# CSE446: Decision Tree Part2

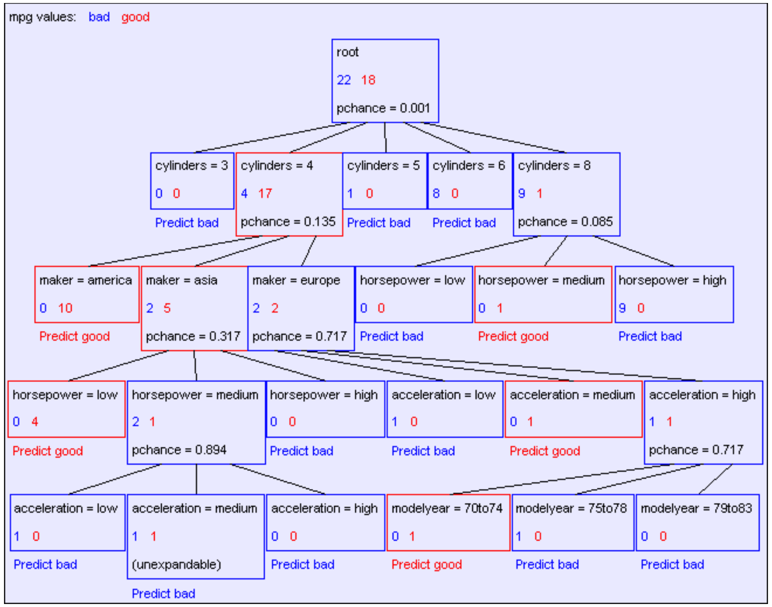
Winter 2016

Ali Farhadi

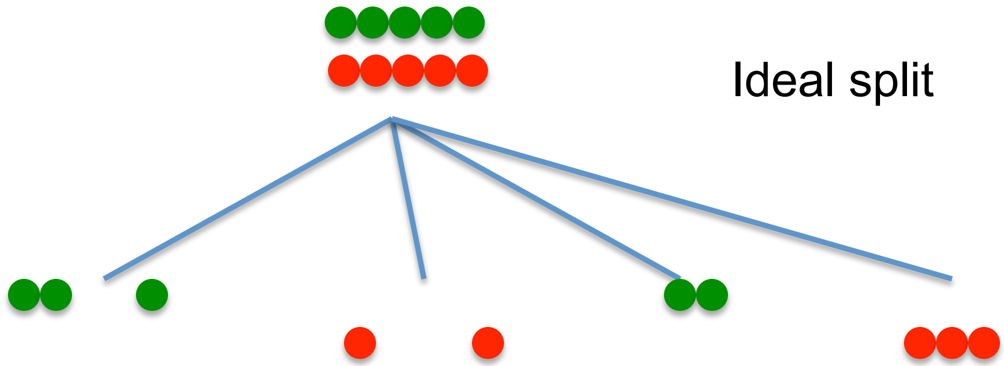
Slides adapted from Carlos Guestrin, Andrew Moore, and Luke Zettelmoyer

# So far …

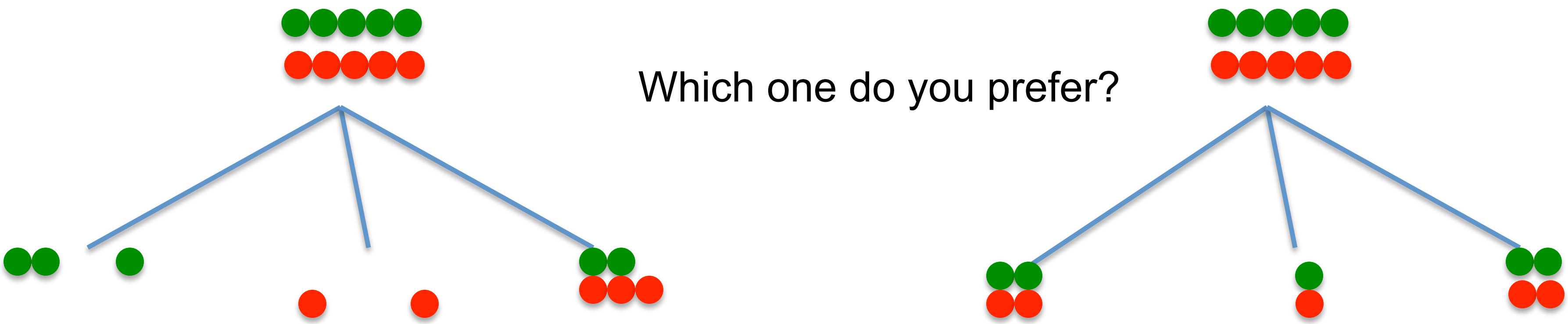
- Decision trees
- They will overfit
- How to split?
- When to stop?



# What defines a good attribute?



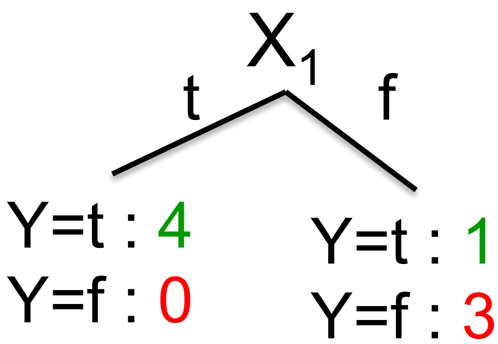
Ideal split

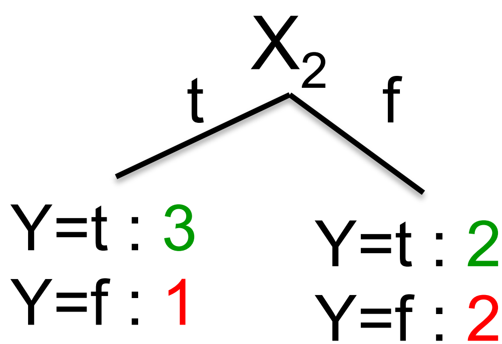


Which one do you prefer?

# 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) = 1/4 | P(Y=B) = 1/4 | P(Y=C) = 1/4 | P(Y=D) = 1/4 |

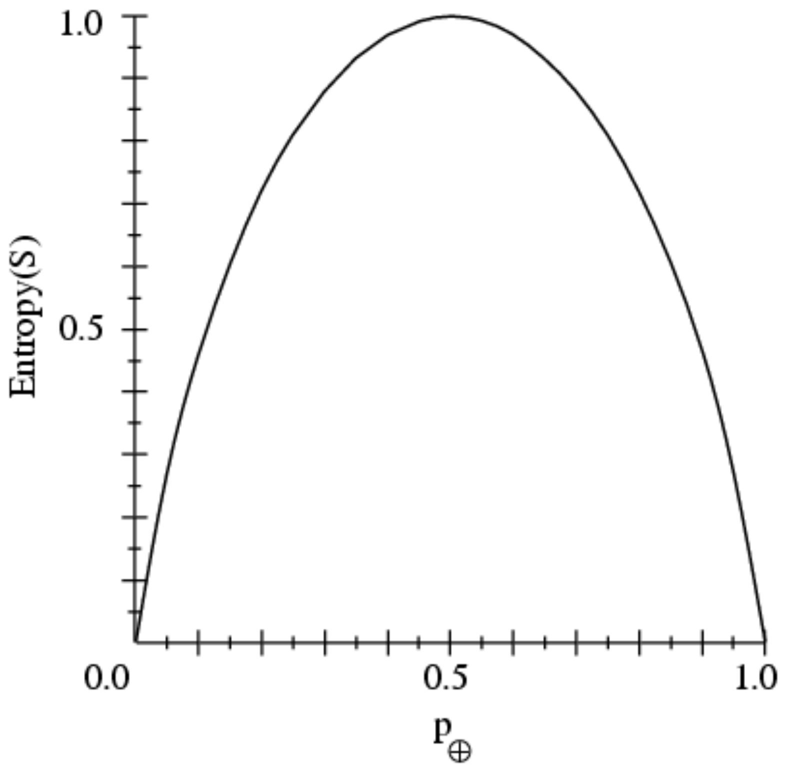
# Entropy

Entropy $H(Y)$ of a random variable $Y$

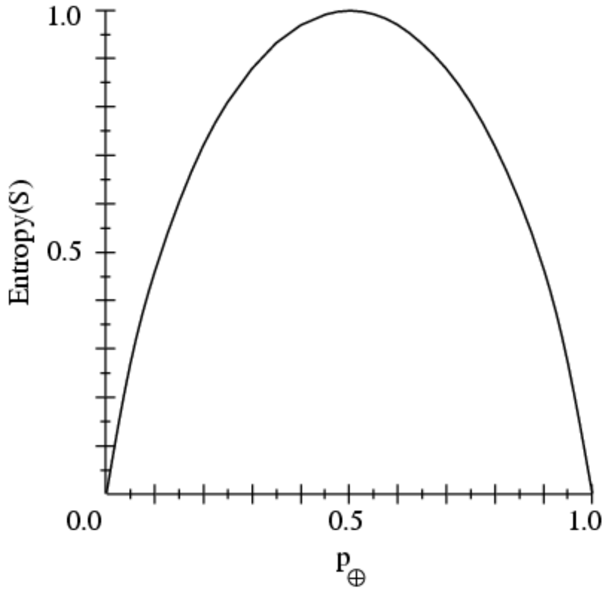
$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

***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




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

P(Y=t) = 5/6

P(Y=f) = 1/6

$H(Y) = -\ 5/6 \log_2 5/6 - 1/6 \log_2 1/6$

$= 0.65$

| $X_1$ | $X_2$ | Y |
|---|---|---|
| T | T | T |
| T | F | T |
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

# Conditional Entropy

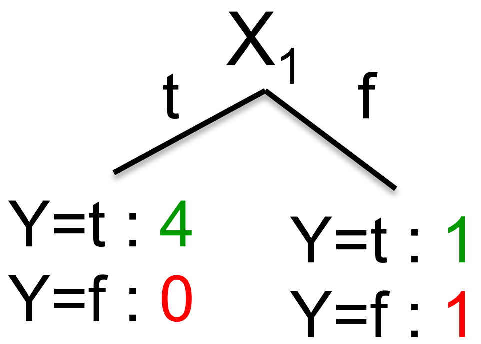
Conditional Entropy $H(Y|X)$ of a random variable $Y$ conditioned on a random variable $X$

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

## 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 |

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

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

$= 2/6$

# Information gain

Decrease in entropy (uncertainty) after splitting

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

- IG(X) is non-negative (>=0)
- Prove by showing H(Y|X) <= H(X), with Jensen's inequality

In our running example:

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

$IG(X_1) > 0$ → 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

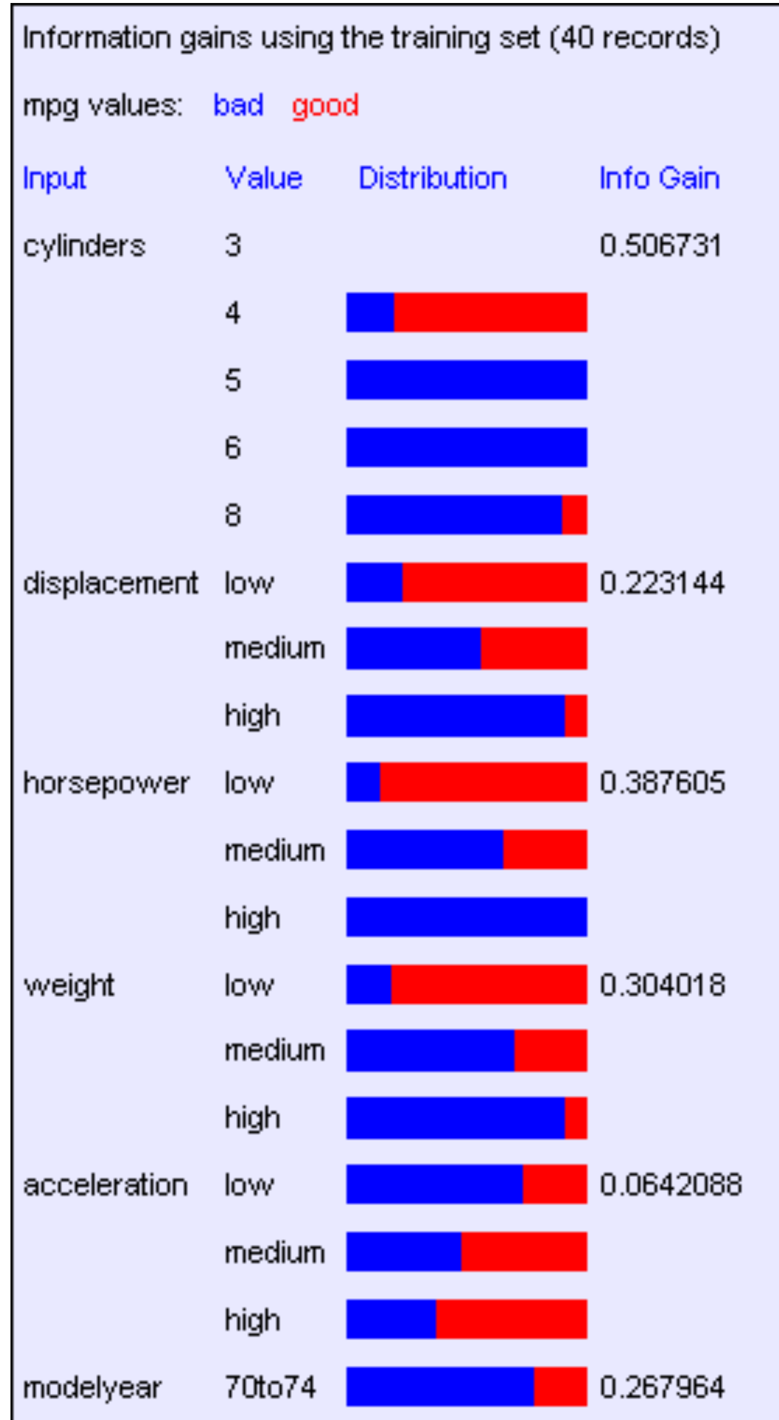
- Start from empty decision tree
- Split on **next best attribute (feature)**
  - Use, for example, information gain to select attribute:

$$\arg\max_i IG(X_i) = \arg\max_i H(Y) - H(Y \mid X_i)$$

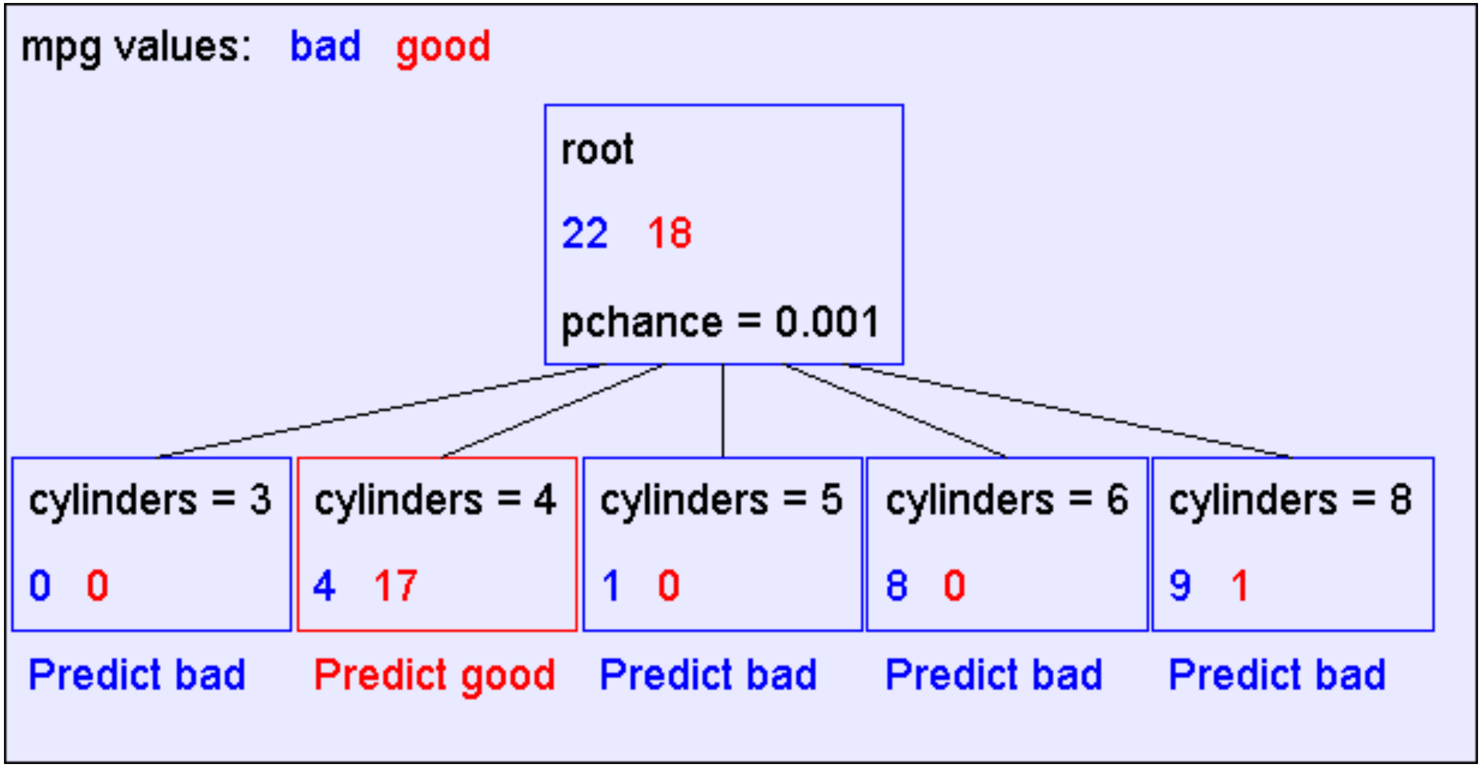
- Recurse

Suppose we want to predict MPG

Look at all the information gains…

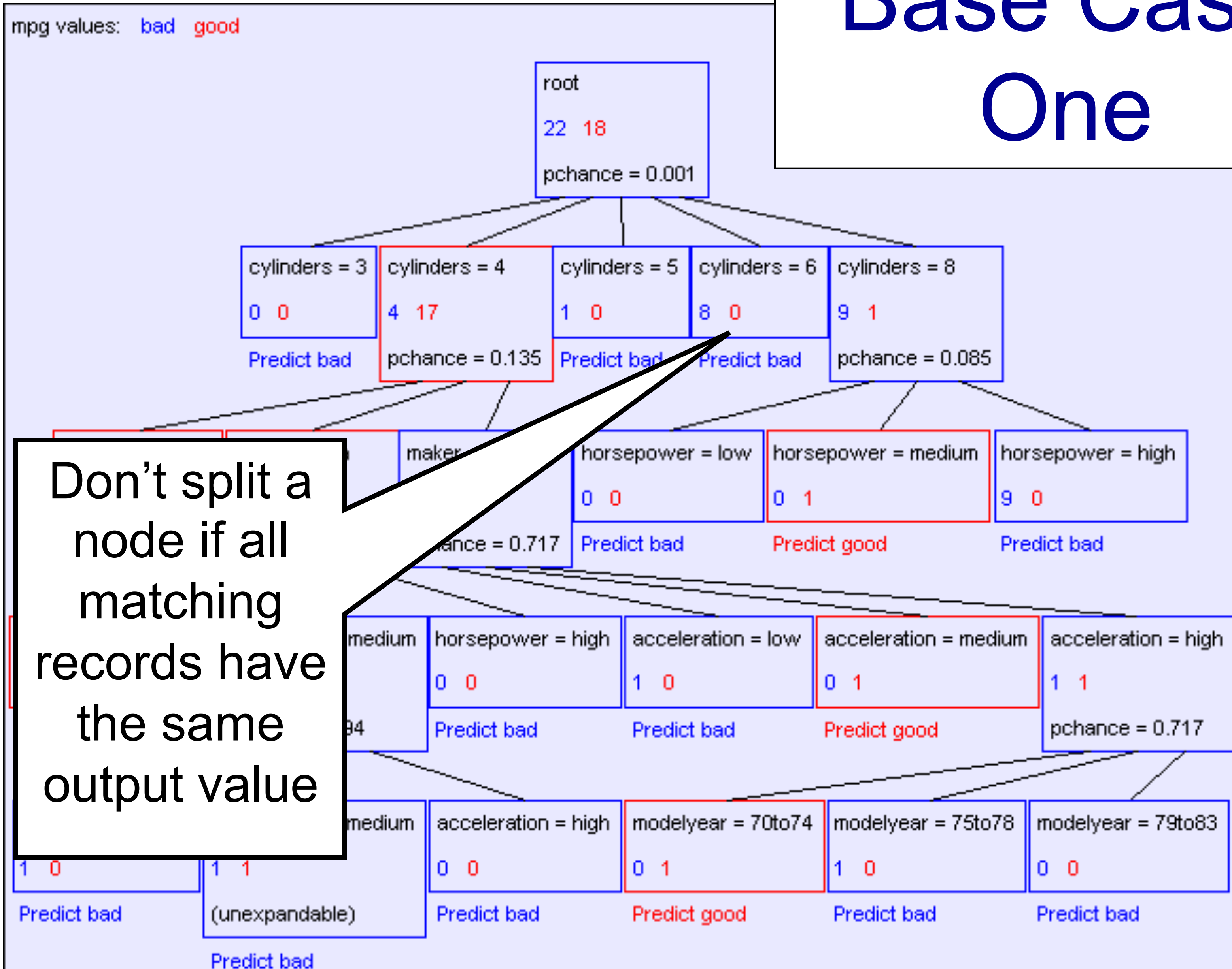


# A Decision Stump



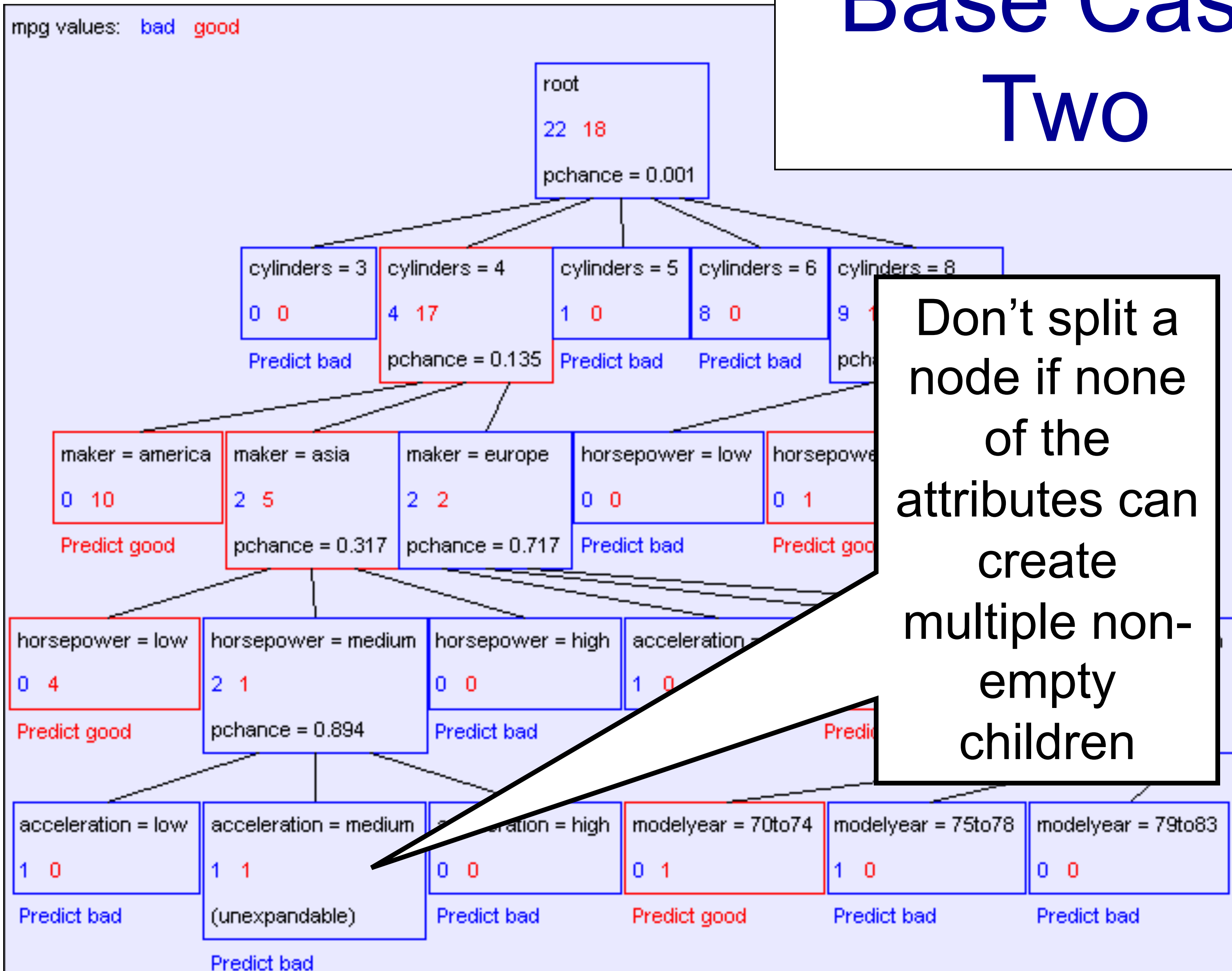

First split looks good! But, when do we stop?

# Base Case One



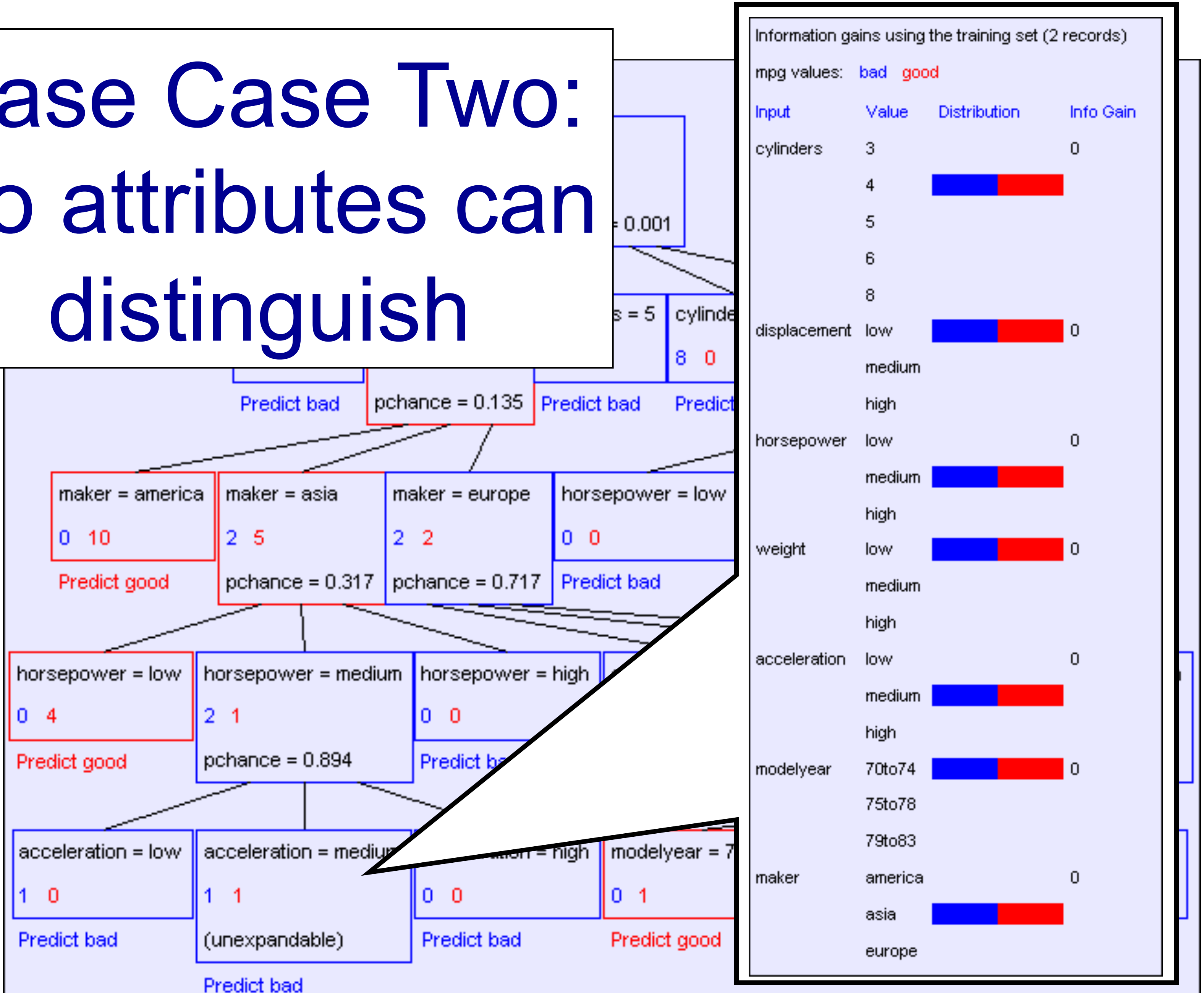

Don't split a node if all matching records have the same output value

# Base Case Two




Don’t split a node if none of the attributes can create multiple non-empty children

# Base Case Two: No attributes can distinguish



# 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

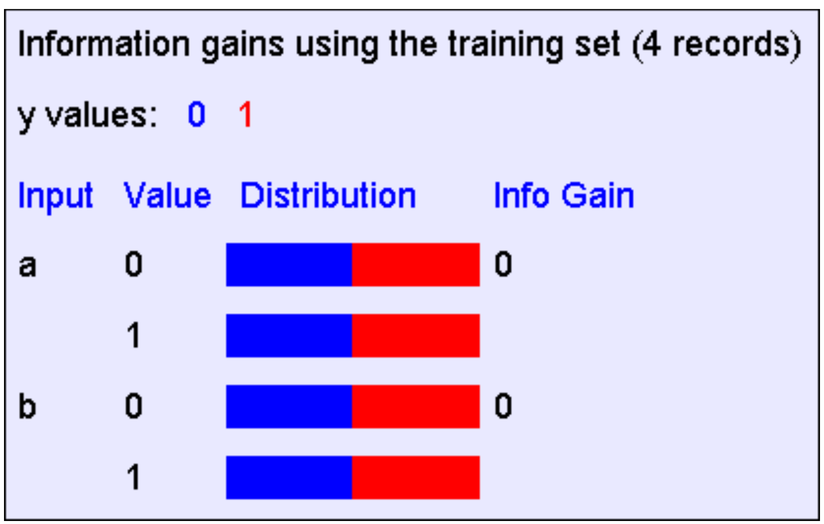
- *Is this a good idea?*

# The problem with Base Case 3

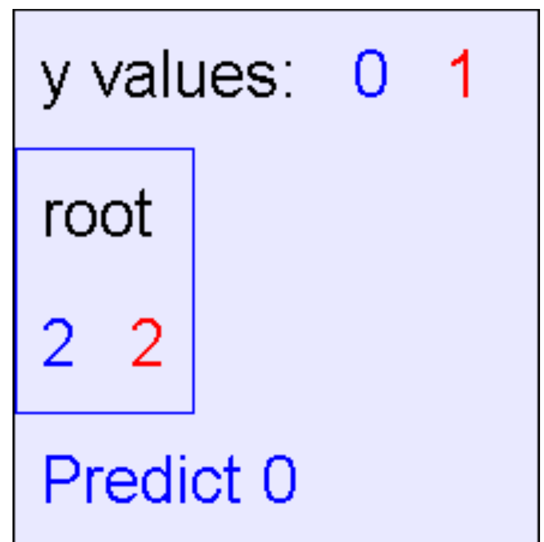
y = a XOR b

| a | b | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

The information gains:




The resulting decision tree:

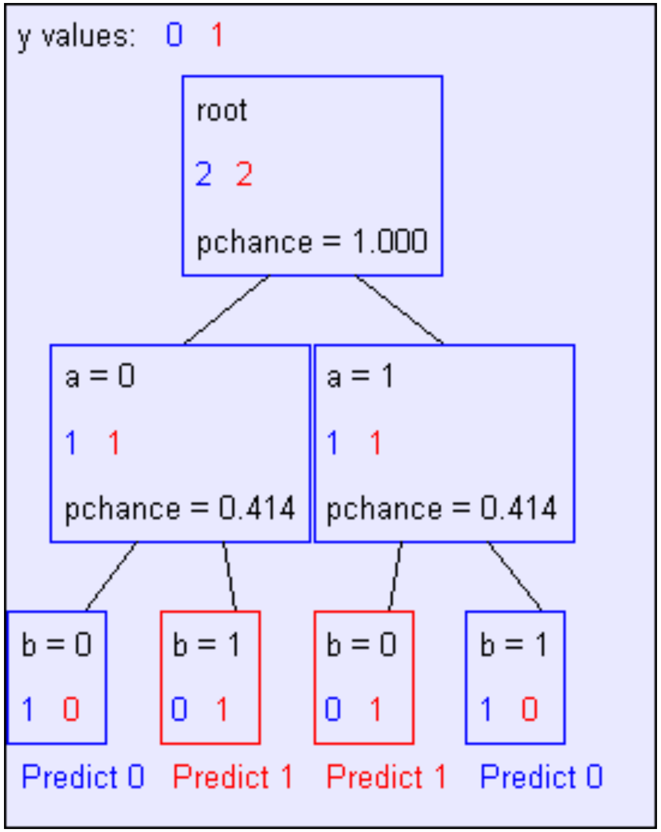


# If we omit Base Case 3:

y = a XOR b

| a | b | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

The resulting decision tree:




Is it OK to omit Base Case 3?

# Summary: Building Decision Trees

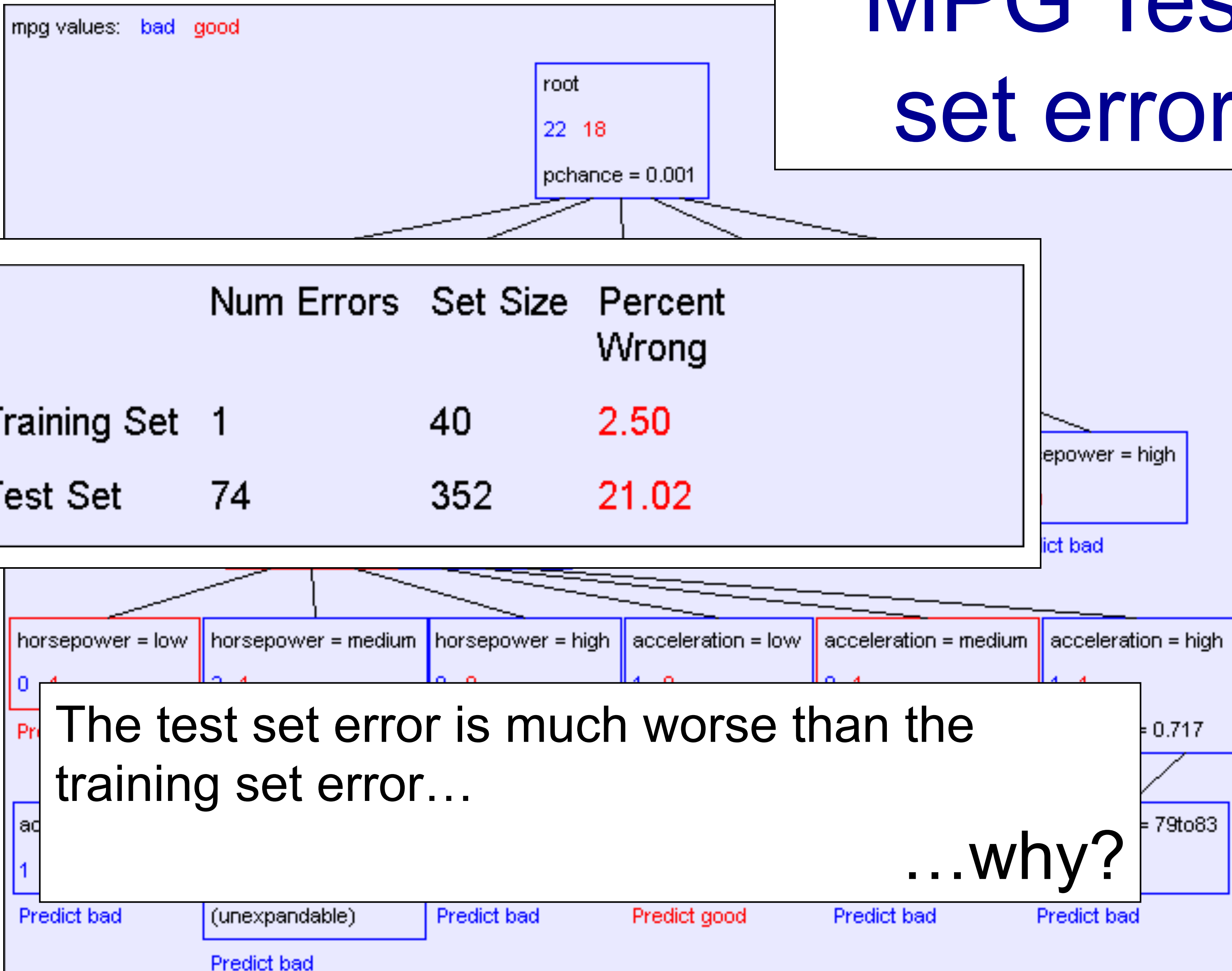
BuildTree(*DataSet,Output*)

- 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 a non-leaf node with $n_X$ children.
  - The *i*'th child should be built by calling

    BuildTree($DS_i$,*Output*)

    Where $DS_i$ contains the records in DataSet where X = *i*th value of X.

# MPG Test set error




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

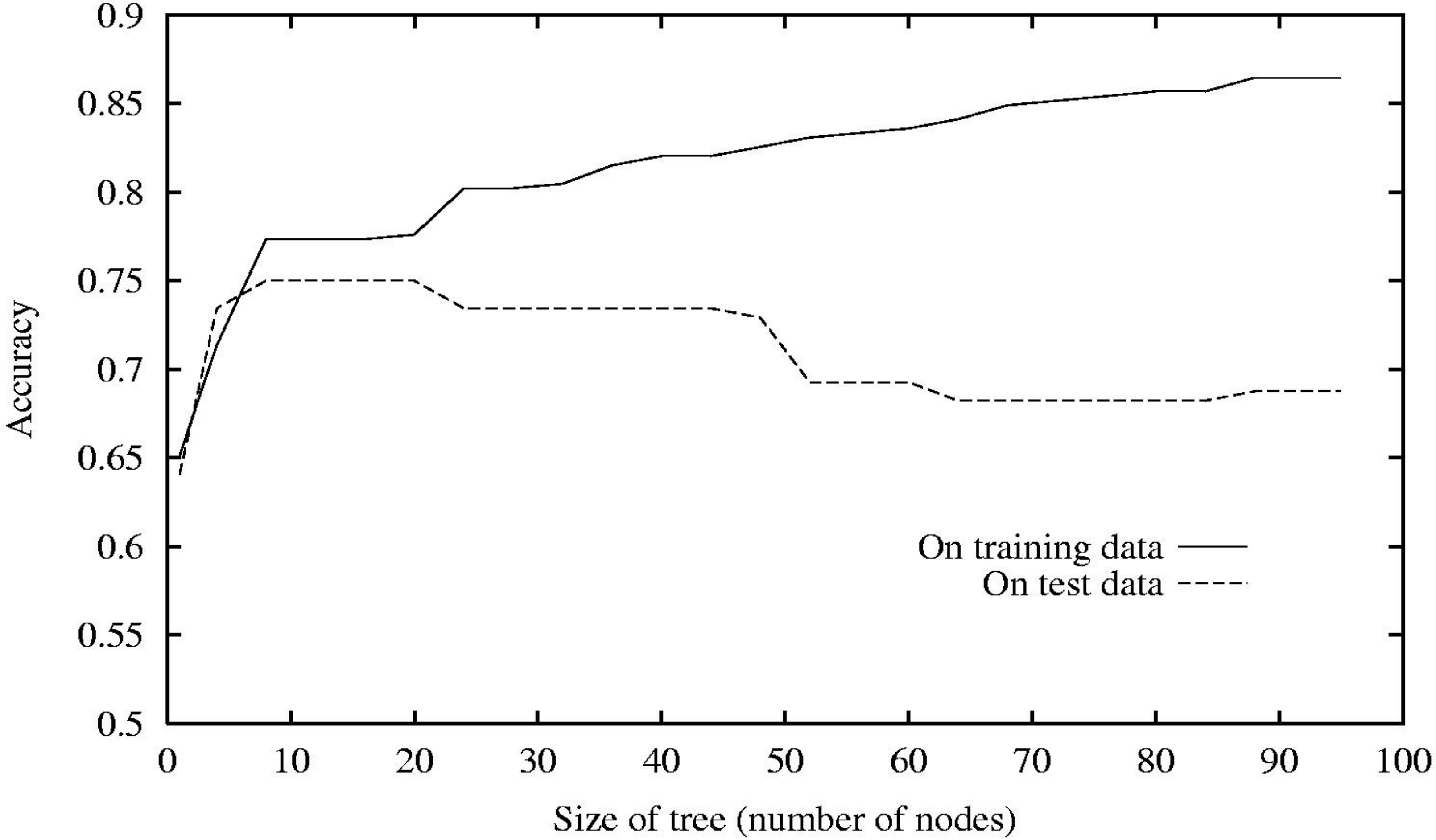
The test set error is much worse than the training set error…

…why?

# Decision trees will overfit!!!

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

# Decision trees will overfit!!!



# One Definition of Overfitting

- Assume:
  - Data generated from distribution $D(X,Y)$
  - A hypothesis space $H$
- Define errors for hypothesis $h \in H$
  - Training error: $error_{train}(h)$
  - Data (true) error: $error_D(h)$

- We say $h$ ***overfits*** the training data if there exists an $h' \in H$ such that:

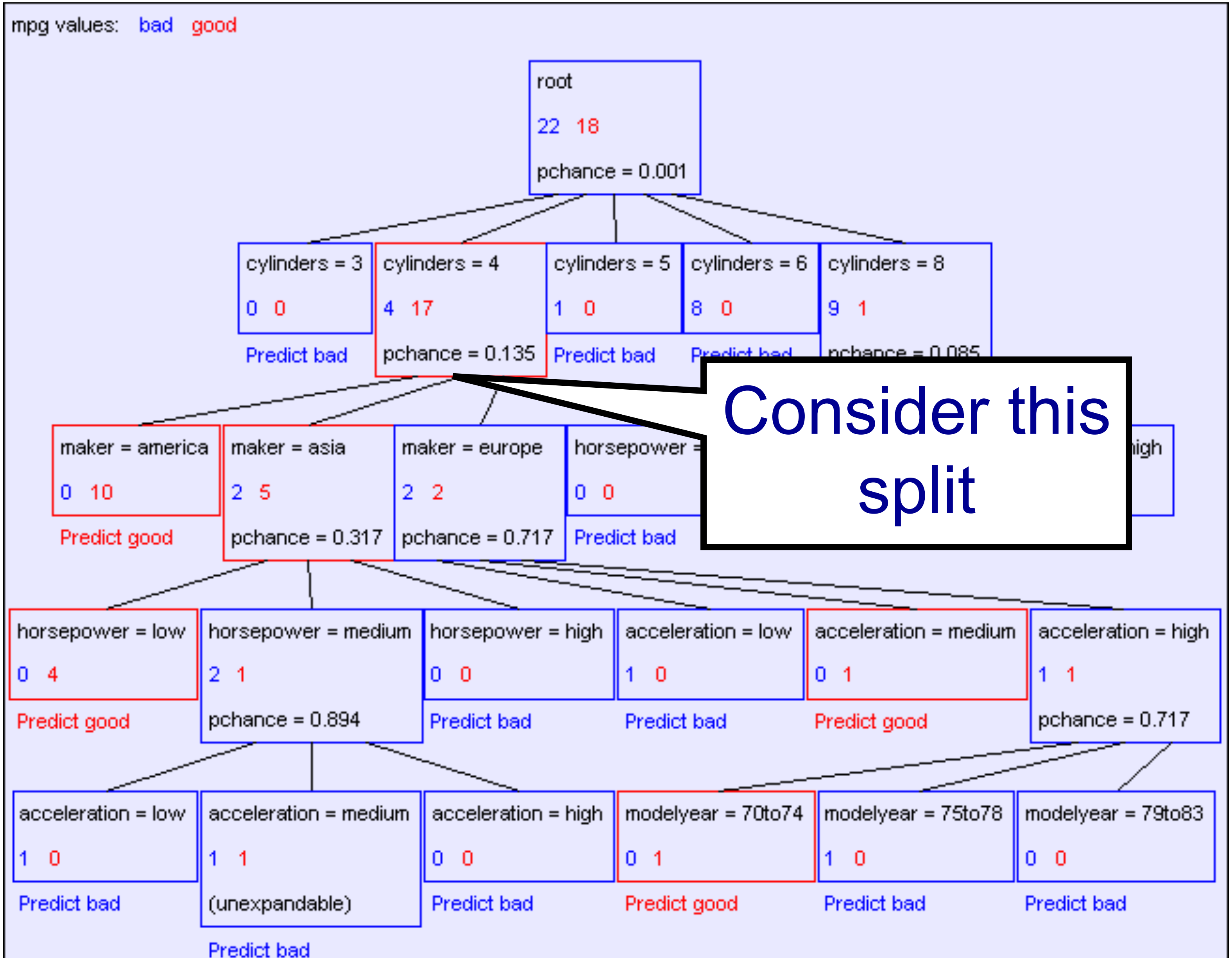
$$error_{train}(h) < error_{train}(h')$$

and

$$error_D(h) > error_D(h')$$

# 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 may might be coincidence
- Arguments against:
  - Argument above really uses the fact that hypothesis space is small!!!
  - What is so special about small sets based on the size of each hypothesis?

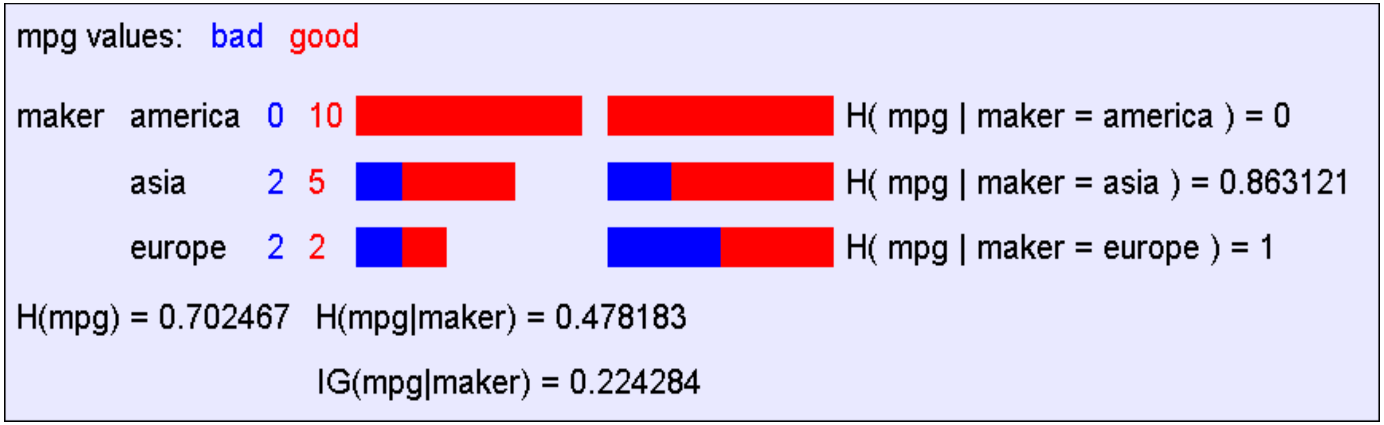

mpg values: bad good
root
22 18
pchance = 0.001
cylinders = 3
0 0
Predict bad
cylinders = 4
4 17
pchance = 0.135
cylinders = 5
1 0
Predict bad
cylinders = 6
8 0
Predict bad
cylinders = 8
9 1
pchance = 0.085
maker = america
0 10
Predict good
maker = asia
2 5
pchance = 0.317
maker = europe
2 2
pchance = 0.717
horsepower =
0 0
Predict bad
high
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
Consider this split

# How to Build Small Trees

Two reasonable approaches:

- Optimize on the held-out (development) set
  - If growing the tree larger hurts performance, then stop growing!!!
  - Requires a larger amount of data…
- Use statistical significance testing
  - Test if the improvement for any split it likely due to noise
  - If so, don't do the 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%

We will not cover Chi Square tests in class. See page 93 of the original ID3 paper [Quinlan, 86].

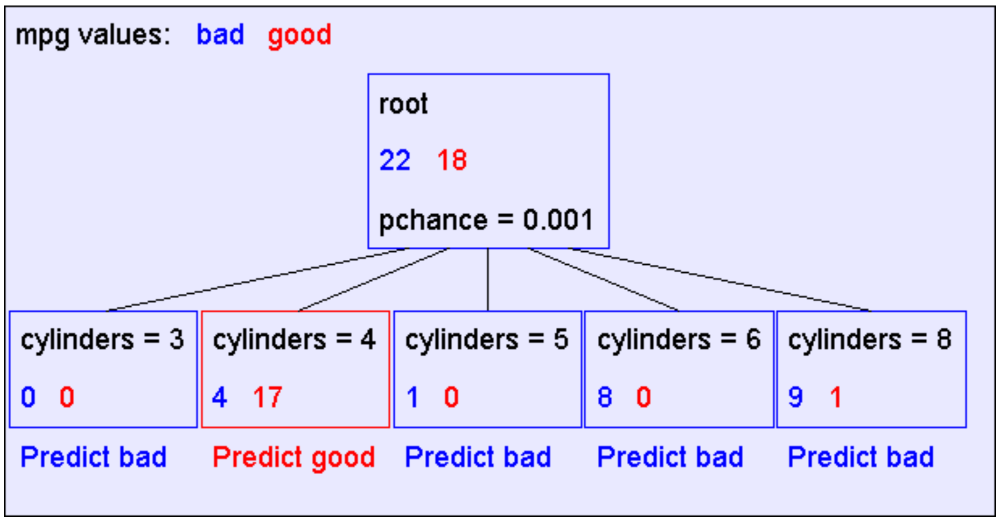
# Using Chi-squared to avoid overfitting

- 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

# Pruning example

- With MaxPchance = 0.05, you will see the following MPG decision tree:

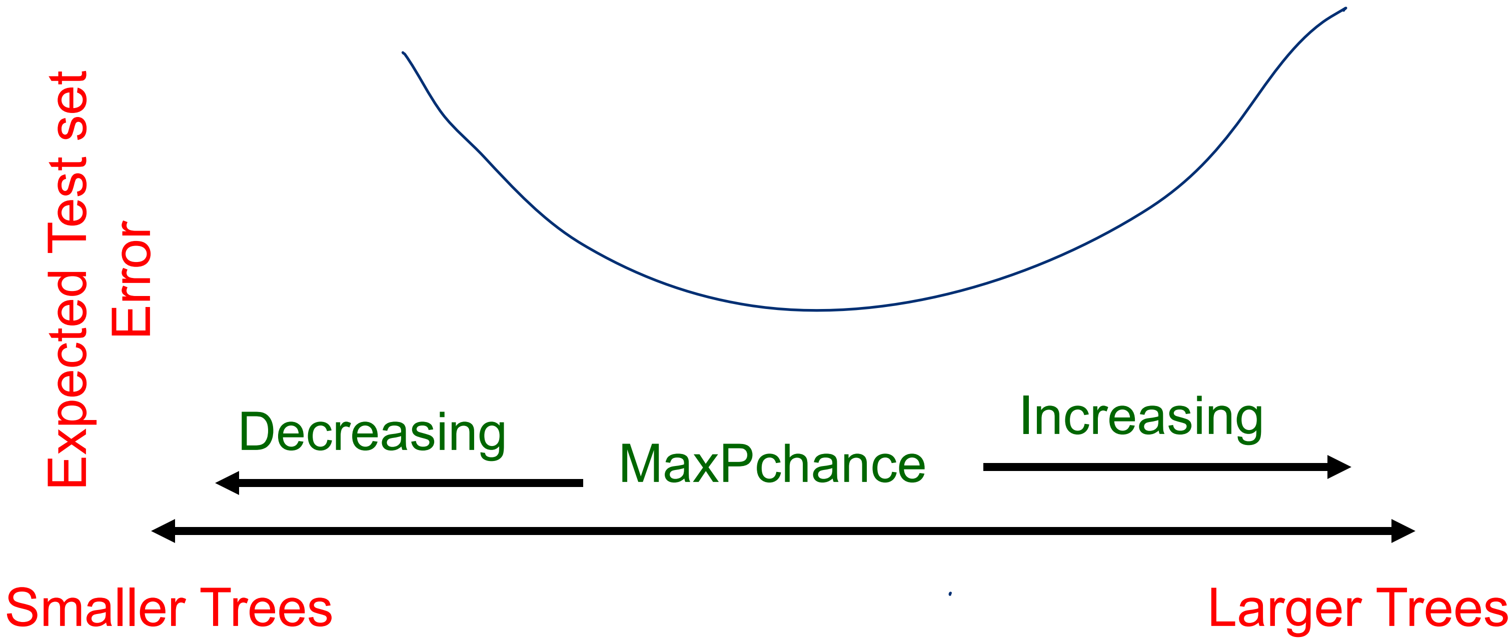



When compared to the unpruned tree
- improved test set accuracy
- worse training accuracy

| | Num Errors | Set Size | Percent Wrong |
|---|---|---|---|
| Training Set | 5 | 40 | 12.50 |
| Test Set | 56 | 352 | 15.91 |

# MaxPchance

- Technical note: MaxPchance is a regularization parameter that helps us bias towards simpler models




We'll learn to choose the value of magic parameters like this one later!

# Real-Valued inputs

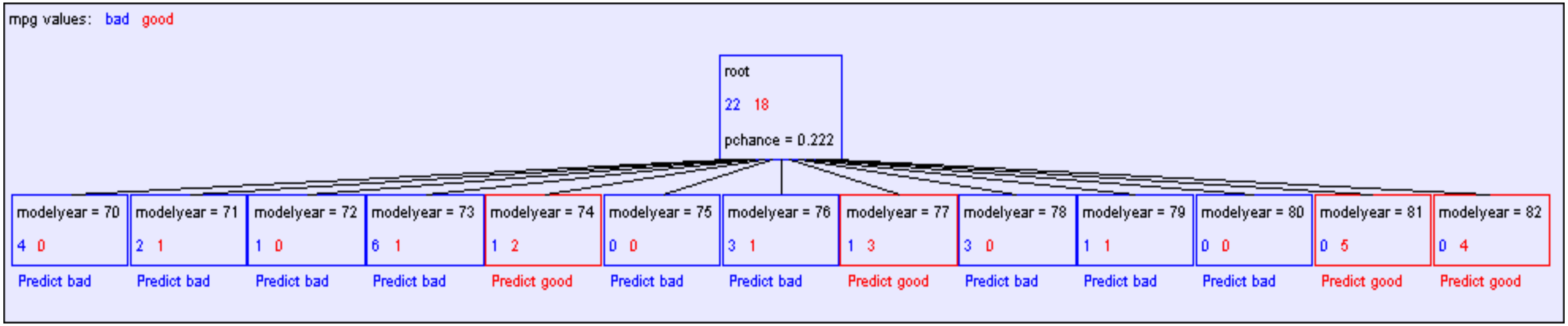
What should we do if some of the inputs are real-valued?

Infinite number of possible split values!!!

Finite dataset, only finite number of relevant splits!

| mpg | cylinders | displacemen | horsepower | weight | acceleration | modelyear | maker |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| good | 4 | 97 | 75 | 2265 | 18.2 | 77 | asia |
| bad | 6 | 199 | 90 | 2648 | 15 | 70 | america |
| bad | 4 | 121 | 110 | 2600 | 12.8 | 77 | europe |
| bad | 8 | 350 | 175 | 4100 | 13 | 73 | america |
| bad | 6 | 198 | 95 | 3102 | 16.5 | 74 | america |
| bad | 4 | 108 | 94 | 2379 | 16.5 | 73 | asia |
| bad | 4 | 113 | 95 | 2228 | 14 | 71 | asia |
| bad | 8 | 302 | 139 | 3570 | 12.8 | 78 | america |
| : | : | : | : | : | : | : | : |
| : | : | : | : | : | : | : | : |
| : | : | : | : | : | : | : | : |
| good | 4 | 120 | 79 | 2625 | 18.6 | 82 | america |
| bad | 8 | 455 | 225 | 4425 | 10 | 70 | america |
| good | 4 | 107 | 86 | 2464 | 15.5 | 76 | europe |
| bad | 5 | 131 | 103 | 2830 | 15.9 | 78 | europe |
| | | | | | | | |

# "One branch for each numeric value" idea:

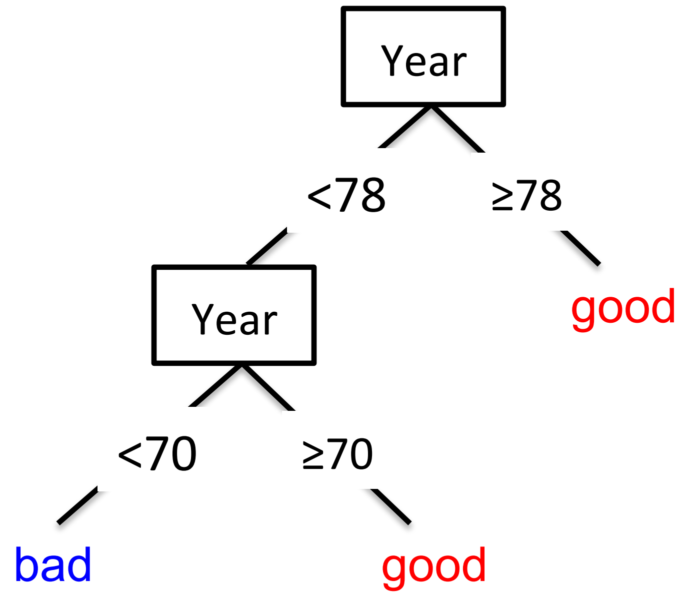



Hopeless: with such high branching factor will shatter the dataset and overfit

# Threshold splits

- Binary tree: split on attribute X at value t
  - One branch: X < t
  - Other branch: X ≥ t

- Requires small change
  - Allow repeated splits on same variable
  - How does this compare to “branch on each value” approach?



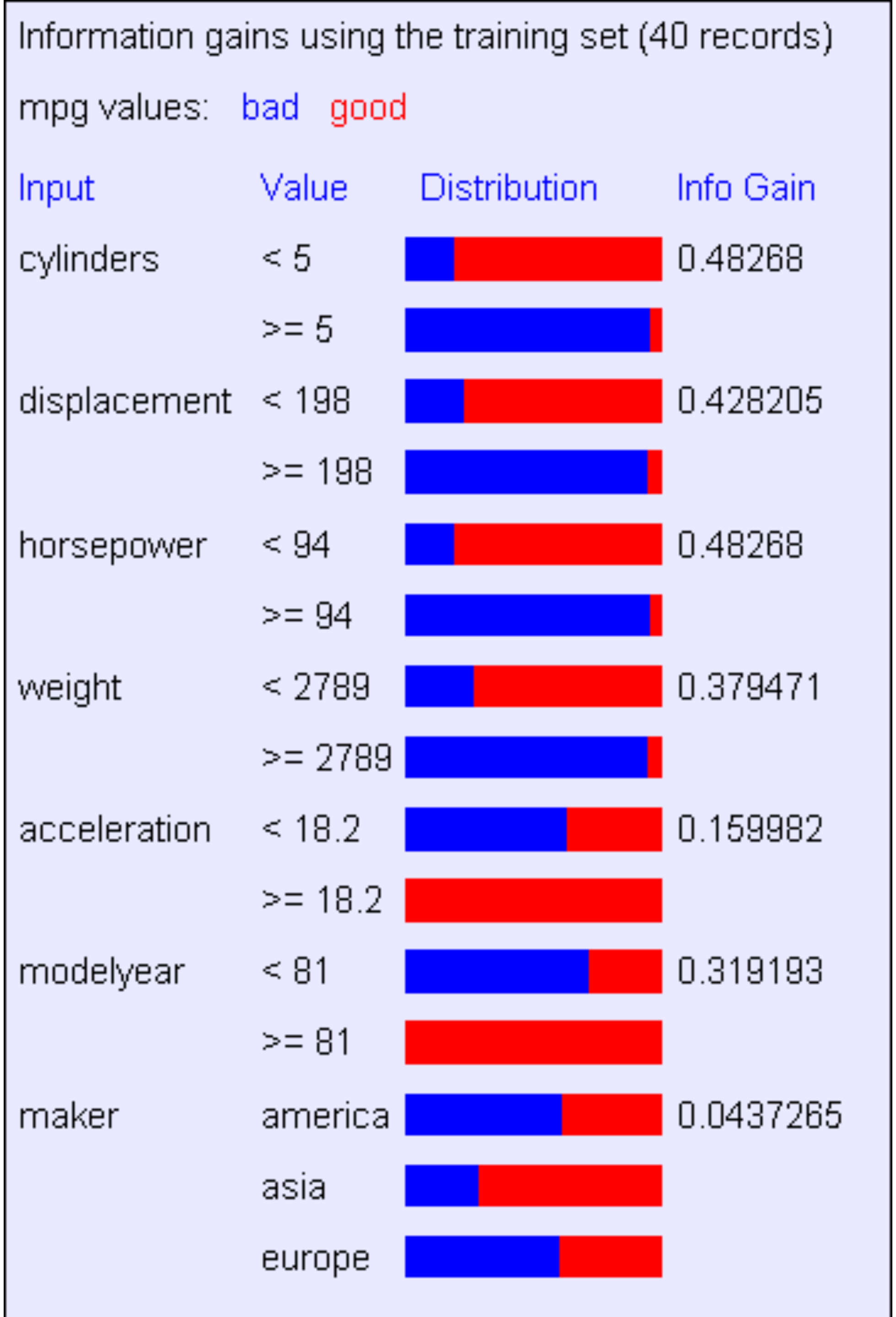
# The set of possible thresholds

- Binary tree, split on attribute X
  - One branch: $X < t$
  - Other branch: $X \geq t$
- Search through possible values of *t*
  - Seems hard!!!
- But only finite number of *t*'s are important
  - Sort data according to X into $\{x_1,\ldots,x_m\}$
  - Consider split points of the form $x_i + (x_{i+1} - x_i)/2$

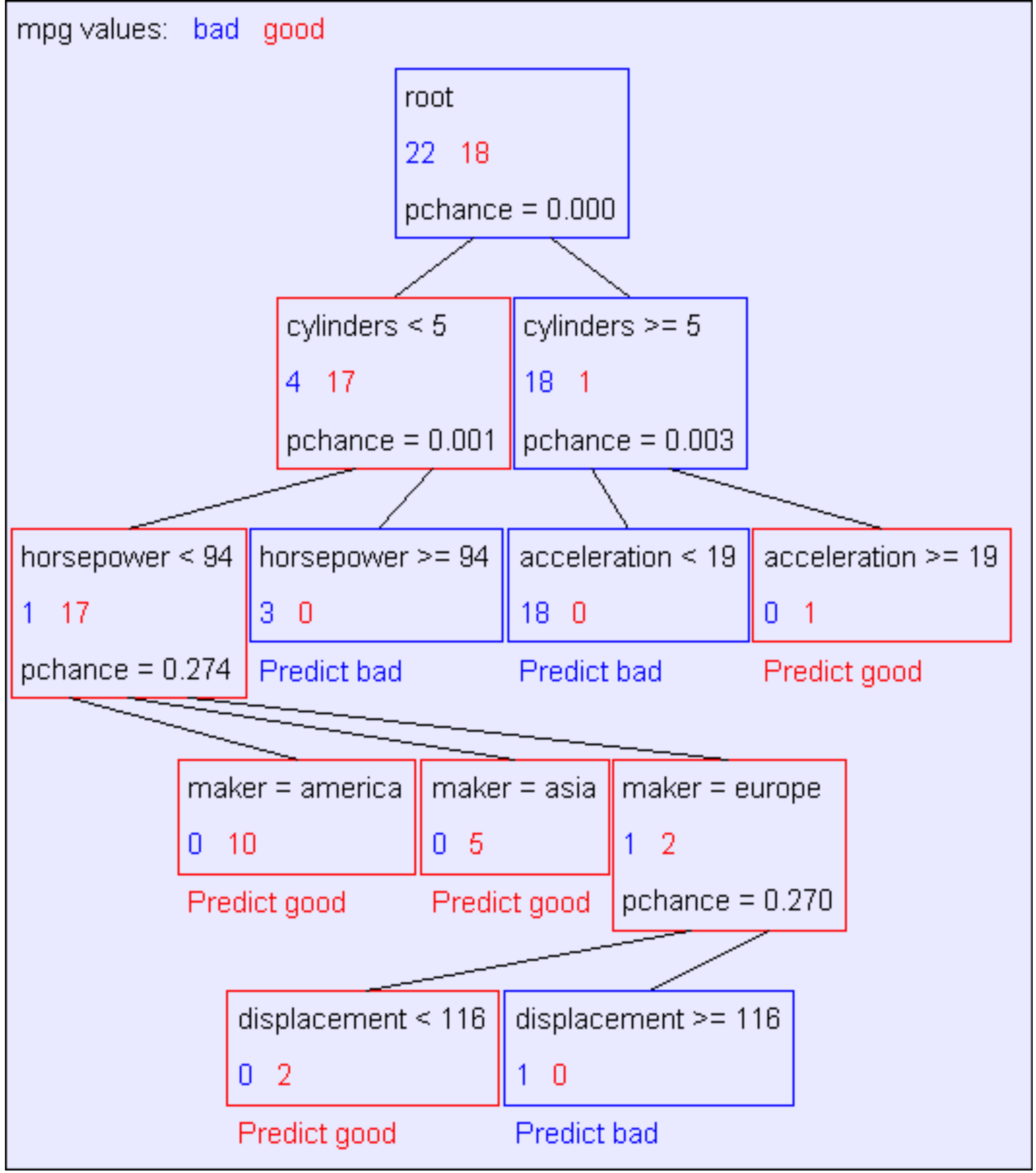
# Picking the best threshold

- Suppose $X$ is real valued with threshold $t$
- Want $IG(Y|X:t)$: the information gain for Y when testing if $X$ is greater than or less than $t$
- Define:
  - $H(Y|X:t) =$
    $$H(Y|X < t)\ P(X < t) + H(Y|X >= t)\ P(X >= t)$$
  - $IG(Y|X:t) = H(Y) - H(Y|X:t)$
  - $IG^*(Y|X) = max_t\ IG(Y|X:t)$
- Use: $IG^*(Y|X)$ for continuous variables

# Example with MPG



# Example tree for our continuous dataset



# What you need to know about decision trees

- Decision trees are one of the most popular ML tools
  - Easy to understand, implement, and 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!!!
  - Must use tricks to find "simple trees", e.g.,
    - Fixed depth/Early stopping
    - Pruning
    - Hypothesis testing

# Acknowledgements


- Some of the material in the decision trees presentation is courtesy of Andrew Moore, from his excellent collection of ML tutorials:
  - http://www.cs.cmu.edu/~awm/tutorials