Decision Trees

Machine Learning - 10601

Geoff Gordon, Miroslav Dudík

[partly based on slides of Carlos Guestrin and Andrew Moore]

http://www.cs.cmu.edu/~ggordon/10601/

October 21, 2009

Non-linear Classifiers

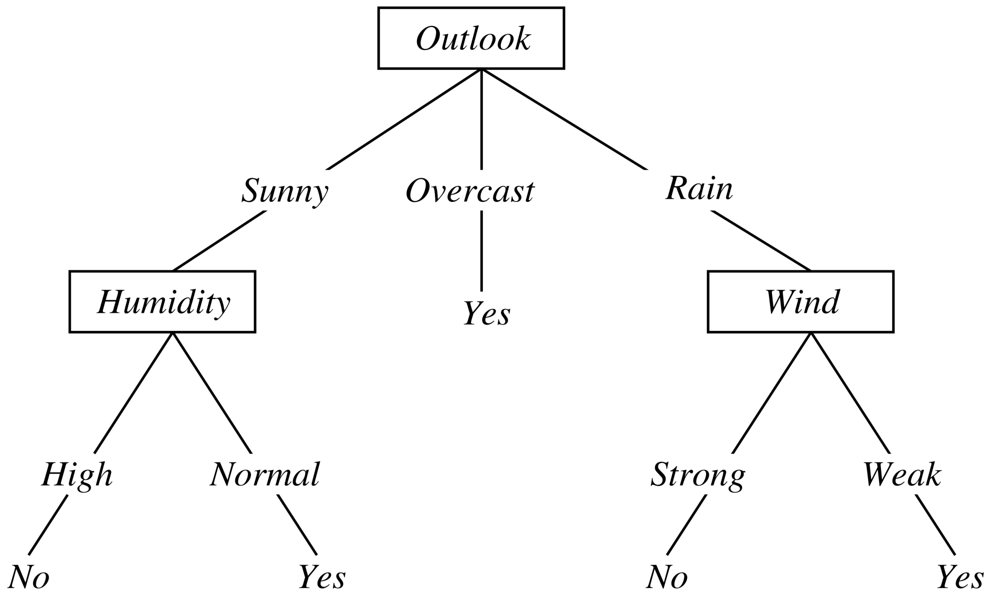
Dealing with non-linear decision boundary

1. add "non-linear" features
   to a linear model (e.g., logistic regression)
2. **use non-linear learners**
   (nearest neighbors, decision trees, artificial neural nets, ...)

k-Nearest Neighbor Classifier

- simple, often a good baseline
- can approximate arbitrary boundary: **non-parametric**
- **downside:** stores all the data

A Decision Tree for *PlayTennis*



Each internal node: **test one feature X_j**
Each branch from a node: **select one value for X_j**
Each leaf node node: **predict Y**
or **$P(Y \mid X \in \text{leaf})$**

Decision trees

How would you represent

$Y = A \vee B$ (A or B)

Decision trees

How would you represent

$Y = (A \wedge B) \vee (\neg A \wedge C)$ (**(A and B) or (not A and C)**)

Optimal Learning of Decision Trees is Hard

- learning the smallest (simplest) decision tree is NP-complete (existing algorithms exponential)
- use "greedy" heuristics:
  - start with an empty tree
  - choose the **next best** attribute (feature)
  - recurse

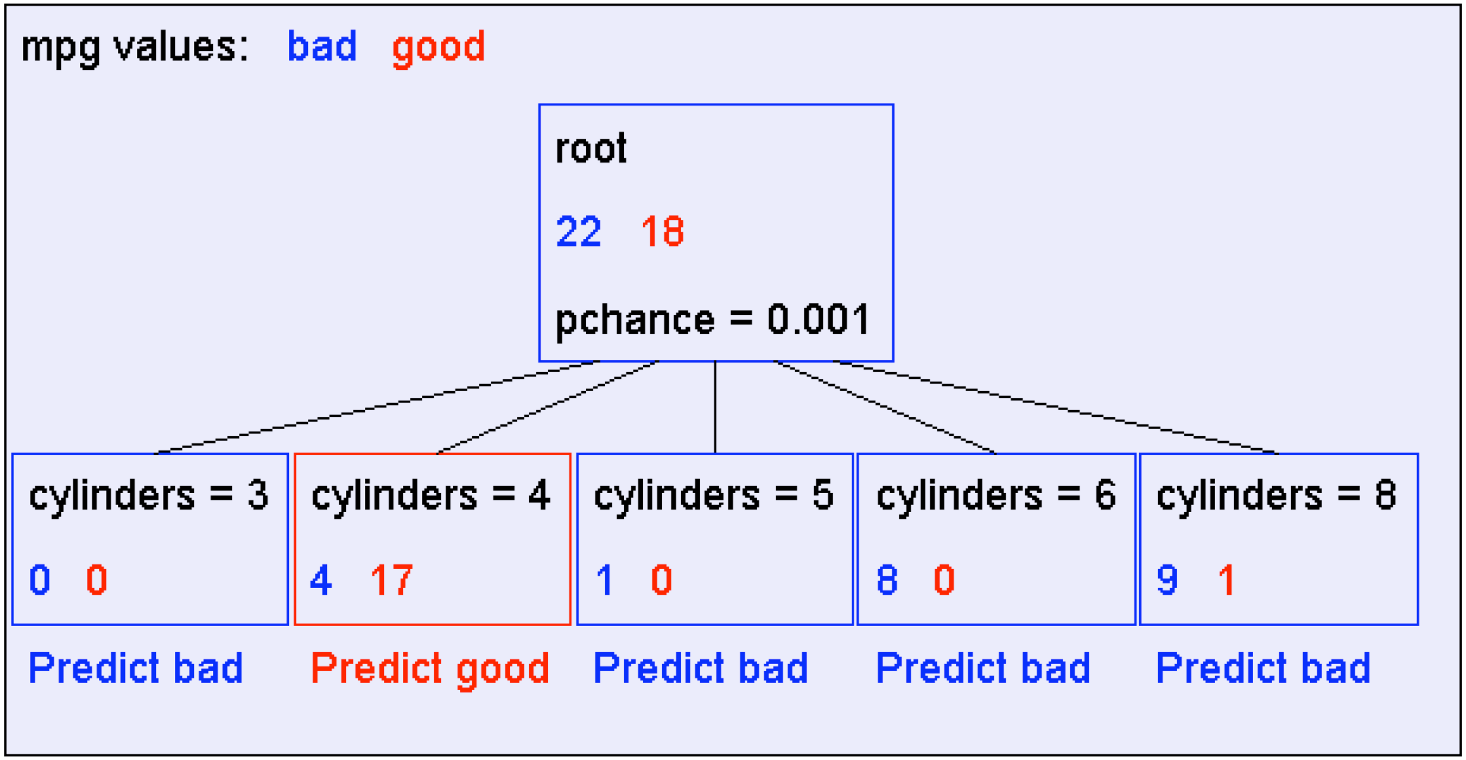
A small dataset: predict miles per gallon (mpg)

40 Records

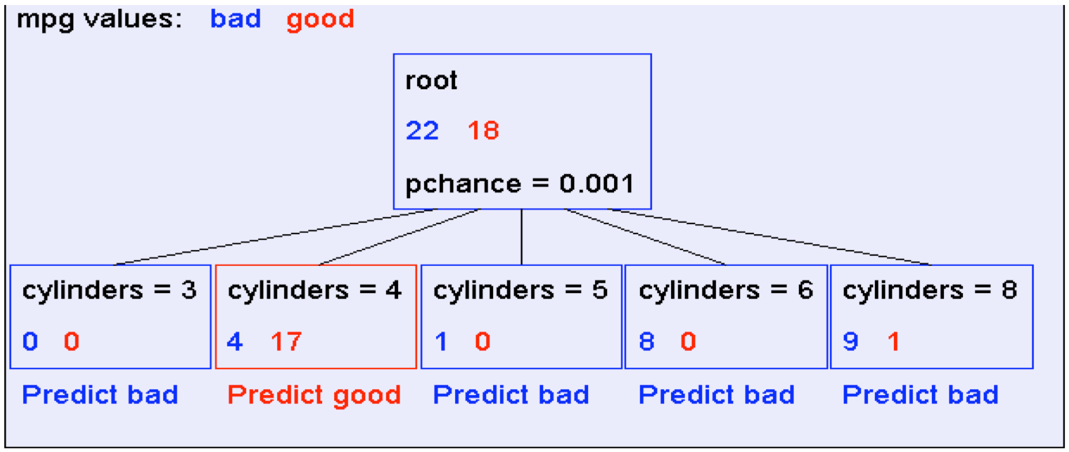
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low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

From the UCI repository (thanks to Ross Quinlan)

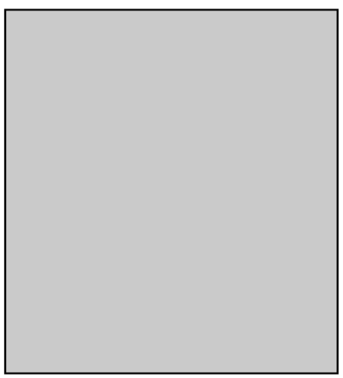
A Decision Stump



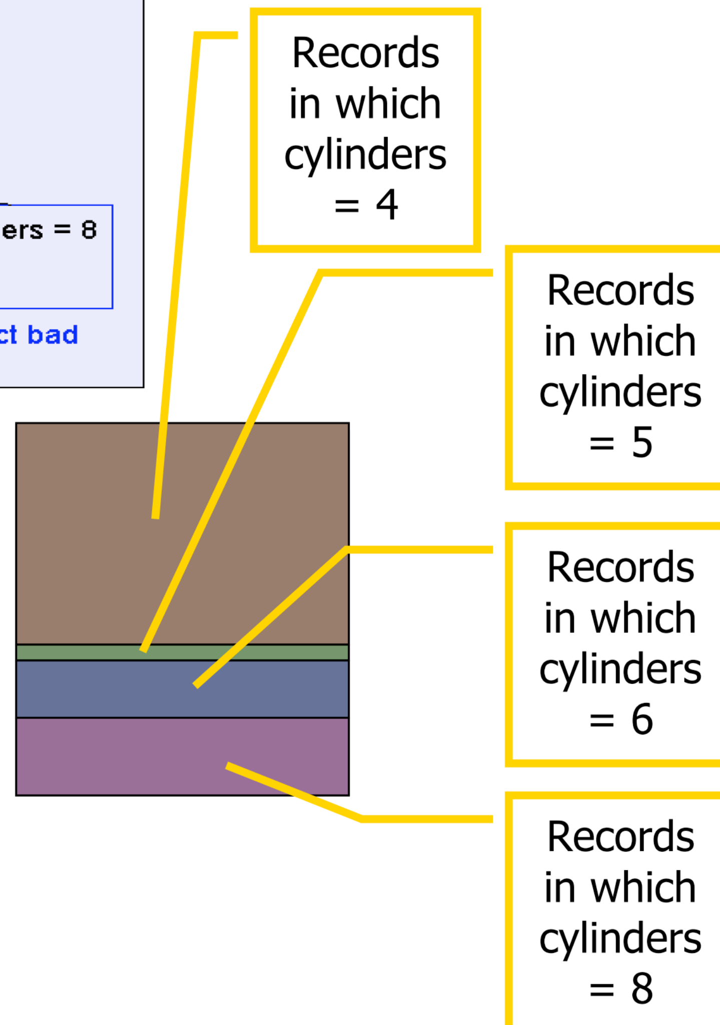
Recursion Step



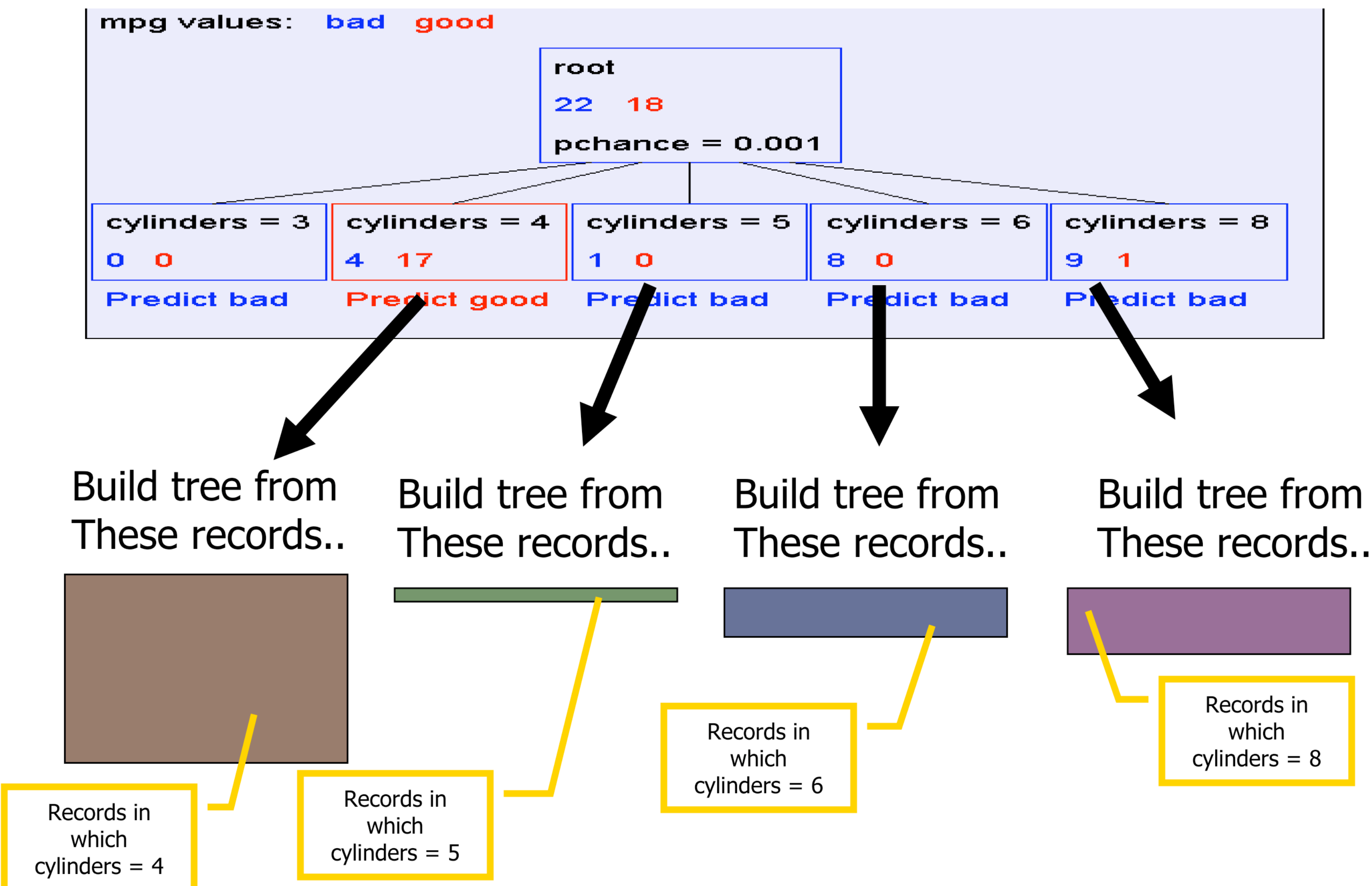
Take the Original Dataset..



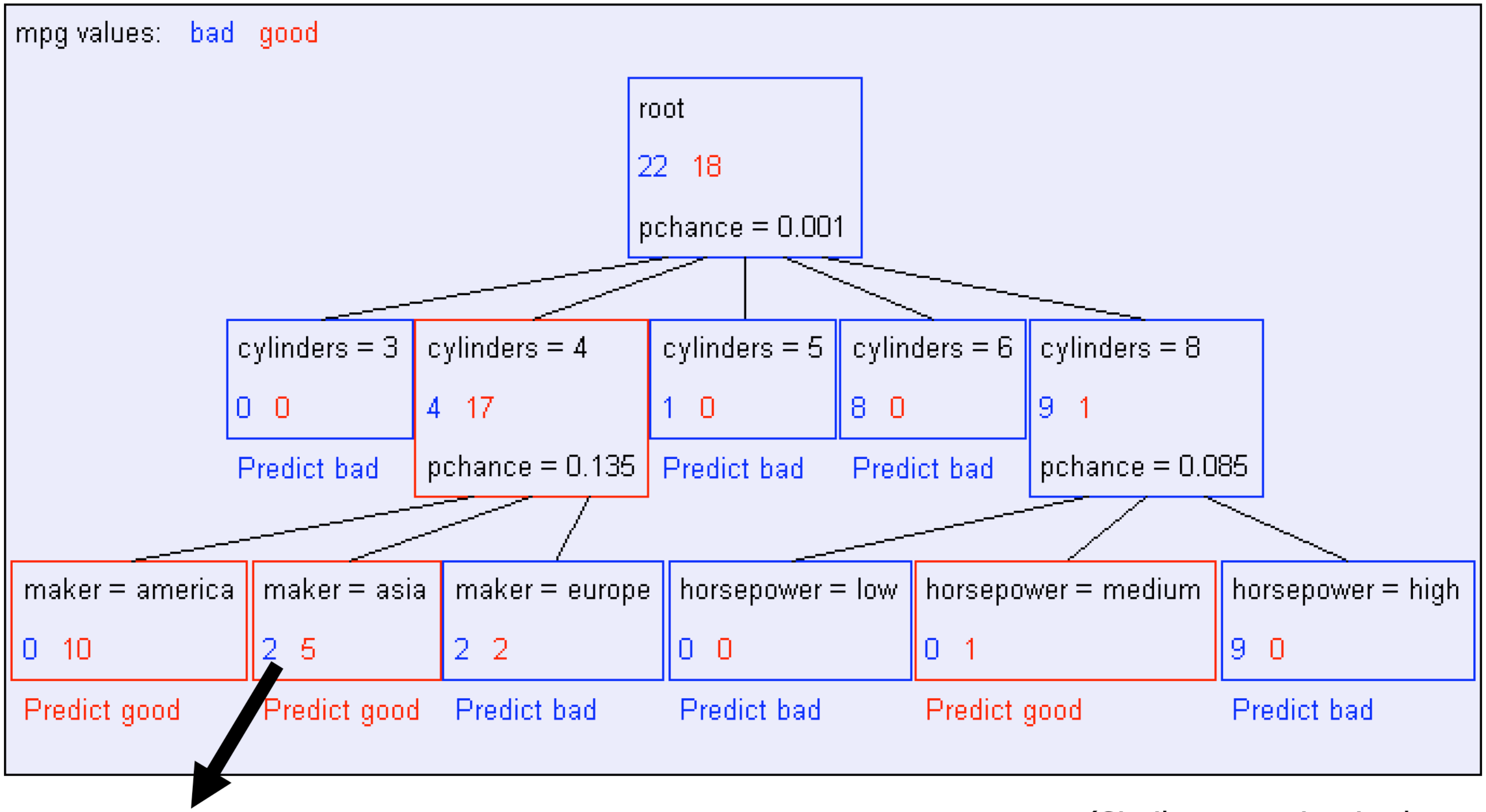
And partition it according to the value of the attribute we split on



Recursion Step



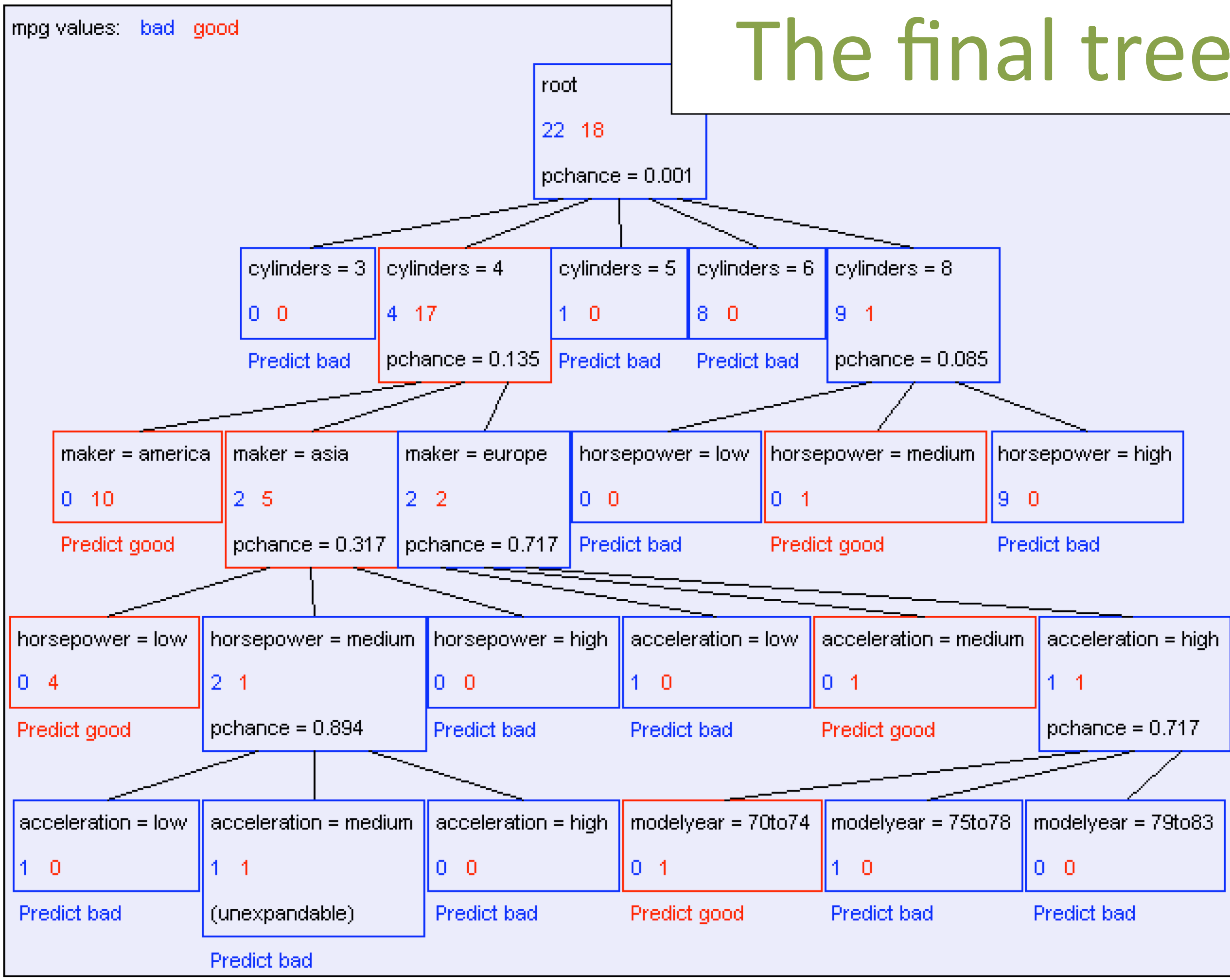
Second Level of Tree



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)

The final tree



Which attribute is the best?

A good split:

increases certainty about classification **after split**

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

Entropy = measure of uncertainty

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

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

H(Y) is the expected number of bits needed to encode a randomly drawn value of **Y**

Entropy = measure of uncertainty

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

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

H(Y) is the expected number of bits needed to encode a randomly drawn value of **Y**

Why?

Entropy = measure of uncertainty

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

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

H(Y) is the expected number of bits needed to encode a randomly drawn value of **Y**

Why?

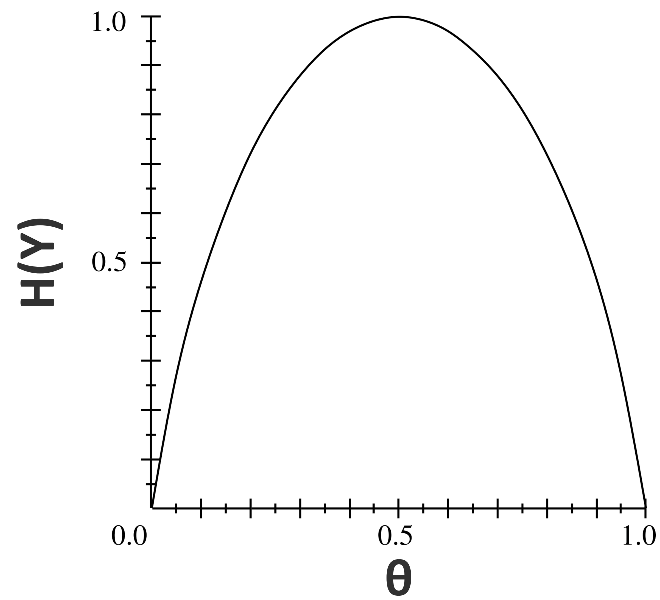
Information Theory: most efficient code assigns $-\log_2 P(Y=y_i)$ **bits** to message $Y=y_i$

Entropy = measure of uncertainty

Y binary

- **$P(Y=t) = \theta$**
- **$P(Y=f) = 1 - \theta$**

$H(Y) = \theta \log_2 \theta + (1 - \theta) \log_2 (1 - \theta)$



Information Gain
= reduction in uncertainty

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

Entropy of Y before split:

$H(Y)$

Entropy of Y after split:

(weighted by probability of each branch)

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

Information gain = difference:

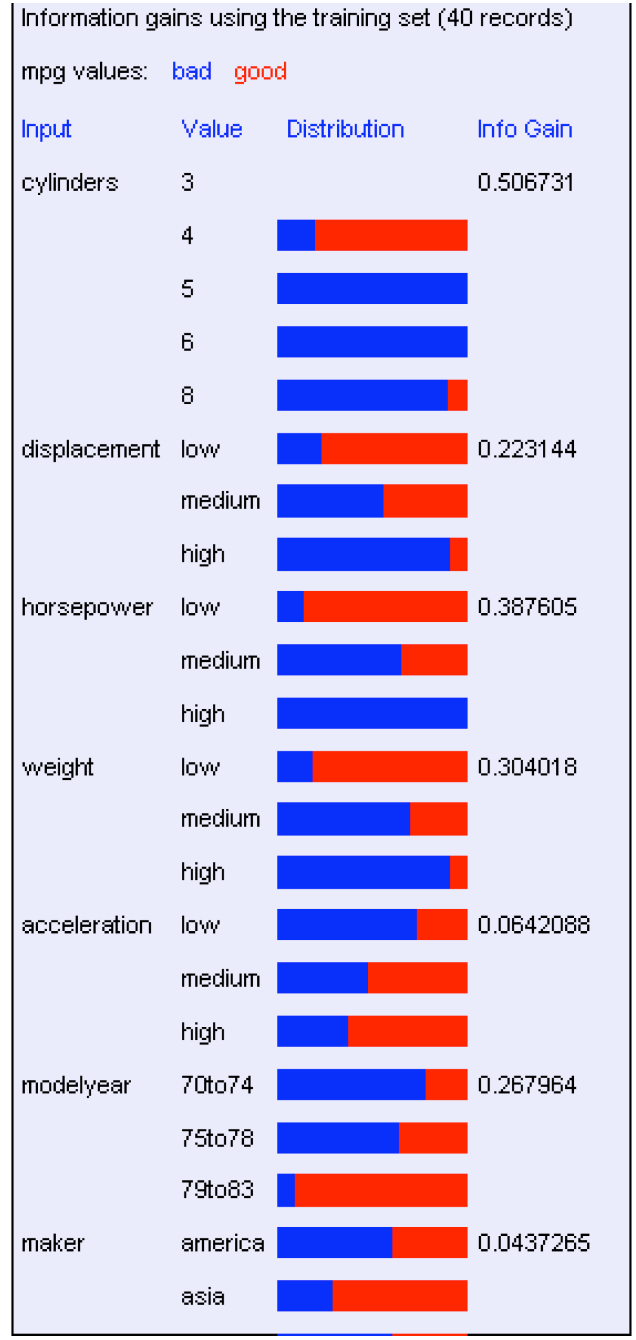
$IG(X) = H(Y) - H(Y|X)$

Learning decision trees

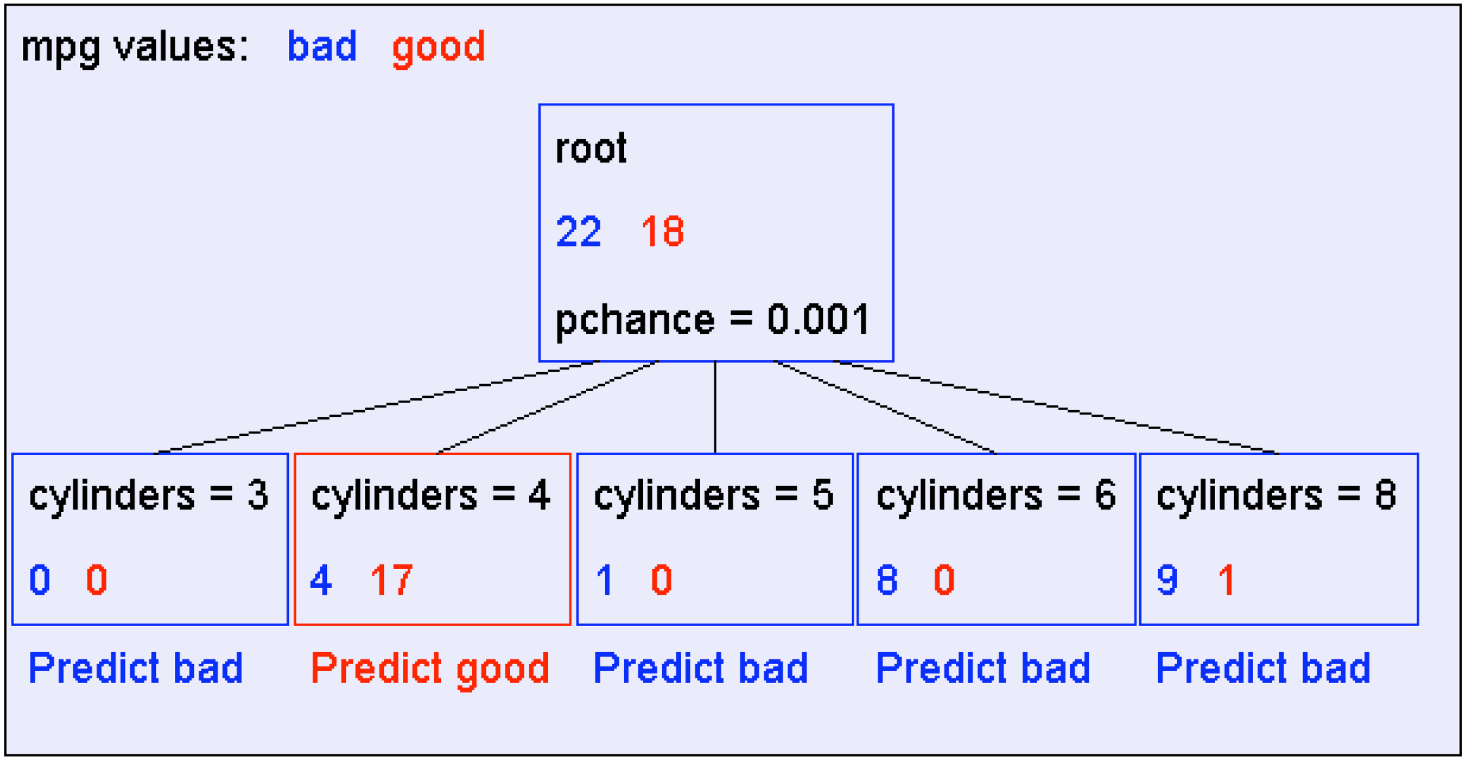
- start with an empty tree
- choose the **next best** attribute (feature)
  - for example, one that maximizes **information gain**
- split
- recurse

Suppose we want to predict MPG.

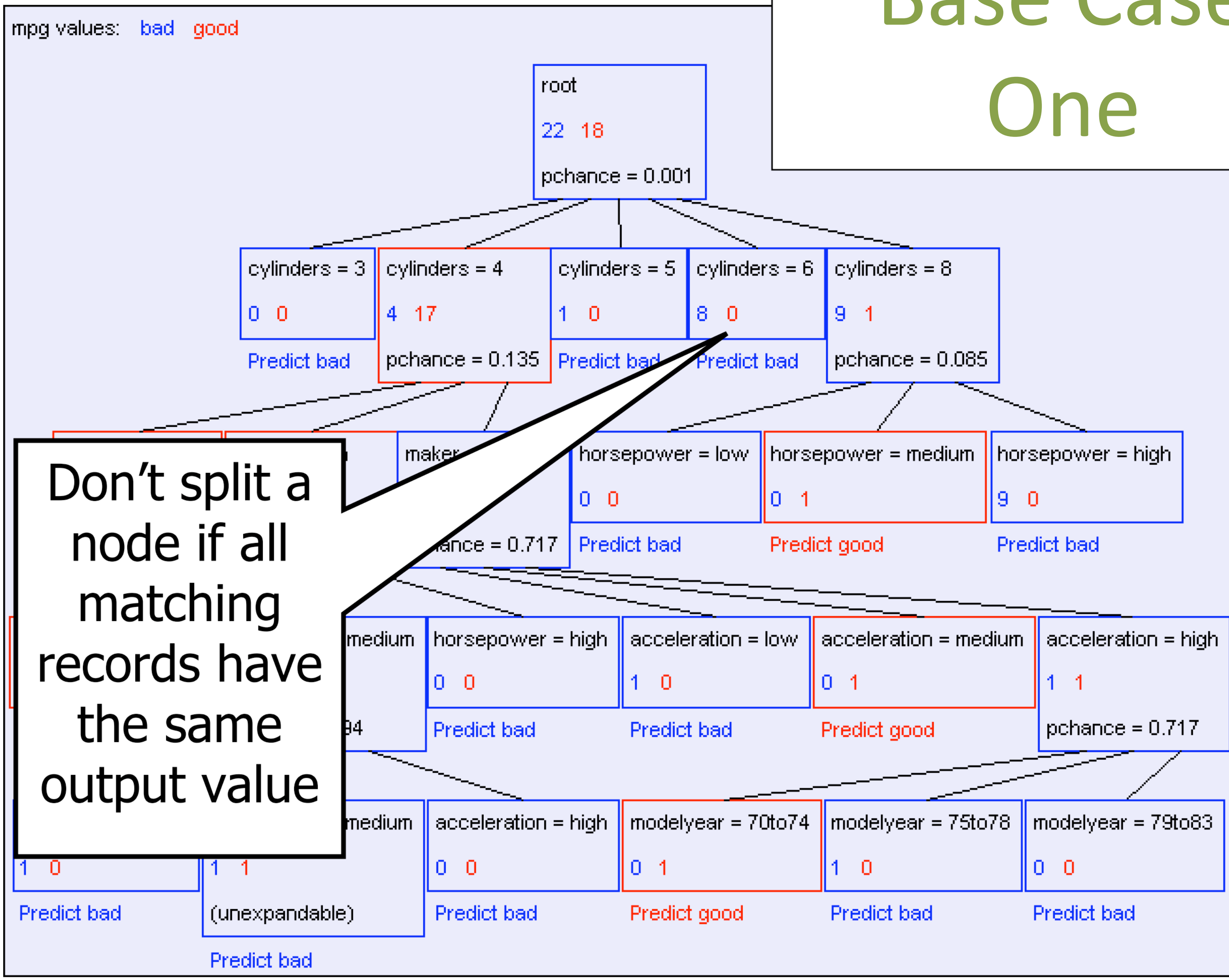
Look at all the information gains…



A Decision Stump

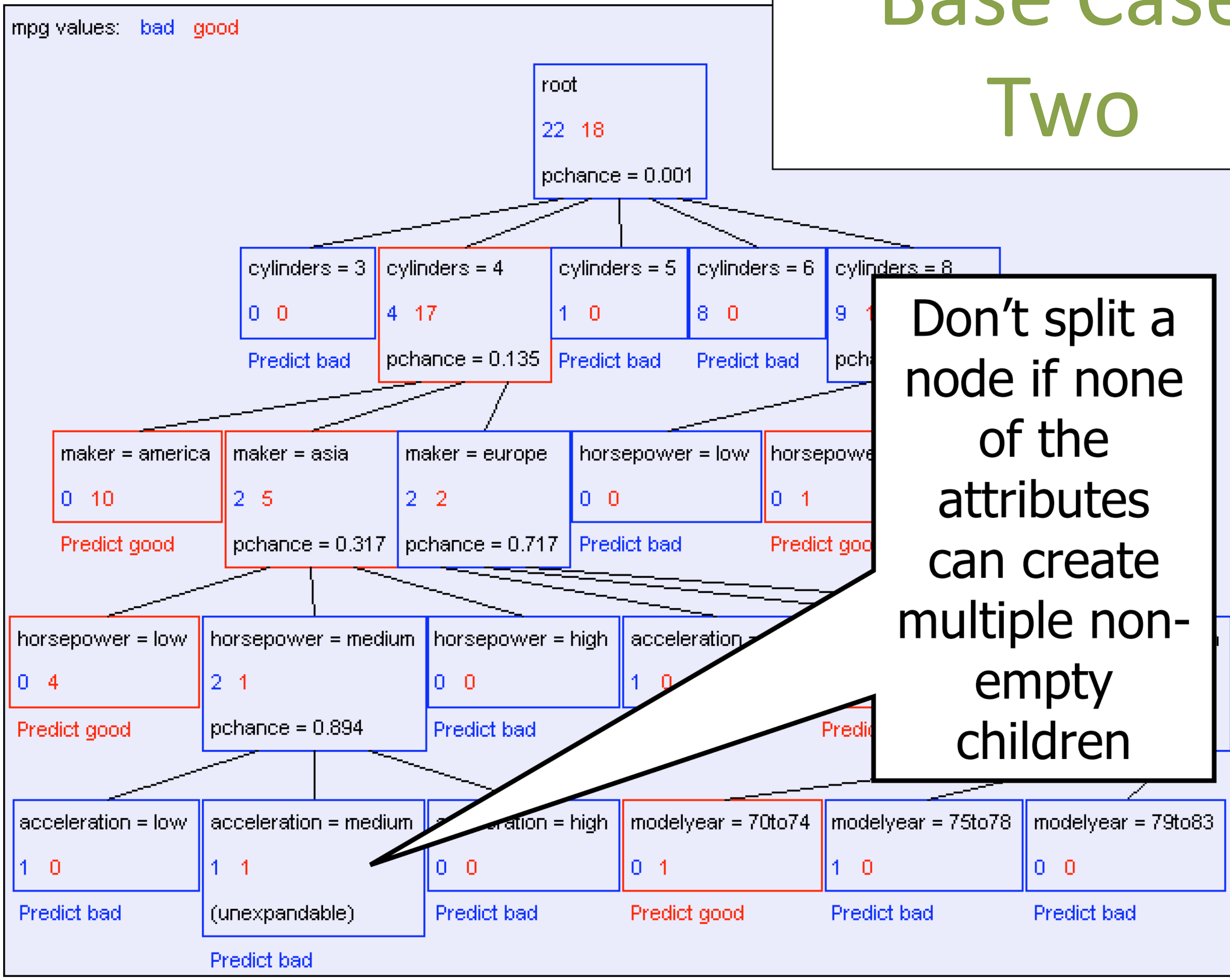


Base Case One

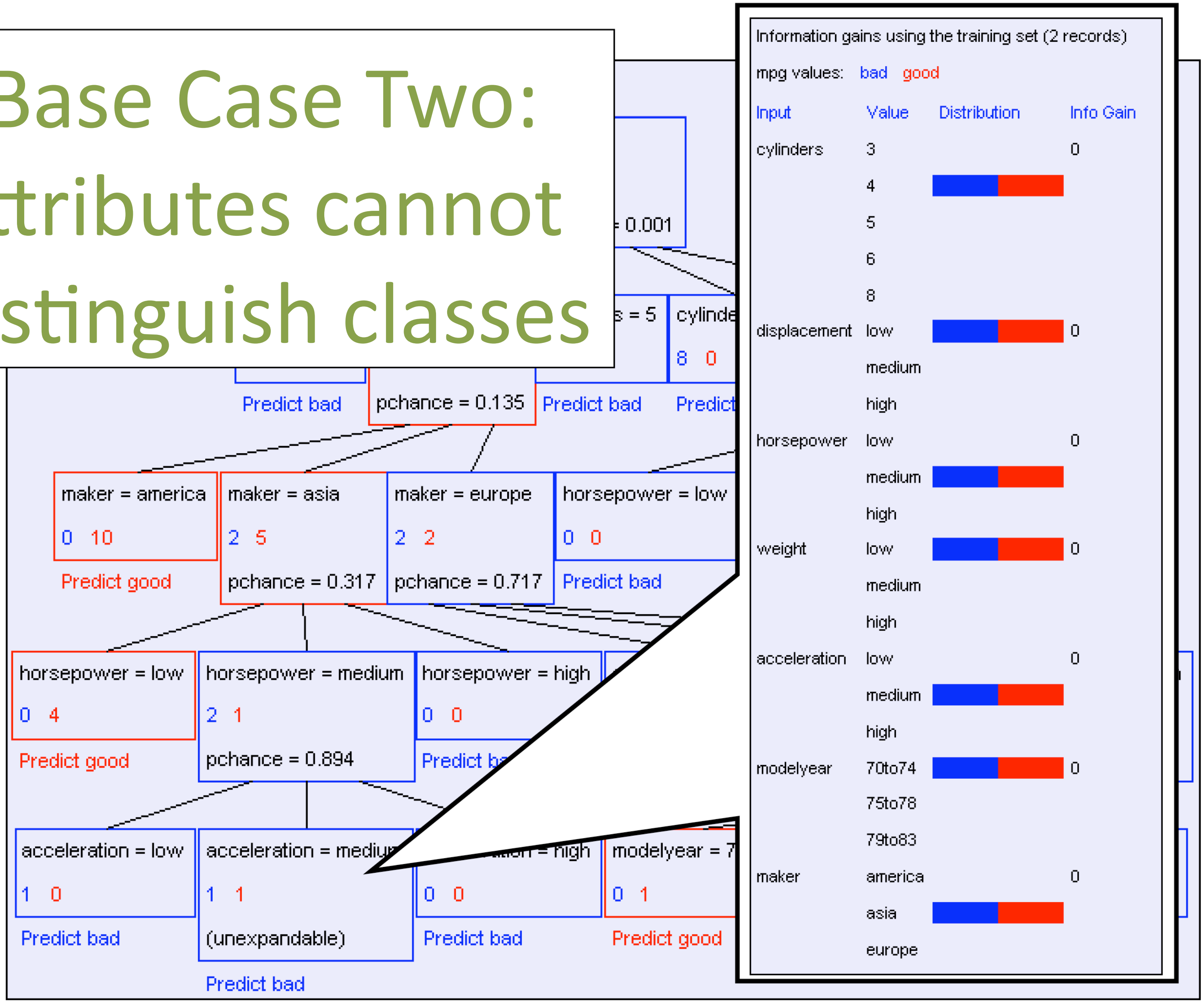


Don't split a node if all matching records have the same output value

Base Case Two



Base Case Two: attributes cannot distinguish classes



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 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

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

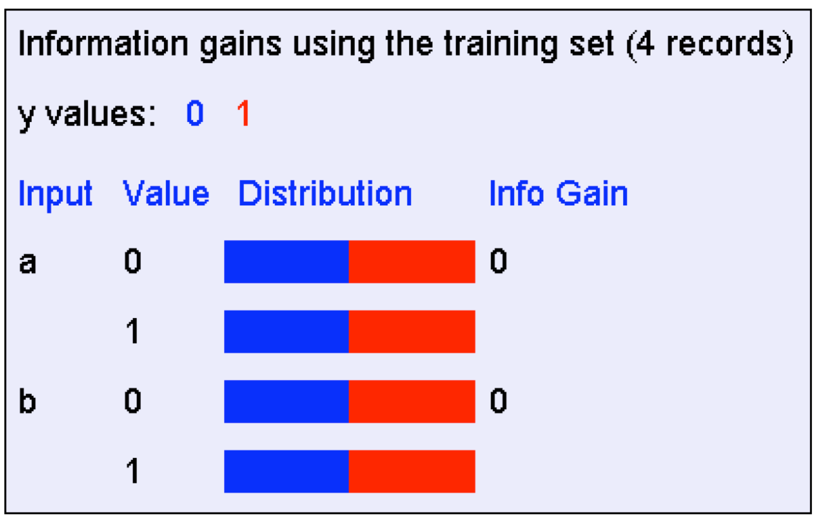
- *Is this a good idea?*

The problem with Base Case 3

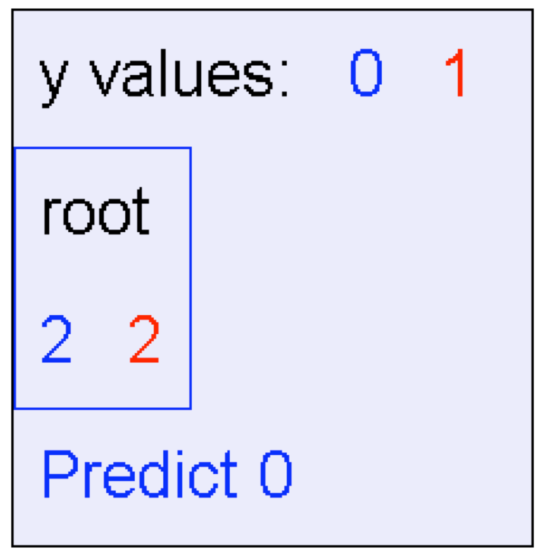
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

y = a XOR b

The information gains:



The resulting decision tree:

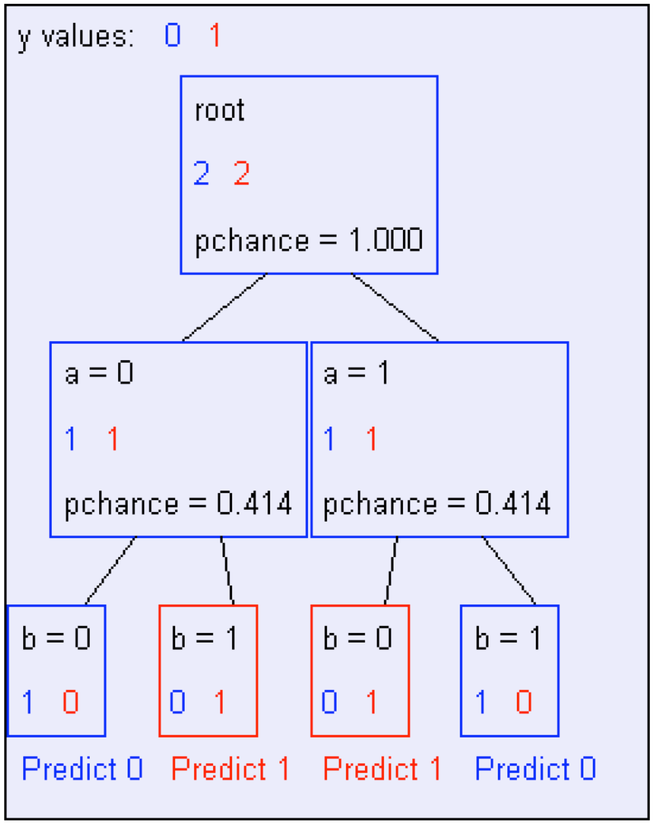


If we omit Base Case 3:

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

y = a XOR b

The resulting decision tree:



Basic Decision-Tree Building Summarized:

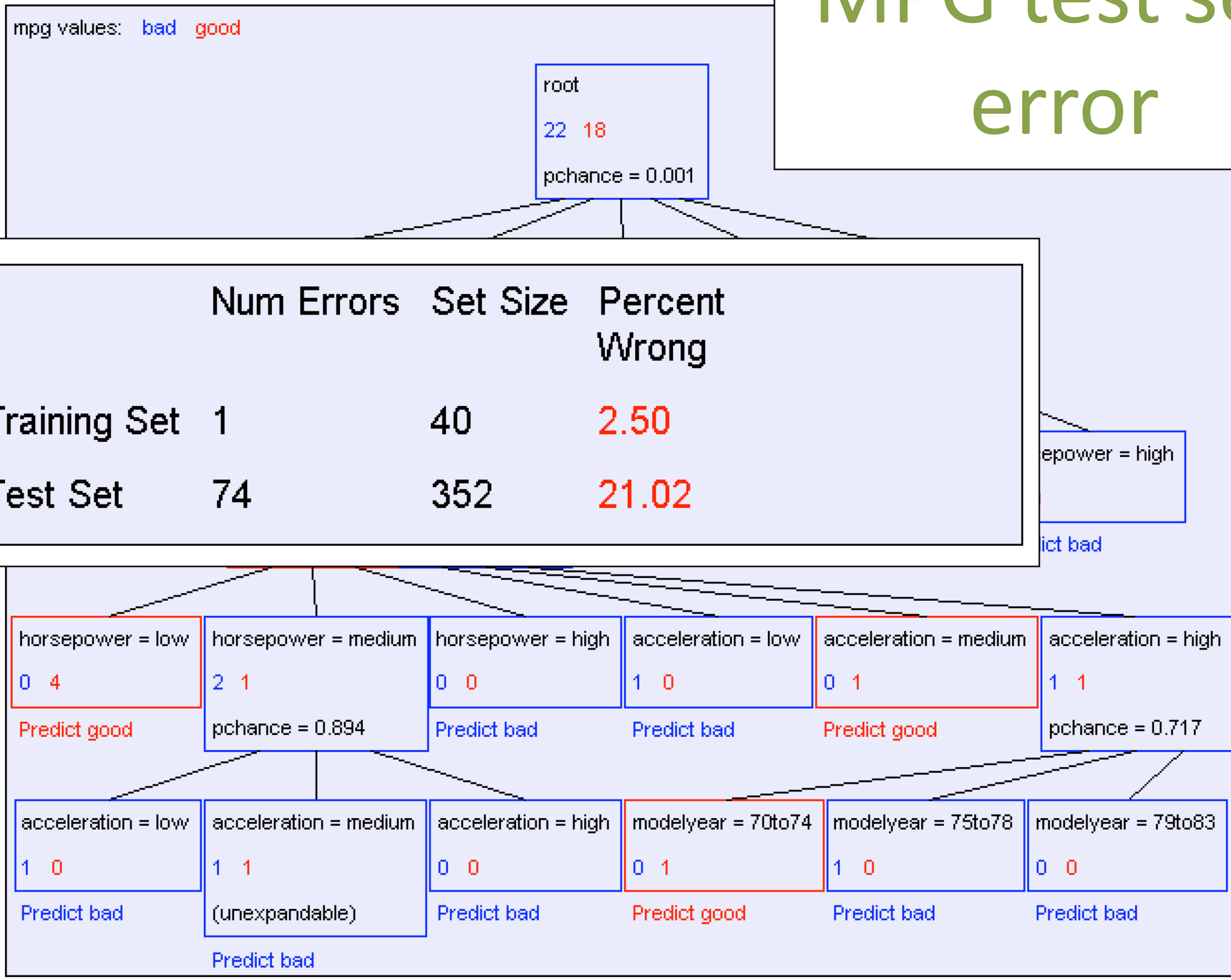
BuildTree(*DataSet,Output*)

- 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(DS_i,*Output*)

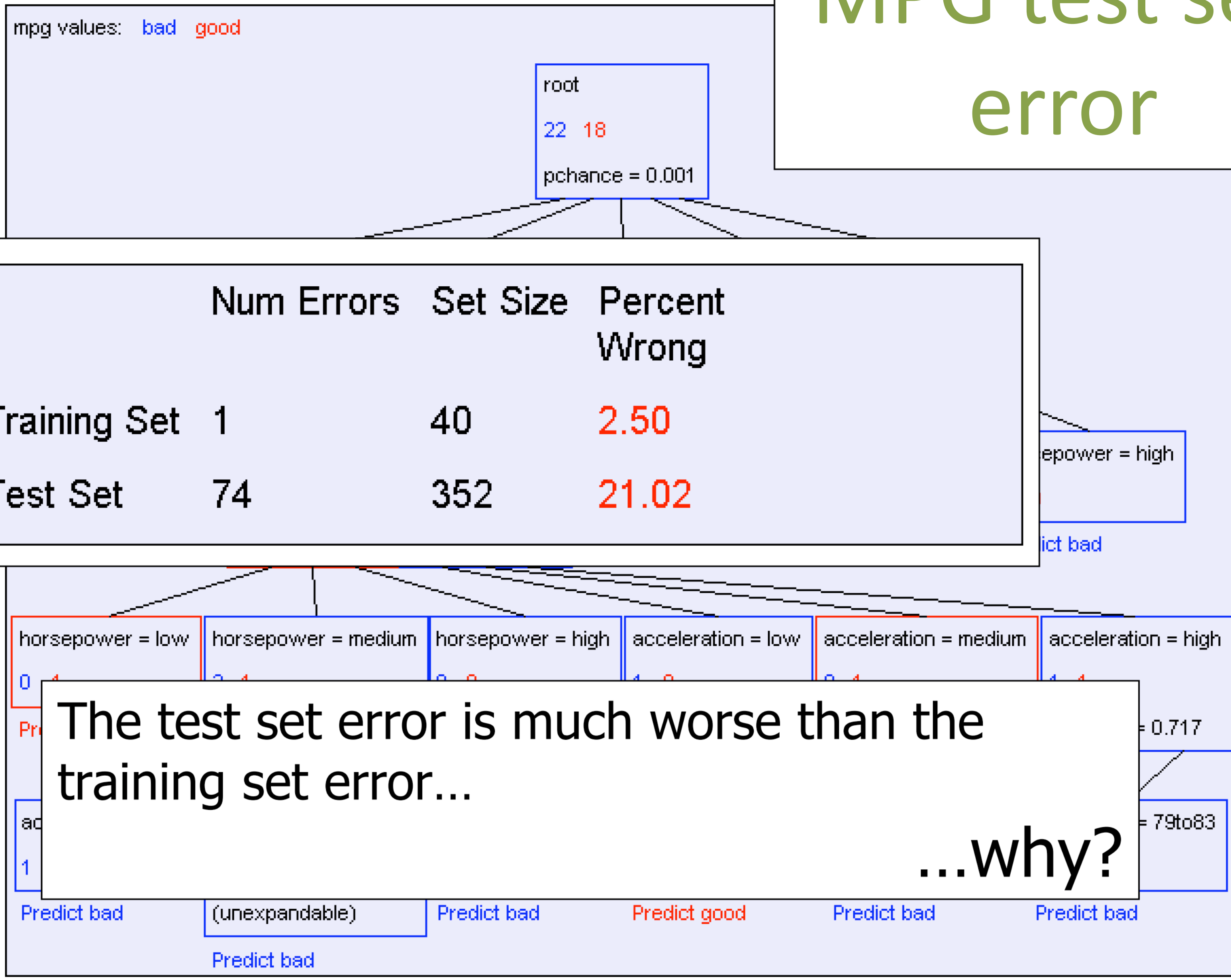
    Where DS_i built consists of all those records in DataSet for which X = *i*th distinct value of X.

MPG test set error



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

MPG test set error

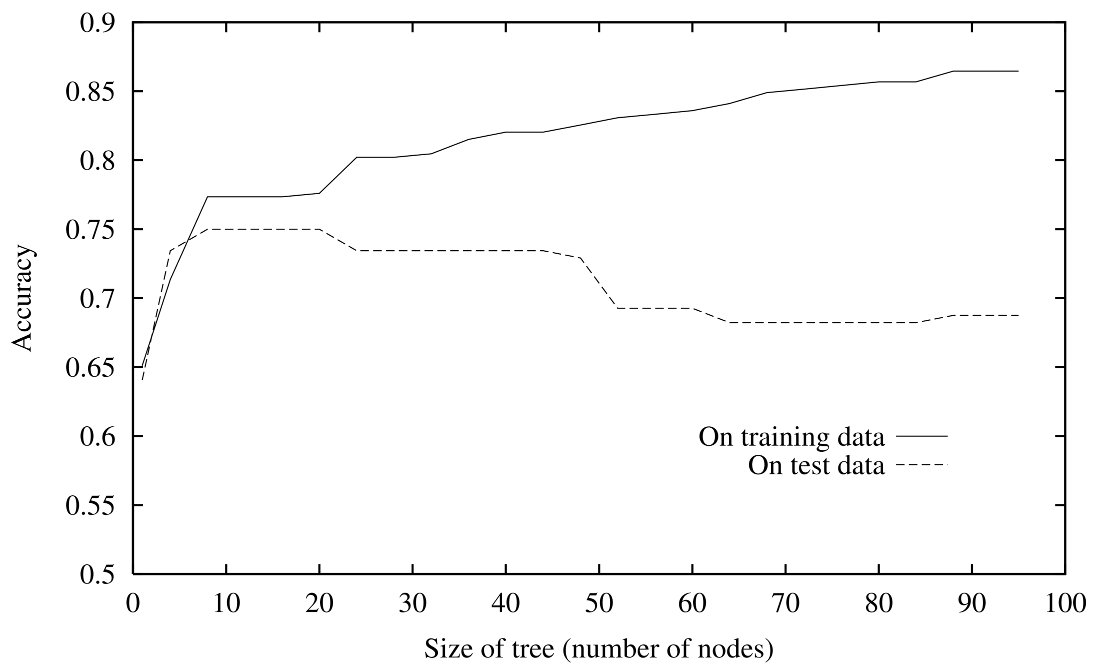


	Num Errors	Set Size	Percent Wrong
Training Set	1	40	2.50
Test Set	74	352	21.02

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

…why?

Decision trees overfit!



Standard decision trees:

- **training error always zero** (if no label noise)
- **lots of variance**

Avoiding overfitting

- fixed depth
- fixed number of leaves
- stop when splits not statistically significant

Avoiding overfitting

- fixed depth
- fixed number of leaves
- stop when splits not statistically significant

OR:

- grow the full tree,
  then **prune**
  (collapse some subtrees)

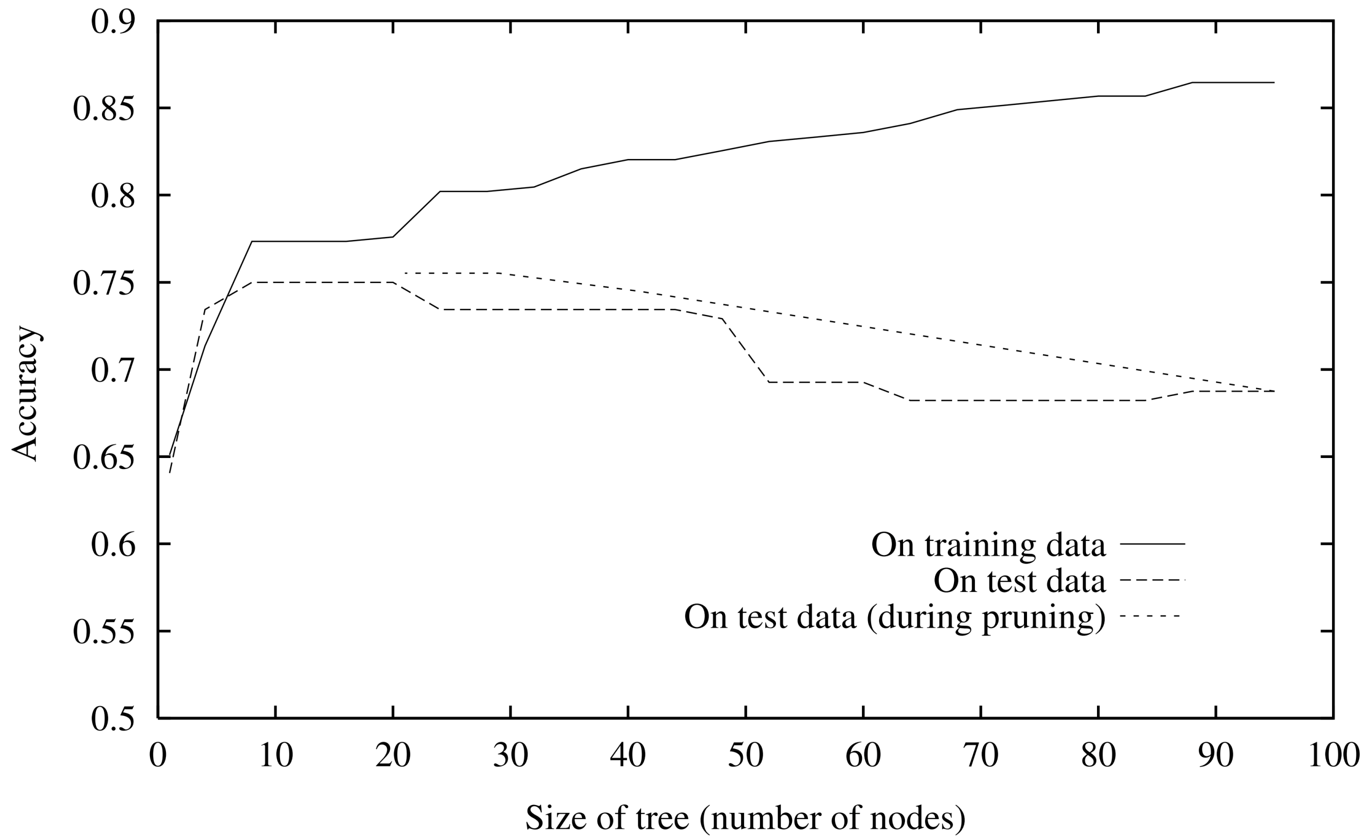
Reduced Error Pruning

Split available data into **training** and **pruning** sets

1. Learn tree that classifies
   training set perfectly
2. Do **until further pruning** is
   harmful over pruning set
   - consider pruning each node
   - collapse the node that best
     improves pruning set accuracy

This produces smallest version of
most accurate tree (over the pruning set)

Impact of Pruning



A Generic Tree-Learning Algorithm

Need to specify:

- an objective to select **splits**
- a criterion for **pruning** (or **stopping**)
- **parameters** for pruning/stopping (usually determined by **cross-validation**)

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

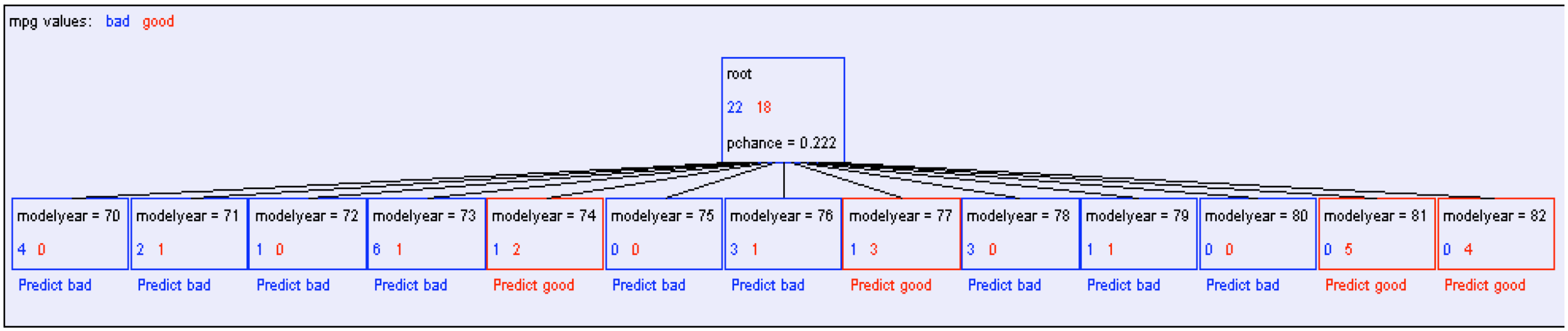
mpg	cylinders	displacemen	horsepower	weight	acceleration	modelyear	maker
good	4	97	75	2265	18.2	77	asia
bad	6	199	90	2648	15	70	america
bad	4	121	110	2600	12.8	77	europe
bad	8	350	175	4100	13	73	america
bad	6	198	95	3102	16.5	74	america
bad	4	108	94	2379	16.5	73	asia
bad	4	113	95	2228	14	71	asia
bad	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

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

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

Idea One: Branch on each possible real value

“One branch for each numeric value” idea:



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

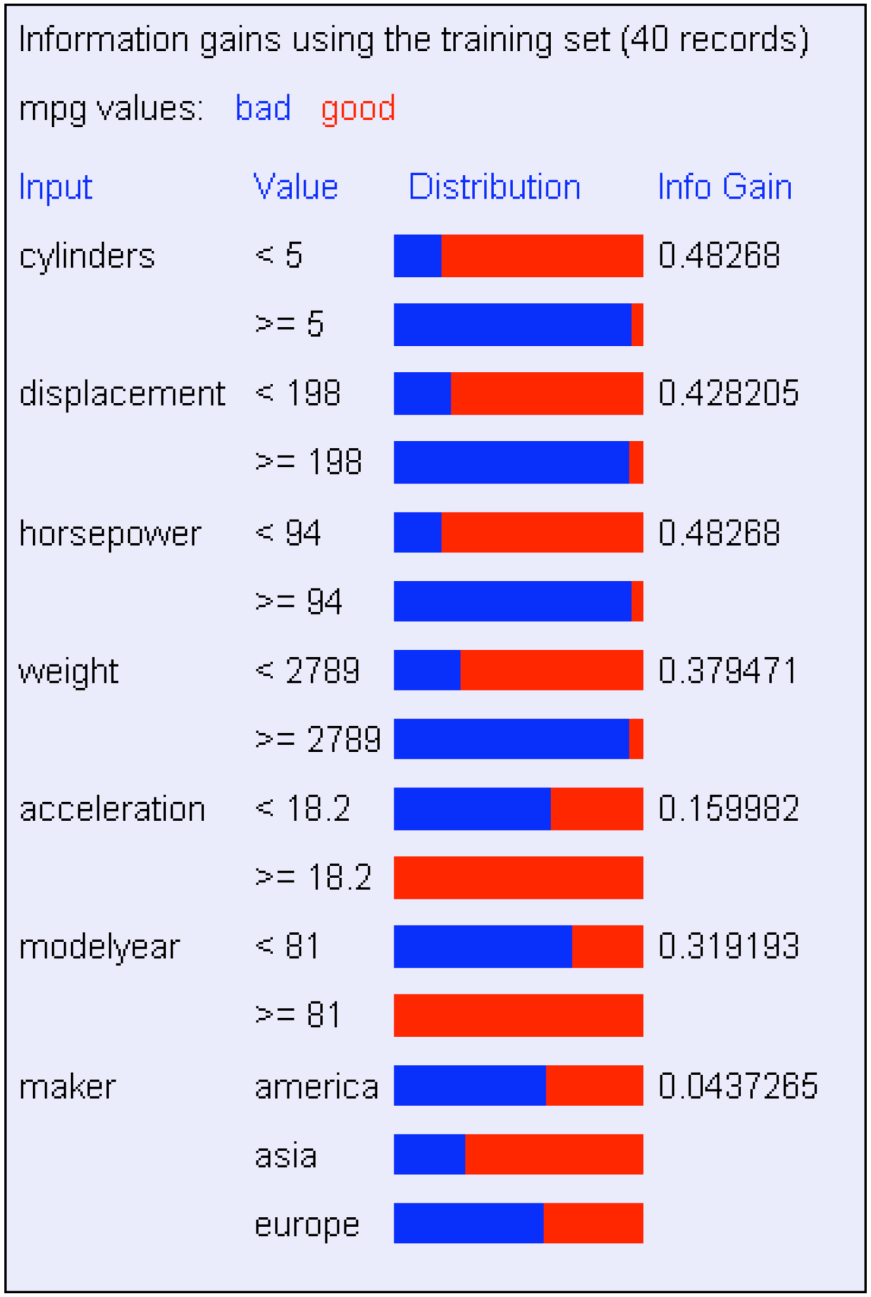
A better idea: **thresholded splits**

- Binary tree, split on attribute X:
  - one branch: **$X < t$**
  - other branch: **$X \geq t$**

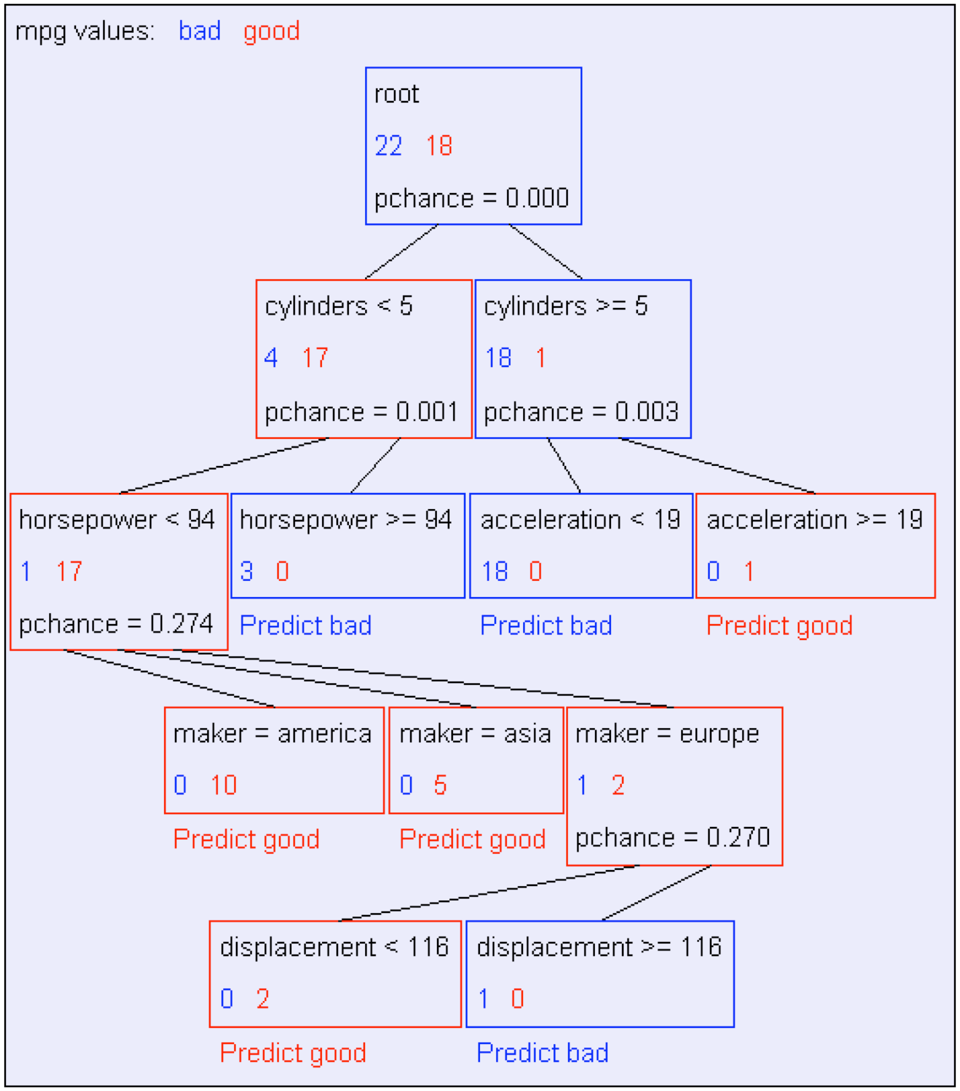
- Search through **all possible values of t**
  - seems hard, but only finite set relevant
  - sort values of X: $\{x_1, \ldots, x_m\}$
  - consider splits at $t = (x_i + x_{i+1})/2$

- Information gain for **each split**
  as if a **binary variable**: "true" for $X < t$
  "false" for $X \geq t$

Example with MPG



Example tree using reals



What you should know about decision trees

- among **most popular** data mining tools:
  - easy to understand
  - easy to implement
  - easy to use
  - computationally fast (but only a greedy heuristic!)
- not only **classification**, also **regression**, **density estimation**
- meaning of information gain
- decision trees overfit!
  - many pruning/stopping strategies

Acknowledgements

Some material in this presentation is courtesy of **Andrew Moore**, from his collection of ML tutorials: http://www.autonlab.org/tutorials/

LEARNING THEORY

Computational Learning Theory

What general laws constrain **"learning"**?

- how many **examples** needed to learn a **target concept** to a given **precision**?
- what is the impact of:
  - complexity of the **target concept**?
  - complexity of our **hypothesis space**?
  - **manner** in which examples presented?
    - random samples—what we mostly consider in this course
    - learner can make queries
    - examples come from an "adversary" (worst-case analysis, no statistical assumptions)