

# CpE/EE 695: Applied Machine Learning

## Fall 2009

### Lecture 3

# Decision Tree Learning

Prof. Haibo He

Department of Electrical and Computer Engineering

Stevens Institute of Technology

Hoboken, NJ 07086



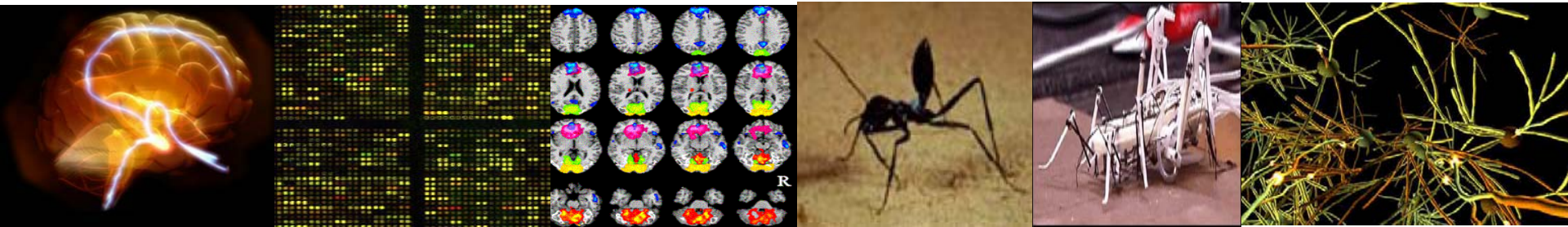
# Review for previous lecture

Learning from examples

General-to-Specific ordering of hypothesis

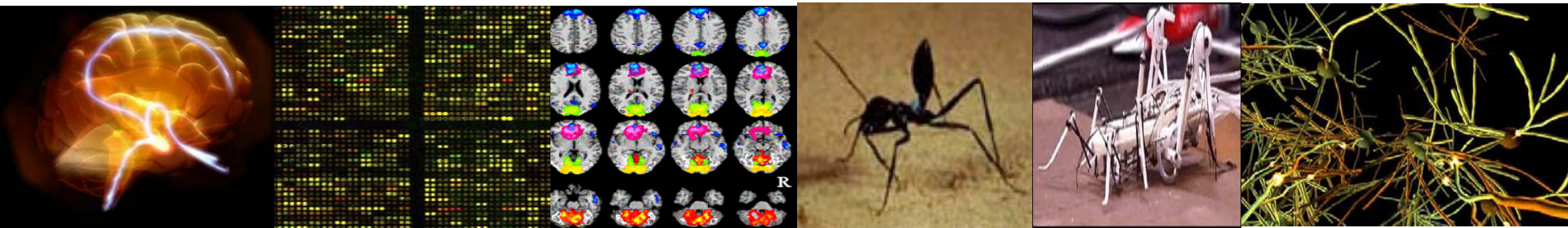
Version spaces and candidate elimination

Inductive bias



# Outline

- Contingency Tables
- Information gain (review of entropy)
- Learning an unpruned decision tree recursively
- Training and testing Error
- Overfitting issue
- Pruning a tree
- Building Decision Trees with real Valued Inputs



# Note

Most of the lecture slides are adopted from the following resources:

[1] Andrew Moore, Statistical Data Mining Tutorials,, [Online],  
available: Andrew's tutorials:

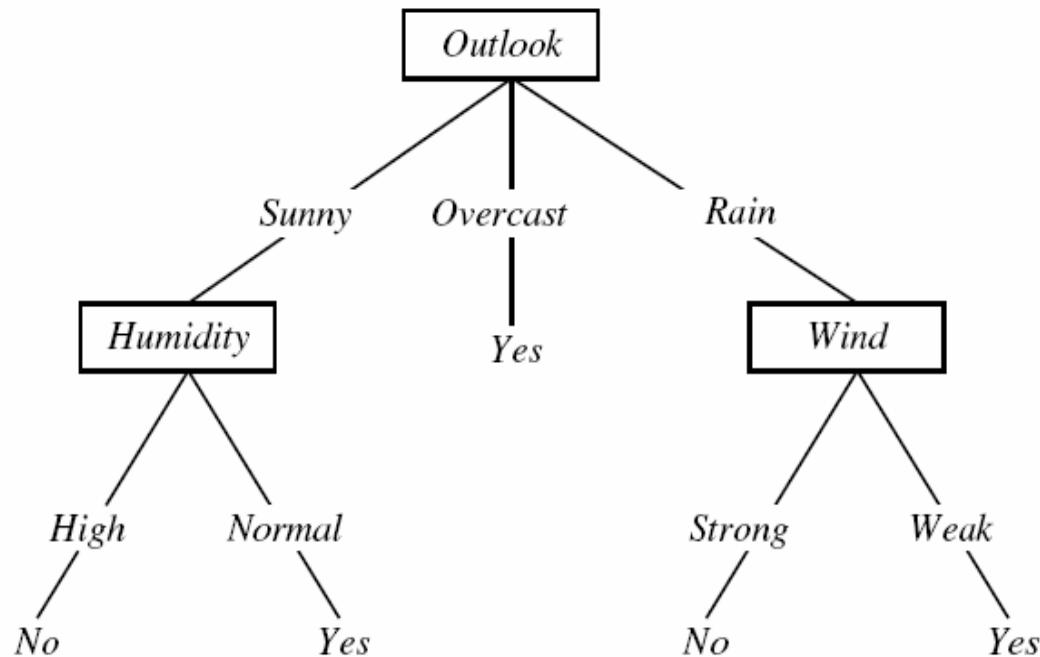
<http://www.autonlab.org/tutorials/dtree18.pdf>

<http://www.autonlab.org/tutorials/infogain11.pdf>

[2] T. M. Mitchell, Machine Learning, McGraw Hill, 1997. ISBN:  
978-0-07-042807-2

# Let's see a simple example first

## Decision tree for *PlayTennis*



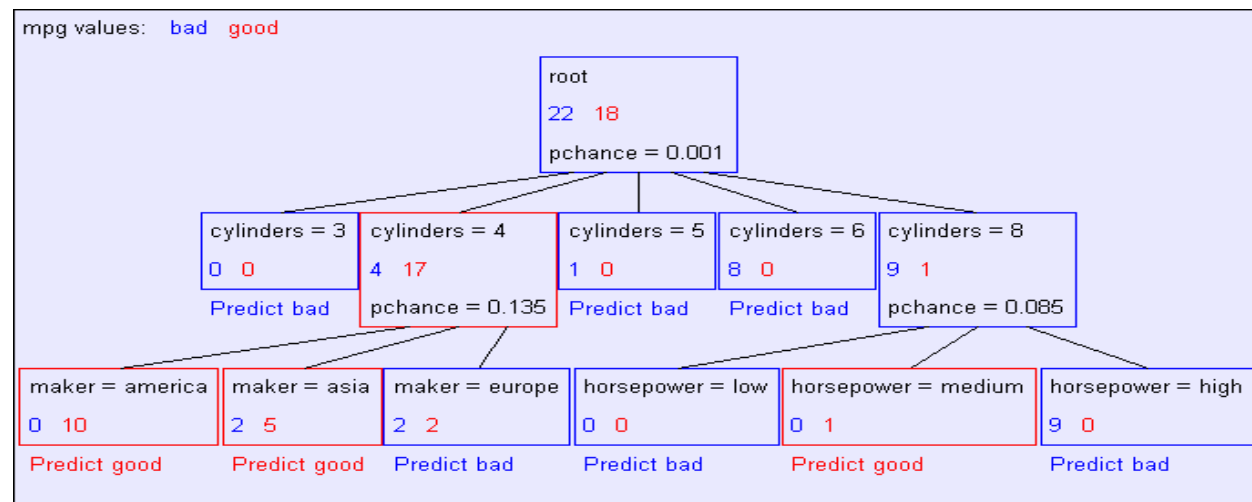
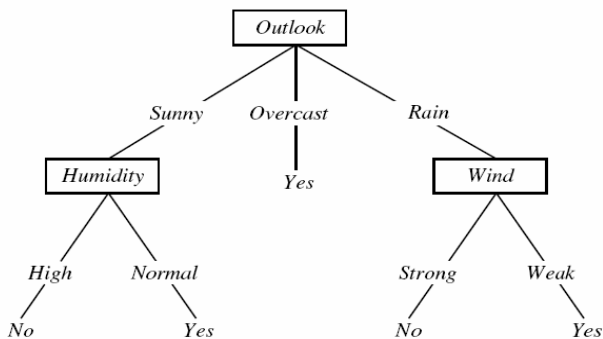
Popular decision tree algorithms:  
ID3, ASSISTANT, C4.5,...

# Decision tree representation

Decision tree representation:

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

## Examples:



# When to consider decision tree

- Instances describable by attribute–value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

Examples:

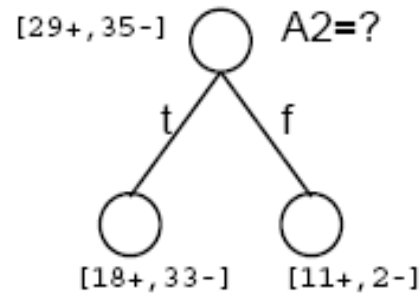
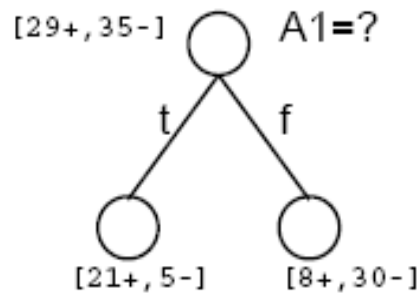
- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

# Top-down induction of decision tree

Main loop:

1.  $A \leftarrow$  the “best” decision attribute for next *node*
2. Assign  $A$  as decision attribute for *node*
3. For each value of  $A$ , create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which attribute is best?



# Let's see another typical machine learning dataset

age	employe	education	edun	marital	...	job	relation	race	gender	hour	country	wealth
39	State_gov	Bachelors	13	Never_mar	...	Adm_cleric	Not_in_fan	White	Male	40	United_Stat	poor
51	Self_emp	Bachelors	13	Married	...	Exec_man	Husband	White	Male	13	United_Stat	poor
39	Private	HS_grad	9	Divorced	...	Handlers_c	Not_in_fan	White	Male	40	United_Stat	poor
54	Private	11th	7	Married	...	Handlers_c	Husband	Black	Male	40	United_Stat	poor
28	Private	Bachelors	13	Married	...	Prof_speci	Wife	Black	Female	40	Cuba	poor
38	Private	Masters	14	Married	...	Exec_man	Wife	White	Female	40	United_Stat	poor
50	Private	9th	5	Married_sp	...	Other_serv	Not_in_fan	Black	Female	16	Jamaica	poor
52	Self_emp	HS_grad	9	Married	...	Exec_man	Husband	White	Male	45	United_Stat	rich
31	Private	Masters	14	Never_mar	...	Prof_speci	Not_in_fan	White	Female	50	United_Stat	rich
42	Private	Bachelors	13	Married	...	Exec_man	Husband	White	Male	40	United_Stat	rich
37	Private	Some_coll	10	Married	...	Exec_man	Husband	Black	Male	80	United_Stat	rich
30	State_gov	Bachelors	13	Married	...	Prof_speci	Husband	Asian	Male	40	India	rich
24	Private	Bachelors	13	Never_mar	...	Adm_cleric	Own_child	White	Female	30	United_Stat	poor
33	Private	Assoc_acc	12	Never_mar	...	Sales	Not_in_fan	Black	Male	50	United_Stat	poor
41	Private	Assoc_voc	11	Married	...	Craft_repai	Husband	Asian	Male	40	*MissingV	rich
34	Private	7th_8th	4	Married	...	Transport_	Husband	Amer_India	Male	45	Mexico	poor
26	Self_emp	HS_grad	9	Never_mar	...	Farming_fi	Own_child	White	Male	35	United_Stat	poor
33	Private	HS_grad	9	Never_mar	...	Machine_c	Unmarried	White	Male	40	United_Stat	poor
38	Private	11th	7	Married	...	Sales	Husband	White	Male	50	United_Stat	poor
44	Self_emp	Masters	14	Divorced	...	Exec_man	Unmarried	White	Female	45	United_Stat	rich
41	Private	Doctorate	16	Married	...	Prof_speci	Husband	White	Male	60	United_Stat	rich
:	:	:	:	:	:	:	:	:	:	:	:	:

48,000 records, 16 attributes [Kohavi 1995]

# Classification

- A Major Data Mining Operation
- Give one attribute (e.g wealth), try to predict the value of new people's wealths by means of some of the other available attributes.
- Applies to categorical outputs
  - Categorical attribute: an attribute which takes on two or more discrete values. Also known as a symbolic attribute.
  - Real attribute: a column of real numbers

## About this dataset

- It is a tiny subset of the 1990 US Census.
- It is publicly available online from the UCI Machine Learning Datasets repository

### Used Attributes

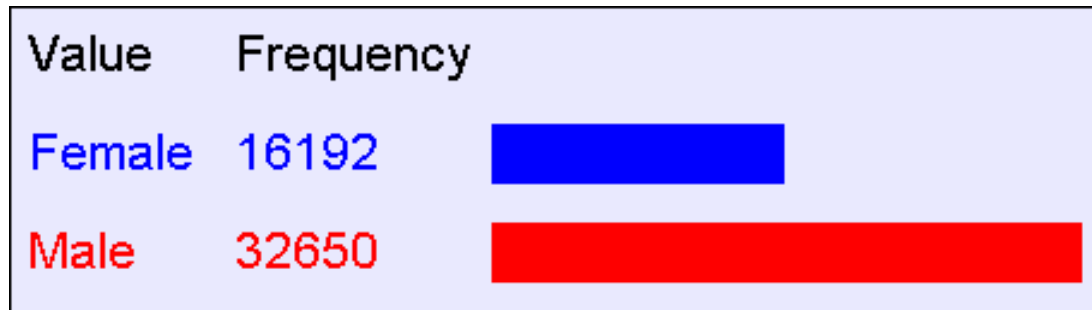
age	edunum	race	hours_worked
employment	marital	gender	country
taxweighting	job	capitalgain	wealth
education	relation	capitalloss	agegroup

This color = Real-valued    This color = Symbol-valued

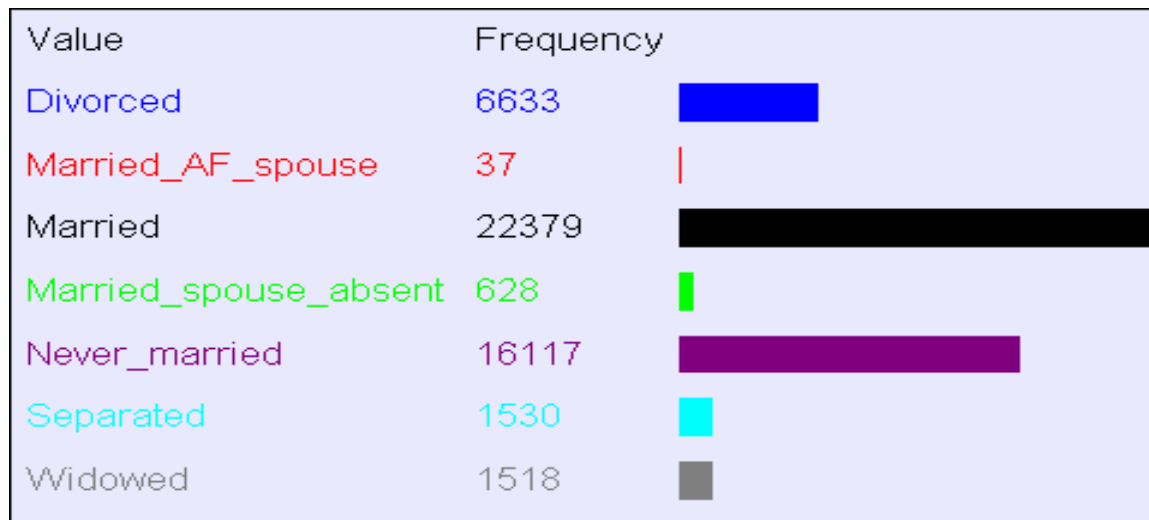
Successfully loaded a new dataset from the file \tadult.fds. It has 16 attributes and 48842 records.

# What can you do with a dataset?

- Well, you can look at histograms...



Gender



Marital  
Status

# Contingency Tables

- A better name for a histogram:  
*A One-dimensional Contingency Table*
- Recipe for making a k-dimensional contingency table:
  1. Pick  $k$  attributes from your dataset. Call them  $a_1, a_2, \dots, a_k$ .
  2. For every possible combination of values,  $a_1 = x_1, a_2 = x_2, \dots, a_k = x_k$ , record how frequently that combination occurs

*Fun fact: A database person would call this a “k-dimensional datacube”*










# A 2-d Contingency Table

wealth values:		poor	rich
agegroup	10s	2507	3
	20s	11262	743
	30s	9468	3461
	40s	6738	3986
	50s	4110	2509
	60s	2245	809
	70s	668	147
	80s	115	16
	90s	42	13

- For each pair of values for attributes (agegroup, wealth) we can see how many records match.

# A 2-d Contingency Table










wealth values: poor rich

agegroup	10s	2507	3	
	20s	11262	743	
	30s	9468	3461	
	40s	6738	3986	
	50s	4110	2509	
	60s	2245	809	
	70s	668	147	
	80s	115	16	
	90s	42	13	

- Easier to appreciate graphically

# A 2-d Contingency Table

wealth values: poor rich

agegroup	10s	2507	3	
	20s	11262	743	
	30s	9468	3461	
	40s	6738	3986	
	50s	4110	2509	
	60s	2245	809	
	70s	668	147	
	80s	115	16	
	90s	42	13	

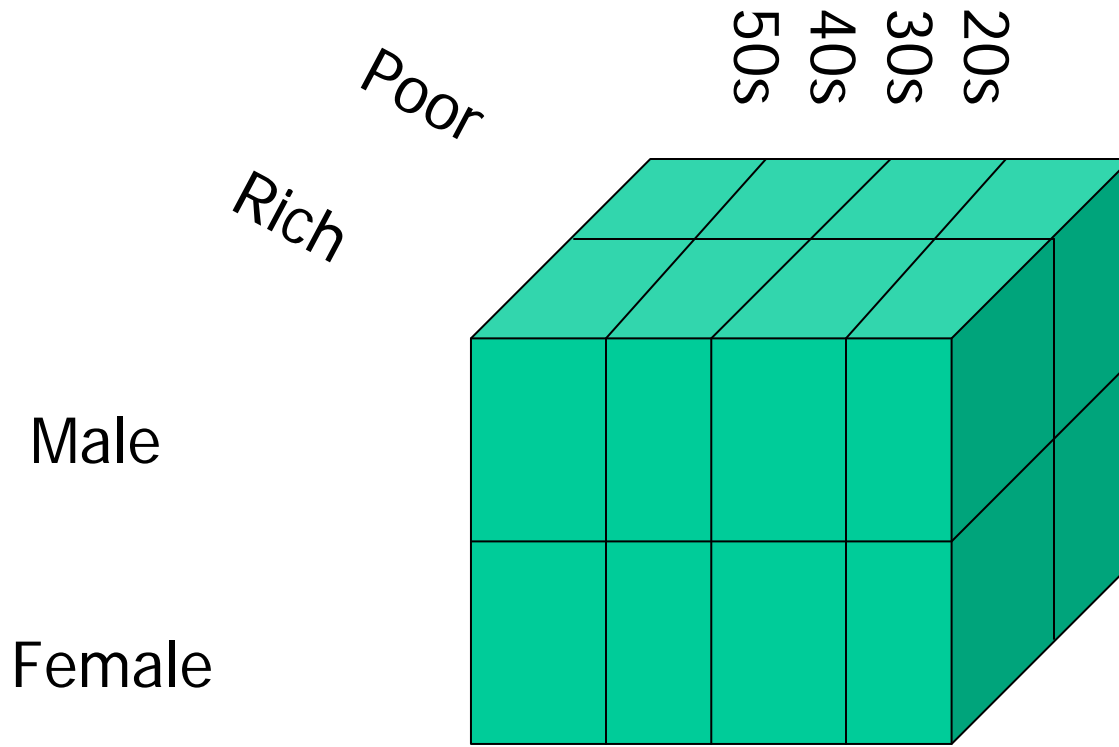
- Easier to see “interesting” things if we stretch out the histogram bars

# A bigger 2-d contingency table

job values:	Adm_clerical	Craft_repair	Farming_fishing	Machine_op_inspct	Priv_house_serv	Protective_serv	Tech_support									
*MissingValue*	Armed_Forces	Exec_managerial	Handlers_cleaners	Other_service	Prof_specialty	Sales	Transport_moving									
marital Divorced	270	1192	0	679	890	90	197	434	762	46	795	121	664	239	254	
Married_AF_spouse	5	6	0	4	3	1	1	1	5	0	4	1	5	0	1	
Married	928	1495	7	3818	3600	869	724	1469	1088	27	3182	583	2491	609	1489	
Married_spouse_absent	45	84	0	77	52	35	32	37	92	9	64	7	55	9	30	
Never_married	1242	2360	8	1301	1260	434	1029	872	2442	99	1849	237	1992	506	486	
Separated	97	224	0	160	126	23	63	123	275	21	145	23	146	48	56	
Widowed	222	250	0	73	155	38	26	86	259	40	133	11	151	35	39	

# 3-d contingency tables

- These are harder to look at!



# On-Line Analytical Processing (OLAP)

- Software packages and database add-ons to do this are known as OLAP tools
- They usually include point and click navigation to view slices and aggregates of contingency tables
- They usually include nice histogram visualization

## Time to stop and think

- Why would people want to look at contingency tables?

## Let's continue to think

- With 16 attributes, how many 1-d contingency tables are there?
- How many 2-d contingency tables?
- How many 3-d tables?
- With 100 attributes how many 3-d tables are there?

## Let's continue to think

- With 16 attributes, how many 1-d contingency tables are there? **16**
- How many 2-d contingency tables?  
 **$16\text{-choose-}2 = 16 * 15 / 2 = 120$**
- How many 3-d tables? **560**
- With 100 attributes how many 3-d tables are there? **161,700**

## Manually looking at contingency tables

- Looking at one contingency table: *can be as much fun as reading an interesting book*
- Looking at ten tables: *as much fun as watching CNN*
- Looking at 100 tables: *as much fun as watching an infomercial*
- Looking at 100,000 tables: *as much fun as a three-week November vacation in Duluth with a dying weasel.*

# Data Mining

- Data Mining is all about automating the process of searching for patterns in the data.

Which patterns are interesting?

Which might be mere illusions?

And how can they be exploited?

## Deciding whether a pattern is interesting

- We will use **information theory**
- A very large topic, originally used for compressing signals
- But more recently used for data mining...

*Let's review some basic knowledge....*

*(you should already know these stuff!)*

# Bits

You are watching a set of independent random samples of  $X$

You see that  $X$  has four possible values

$P(X=A) = 1/4$	$P(X=B) = 1/4$	$P(X=C) = 1/4$	$P(X=D) = 1/4$
----------------	----------------	----------------	----------------

So you might see: BAACBADCDADDDA...

You transmit data over a binary serial link. You can encode each reading with two bits (e.g.  $A = 00$ ,  $B = 01$ ,  $C = 10$ ,  $D = 11$ )

0100001001001110110011111100...

## Fewer Bits

Someone tells you that the probabilities are not equal

$P(X=A) = 1/2$	$P(X=B) = 1/4$	$P(X=C) = 1/8$	$P(X=D) = 1/8$
----------------	----------------	----------------	----------------

It's possible...

...to invent a coding for your transmission that only uses 1.75 bits on average per symbol. How?

# Fewer Bits

Someone tells you that the probabilities are not equal

$P(X=A) = 1/2$	$P(X=B) = 1/4$	$P(X=C) = 1/8$	$P(X=D) = 1/8$
----------------	----------------	----------------	----------------

It's possible...

...to invent a coding for your transmission that only uses 1.75 bits on average per symbol. How?

A	0
B	10
C	110
D	111

(This is just one of several ways)

$$(1 \cdot (1/2) + 2 \cdot (1/4) + 3 \cdot (1/8) + 3 \cdot (1/8)) = 1.75 \text{ bits/symbol}$$

# Fewer Bits

Suppose there are three equally likely values...

$P(X=A) = 1/3$	$P(X=B) = 1/3$	$P(X=C) = 1/3$
----------------	----------------	----------------

Here's a naïve coding, costing 2 bits per symbol

A	00
B	01
C	10

Can you think of a coding that would need only 1.6 bits per symbol on average?

In theory, it can in fact be done with 1.58496 bits per symbol.  
(why?)

# General Case

Suppose  $X$  can have one of  $m$  values...  $V_1, V_2, \dots, V_m$

$P(X=V_1) = p_1$	$P(X=V_2) = p_2$	....	$P(X=V_m) = p_m$
------------------	------------------	------	------------------

What's the smallest possible number of bits, on average, per symbol, needed to transmit a stream of symbols drawn from  $X$ 's distribution?  
It's

$$\begin{aligned} H(X) &= -p_1 \log_2 p_1 - p_2 \log_2 p_2 - \dots - p_m \log_2 p_m \\ &= -\sum_{j=1}^m p_j \log_2 p_j \end{aligned}$$

$H(X)$  = The entropy of  $X$  -- measure of the uncertainty associated with a random variable

- “High Entropy” means  $X$  is from a uniform (boring) distribution
- “Low Entropy” means  $X$  is from varied (peaks and valleys) distribution

# Entropy in a nut-shell



Low Entropy



High Entropy

# Entropy in a nut-shell



Low Entropy

..the values (locations of soup) sampled entirely from within the soup bowl



High Entropy

..the values (locations of soup) unpredictable... almost uniformly sampled throughout our dining room

# Specific Conditional Entropy $H(Y|X=v)$

Suppose I'm trying to predict output Y and I have input X

X = College Major

Y = Likes "Gladiator"

Let's assume this reflects the true probabilities

E.G. From this data we estimate

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

- $P(\text{LikeG} = \text{Yes}) = 0.5$
- $P(\text{Major} = \text{Math} \ \& \ \text{LikeG} = \text{No}) = 0.25$
- $P(\text{Major} = \text{Math}) = 0.5$
- $P(\text{LikeG} = \text{Yes} \mid \text{Major} = \text{History}) = 0$

Note:

- $H(X) = 1.5$
- $H(Y) = 1$

# Specific Conditional Entropy $H(Y|X=v)$

**X = College Major**

**Y = Likes "Gladiator"**

**Definition of Specific Conditional Entropy:**

$H(Y | X=v)$  = The entropy of  $Y$  among only those records in which  $X$  has value  $v$

<b>X</b>	<b>Y</b>
<b>Math</b>	<b>Yes</b>
<b>History</b>	<b>No</b>
<b>CS</b>	<b>Yes</b>
<b>Math</b>	<b>No</b>
<b>Math</b>	<b>No</b>
<b>CS</b>	<b>Yes</b>
<b>History</b>	<b>No</b>
<b>Math</b>	<b>Yes</b>

# Specific Conditional Entropy $H(Y|X=v)$

$X$  = College Major

$Y$  = Likes "Gladiator"

**Definition of Specific Conditional Entropy:**

$H(Y | X=v)$  = The entropy of  $Y$  among only those records in which  $X$  has value  $v$

**Example:**

- $H(Y|X=Math) = 1$
- $H(Y|X=History) = 0$
- $H(Y|X=CS) = 0$

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

# Conditional Entropy $H(Y|X)$

$X$  = College Major

$Y$  = Likes "Gladiator"

$X$	$Y$
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

## Definition of Conditional Entropy:

$H(Y | X)$  = The average specific conditional entropy of  $Y$

= if you choose a record at random what will be the conditional entropy of  $Y$ , conditioned on that row's value of  $X$

= Expected number of bits to transmit  $Y$  if both sides will know the value of  $X$

$$= \sum_j \text{Prob}(X=v_j) H(Y | X = v_j)$$

# Conditional Entropy

X = College Major

**Definition of Conditional Entropy:**

Y = Likes "Gladiator"

$H(Y|X)$  = The average conditional entropy of Y

$$= \sum_j \text{Prob}(X=v_j) H(Y | X = v_j)$$

**Example:**

$v_j$	$\text{Prob}(X=v_j)$	$H(Y   X = v_j)$
Math	0.5	1
History	0.25	0
CS	0.25	0

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

$$H(Y|X) = 0.5 * 1 + 0.25 * 0 + 0.25 * 0 = 0.5$$

# Information Gain

**X = College Major**

**Y = Likes "Gladiator"**

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

**Definition of Information Gain:**

$IG(Y|X)$  = I must transmit  $Y$ .  
How many bits on average  
would it save me if both ends of  
the line knew  $X$ ?

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

**Example:**

- $H(Y) = 1$
- $H(Y|X) = 0.5$
- Thus  $IG(Y|X) = 1 - 0.5 = 0.5$

# Information Gain Example

wealth values: poor rich

gender Female 14423 1769   $H(\text{wealth} | \text{gender} = \text{Female}) = 0.497654$

Male 22732 9918   $H(\text{wealth} | \text{gender} = \text{Male}) = 0.885847$

$H(\text{wealth}) = 0.793844$   $H(\text{wealth} | \text{gender}) = 0.757154$

$IG(\text{wealth} | \text{gender}) = 0.0366896$

For example:

Wealth: poor:  $14423 + 22732 = 37155$ ;

rich:  $1769 + 9918 = 11687$ ;

$P_{\text{poor}} = 0.7607$ ;  $P_{\text{rich}} = 0.2393$

$H(\text{wealth}) = 0.7607 * \log_2(1/0.7607) + 0.2393 * \log_2(1/0.2393) = 0.793844$

$H(\text{wealth} | \text{gender}) = (14423 + 1769) / (14423 + 1769 + 22732 + 9918) * 0.497654 +$   
 $(22732 + 9918) / (14423 + 1769 + 22732 + 9918) * 0.885847$   
 $= 0.757154$

# Another example

wealth values: poor rich

agegroup	10s	2507	3		$H(\text{wealth} \mid \text{agegroup} = 10\text{s}) = 0.0133271$
	20s	11262	743		$H(\text{wealth} \mid \text{agegroup} = 20\text{s}) = 0.334906$
	30s	9468	3461		$H(\text{wealth} \mid \text{agegroup} = 30\text{s}) = 0.838134$
	40s	6738	3986		$H(\text{wealth} \mid \text{agegroup} = 40\text{s}) = 0.951961$
	50s	4110	2509		$H(\text{wealth} \mid \text{agegroup} = 50\text{s}) = 0.957376$
	60s	2245	809		$H(\text{wealth} \mid \text{agegroup} = 60\text{s}) = 0.834049$
	70s	668	147		$H(\text{wealth} \mid \text{agegroup} = 70\text{s}) = 0.680882$
	80s	115	16		$H(\text{wealth} \mid \text{agegroup} = 80\text{s}) = 0.535474$
	90s	42	13		$H(\text{wealth} \mid \text{agegroup} = 90\text{s}) = 0.788941$

$H(\text{wealth}) = 0.793844$      $H(\text{wealth} \mid \text{agegroup}) = 0.709463$

$IG(\text{wealth} \mid \text{agegroup}) = 0.0843813$

## What is Information Gain used for?

Suppose you are trying to predict whether someone is going live past 80 years. From historical data you might find...







- $IG(\text{LongLife} \mid \text{HairColor}) = 0.01$
- $IG(\text{LongLife} \mid \text{Smoker}) = 0.2$
- $IG(\text{LongLife} \mid \text{Gender}) = 0.25$
- $IG(\text{LongLife} \mid \text{LastDigitOfSSN}) = 0.00001$

IG tells you how interesting a 2-d contingency table is going to be.

# Searching for High Info Gains

- Given something (e.g. wealth) you are trying to predict, it is easy to ask the computer to find which attribute has highest information gain for it.

wealth values: poor rich

relation	Husband	10870	8846		$H(\text{wealth} \mid \text{relation} = \text{Husband}) = 0.992385$
	Not_in_family	11307	1276		$H(\text{wealth} \mid \text{relation} = \text{Not\_in\_family}) = 0.473439$
	Other_relative	1454	52		$H(\text{wealth} \mid \text{relation} = \text{Other\_relative}) = 0.216617$
	Own_child	7470	111		$H(\text{wealth} \mid \text{relation} = \text{Own\_child}) = 0.110192$
	Unmarried	4816	309		$H(\text{wealth} \mid \text{relation} = \text{Unmarried}) = 0.328606$
	Wife	1238	1093		$H(\text{wealth} \mid \text{relation} = \text{Wife}) = 0.997207$

$H(\text{wealth}) = 0.793844$   $H(\text{wealth} \mid \text{relation}) = 0.628421$

$IG(\text{wealth} \mid \text{relation}) = 0.165423$

# Learning Decision Trees

- A Decision Tree is a tree-structured plan of a set of attributes to test in order to predict the output.
- To decide which attribute should be tested first, simply find the one with the highest information gain.
- Then recurse...

# A small dataset: Miles Per Gallon

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	europa
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	europa
bad	5	medium	medium	medium	medium	75to78	europa

From the UCI repository (thanks to Ross Quinlan)

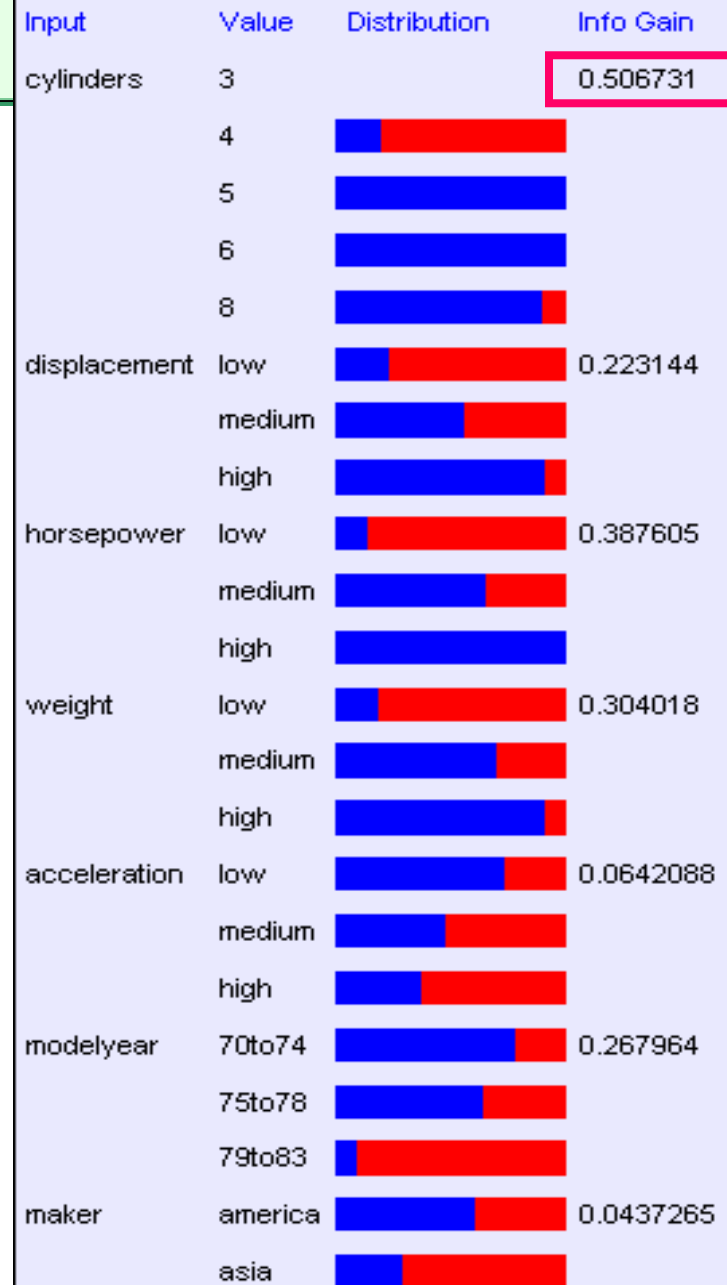
Suppose we want to predict MPG.

Look at all the information gains...

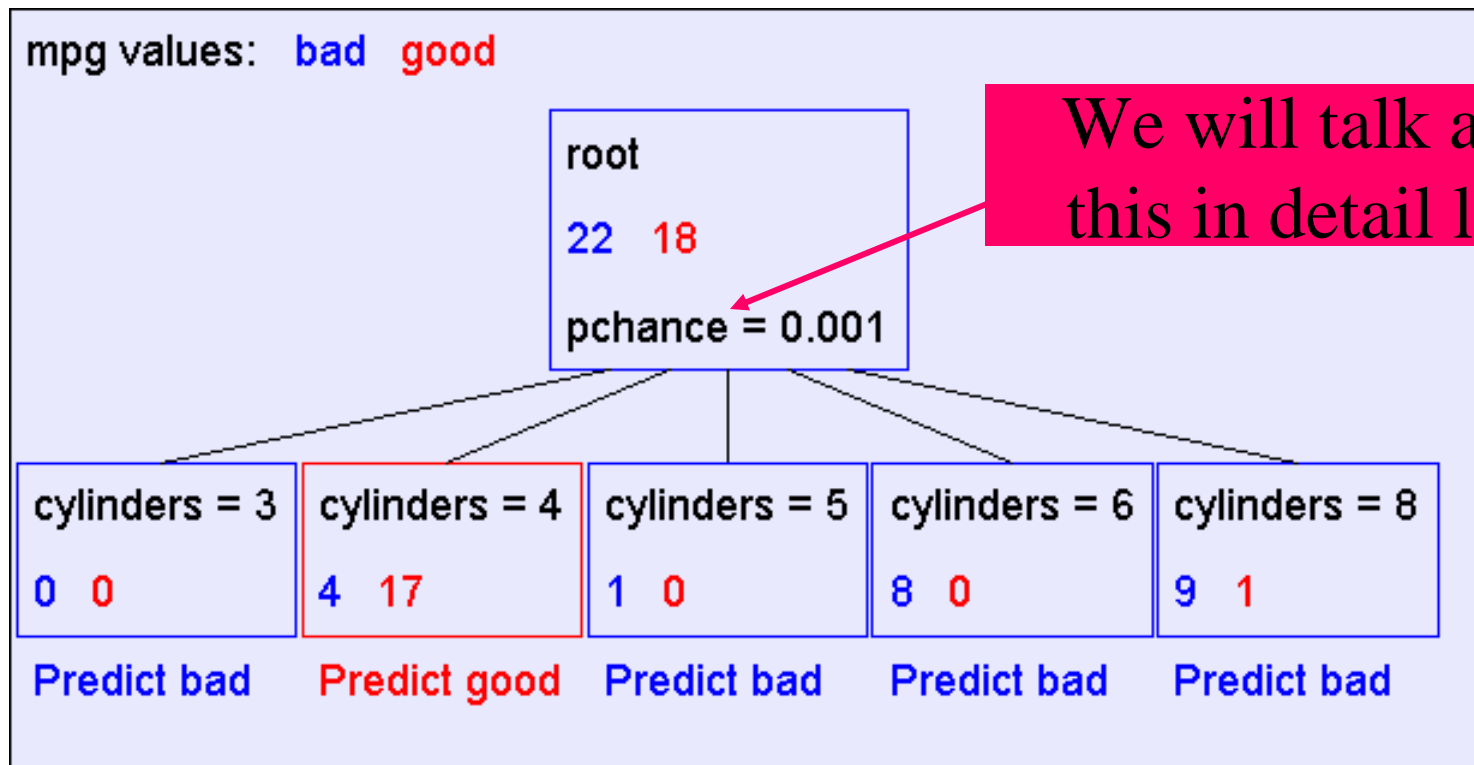
“Cylinders” has the highest Information Gain

Information gains using the training set (40 records)

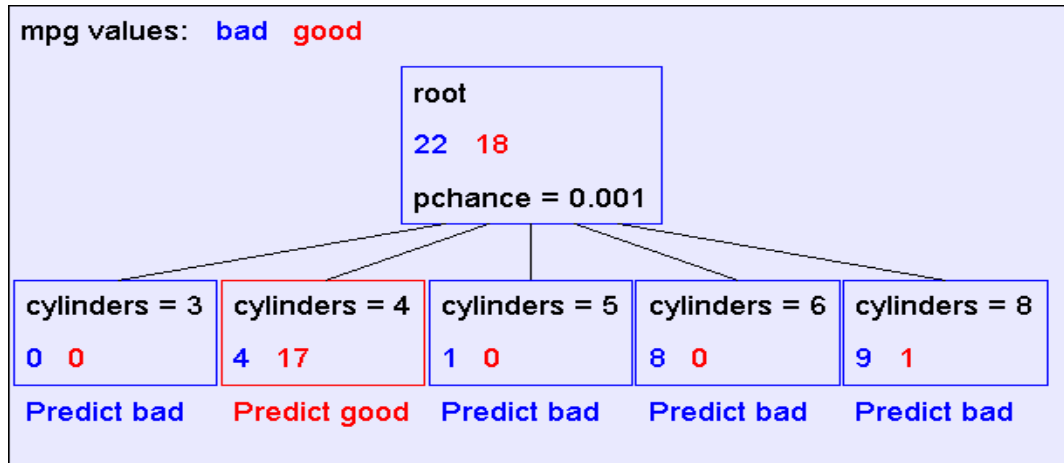
mpg values: bad good



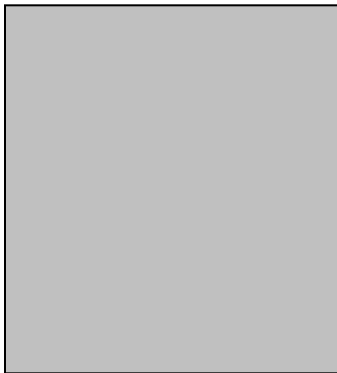
# A Decision Stump



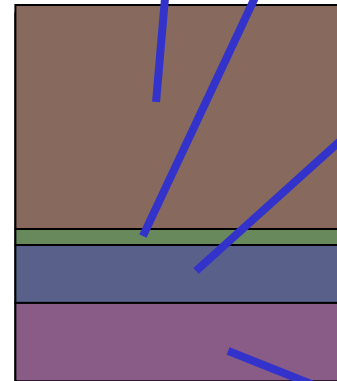
# Recursion Step



Take the Original Dataset..



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



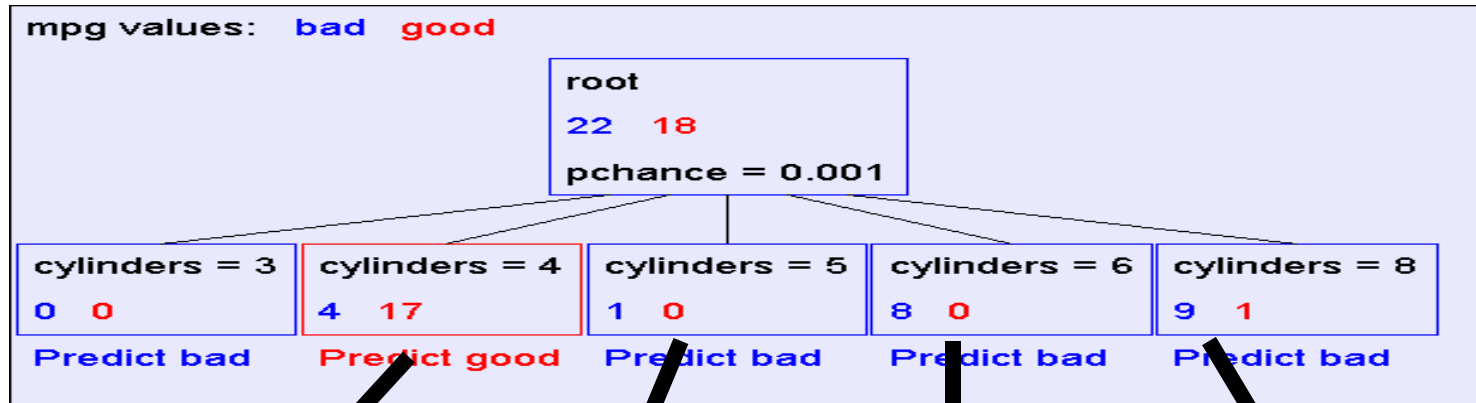
Records in which cylinders = 4

Records in which cylinders = 5

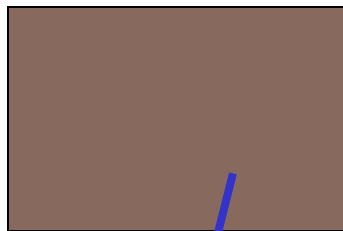
Records in which cylinders = 6

Records in which cylinders = 8  
47

# Recursion Step



Build tree from  
These records..



Records in  
which  
cylinders = 4

Build tree from  
These records..



Records in  
which  
cylinders = 5

Build tree from  
These records..



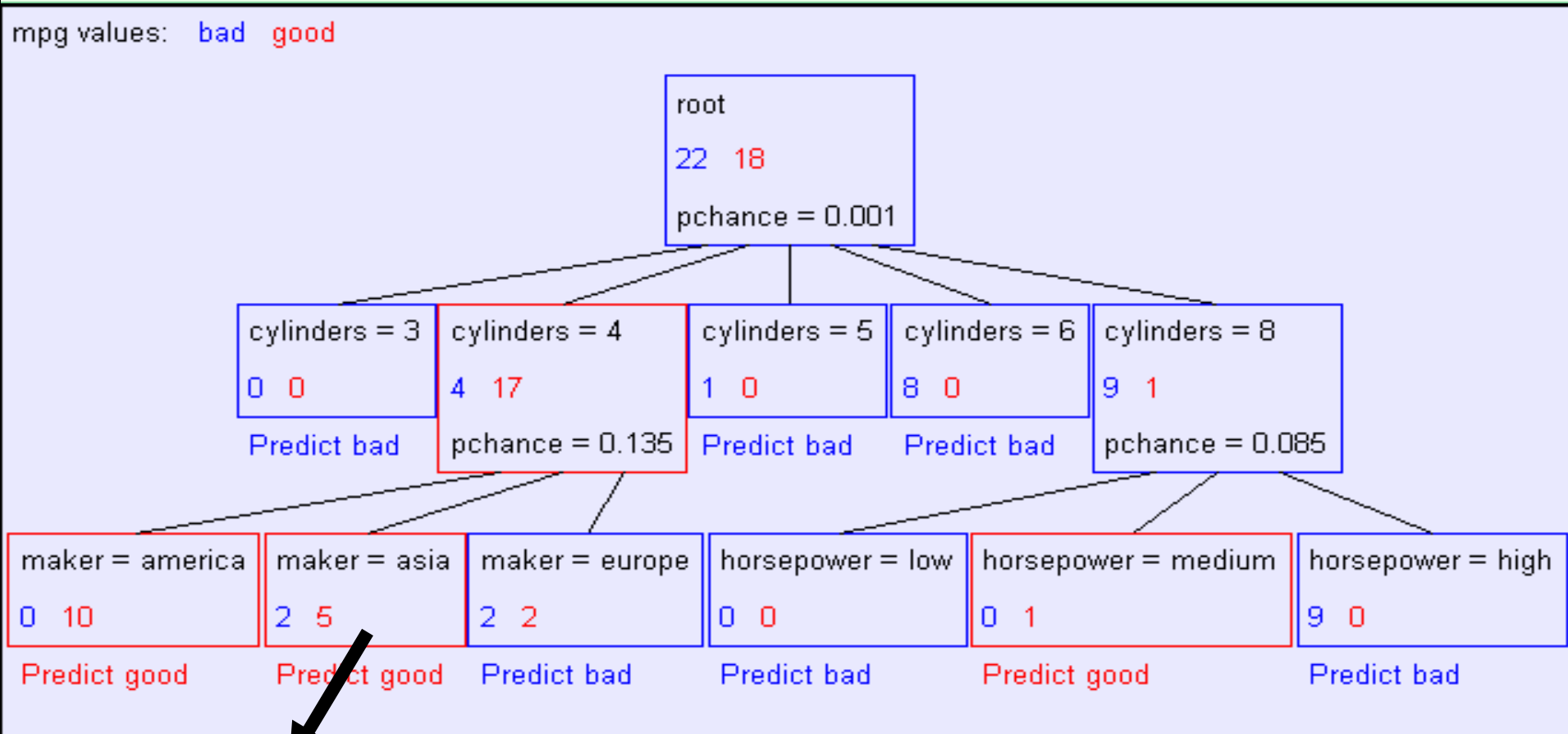
Records in  
which  
cylinders = 6

Build tree from  
These records..



Records in  
which  
cylinders = 8

# Second level of tree

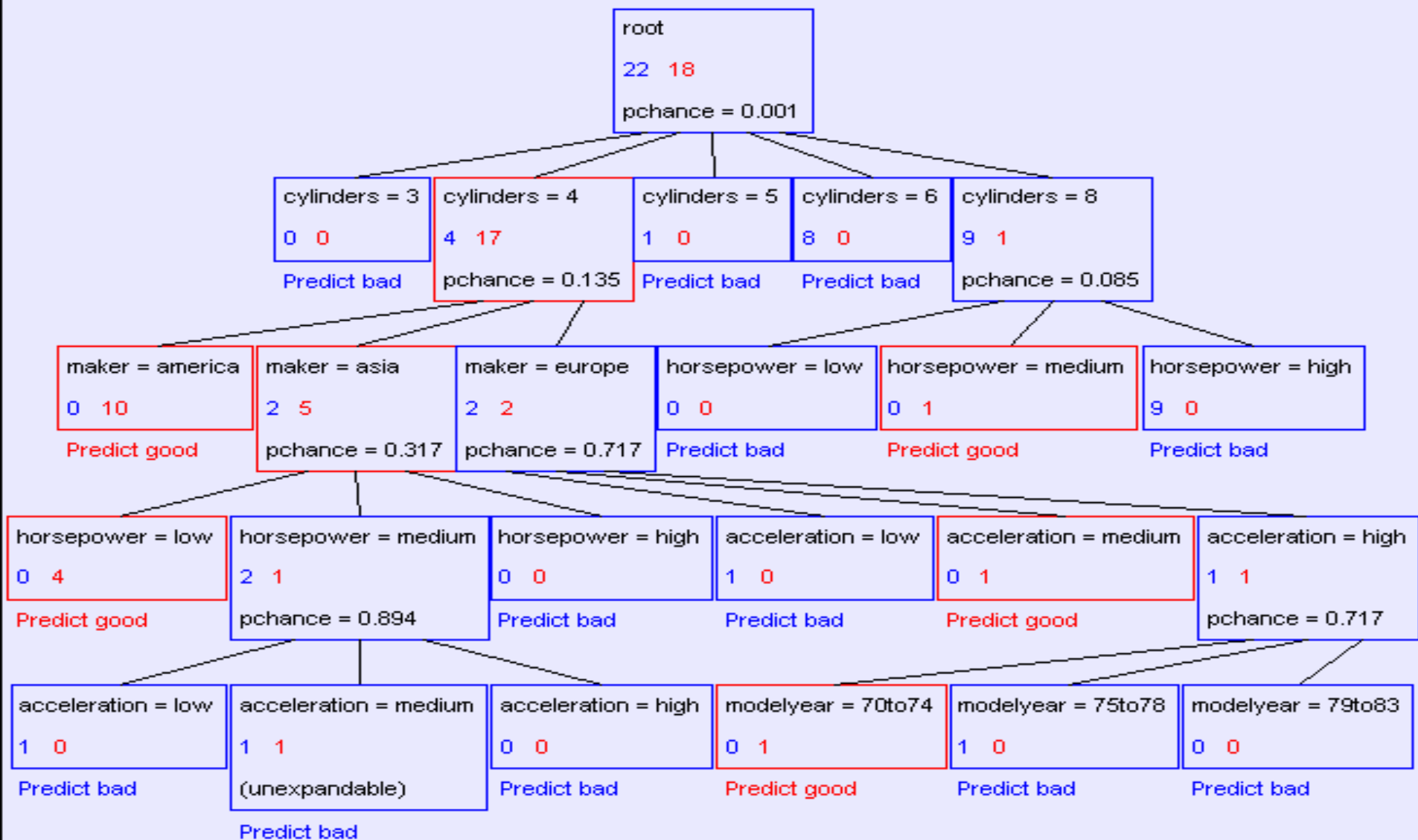


Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

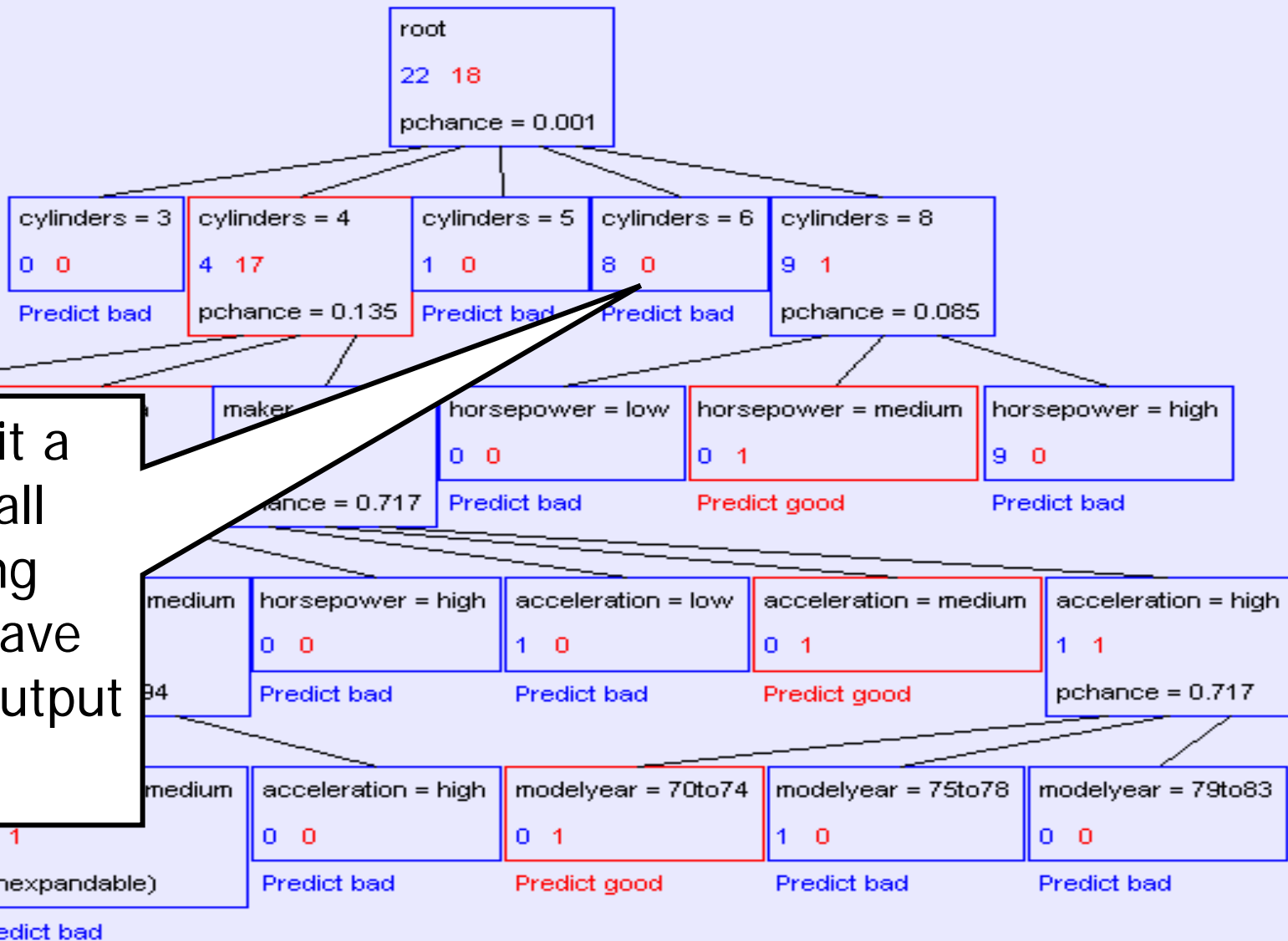
# The final tree

mpg values: bad good



# Base Case One

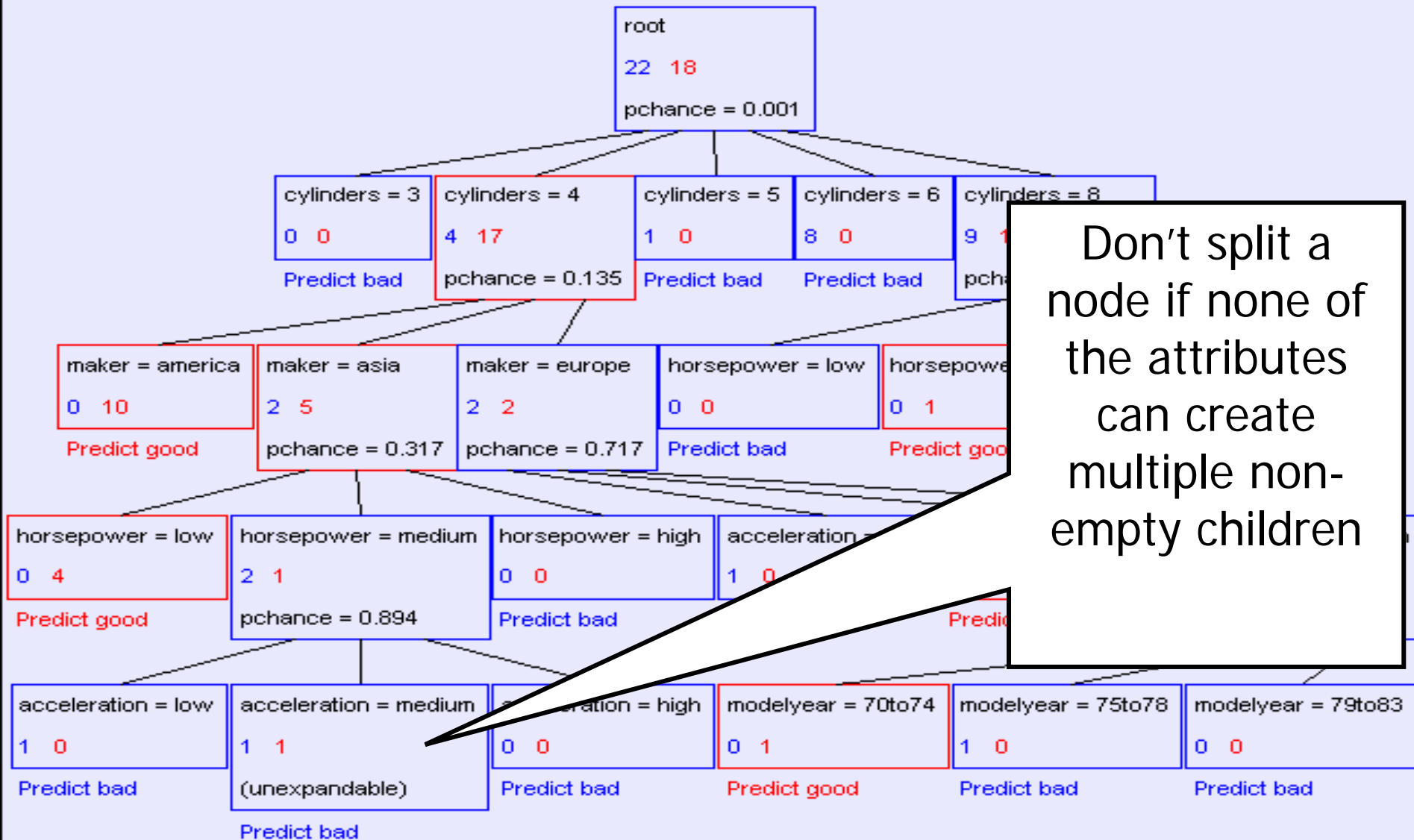
mpg values: bad good



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

# Base Case Two

mpg values: bad good



## When to stop?

Continue developing the tree until either of the two conditions is met:

- (1) Every attribute has already been included along this path through the tree
- (2) The training examples associated with this leaf node all have the same target attribute value (i.e. their entropy is zero)

# Summary of ID3 algorithm

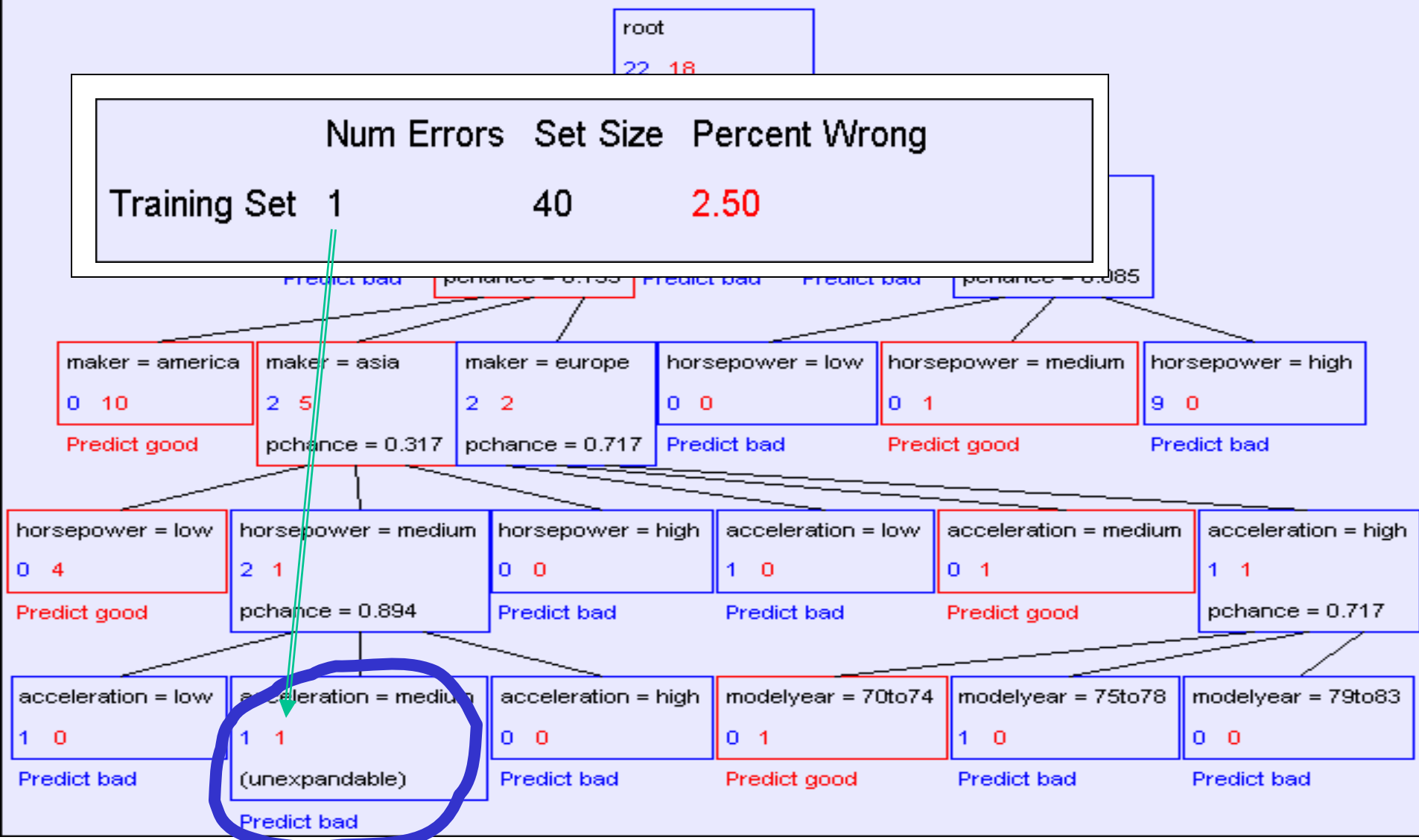
- ID3 (Examples, Target\_Attribute, Attributes)
- Create a root node for the tree
- IF all examples are positive, Return the single-node tree Root, with label = +
- If all examples are negative, Return the single-node tree Root, with label = -
- If attributes is empty, then Return the single node tree Root, with label = *most common value of the target attribute in the examples*
- Otherwise Begin
  - $A \leftarrow$  The Attribute that best classifies examples (according to information gain)
  - The decision tree attribute for Root  $\leftarrow A$
  - For each possible value,  $vi$ , of  $A$ ,
    - Add a new tree branch below Root, corresponding to the test  $A = vi$
    - Let Examples( $vi$ ) be the subset of examples that have the value  $vi$  for  $A$
    - If Examples( $vi$ ) is empty
      - Then below this new branch add a leaf node with label = most common target value in the examples
    - Else below this new branch add the subtree  
ID3 (Examples( $vi$ ), Target\_Attribute, Attributes – { $A$ })
- End
- Return Root

# Training Set Error

- For each record, follow the decision tree to see what it would predict  
For what number of records does the decision tree's prediction disagree with the true value in the database?
- This quantity is called the *training set error*. The smaller the better.

# MPG Training error

mpg values: bad good



# Stop and reflect: Why are we doing this learning anyway?

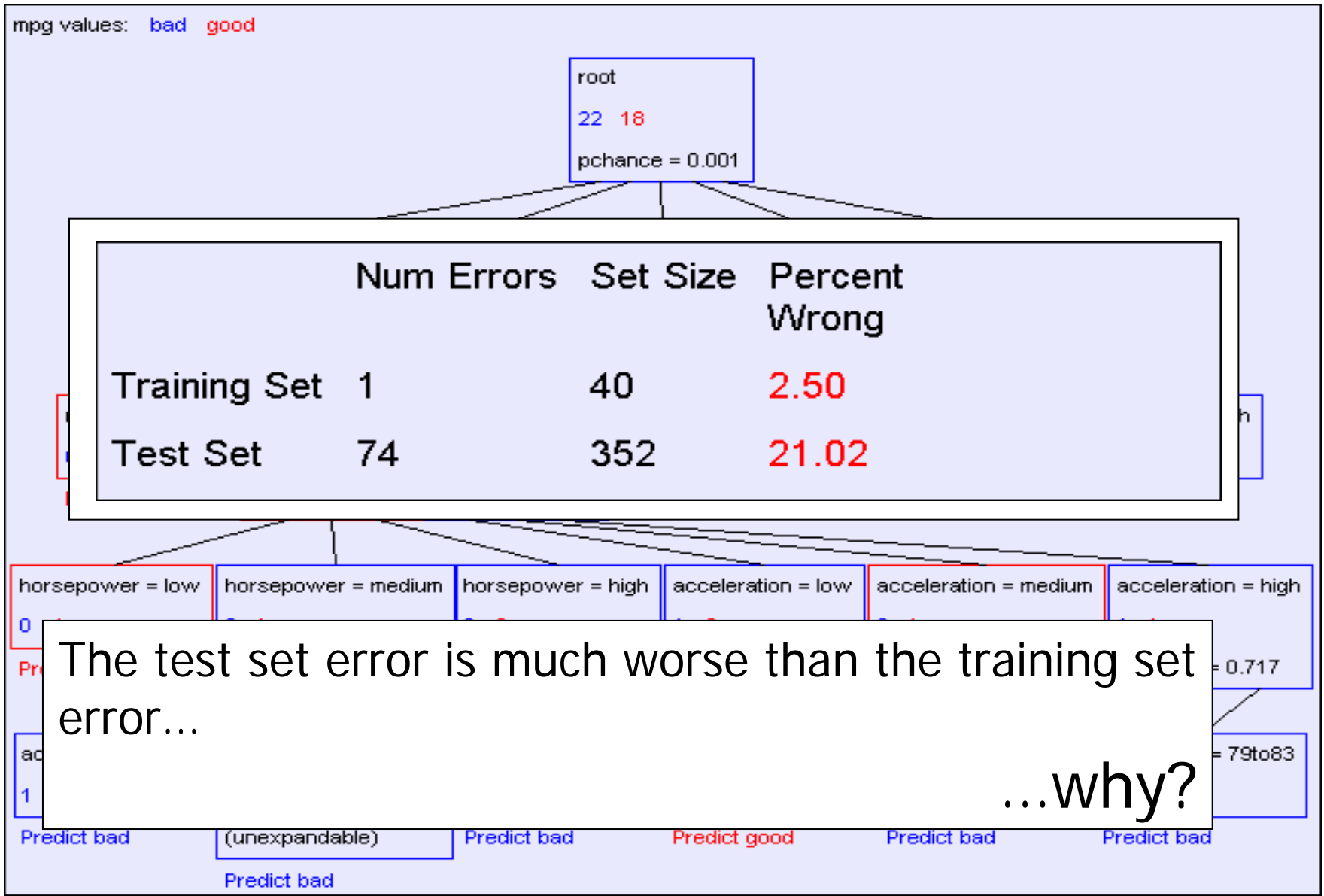
- It is not usually in order to predict the training data's output on data we have already seen.
- It is more commonly in order to predict the output value for **future data** we have not yet seen.

*Warning: A common data mining misperception is that the above two bullets are the only possible reasons for learning. There are at least a dozen others.*

# Test Set Error

- Suppose we are forward thinking.
- We hide some data away when we learn the decision tree.
- But once learned, we see how well the tree predicts that data.
- This is a good simulation of what happens when we try to predict future data.
- And it is called **Test Set Error**.

# MPG Test set error



# The overfitting issue

•Definition: If your machine learning algorithm fits noise (i.e. pays attention to parts of the data that are irrelevant) it is **overfitting**.

Consider error of hypothesis  $h$  over

- training data:  $error_{train}(h)$
- entire distribution  $\mathcal{D}$  of data:  $error_{\mathcal{D}}(h)$

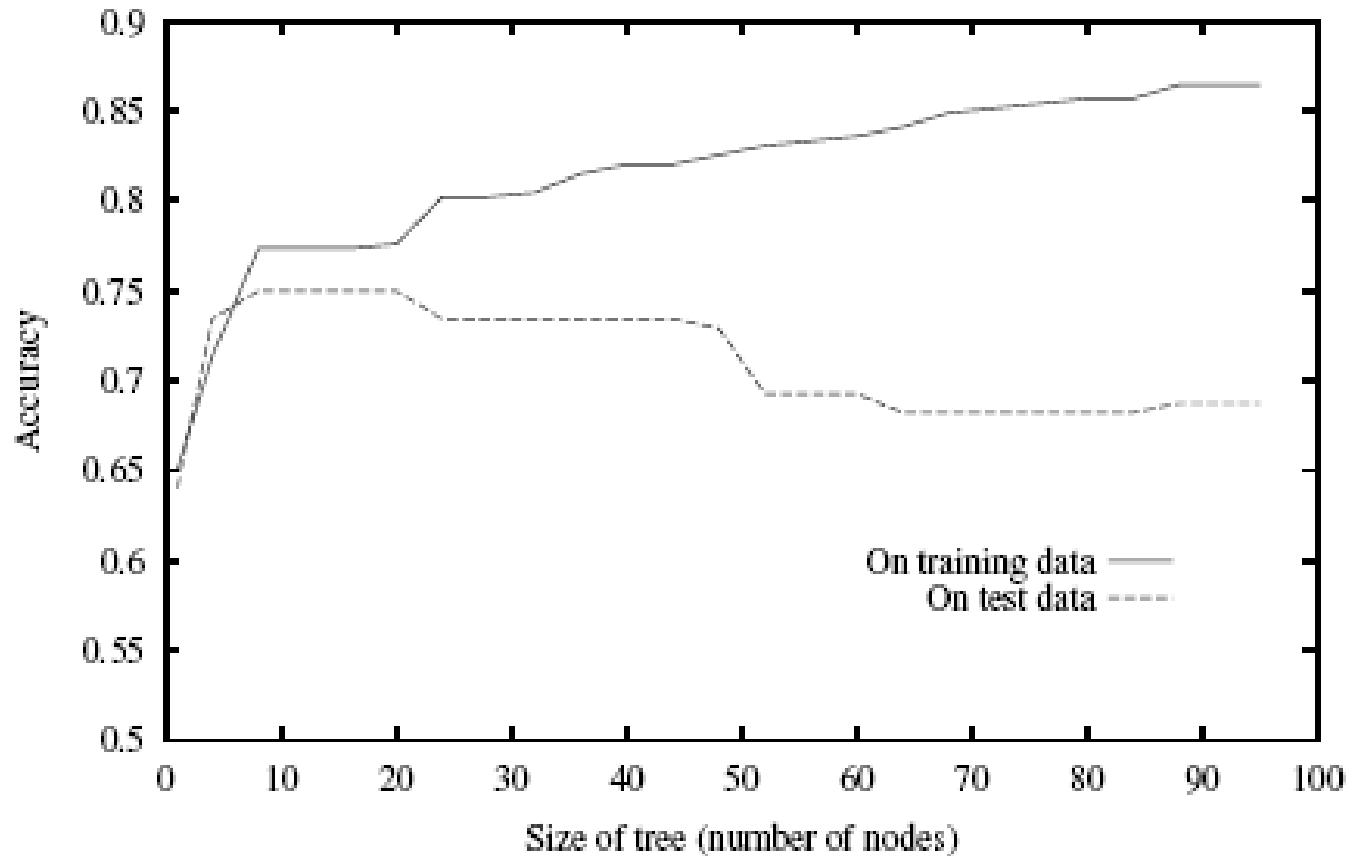
Hypothesis  $h \in H$  **overfits** training data if there is an alternative hypothesis  $h' \in H$  such that

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

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

# Overfitting in decision tree learning (ID3 algorithm)



# Avoiding overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune

How to select “best” tree:

- Measure performance over training data
- Measure performance over separate validation data set

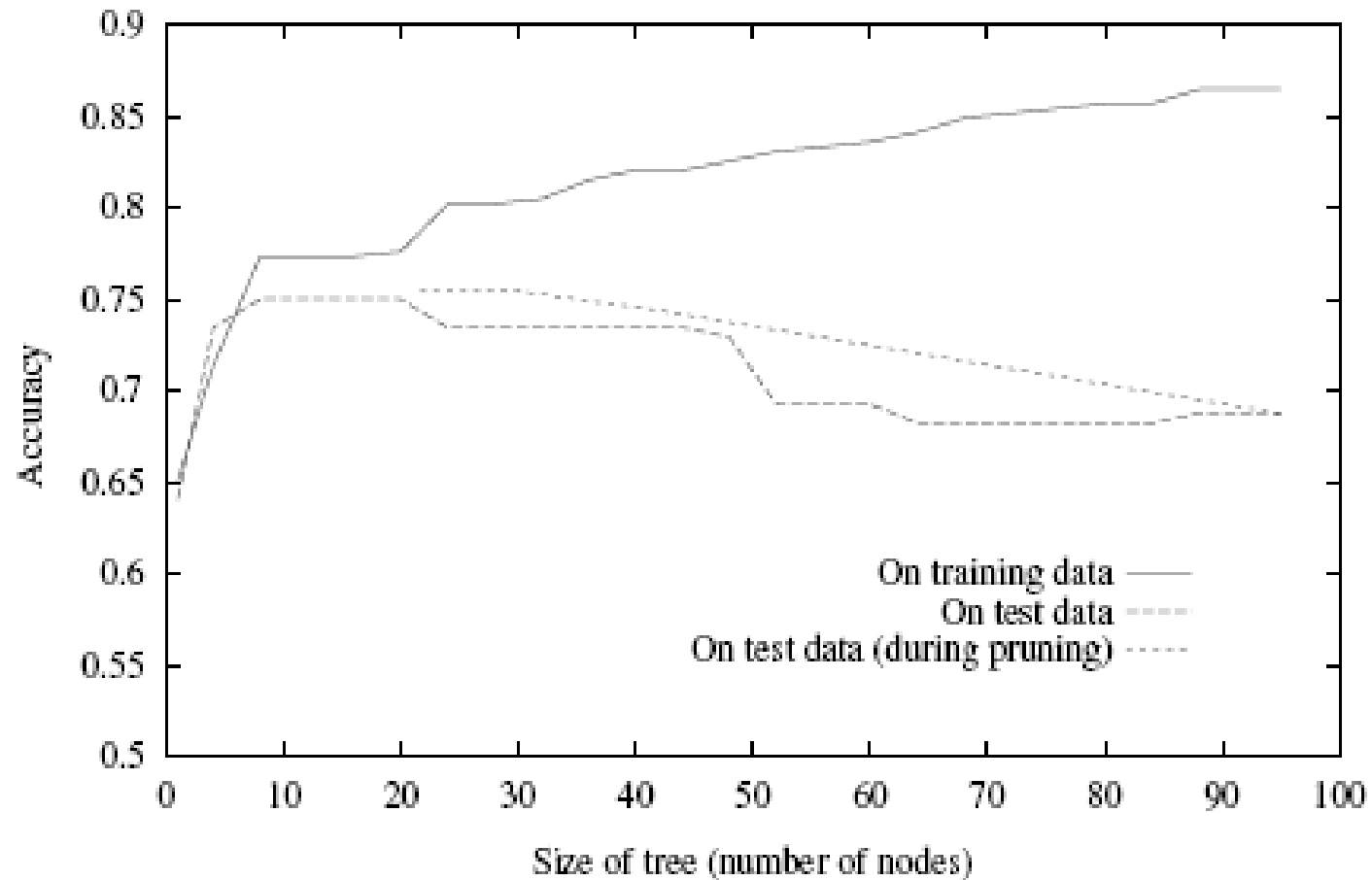
# Reduced error pruning

Split data into *training* and *validation* set

Do until further pruning is harmful:

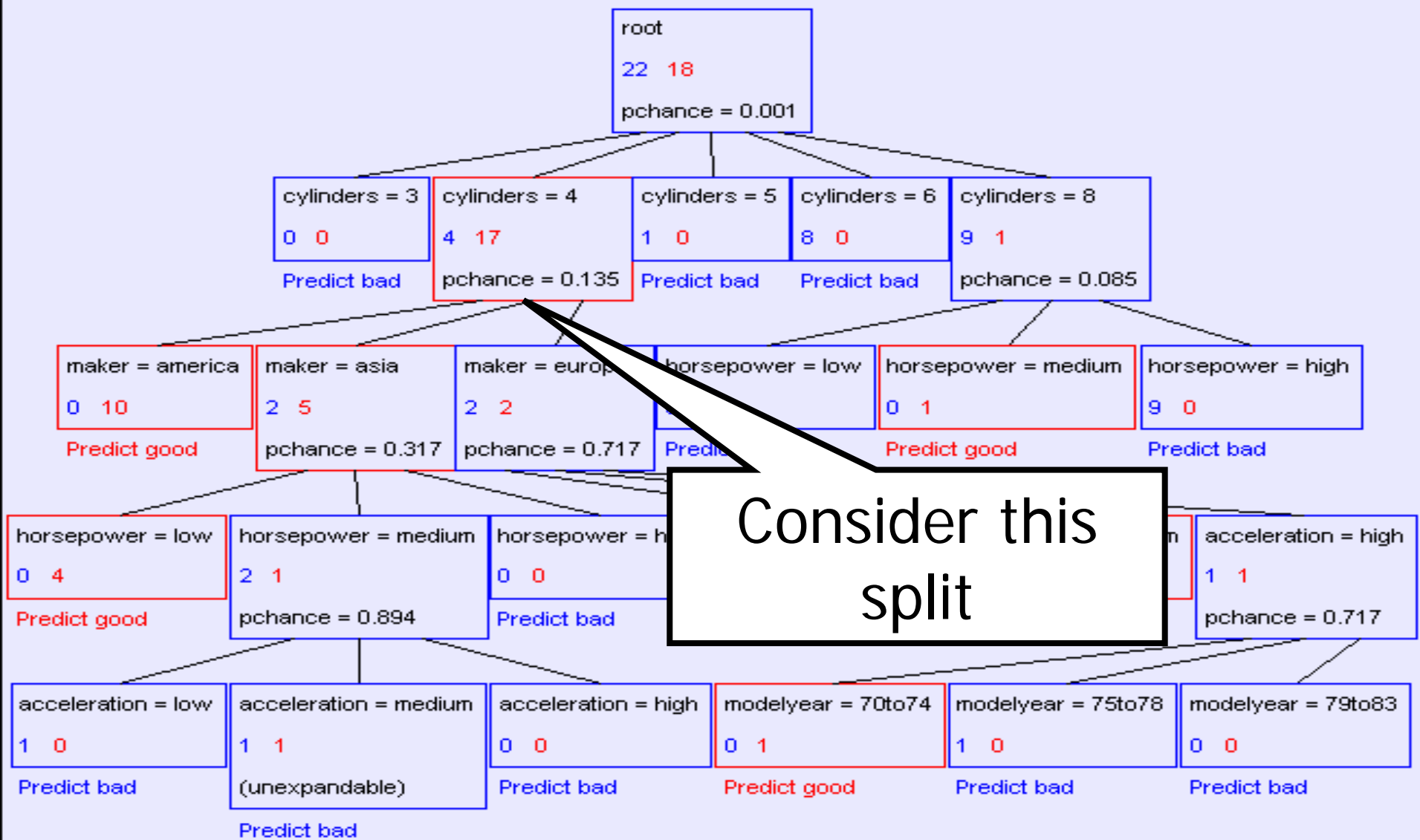
1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
  2. Greedily remove the one that most improves *validation* set accuracy
- produces smallest version of most accurate subtree
  - What if data is limited?

# Effects of reduced error pruning

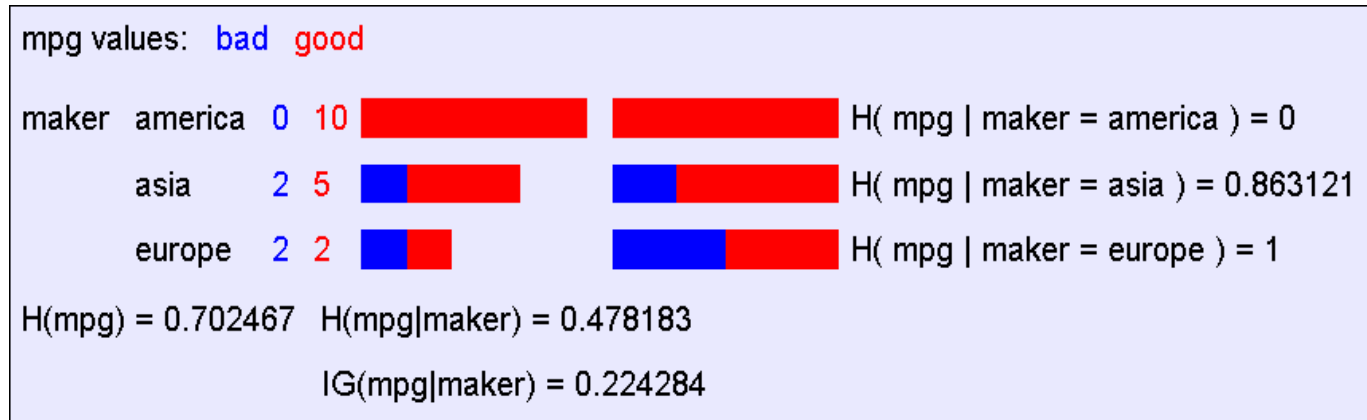


# Let's see another method for pruning: Chi-squared pruning

mpg values: bad good



# A chi-squared test



- Suppose that mpg was completely uncorrelated (irrelevant) 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-squared test, the answer is  $0.135 = 13.5\%$ .

how to do chi-squared test???

## Review of chi-squared test

Chi-squared pruning: to test whether splitting on an attribute contributes a statistically significant amount of information.

**-- Use the statistical significance test to Find the irrelevant attribute.**

ID3 used Chi-squared test

## Review of chi-squared test

Suppose at one node in the tree have training data  $S$ , the number of positive examples in  $S$  is  $p$ , and the number of negative examples is  $n$ , then  $p+n=S$

By the information gain analysis, we select an attribute that splits  $S$  into a number of subsets  $S_i$ , each of these subsets has  $p_i$  positive examples and  $n_i$  negative examples. ( $p_i + n_i = |S_i|$ )

If this attribute is irrelevant, then we would expect that the number of positive and negative examples in would be:

$$\hat{p}_i = \frac{p}{p+n} |S_i| \quad \hat{n}_i = \frac{n}{p+n} |S_i|$$

Then we can calculate how much “error” there is between the actual number of positive and negative examples in each  $S_i$  and the expected number.

$$Q = \sum_i \frac{(p_i - \hat{p}_i)^2}{\hat{p}_i} + \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i}$$

Note: this sum is over each subset created by the split on this attribute.

## Review of chi-squared test

Small  $Q$  : this attribute is not relevant -- the data in each split are following the same distribution as the data before splitting on this attribute.

Large  $Q$  , means that there is a lot of "error" between what we would have expected (under the irrelevant attribute hypothesis) and the actual distribution of examples.

The  $\chi^2$  distribution (chi-squared distribution) will determine the probability that the attribute is not relevant.

Let's see how to calculate the probability based on the  $Q$  and chi-squared distribution!

# Review of chi-square distribution

If  $X_i$  are  $k$  independent, normally distributed random variables with mean 0 and variance

1, then the random variable  $Q = \sum_{i=1}^k X_i^2$

is distributed according to the chi-square distribution:

$$Q \sim \chi_k^2.$$

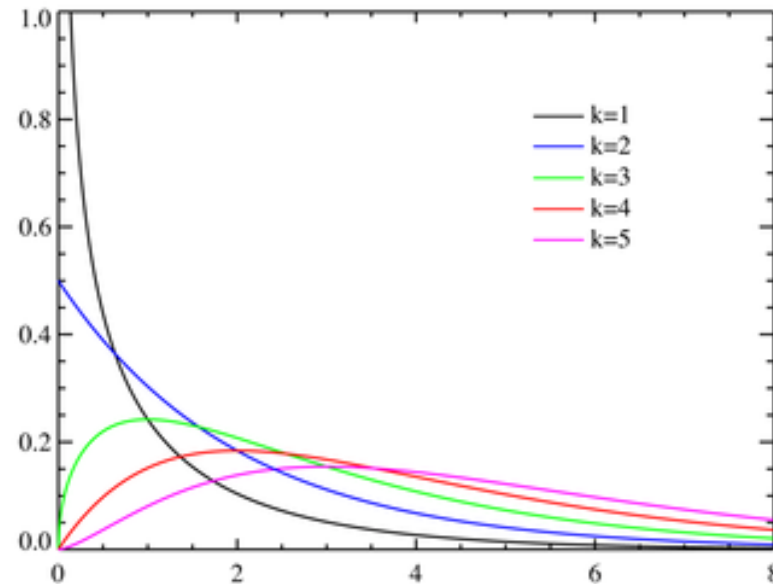
The chi-square distribution has one parameter:  $k$  - a positive integer that specifies the number of degrees of freedom.

A probability density function of the chi-square distribution is

$$f(x; k) = \begin{cases} \frac{1}{2^{k/2}\Gamma(k/2)} x^{k/2-1} e^{-x/2} & \text{for } x > 0, \\ 0 & \text{for } x \leq 0, \end{cases}$$

Where the Gamma function is:

$$\Gamma(z) = \int_0^{\infty} t^{z-1} e^{-t} dt$$



# Review of chi-squared test

You can use a statistical software or Chi-Square Table to estimate the probability that the attribute is really irrelevant or not.

df\area	.995	.990	.975	.950	.900	.750	.500	.250	.100	.050	.025	.010	.005
1	0.00004	0.00016	0.00098	0.00393	0.01579	0.10153	0.45494	1.32330	2.70554	3.84146	5.02389	6.63490	7.87944
2	0.01003	0.02010	0.05064	0.10259	0.21072	0.57536	1.38629	2.77259	4.60517	5.99146	7.37776	9.21034	10.59663
3	0.07172	0.11483	0.21580	0.35185	0.58437	1.21253	2.36597	4.10834	6.25139	7.81473	9.34840	11.34487	12.83816
4	0.20699	0.29711	0.48442	0.71072	1.06362	1.92256	3.35669	5.38527	7.77944	9.48773	11.14329	13.27670	14.86026
5	0.41174	0.55430	0.83121	1.14548	1.61031	2.67460	4.35146	6.62568	9.23636	11.07050	12.83250	15.08627	16.74960
6	0.67573	0.87209	1.23734	1.63538	2.20413	3.45460	5.34812	7.84080	10.64464	12.59159	14.44938	16.81189	18.54758
7	0.98926	1.23904	1.68987	2.16735	2.83311	4.25485	6.34581	9.03715	12.01704	14.06714	16.01276	18.47531	20.27774
8	1.34441	1.64650	2.17973	2.73264	3.48954	5.07064	7.34412	10.21885	13.36157	15.50731	17.53455	20.09024	21.95495
9	1.73493	2.08790	2.70039	3.32511	4.16816	5.89883	8.34283	11.38875	14.68366	16.91898	19.02277	21.66599	23.58935
10	2.15586	2.55821	3.24697	3.94030	4.86518	6.73720	9.34182	12.54886	15.98718	18.30704	20.48318	23.20925	25.18818

What you need:

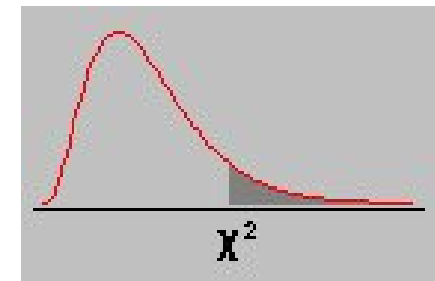
Q and degree of freedom (df)

$df = (\text{number of values of this attribute}) - 1$

$= (\text{number of subsets split by this attribute}) - 1$

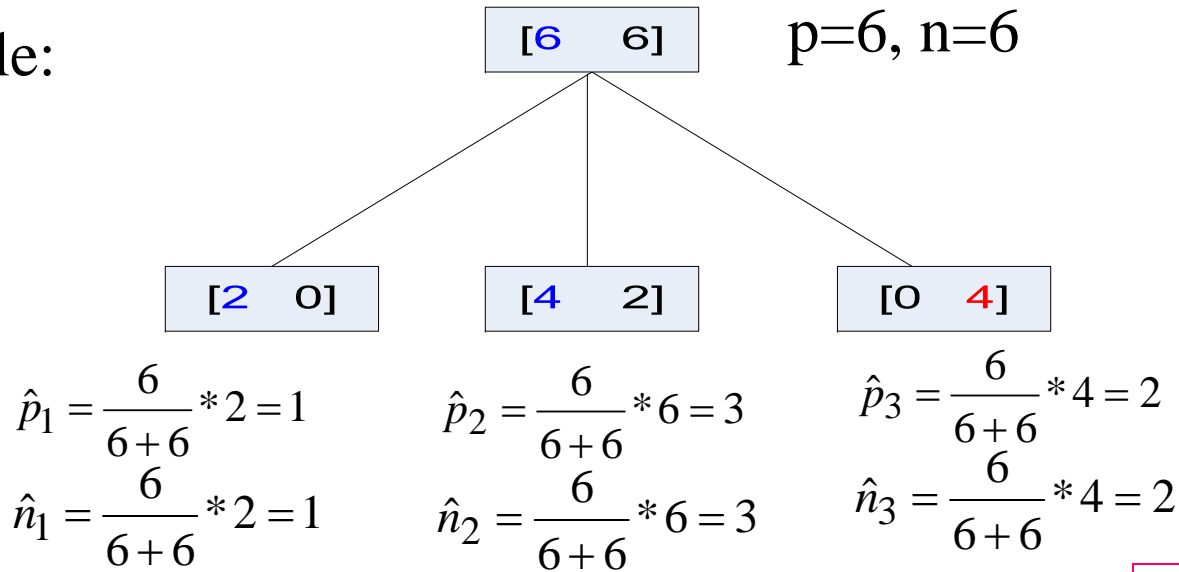
You can find such Chi-square table in many places, such as

<http://www.statsoft.com/textbook/sttable.html#chi>



# Prune based on Chi-squared test

Example:



$$Q = \sum_i \frac{(p_i - \hat{p}_i)^2}{\hat{p}_i} + \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i}$$

$$= \frac{(2-1)^2}{1} + \frac{(0-1)^2}{1} + \frac{(4-3)^2}{3} + \frac{(2-3)^2}{3} + \frac{(0-2)^2}{2} + \frac{(4-2)^2}{2}$$

$$= 1+1+1/3+1/3+2+2 = 6.6667$$

Compare this to a MaxPchance



Degree of freedom (df) = 3-1=2, so  $p_{chance} = 0.03567$

You can use the Matlab function to calculate: 1-chi2cdf(6.67,2)

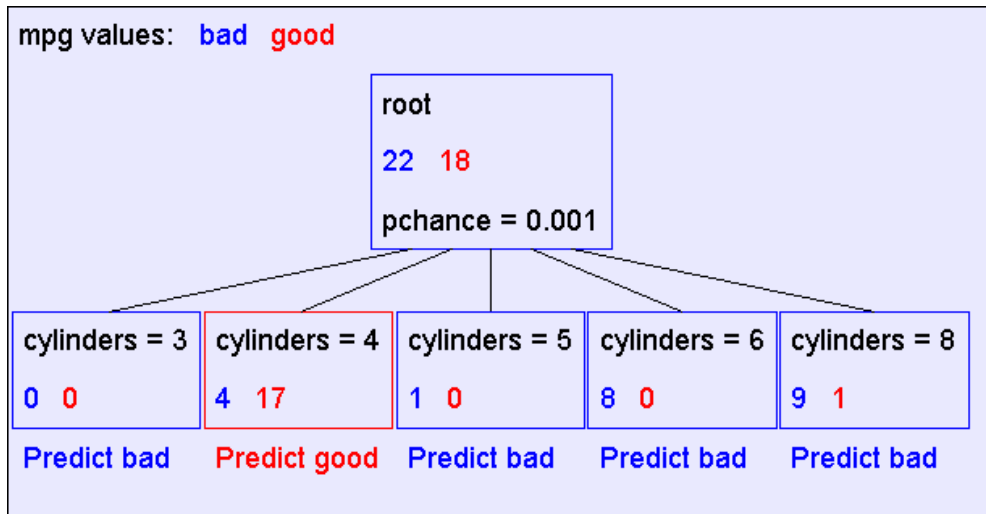
## Using Chi-squared to avoid overfitting

- 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  $p_{chance} > \textit{MaxPchance}$ .
  - Continue working your 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  $\text{MaxPchance} = 0.1$ , you will see the following MPG decision tree:



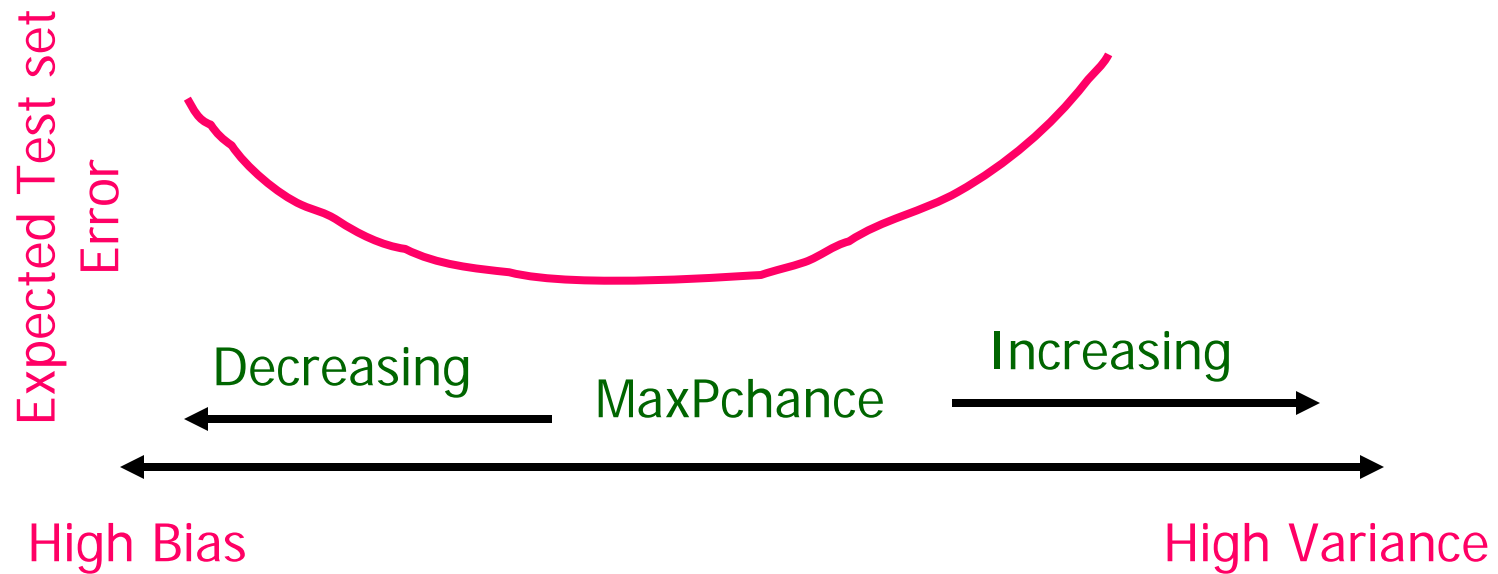
Note the improved test set accuracy compared with the unpruned tree

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

# MaxPchance

- **Good news:** The decision tree can automatically adjust its pruning decisions according to the amount of apparent noise and data.
- **Bad news:** The user must come up with a good value of MaxPchance. (for instance, 0.05 as a magic parameter).
- **Good news:** But with extra work, the best MaxPchance value can be estimated automatically by *cross-validation*.

# MaxPchance



## Other methods: estimating error rate

There are many other methods for pruning: C4.5 used the confidence level estimation for pruning

- Bottom-up pruning: at each non-leaf node  $v$ , if merging the subtree at  $v$  into a leaf node improves accuracy, perform the merging.
  - Method 1: compute accuracy using examples not seen by the algorithm.
  - Method 2: estimate accuracy using the training examples:
- Consider classifying  $E$  examples incorrectly out of  $N$  examples as observing  $E$  events in  $N$  trials in the binomial distribution.
  - For a given confidence level  $CF$ , the upper limit on the error rate over the whole population is  $U_{CF}(E, N)$  with  $CF\%$  confidence.

## Other methods: estimating error rates

- Error estimate for subtree is weighted sum of error estimates for all its leaves
- Error estimate for a node:

$$e = \left( f + \frac{z^2}{2N} + z \sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right) / \left( 1 + \frac{z^2}{N} \right)$$

- If confidence level  $c = 25\%$  then  $z = 0.69$  (from normal distribution)
- $f$  is the error on the training data
- $N$  is the number of instances covered by the leaf

Reference: Dr. Qiang Yang, Decision tree, [online], available:

<http://www.cs.ust.hk/~qyang/521/PPT/dtrees2.ppt#294,1>, Classification with Decision Trees II

## Example for Estimating Error

- Consider a subtree rooted at Outlook with 3 leaf nodes:
  - Sunny: Play = yes : (0 error, 6 instances)
  - Overcast: Play= yes: (0 error, 9 instances)
  - Cloudy: Play = no (0 error, 1 instance)

$$U_{0.25}(6,0) = 0.206, U_{0.25}(9,0) = 0.143, U_{0.25}(1,0) = 0.750$$

- The estimated error for this subtree is

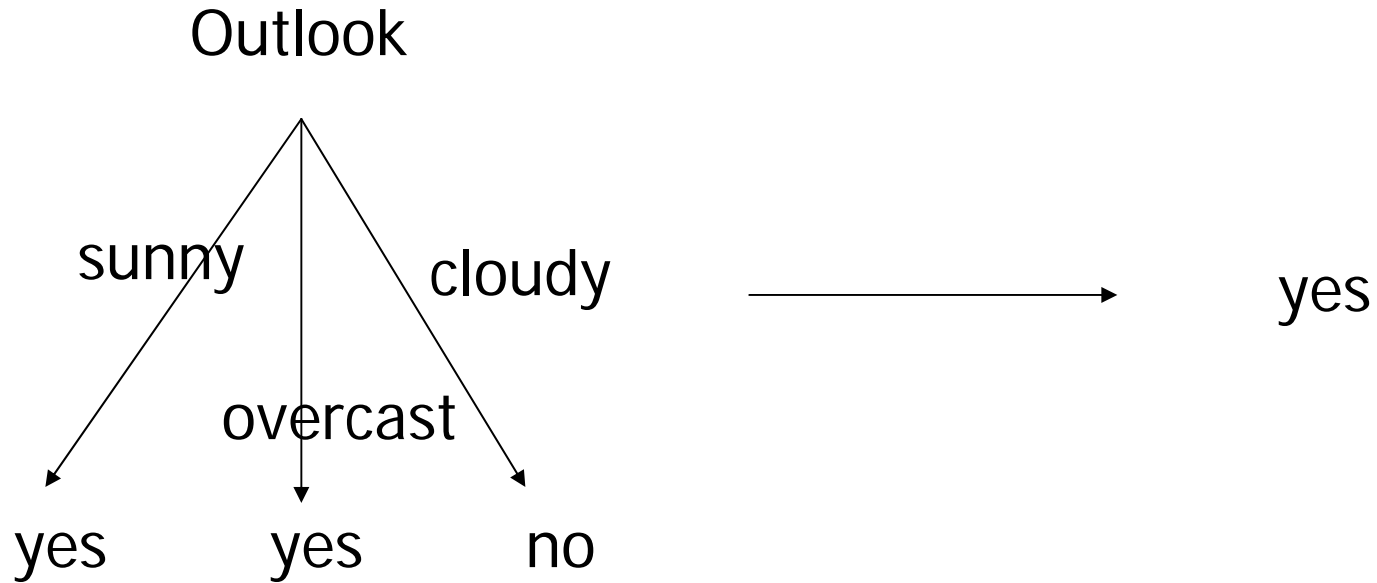
$$(6/16)*0.206+(9/16)*0.143+(1/16)*0.750=0.2046$$

- If the subtree is replaced with the leaf "yes", the estimated error is

$$U_{0.25}(16,1) = 0.157$$

- So the pruning is performed and the tree is merged

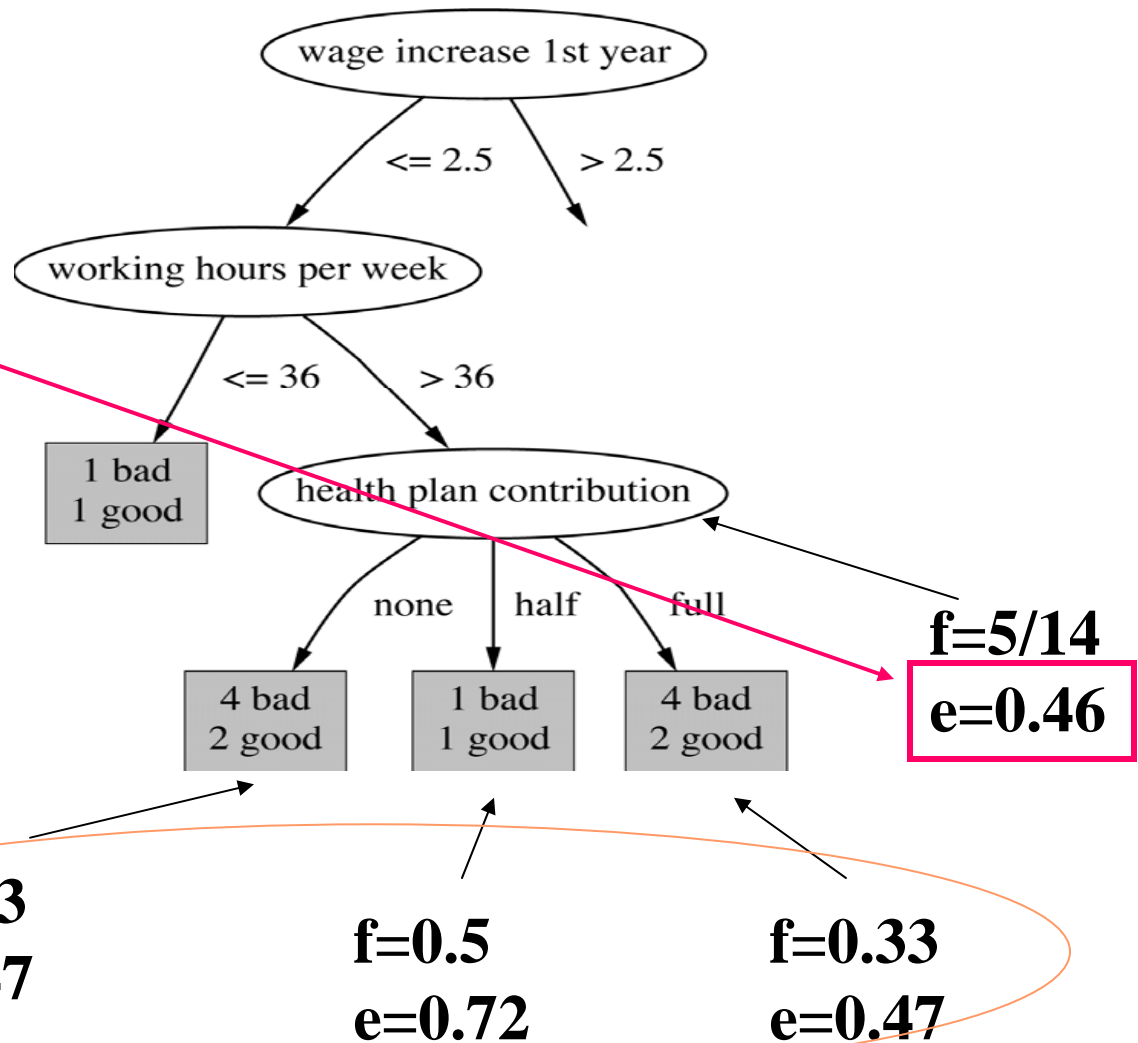
## Example (cont')



# Another example

So prune!

Combined using ratios 6:2:6  
this gives **0.51**



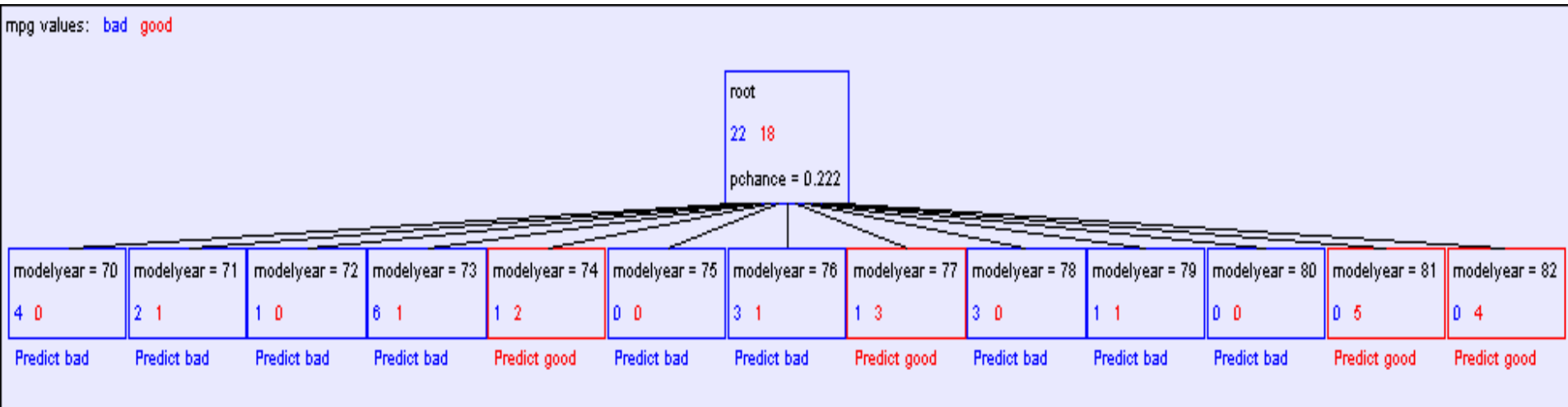
# Real-Valued inputs

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

mpg	cylinders	displacemen	horsepower	weight	acceleration	modeleyear	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	europa
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	europa
bad	5	131	103	2830	15.9	78	europa

Idea One: Branch on each possible real value

“One branch for each numeric value” idea:



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

Note pchance is 0.222 in the above...if MaxPchance was 0.05 that would end up pruning away to a single root node.

## A better idea: thresholded splits

- Suppose  $X$  is real valued.
- Define  $IG(Y/X:t)$  as  $H(Y) - H(Y/X:t)$
- Define  $H(Y/X:t) =$   
 $H(Y/X < t) P(X < t) + H(Y/X \geq t) P(X \geq t)$ 
  - $IG(Y/X:t)$  is the information gain for predicting  $Y$  if all you know is whether  $X$  is greater than or less than  $t$
- Then define  $IG^*(Y/X) = \max_t IG(Y/X:t)$
- For each real-valued attribute, use  $IG^*(Y/X)$  for assessing its suitability as a split

# Summary

- Contingency Tables
- Information gain (review of entropy)
- Learning an unpruned decision tree recursively
- Training and testing Error
- Overfitting issue
- Pruning a tree
- Building Decision Trees with real Valued Inputs

# Reference

*The lecture notes in this lecture are adopted and based on the following information:*

- T. M. Mitchell, Machine Learning, McGraw Hill, 1997. ISBN: 978-0-07-042807-2
- Learning Systems (course CD5720), Department of Computer Science and Electronics, Mälardalen University. [Online], available: <http://www.idt.mdh.se/kurser/cd5720/rjn/2006lp1/>
- Statistical Data Mining Tutorials, Dr. Andrew Moore, [Online], available: Andrew's tutorials: <http://www.cs.cmu.edu/~awm/tutorials>
- Dr. Qiang Yang, Decision tree, [online], available: <http://www.cs.ust.hk/~qyang/521/PPT/dtrees2.ppt#294,1,Classification with Decision Trees II>
- Ian H. Witten and Eibe Frank, "Data mining: practical machine learning tools and techniques", Morgan Kaufmann series. 2005
- C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006, ISBN: 978-0-387-31073-2.
- E. Alpaydin, Introduction to Machine Learning, MIT Press, 2004, ISBN 0-262-01211-1
- J. Hawkins, S. Blakeslee, "On Intelligence," Times Books, 2004;
- S. Haykin, "Neural Networks: A Comprehensive Foundation," Prentice Hall, 2nd edition, 1999, ISBN: 0-13-273350-1
- R. Pfeifer, C. Scheier, "Understanding Intelligence," The MIT Press, 2001.
- R. S. Sutton, A. G. Barto, "Reinforcement Learning: An Introduction," MIT Press, 1998.
- The AI lectures from Tokyo: <http://tokyolectures.org/>