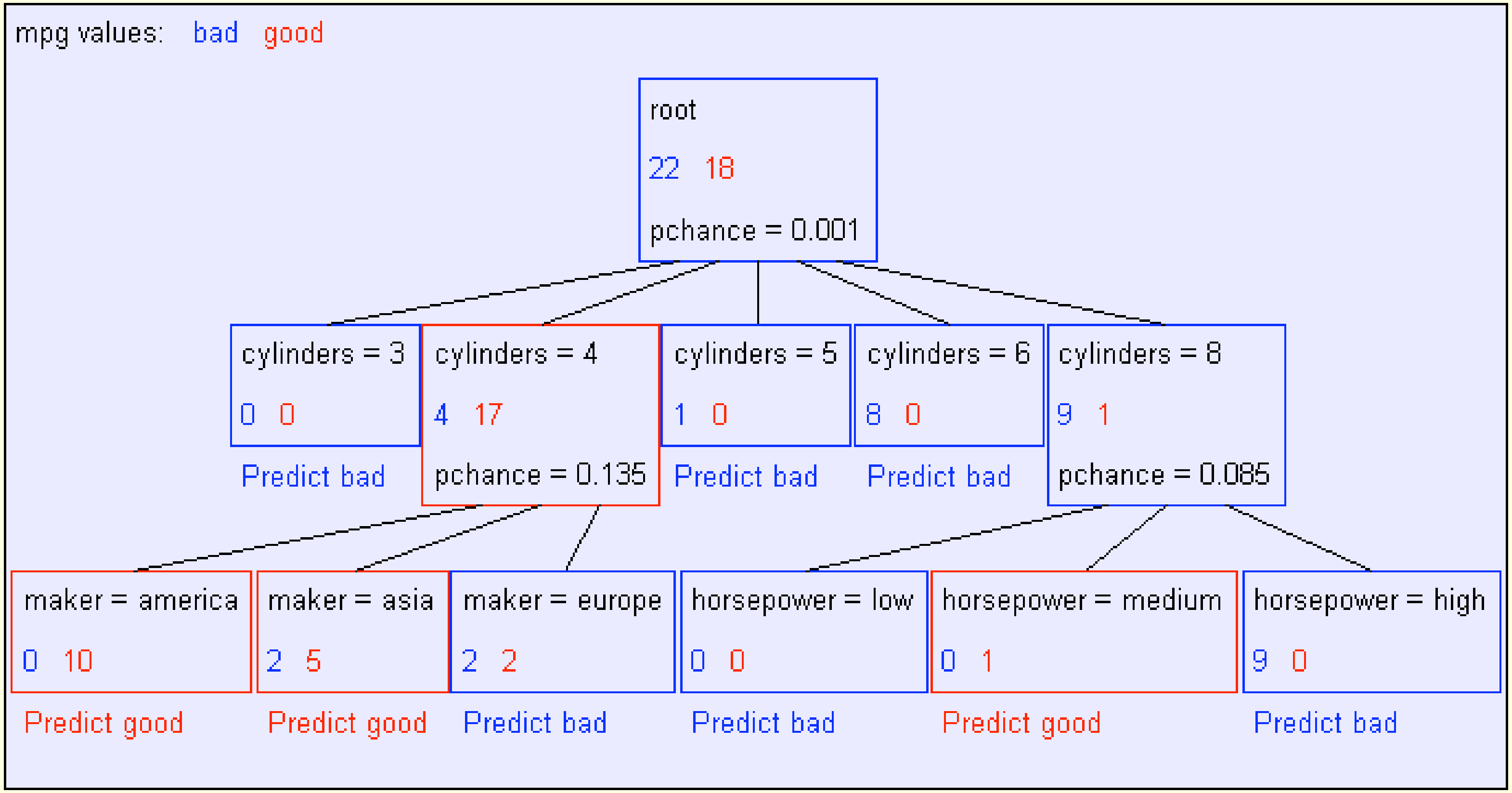


And partition it according to the value of the attribute we split on



Build Tree from these Records

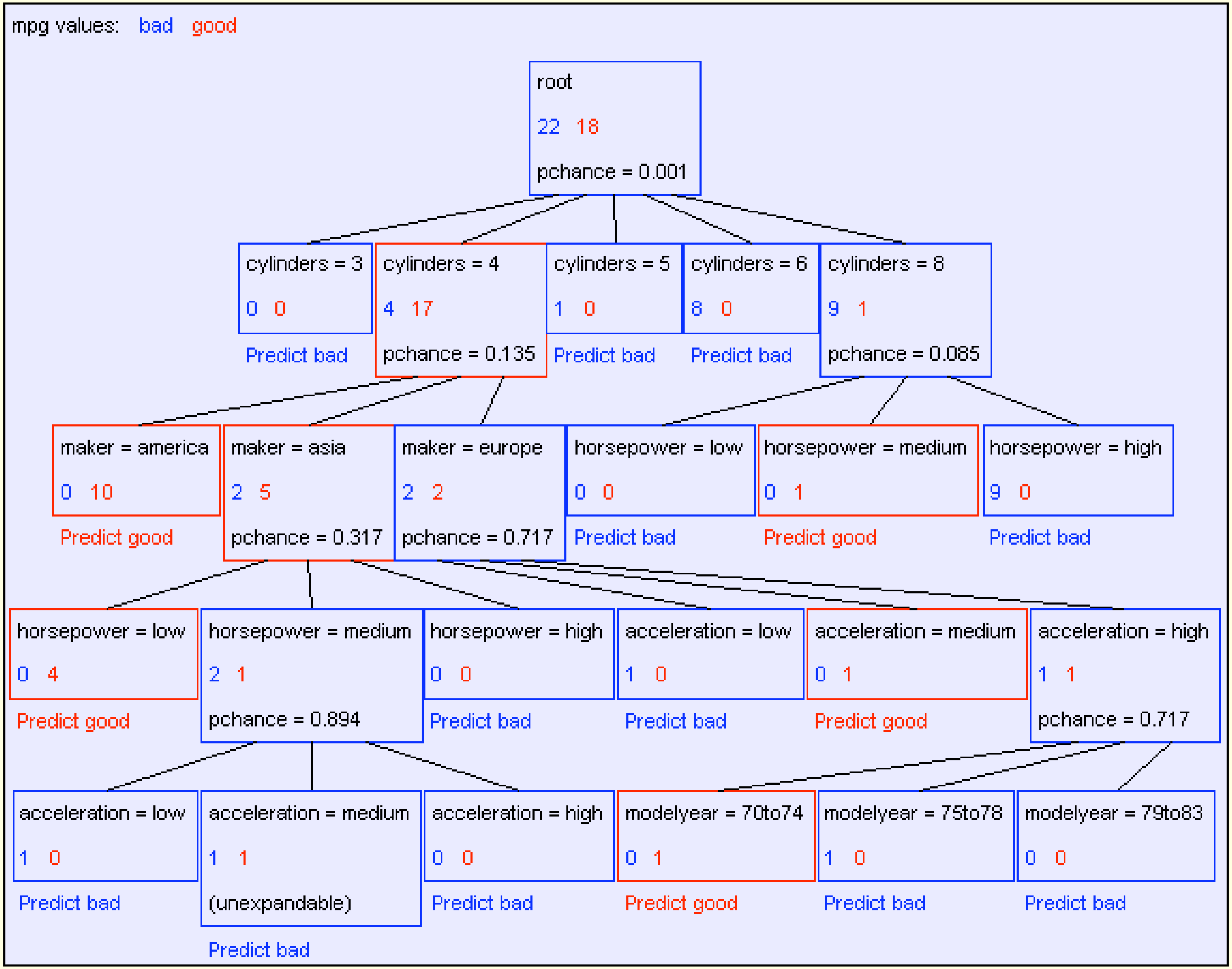
Second level of tree



Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

The final tree



Classification of a new example

- Classifying a test example
- Traverse tree
- Report leaf label

Learning decision trees is hard!!!

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest '76]

- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on **next best attribute (feature)**
  - Recurse
- How to choose the best attribute and the value for a split?

Entropy

- Entropy characterizes our uncertainty about our source of information
- ***More uncertainty, more entropy!***
  - *Information Theory interpretation: H(Y) is the expected number of bits needed* to encode a randomly drawn value of *Y (under most efficient code)*

Information gain

- Advantage of attribute – decrease in uncertainty
  - Entropy of Y before you split

$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

  - Entropy after split
    - Weight by probability of following each branch, i.e., normalized number of records

$$H(Y \mid X) = -\sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i \mid X = x_j) \log_2 P(Y = y_i \mid X = x_j)$$

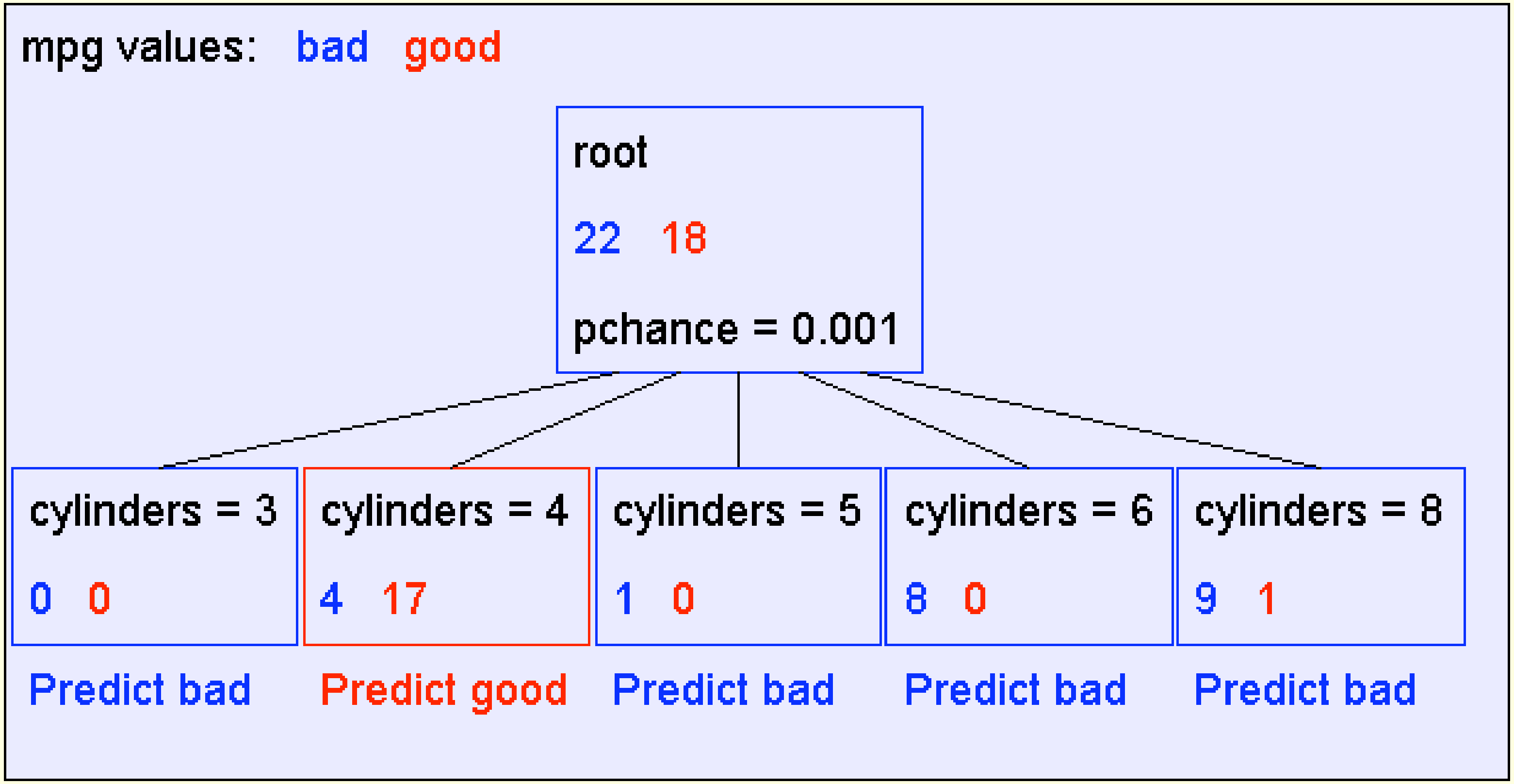
- Information gain is difference

$$IG(X) = H(Y) - H(Y \mid X)$$

Learning decision trees

- Start from empty decision tree
- Split on **next best attribute (feature)**
  - Use, for example, information gain to select attribute
  - Split on $\arg\max_i IG(X_i) = \arg\max_i H(Y) - H(Y \mid X_i)$
- Recurse

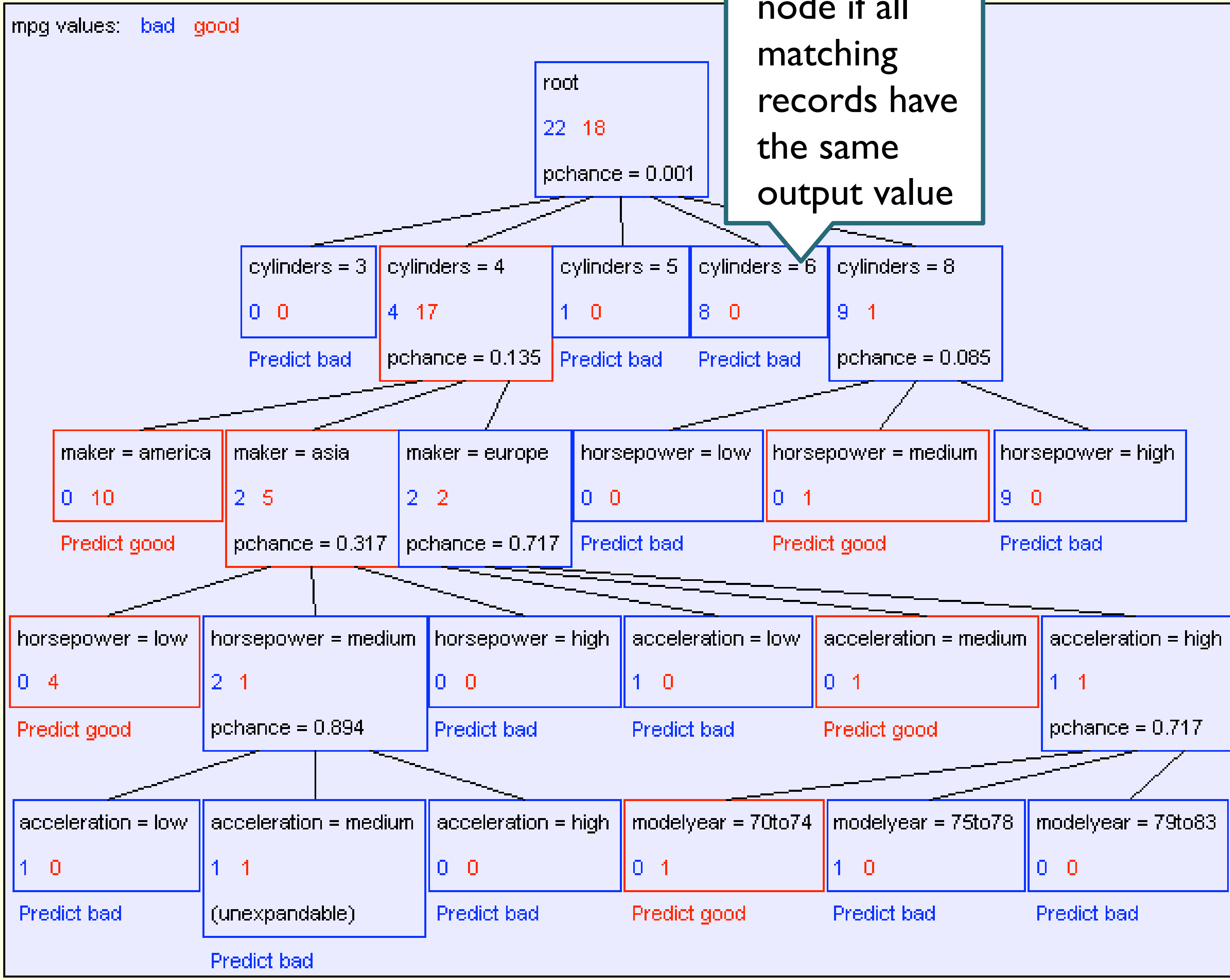
A Decision Stump



Base Cases

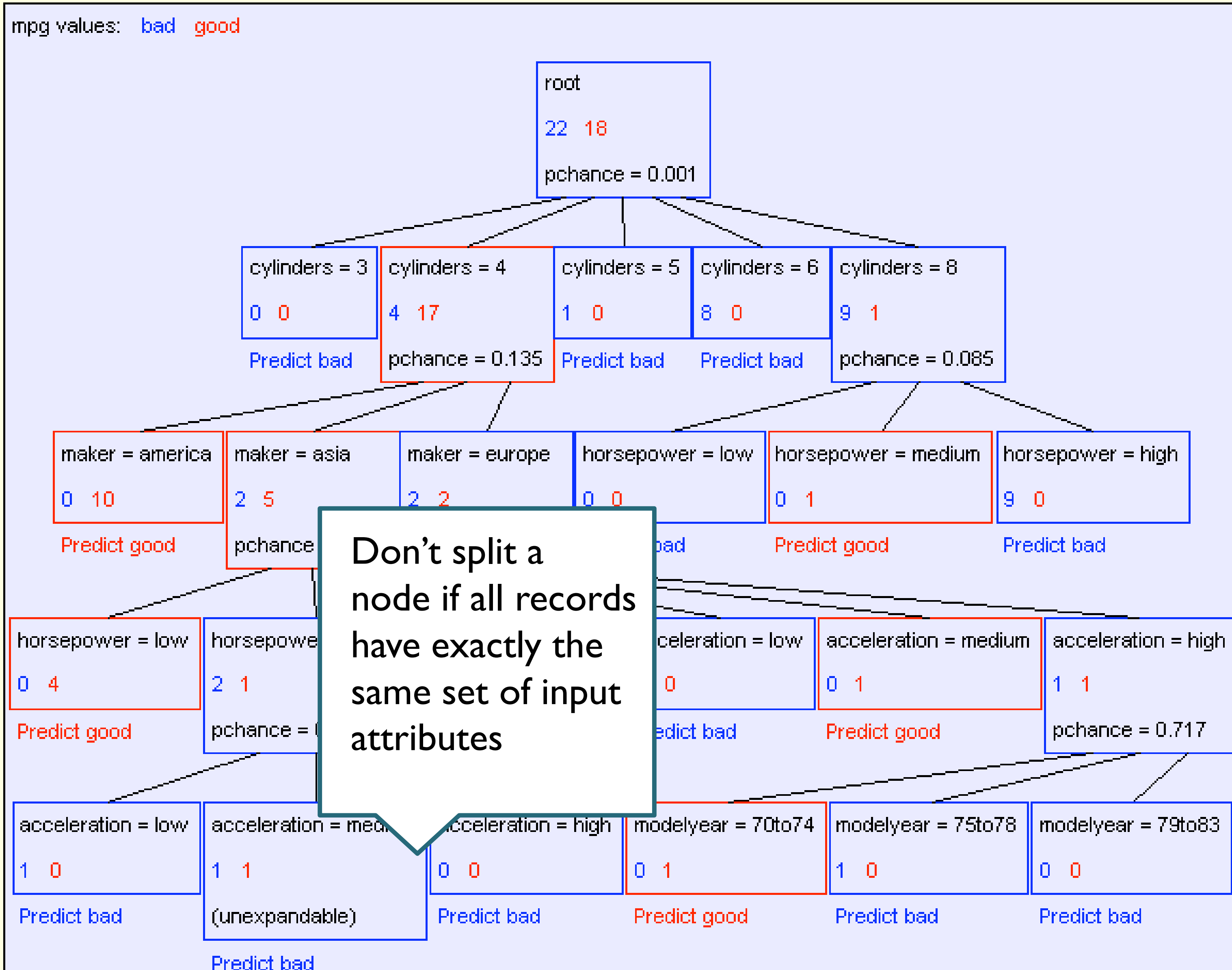
- Base Case One: If all records in current data subset have the same output then **don't recurse**
- Base Case Two: If all records have exactly the same set of input attributes then **don't recurse**

Base Case 1



Don't split a node if all matching records have the same output value

Base Case 2



Don't split a node if all records have exactly the same set of input attributes

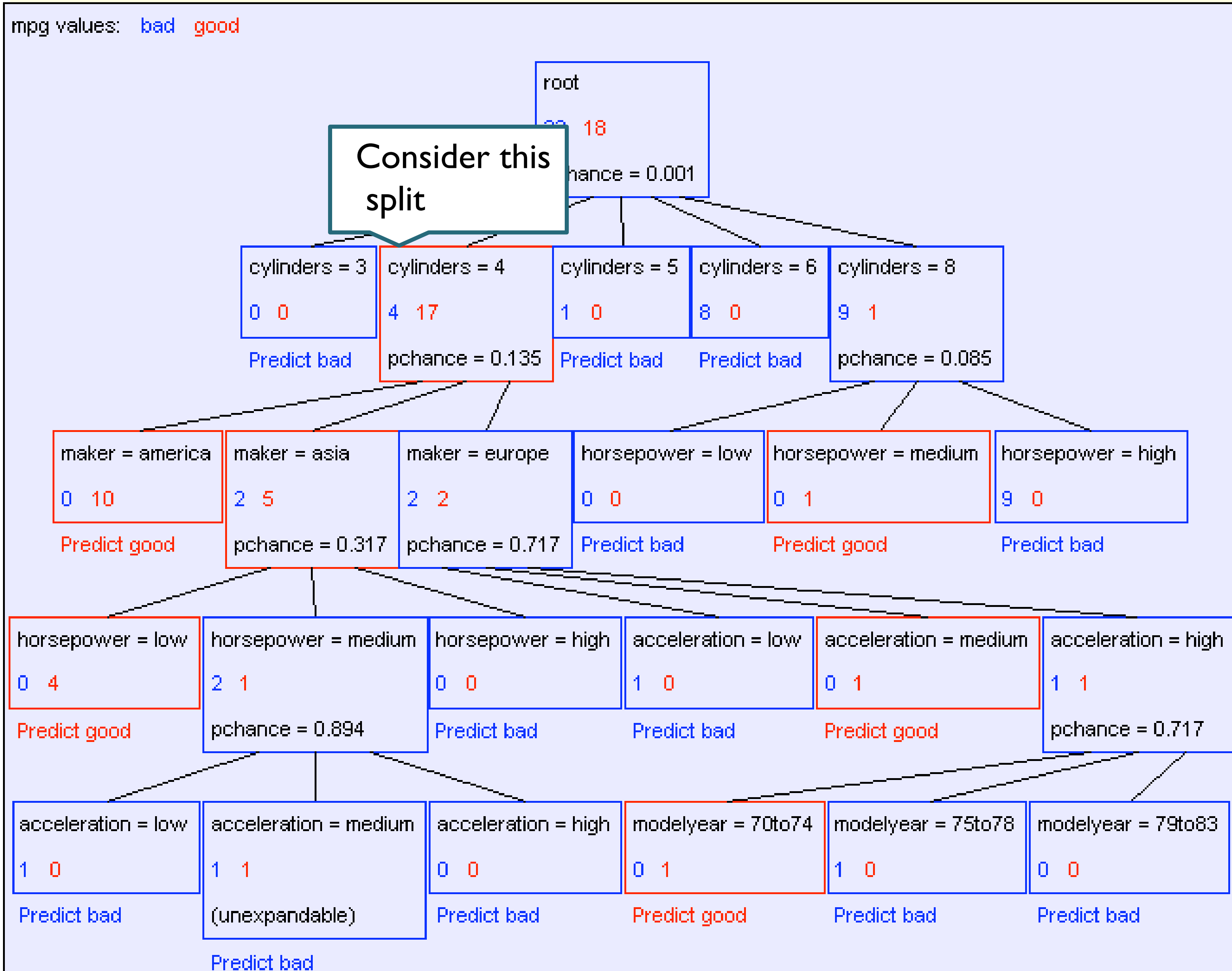
Basic Decision Tree Building Summarized

- BuildTree(*DataSet,Output*)
- If all output values are the same in *DataSet, return a leaf node that says* "predict this unique output"
- If all input values are the same, return a leaf node that says "predict the majority output"
- Else find attribute *X with highest Info Gain*
- Suppose *X has* n_X *distinct values (i.e. X has arity* n_X*).*
  - Create and return a non-leaf node with n_X *children.*
  - The *i'th child should be built by calling* BuildTree(*DSi,Output*)

  Where *DSi built consists of all those records in DataSet for which X = ith* distinct value of X.

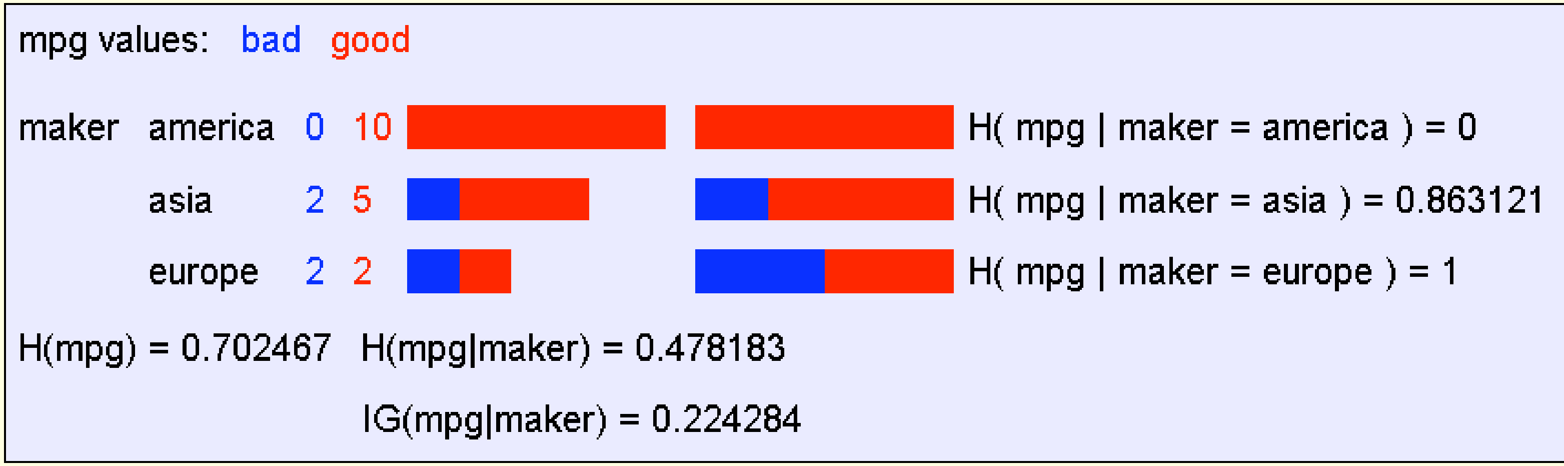
Decision trees will overfit

- Standard decision trees are have no learning biased
  - Training set error is always zero!
    - (If there is no label noise)
  - Lots of variance
  - Will definitely overfit!!!
  - Must bias towards simpler trees
- Many strategies for picking simpler trees:
  - Fixed depth
  - Fixed number of leaves
  - Or something smarter…



mpg values: bad good
root
18
Consider this split
cylinders = 3
0 0
Predict bad
cylinders = 4
4 17
pchance = 0.135
cylinders = 5
1 0
Predict bad
cylinders = 6
8 0
Predict bad
cylinders = 8
9 1
pchance = 0.085
maker = america
0 10
Predict good
maker = asia
2 5
pchance = 0.317
maker = europe
2 2
pchance = 0.717
horsepower = low
0 0
Predict bad
horsepower = medium
0 1
Predict good
horsepower = high
9 0
Predict bad
horsepower = low
0 4
Predict good
horsepower = medium
2 1
pchance = 0.894
horsepower = high
0 0
Predict bad
acceleration = low
1 0
Predict bad
acceleration = medium
0 1
Predict good
acceleration = high
1 1
pchance = 0.717
acceleration = low
1 0
Predict bad
acceleration = medium
1 1
(unexpandable)
Predict bad
acceleration = high
0 0
Predict bad
modelyear = 70to74
0 1
Predict good
modelyear = 75to78
1 0
Predict bad
modelyear = 79to83
0 0
Predict bad

A statistical test



- Suppose that mpg was completely uncorrelated with maker.
- What is the chance we'd have seen data of at least this apparent level of association anyway?

Using to avoid overfitting

- Build the full decision tree as before
- But when you can grow it no more, start to prune:
    - Beginning at the bottom of the tree, delete splits in which have extreme low chance to appear//*pchance* > *MaxPchance*
    - Continue working your way up until there are no more prunable nodes

What you need to know about decision trees

- Decision trees are one of the most popular data mining tools
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap (to solve heuristically)
- Information gain to select attributes (ID3, C4.5,…)
- Presented for classification, can be used for regression and density estimation too
- Decision trees will overfit!!!
  - Zero bias classifier ! Lots of variance
  - Must use tricks to find "simple trees", e.g.,
    - Fixed depth/Early stopping
    - Pruning

Decision trees in Matlab

- Use classregtree class
- Create a new tree:

  t=classregtree(X,Y), X is a matrix of predictor values, y is a vector of n response values
- Prune the tree:

  tt = prune(t, alpha, pChance) alpha defines the level of the pruning
- Predict a value

  y= eval(tt, X)