Artificial Intelligence
CE-417, Group 1
Computer Eng. Department
Sharif University of Technology

Fall 2023

By Mohammad Hossein Rohban, Ph.D.

Courtesy: Most slides are adopted from CSE-573 (Washington U.), original slides for the textbook, and CS-188 (UC. Berkeley).

Classification

(Decision Tree)

A learning problem: predict fuel efficiency

- From the UCI repository (thanks to Ross Quinlan)
  - 40 records
  - Discrete data (for now)
  - Predic MPG

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	aisa
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

Need to find "Hypothesis": $f: X \to Y$

$\underbrace{}_{Y}$ $\underbrace{}_{X}$

How Represent Function?

f (

cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
4	low	low	low	high	75to78	asia

) $\rightarrow$

mpg
good

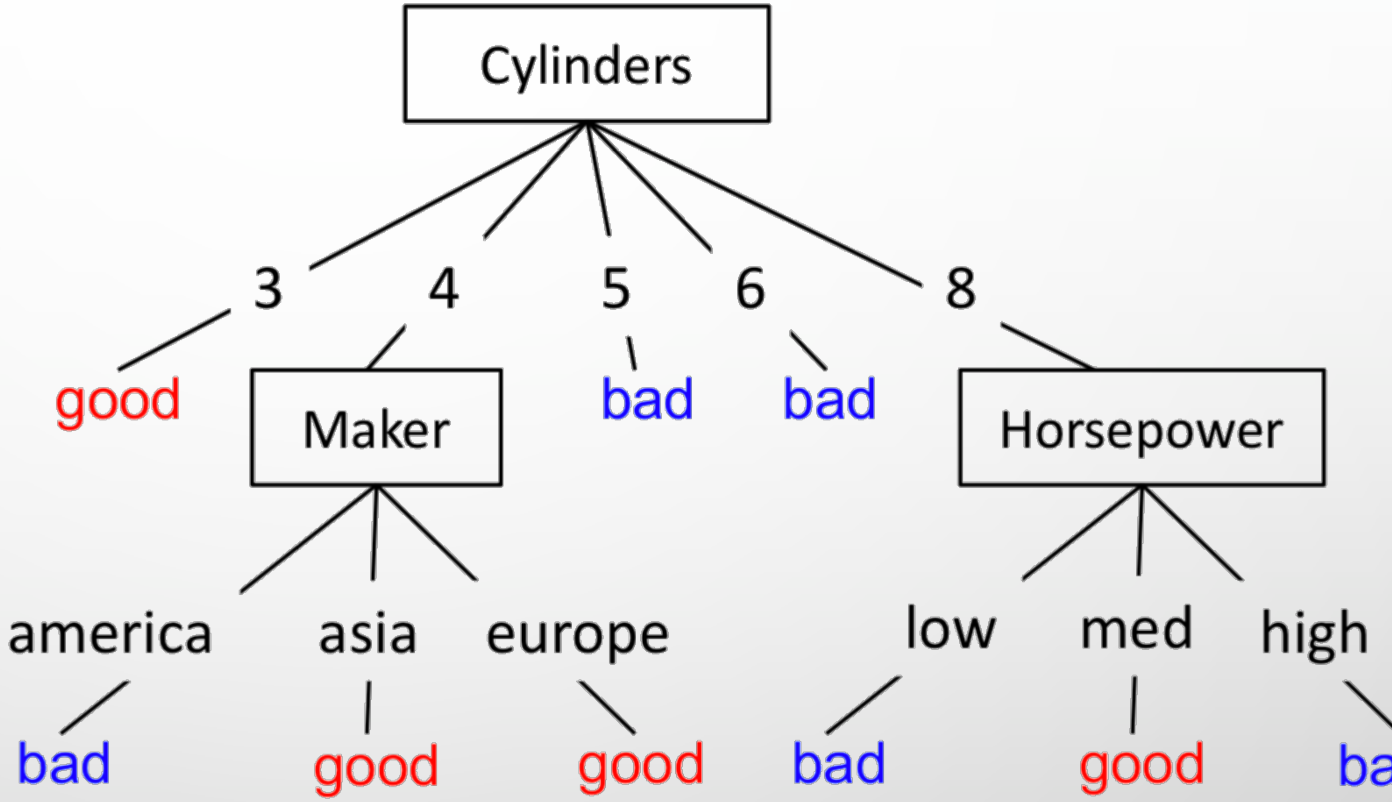
General Propositional Logic?

maker=asia $\vee$ weight=low

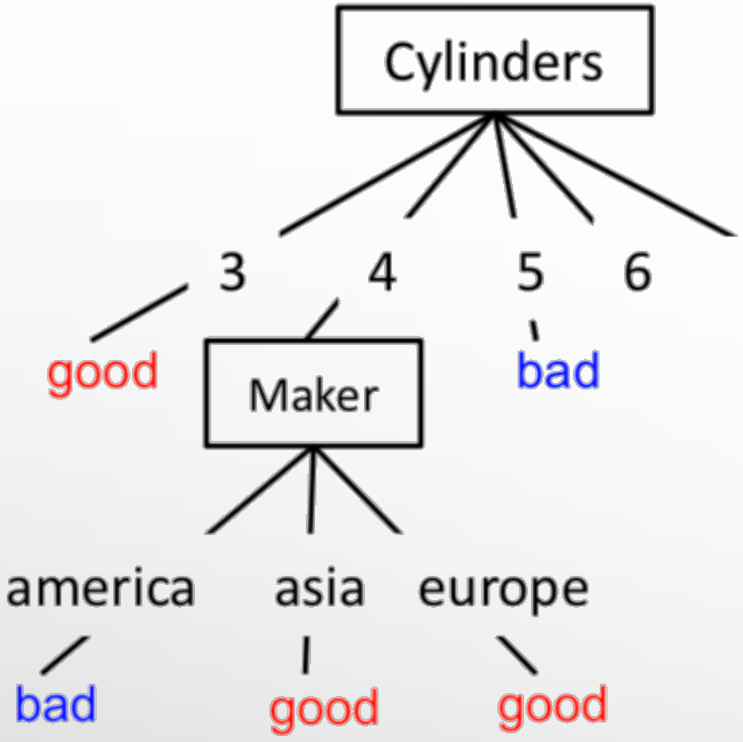
Need to find "Hypothesis": $f : X \rightarrow Y$

Hypotheses: decision trees $f : X \rightarrow Y$

- Each internal node tests an attribute x_i
- Each branch assigns an attribute value $x_i = v$
- Each leaf assigns a class y
- To classify input x?
- traverse the tree from root to leaf, output the labeled y



What functions can be represented?



$$cy = 3 \vee (cy = 4 \wedge (maker = asia \vee maker = europe)) \vee \cdots$$

Learning as Search

- Nodes?
- Operators?
- Start State?
- Goal?
  Search Algorithm?
- Heuristic?

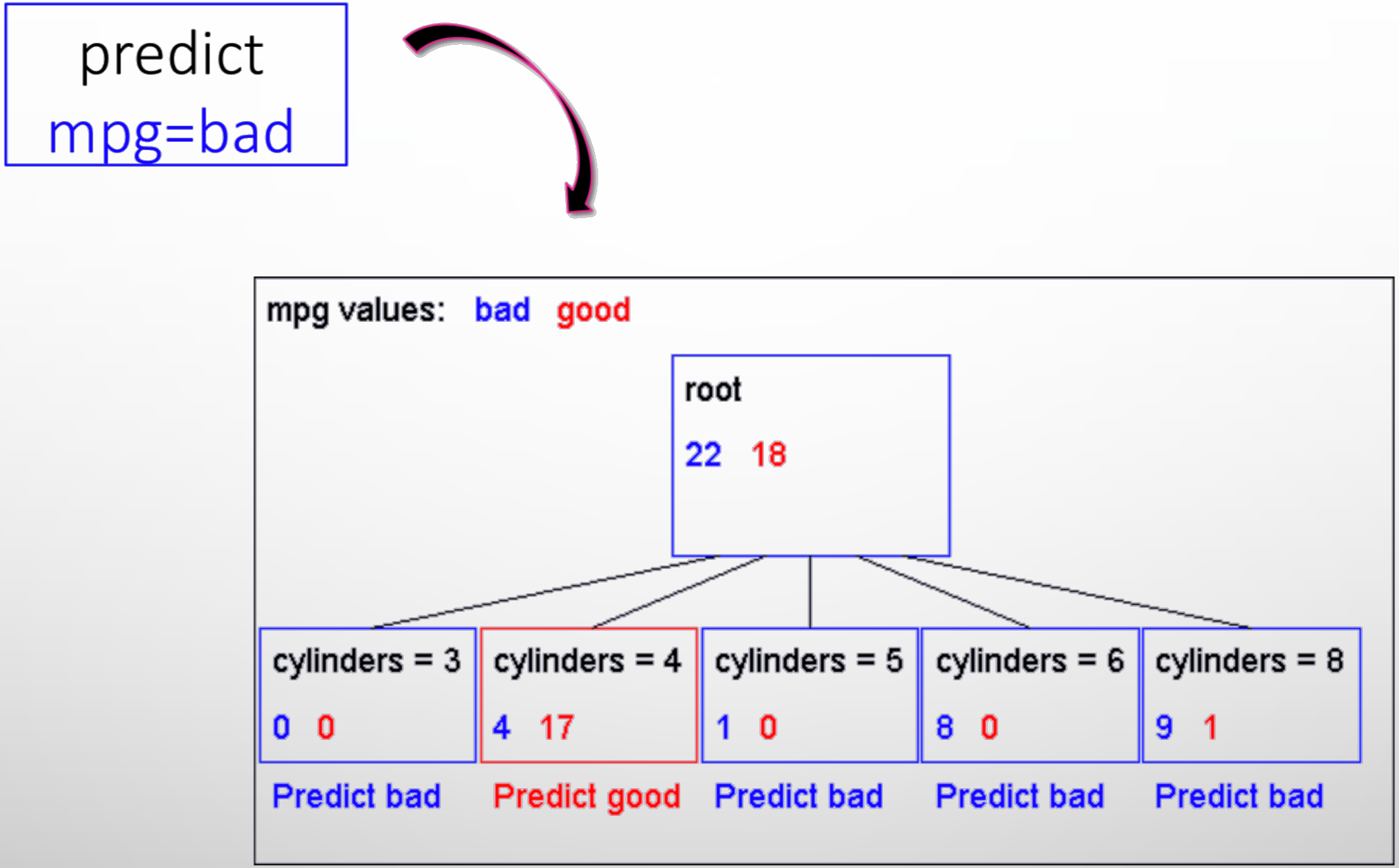
The Starting Node: What is the Simplest Tree?

predict
mpg=bad

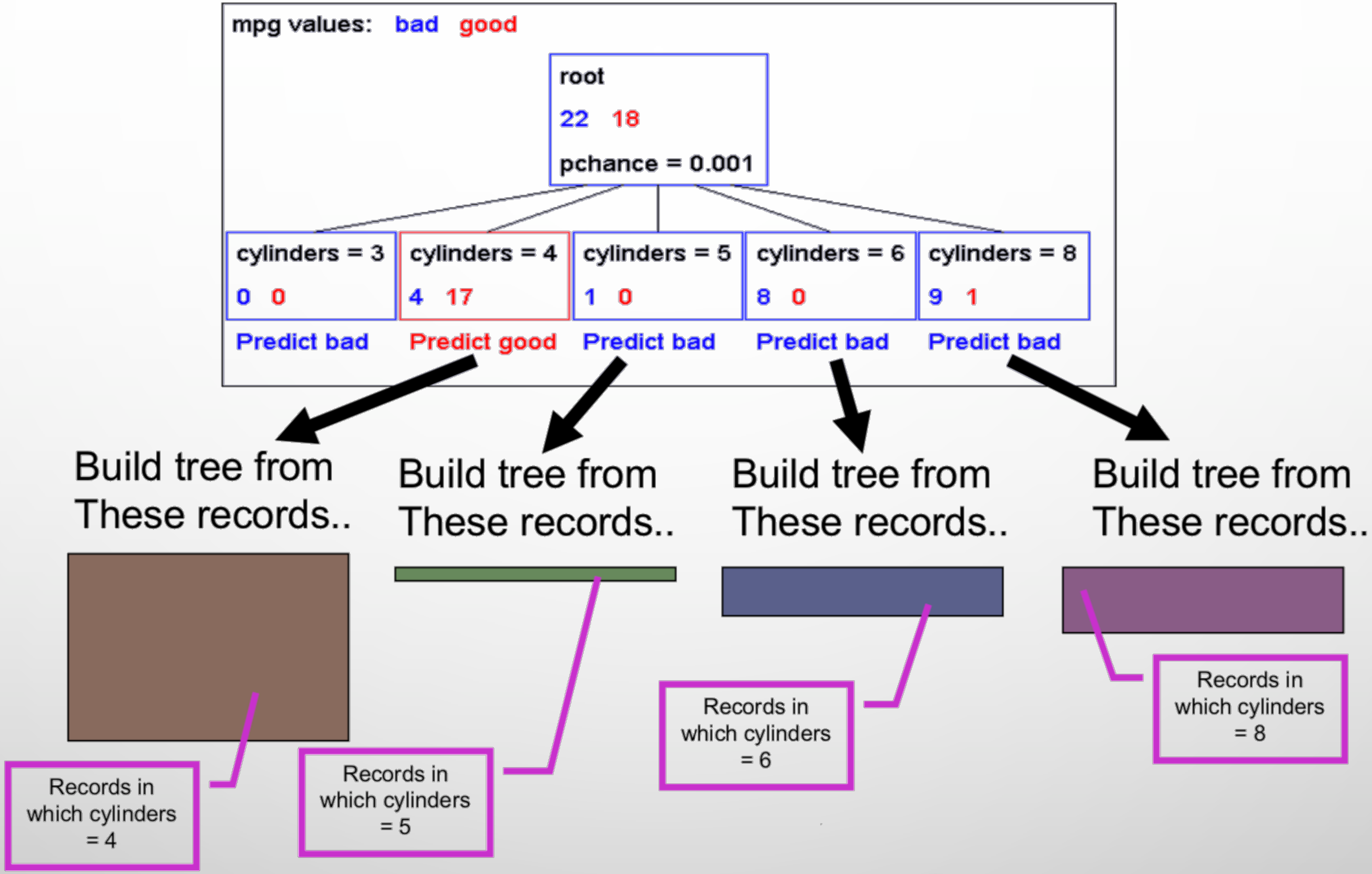
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	aisa
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	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

- Is this a good tree?
- [22+, 18-] : Means: correct on 22 examples incorrect on 18 examples.

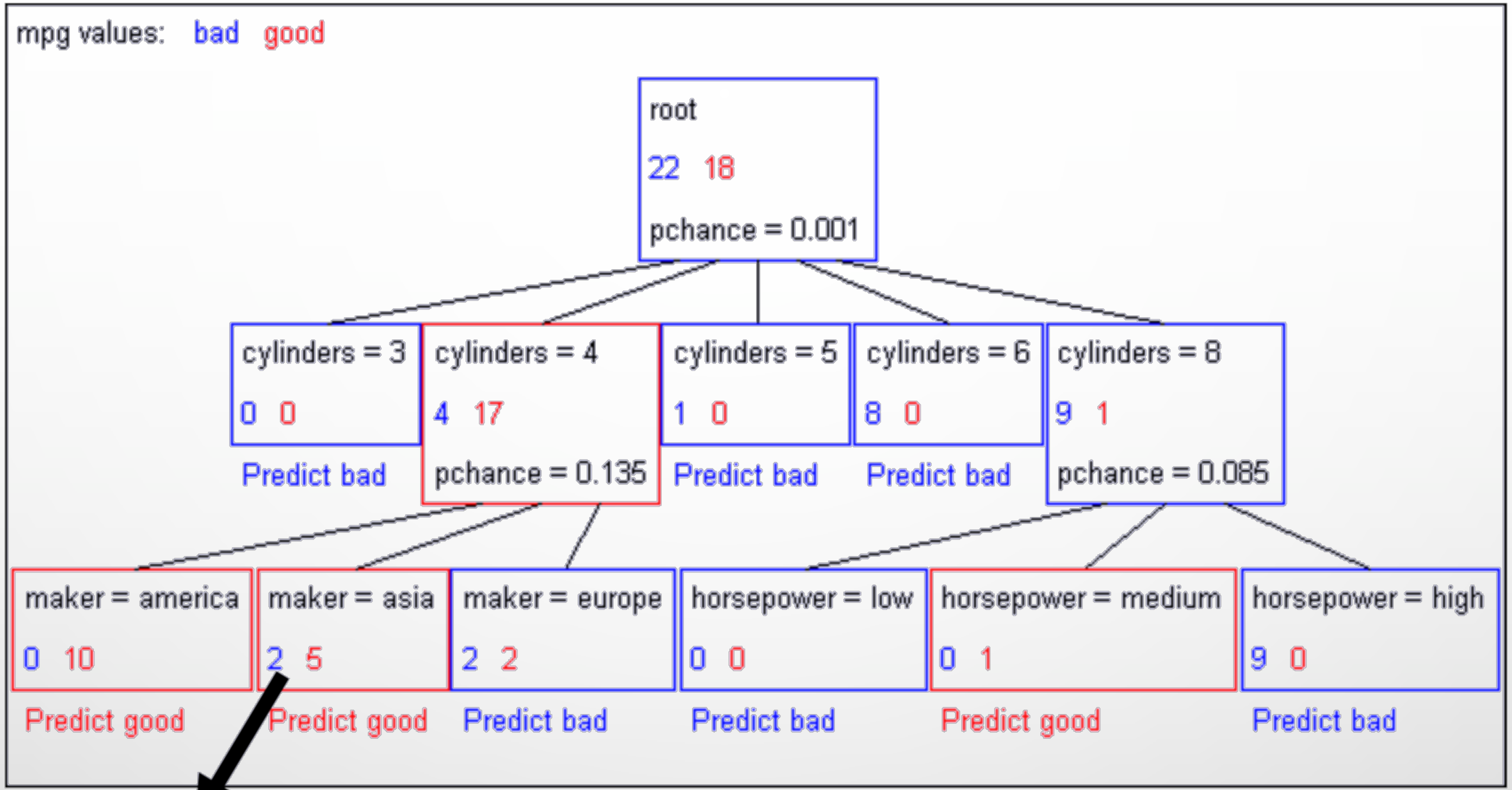
Operators: Improving the Tree



Recursive Step



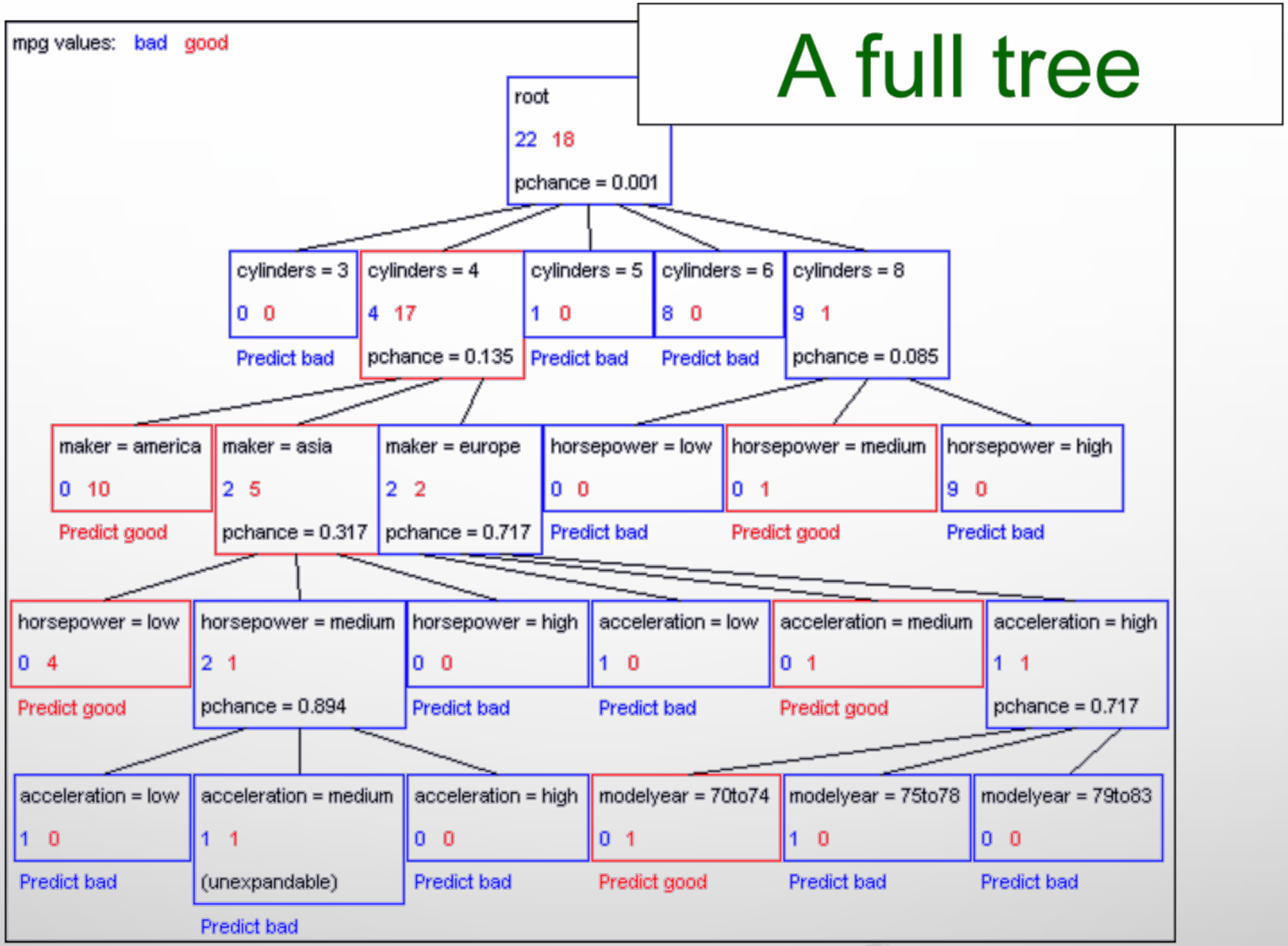
Second level of Tree



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)

A Full Tree



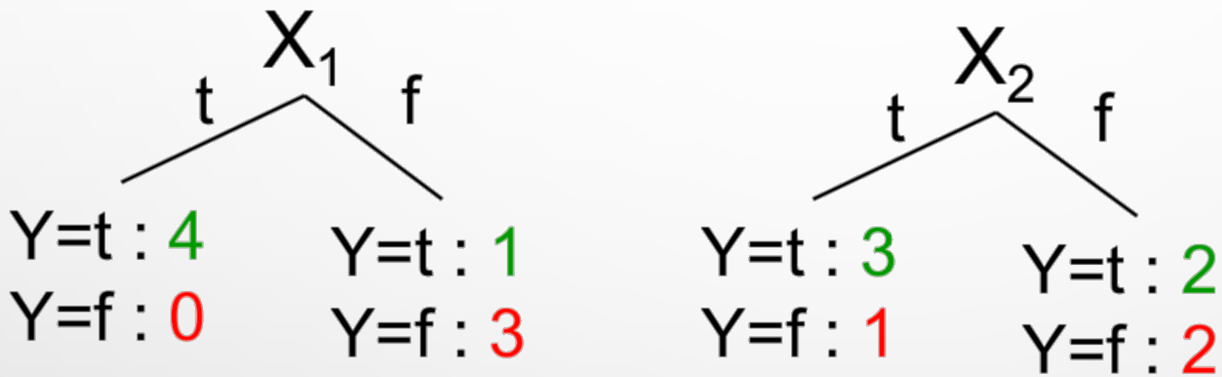
mpg values: bad good
A full tree
root
22 18
pchance = 0.001
cylinders = 3
0 0
Predict bad
cylinders = 4
4 17
pchance = 0.135
cylinders = 5
1 0
Predict bad
cylinders = 6
8 0
Predict bad
cylinders = 8
9 1
pchance = 0.085
maker = america
0 10
Predict good
maker = asia
2 5
pchance = 0.317
maker = europe
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

Two Questions

- Hill Climbing Algorithm:
  - Start from empty decision tree
  - Split on the **best attribute (feature)** – Recurse
- Which attribute gives the best split?
- When to stop recursion?

Splitting: choosing a good attribute

Would we prefer to split on X_1 or X_2?



X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

Idea: use counts at leaves to define probability distributions so we can measure uncertainty!

Measuring uncertainty

- Good split if we are more certain about classification after split
  - Deterministic good (all true or all false)
  - Uniform distribution?   BAD
  - What about distributions in between?

P(Y=A) = 1/2	P(Y=B) = 1/4	P(Y=C) = 1/8	P(Y=D) = 1/8

P(Y=A) = 1/3	P(Y=B) = 1/4	P(Y=C) = 1/4	P(Y=D) = 1/6

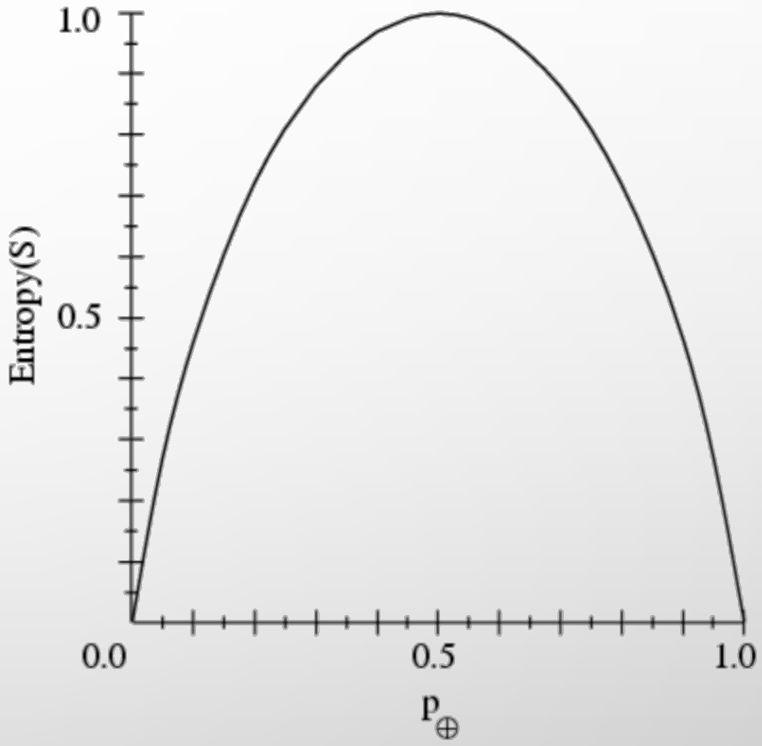
Which attribute gives the best split?

- A1: The one with the highest ***information gain***
  - Defined in terms of ***entropy***
- A2: Actually many alternatives,
  - e.g., ***accuracy***. Seeks to reduce the ***misclassification rate***

Entropy

- Entropy *H(Y)* of a random variable *Y*

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



- ***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)

Entropy Example

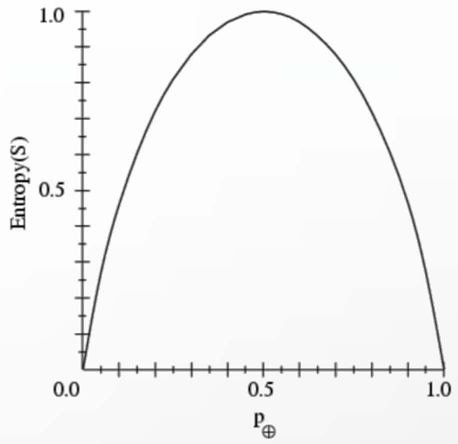
- $P(Y=t)=5/6, P(Y=f)=1/6$

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

$$P(Y=t)=\frac{5}{6}$$

$$P(Y=f)=\frac{1}{6}$$

$$H(Y) = -\frac{5}{6}\log_2\frac{5}{6} - \frac{1}{6}\log_2\frac{1}{6} \approx 0.65$$



X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

Conditional Entropy

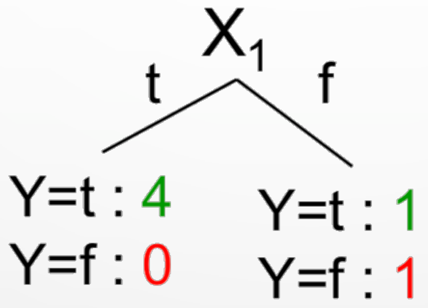
- Conditional Entropy $H(Y|X)$ of a random variable Y conditioned on a random variable X

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

Example:

$P(X_1=t) = 4/6$

$P(X_1=f) = 2/6$



X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

$H(Y|X_1) = -\ 4/6\ (1 \log_2 1 + 0 \log_2 0)$

$\quad - 2/6\ (1/2 \log_2 1/2 + 1/2 \log_2 1/2)$

$= 2/6$

$= 0.33$

Information Gain

- Advantage of attribute – decrease in entropy (uncertainty) after splitting

$$IG(X) = H(Y) - H(Y|X)$$

In our running example:

$IG(X_1) = H(Y) – H(Y|X_1)$
$= 0.65 – 0.33$

$IG(X_1) > 0 \rightarrow$ we prefer the split!

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

Learning Decision Trees

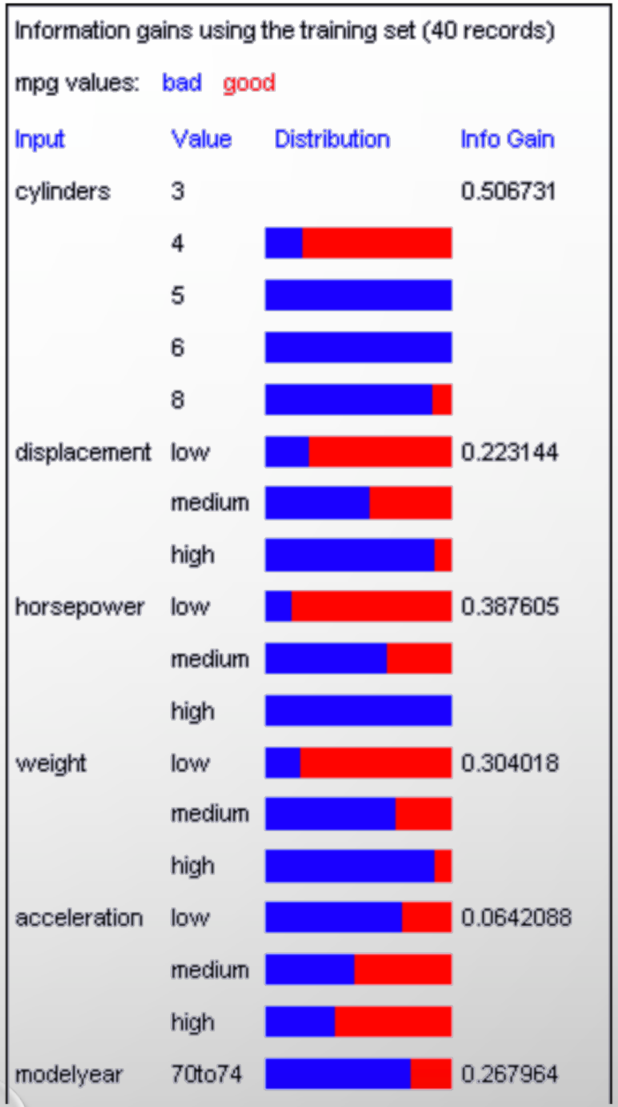
- Start from empty decision tree
- Split on **next best attribute (feature)**
- Use information gain (or…?) to select attribute:

$$\underset{i}{\operatorname{argmax}} IG(X_i) = \underset{i}{\operatorname{argmax}} [H(Y) - H(Y|X_i)]$$

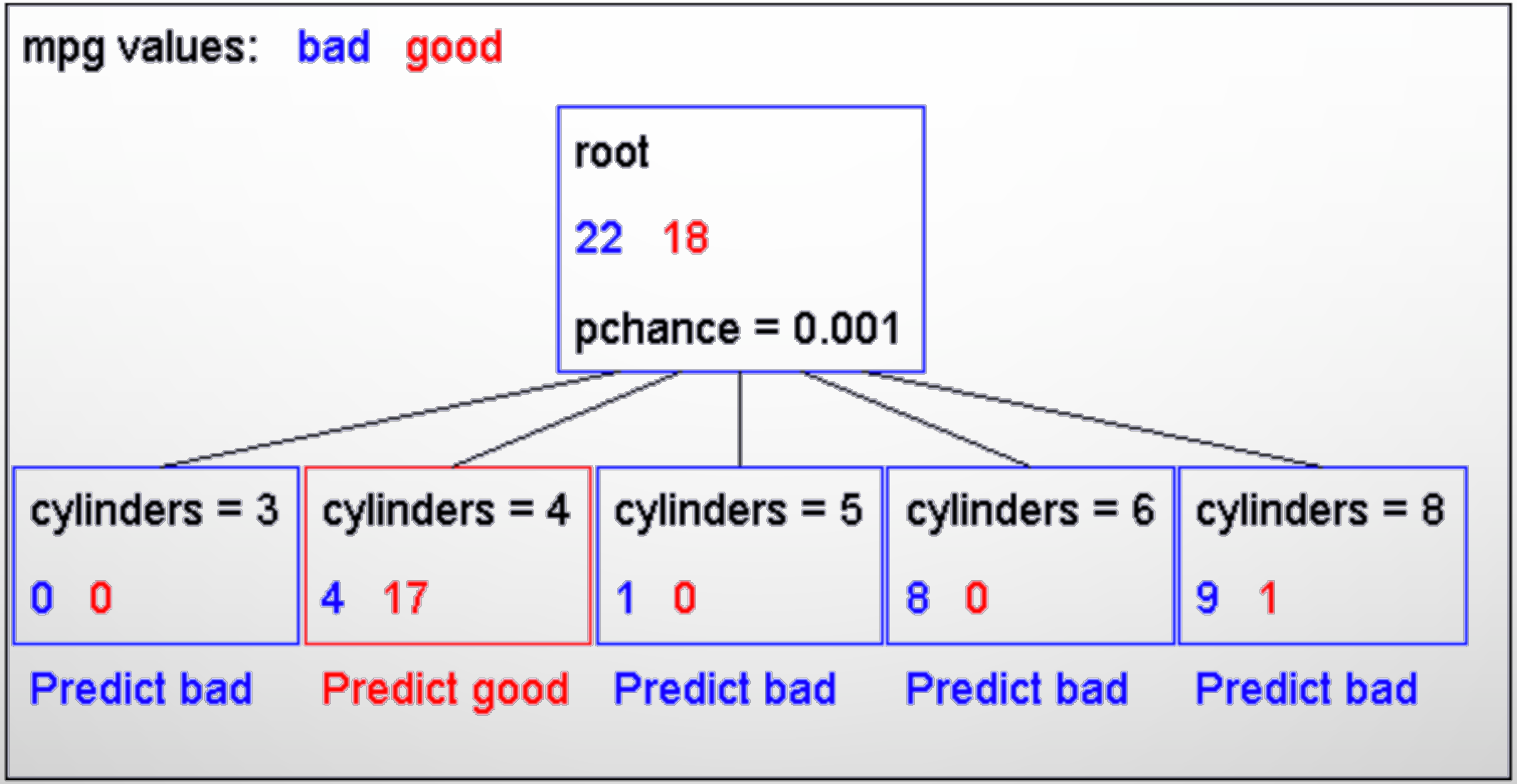
- Recurse.

Learning Decision Trees (cont.)

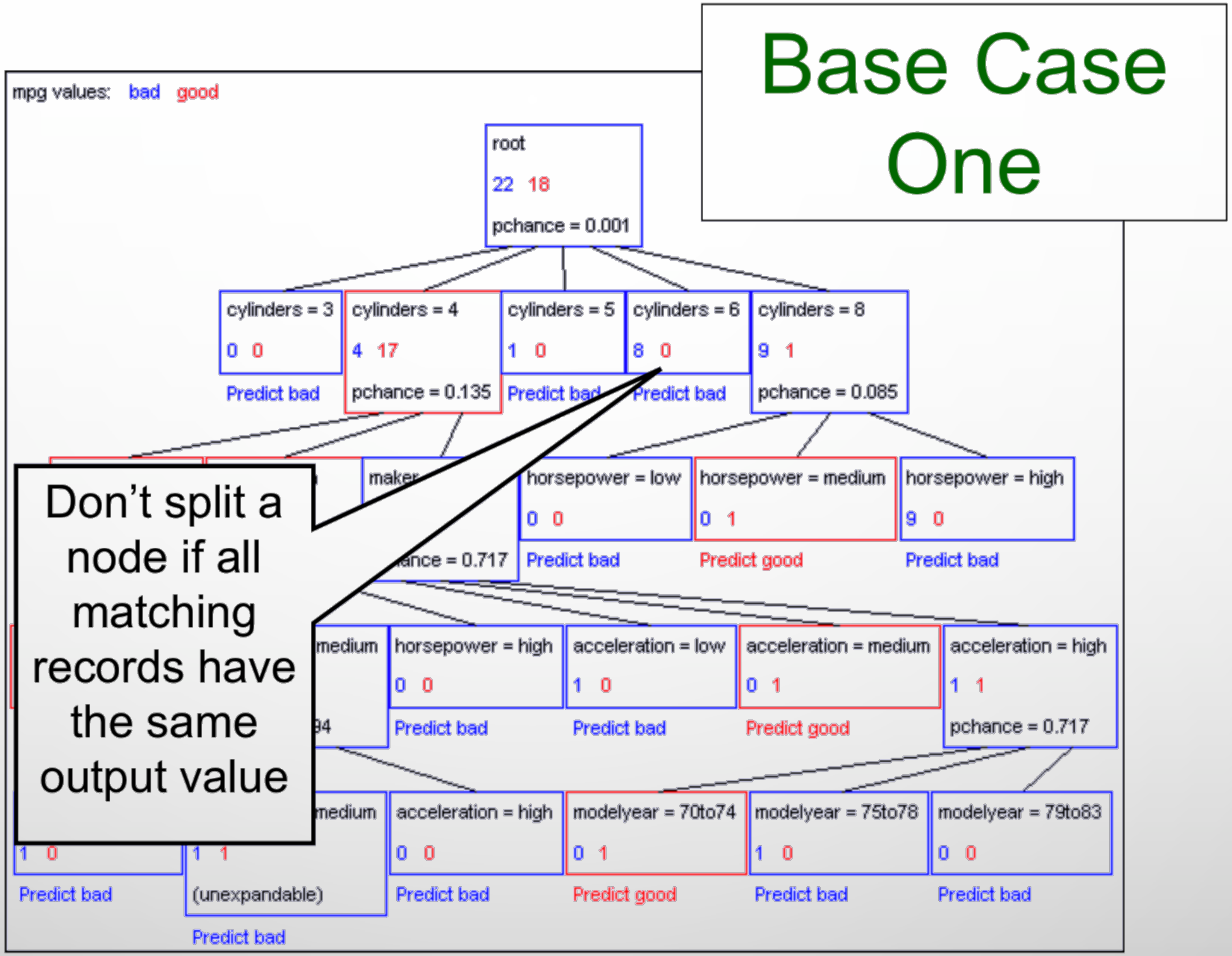
- Suppose we want to predict MPG.
- Now, Look at all the information gains…



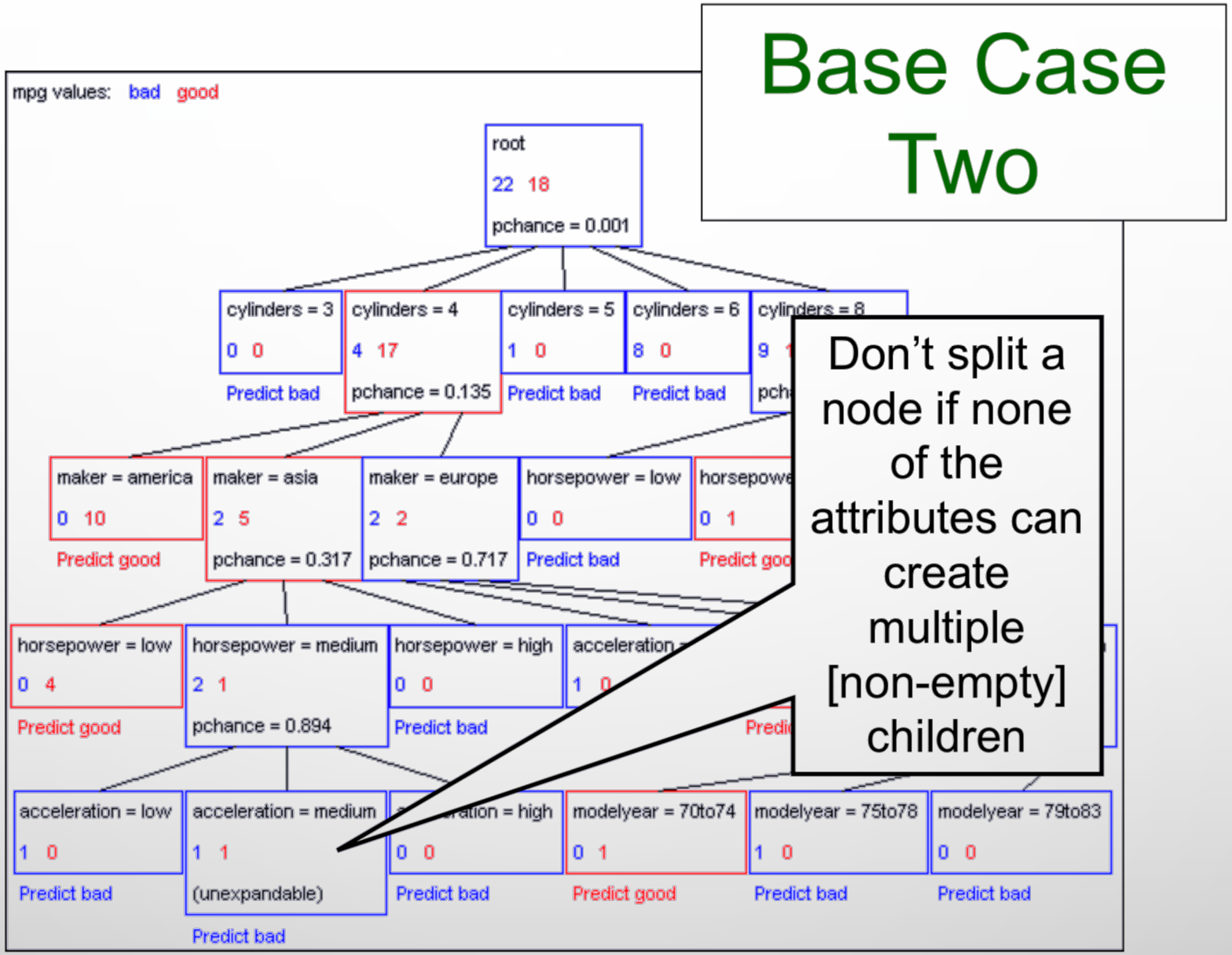
Tree After One Iteration



When to Terminate?



When to terminate? (cont.)



Base Cases: An idea

- 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.

Proposed Base Case 3:
If all attributes have zero information gain then don't recurse

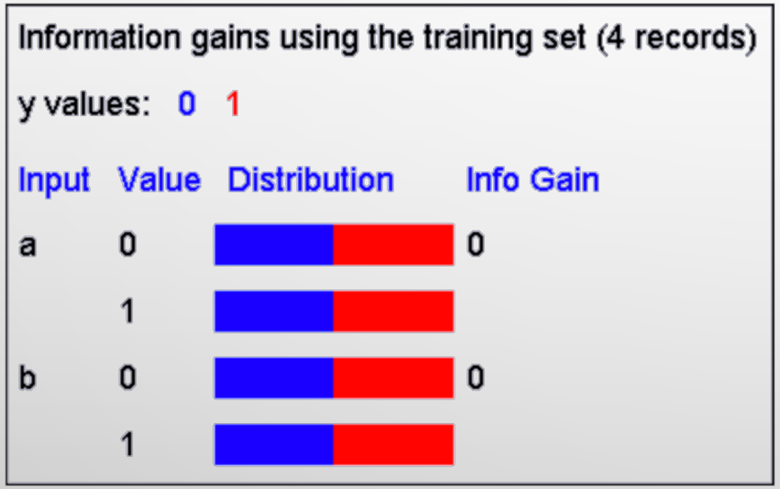
Is this a good idea?

The problem with Base Case 3

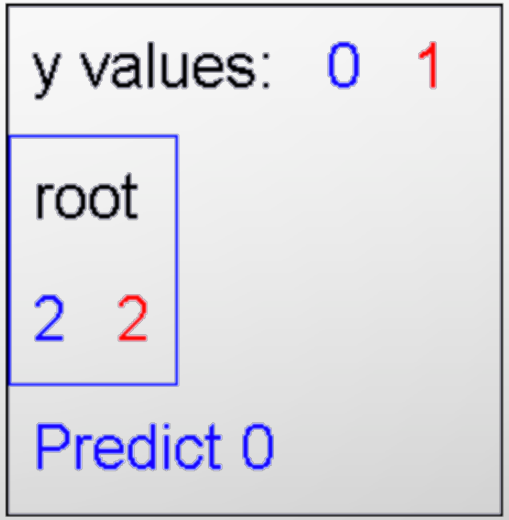
y = a XOR b

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

The information gains:



The resulting decision tree:

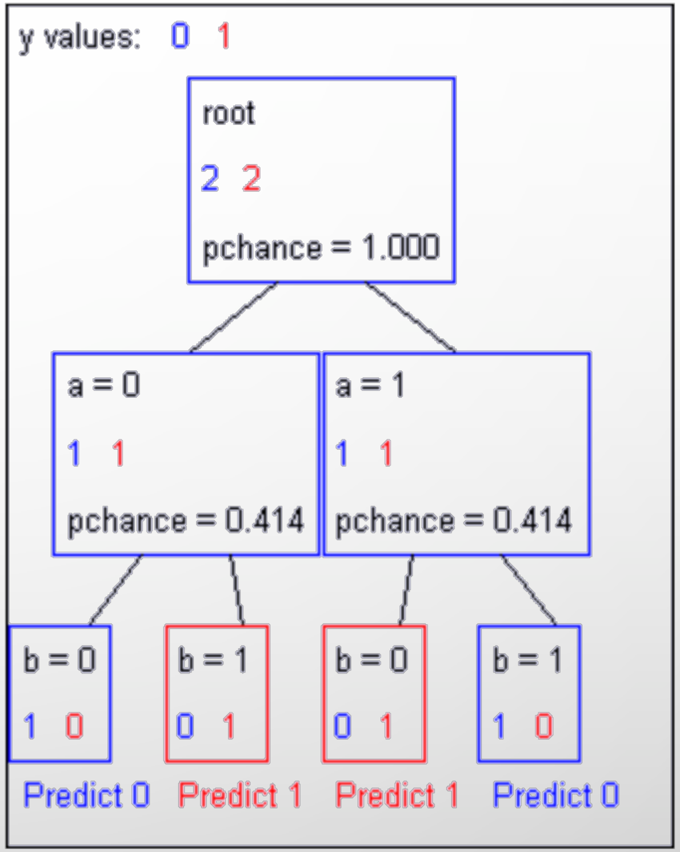


But *Without* Base Case 3:

y = a XOR b

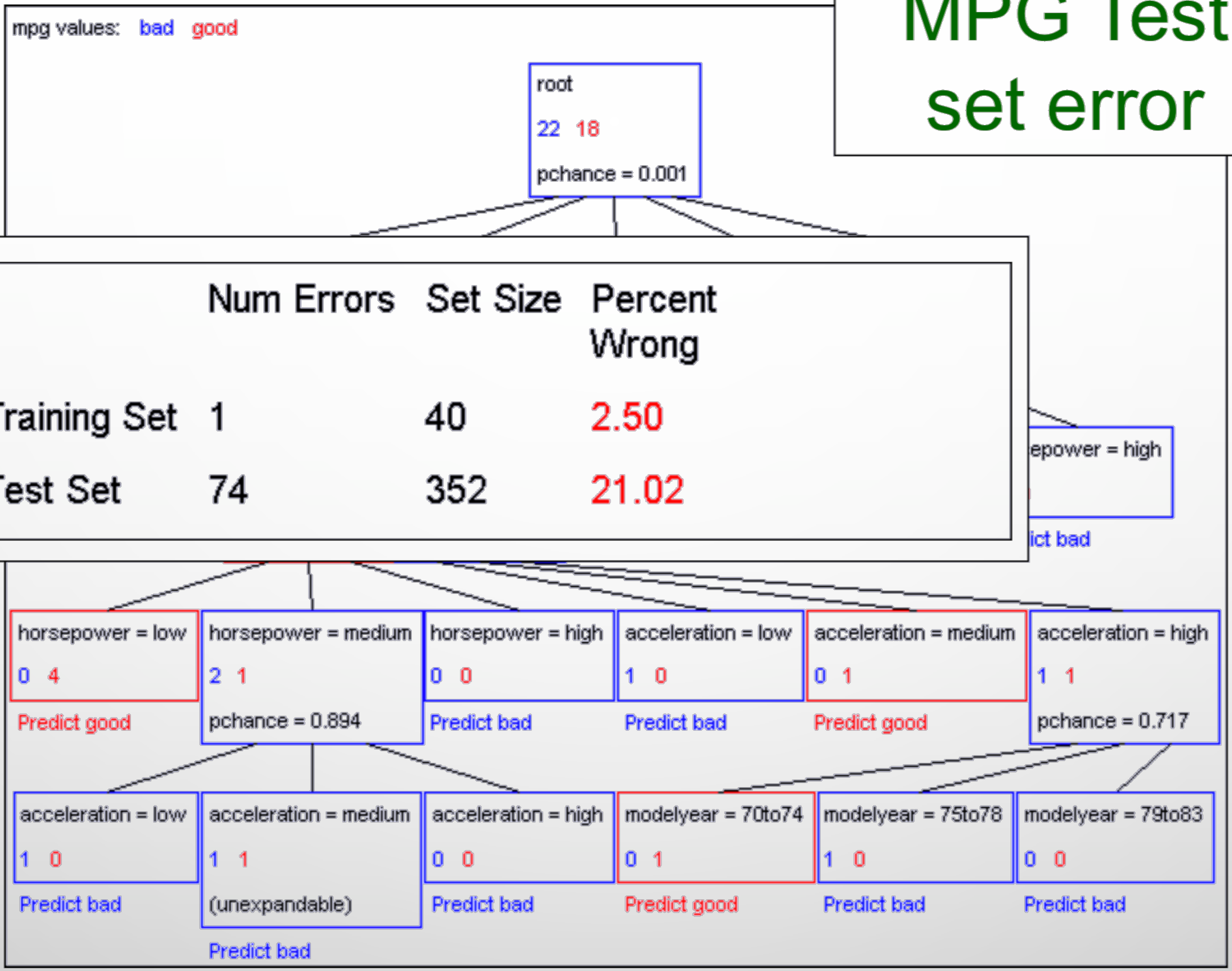
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

The resulting decision tree:



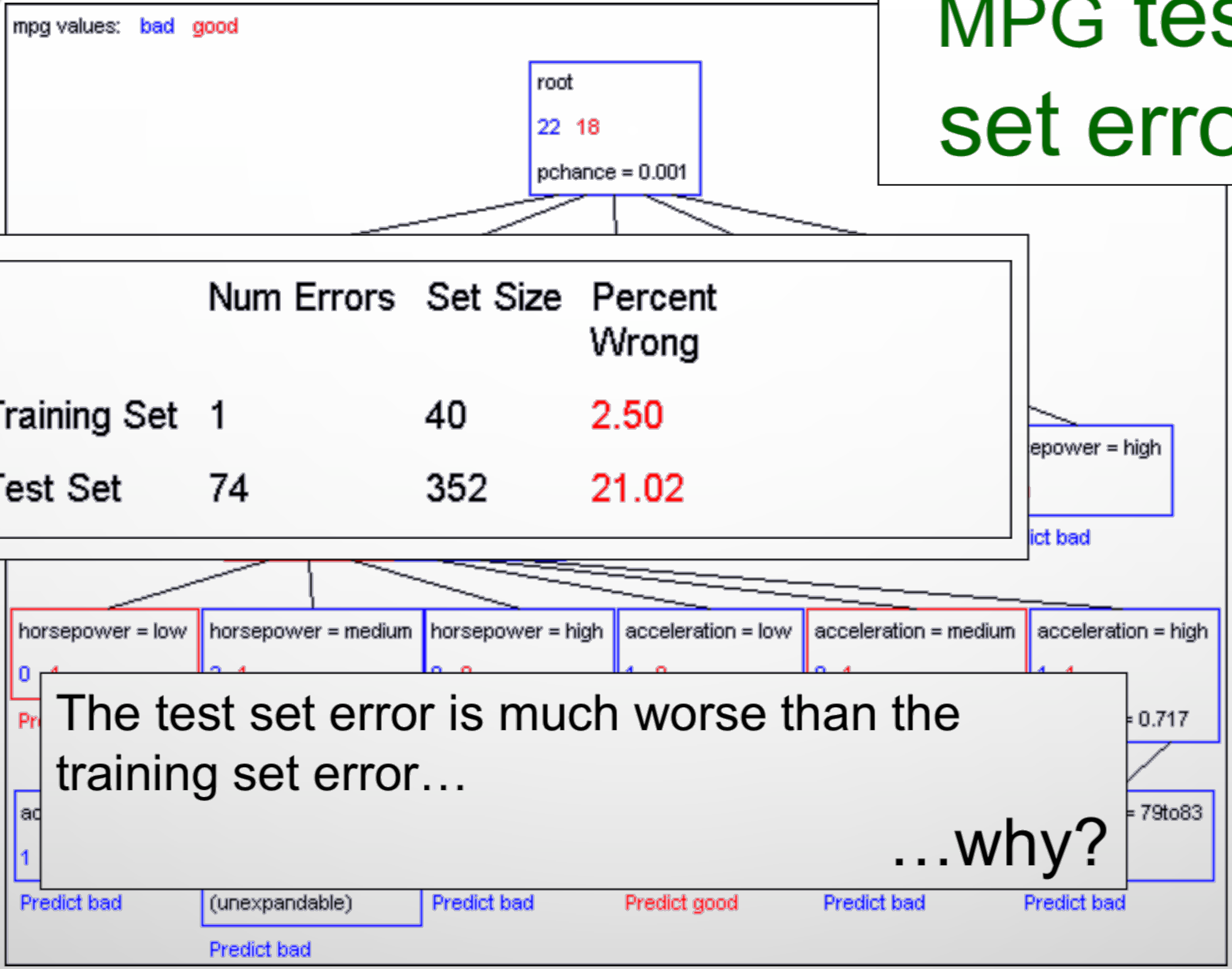
So: Base Case 3?
Include or Omit?

MPG Test set error



	Num Errors	Set Size	Percent Wrong
Training Set	1	40	2.50
Test Set	74	352	21.02

MPG test set error



	Num Errors	Set Size	Percent Wrong
Training Set	1	40	2.50
Test Set	74	352	21.02

The test set error is much worse than the training set error…

…why?

Decision trees will overfit

- Our decision trees have no learning bias
  - Training set error is always zero!
    - (If there is no label noise)
  - Lots of variance
  - Will definitely overfit!!!
  - Must introduce some bias towards *simpler* trees
- Why might one pick simpler trees?

Inductive bias

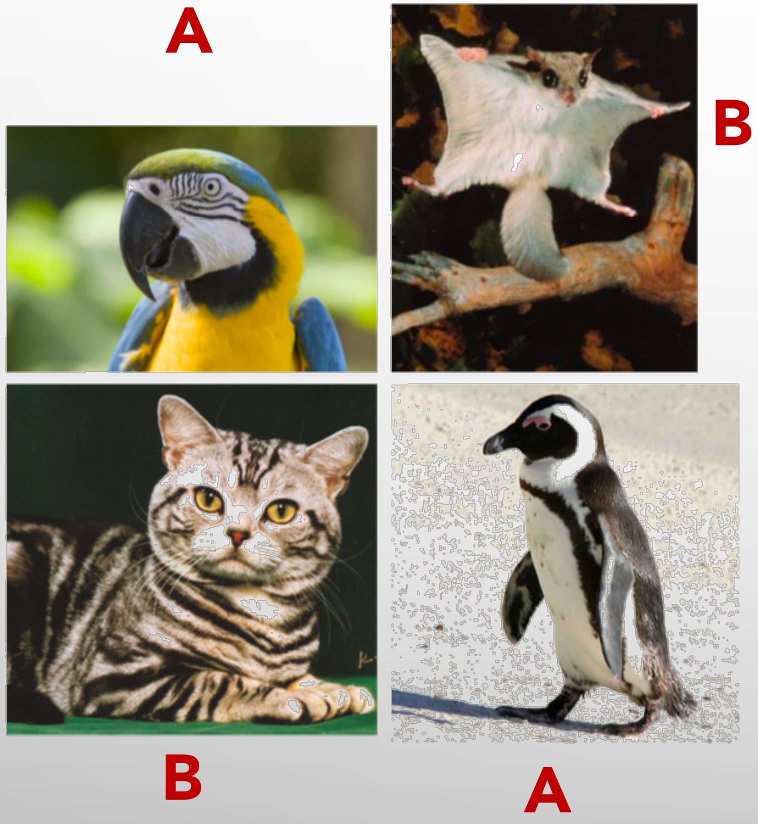
- Suppose that you are given 8 training samples for two classes A and B.

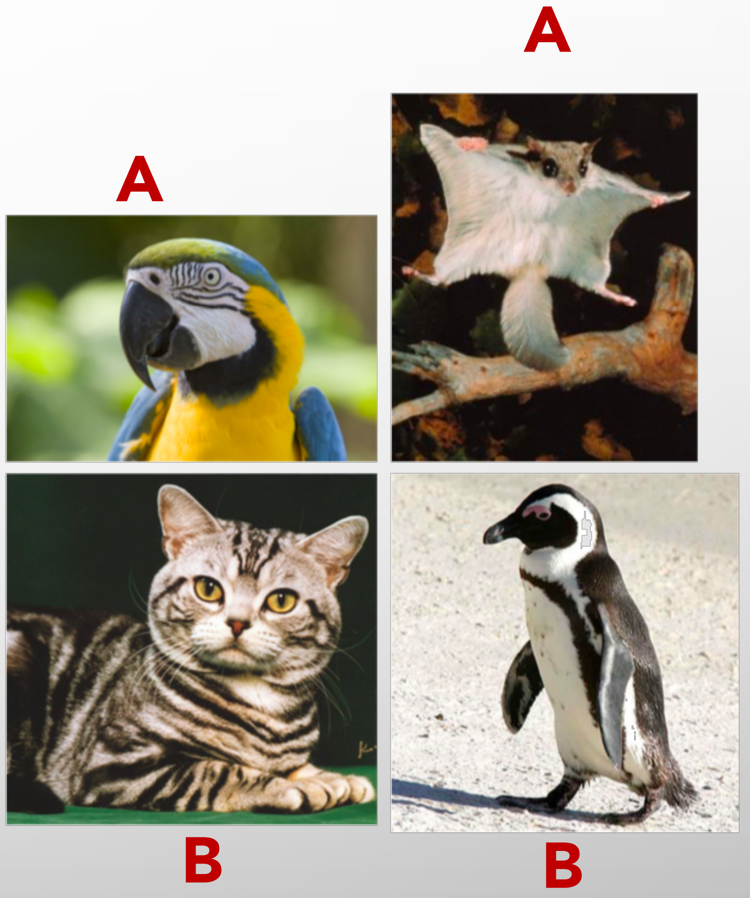
class A

class B

Inductive bias (cont.)

- What is your guess on the classes of the following test data?



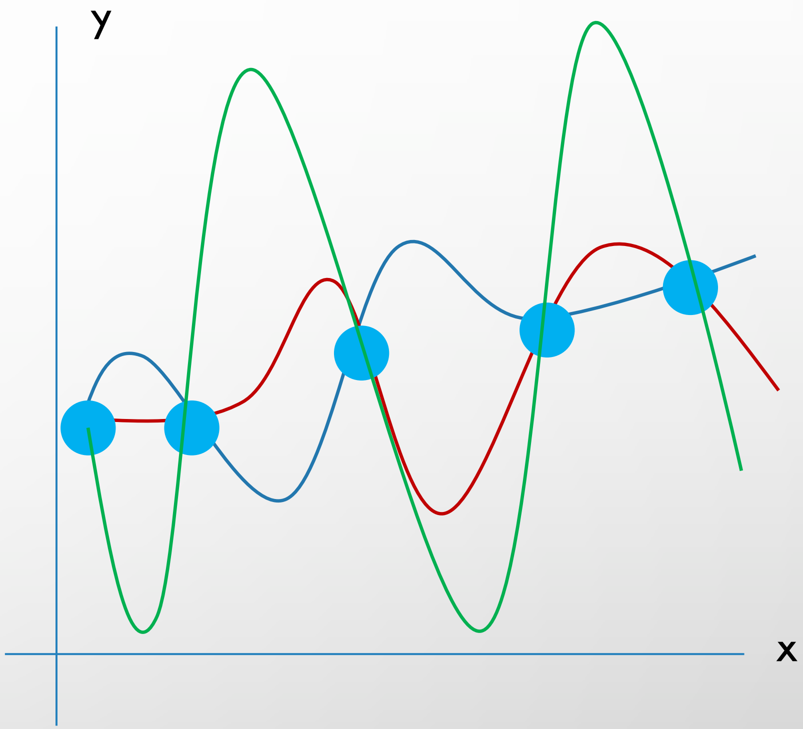


Inductive bias (cont.)

- Each person has a bias in learning (bird vs. Non-bird or flying vs. Non-flying).
- In the **absence** of data that narrows down the relevant concept, what type of solutions are we more likely to prefer?
- Different approaches that we introduce in this course are different types of biases.
- Suppose that we restrict depth of a decision tree. What would be the inductive bias?
- **Correct inductive bias is necessary for a problem to be learnable.**

“No free lunch” theorem

- Suppose that all the functions that are consistent with any given training data are **equally likely a solution to our induction**.
- Then all learning algorithms would have the same average true error on out-of-training-sample (D_o), where average is taken across different problems.
- This includes random guessing!
  - So in absence of any sense on what functions are more likely, learning is impossible!



Occam’s Razor

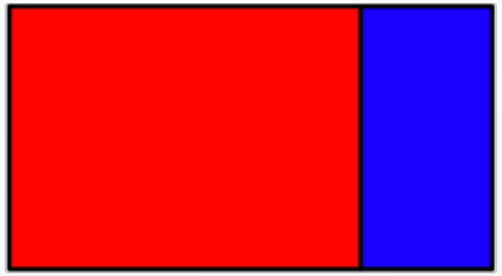
- Why Favor Short Hypotheses?
- Arguments for:
  - Fewer short hypotheses than long ones
  - → A short hyp. less likely to fit data by coincidence
  - → Longer hyp. that fit data might be coincidence

How to Build Small Trees

- Several reasonable approaches:
- **Stop growing tree before overfit**
  - Bound depth or # leaves
  - Base Case 3
  - *Doesn't work well in practice*
- **Grow full tree; then prune**
  - **Optimize on a held-out (development set)**
    - If growing the tree hurts performance, then cut back
    - Con: Requires a larger amount of data…
  - **Use statistical significance testing**
    - Test if the improvement for any split is likely due to noise
    - If so, then prune the split!

Reduced Error Pruning

- Split data into ***training*** & ***validation*** sets (10-33%)



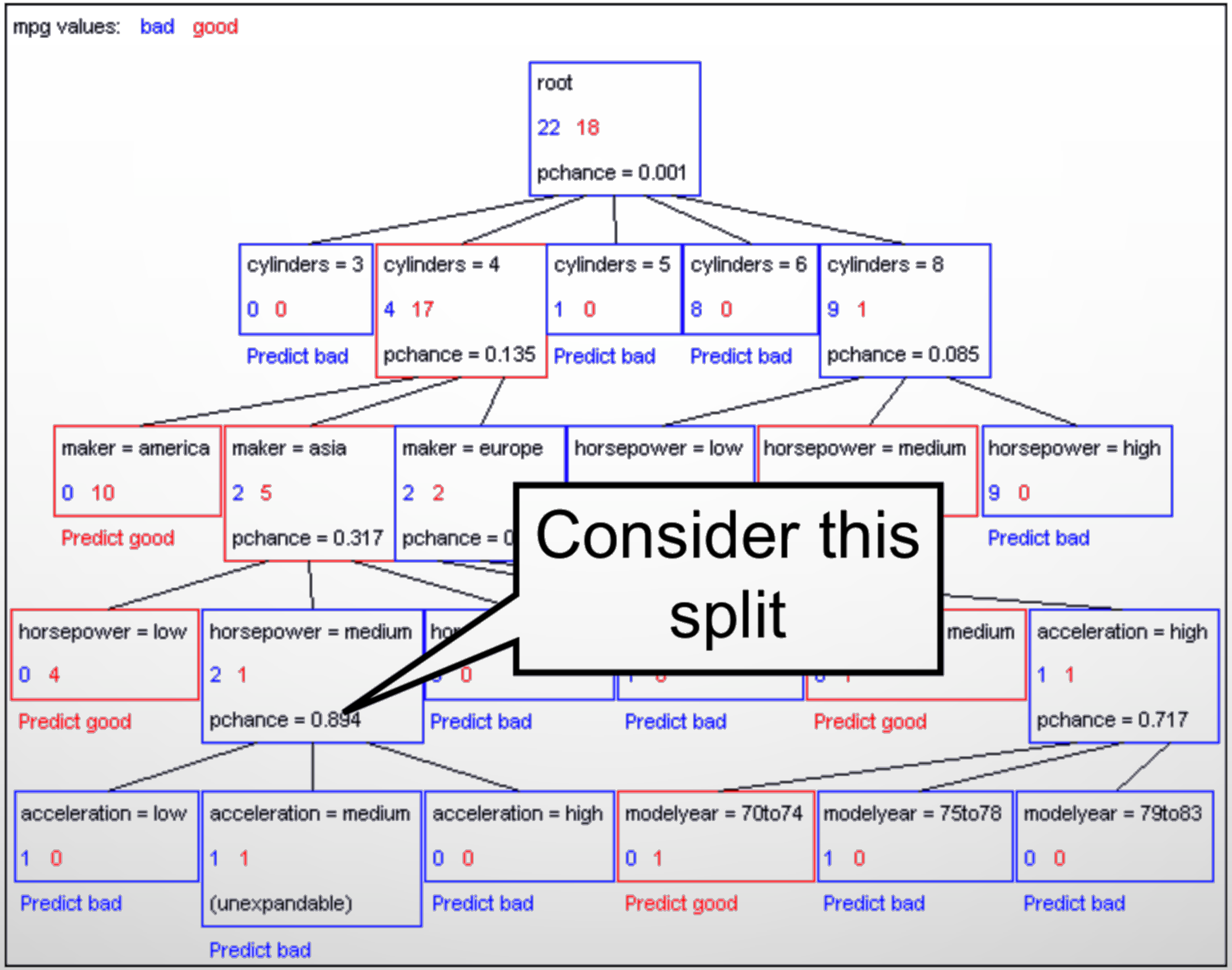
- Train on training set (overfitting)
- Do until further pruning is harmful:

  1) Evaluate effect on validation set of pruning ***each*** possible node (and tree below it)

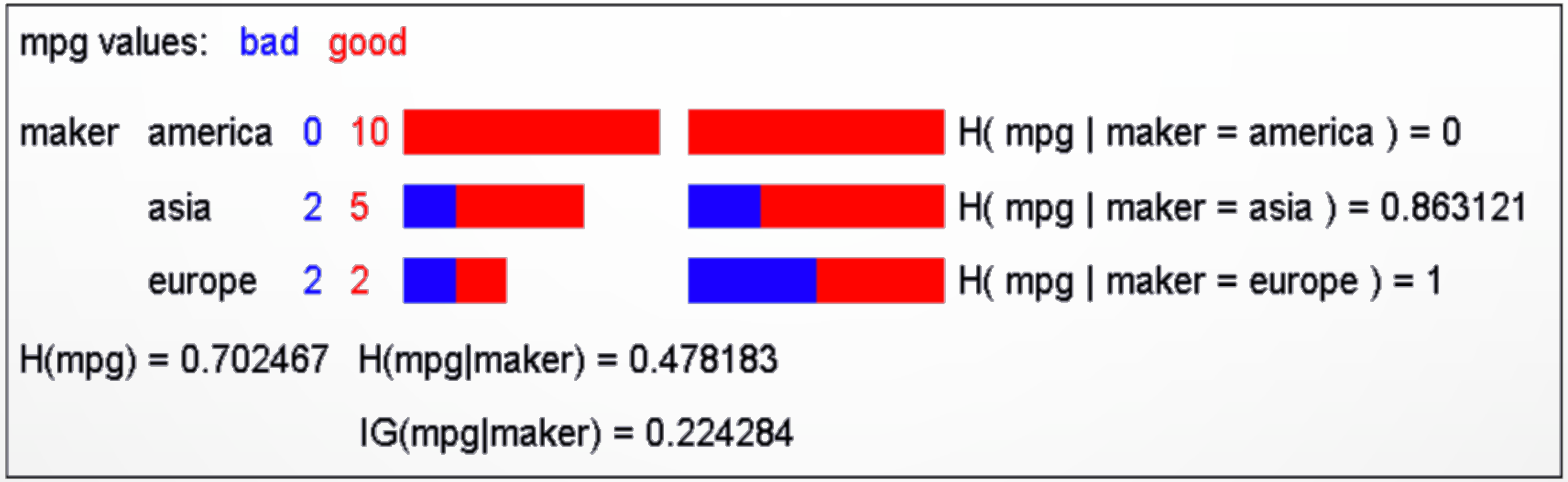
  2) Greedily remove the node that ***most improves accuracy of validation set***

Alternatively

- Chi-squared pruning
  - Grow tree fully
  - Consider leaves in turn
    - Is parent split worth it?


mpg values: bad good
root
22 18
pchance = 0.001
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
horsepower = low
horsepower = medium
horsepower = high
9 0
Predict bad
horsepower = low
0 4
Predict good
horsepower = medium
2 1
pchance = 0.894
Predict bad
Predict bad
medium
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
Consider this split

A chi-square 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?
- By using a particular kind of chi-square test, the answer is 13.5%. Such hypothesis tests are relatively easy to compute, but involved

Using Chi-squared 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 $p_{chance} > MaxPchance$
  - Continue working you way up until there are no more prunable nodes

- *MaxPchance* is a magic parameter you must specify to the decision tree, indicating your willingness to risk fitting noise

Regularization