



# Classification: Decision Trees

Data Mining and Text Mining (UIC 583 @ Politecnico di Milano)

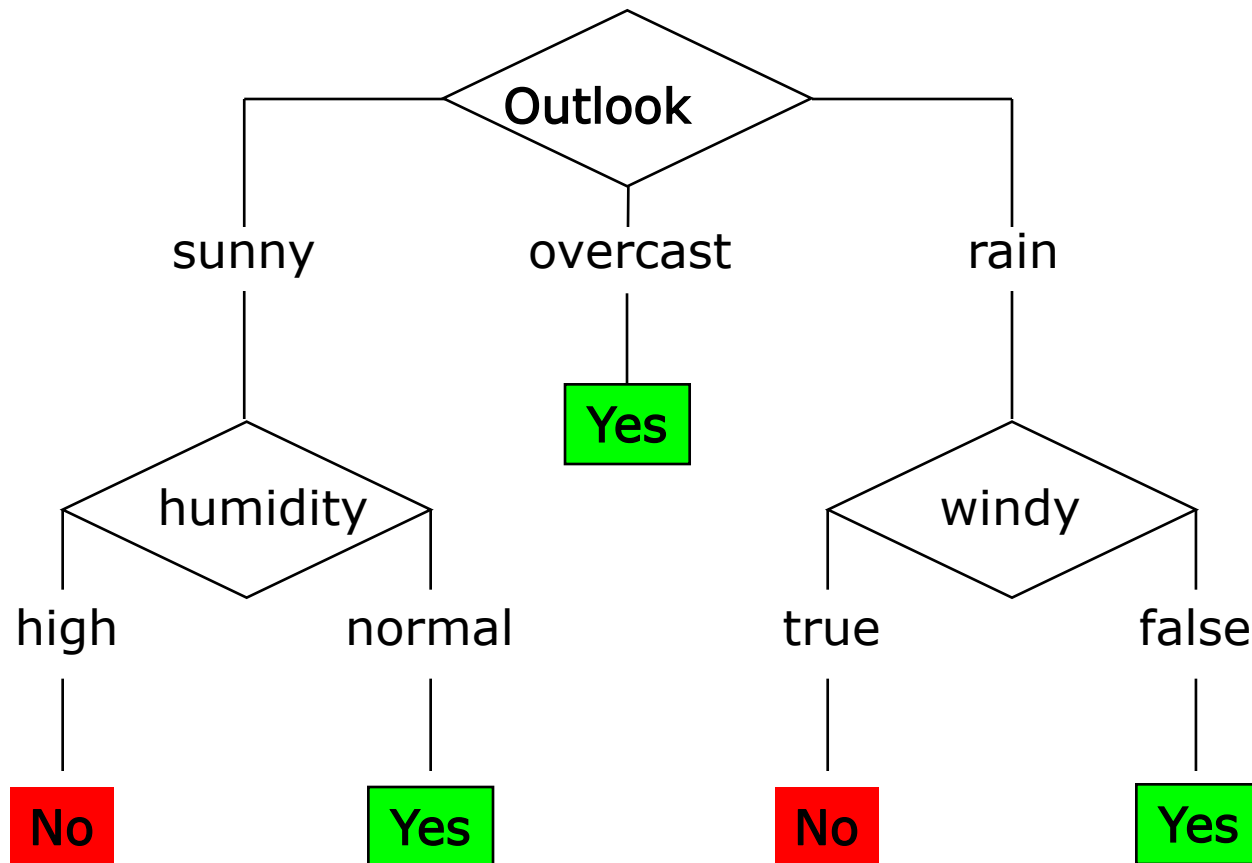
- ❑ What is a decision tree?
- ❑ Building decision trees
  - ▶ Choosing the Splitting Attribute
  - ▶ Choosing the Purity Measure
- ❑ The C4.5 Algorithm
- ❑ Pruning
- ❑ From Trees to Rules

# Decision trees

# Example: to play or not to play?

4

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



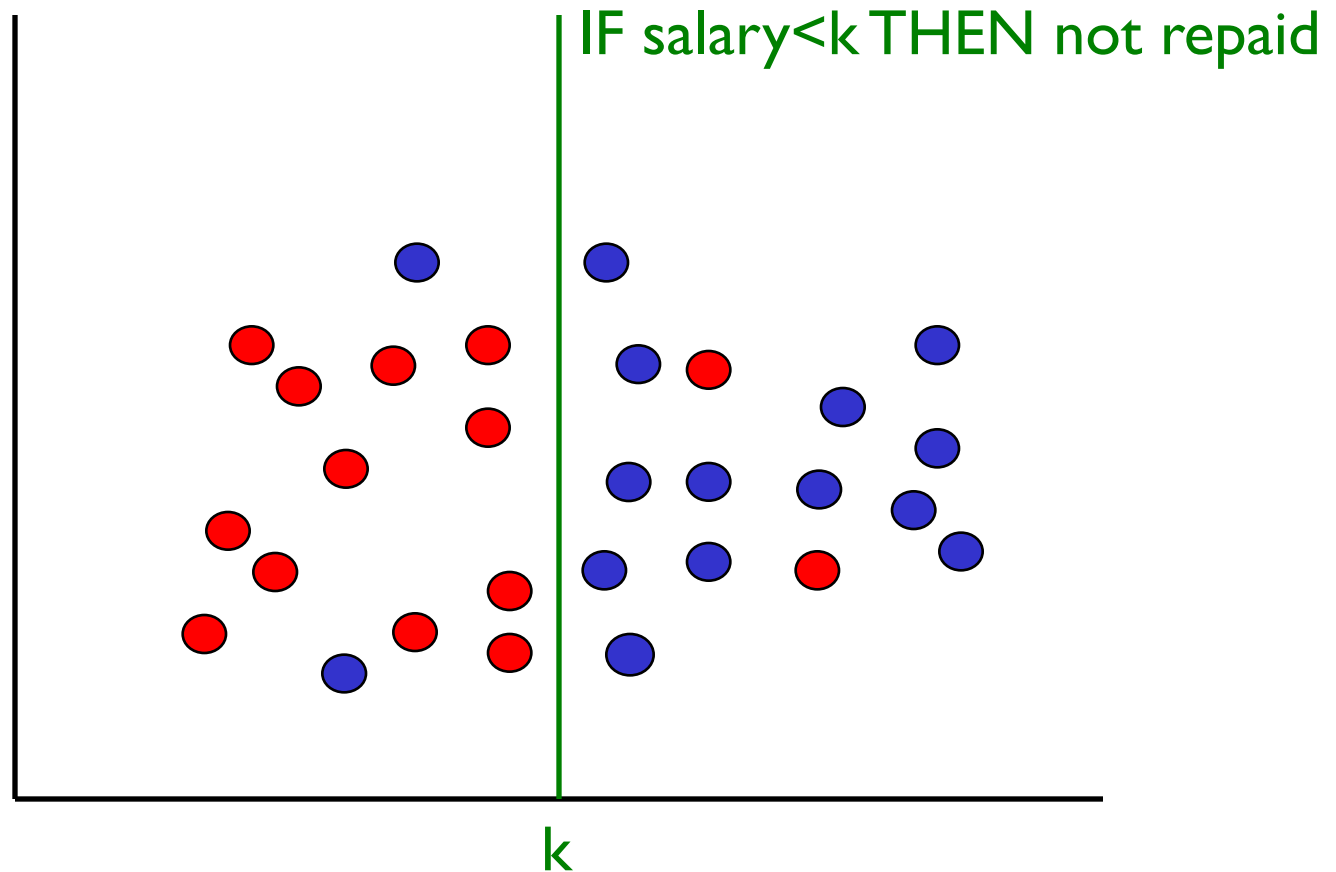
- ❑ An internal node is a test on an attribute
- ❑ A branch represents an outcome of the test, e.g.,  
outlook=windy
- ❑ A leaf node represents a class label or class label distribution
- ❑ At each node, one attribute is chosen to split training examples into distinct classes as much as possible
- ❑ A new case is classified by following a matching path to a leaf node

- ❑ Top-down tree construction
  - ▶ At start, all training examples are at the root
  - ▶ Partition the examples recursively by choosing one attribute each time
  
- ❑ Bottom-up tree pruning
  - ▶ Remove subtrees or branches, in a bottom-up manner, to improve the estimated accuracy on new cases.

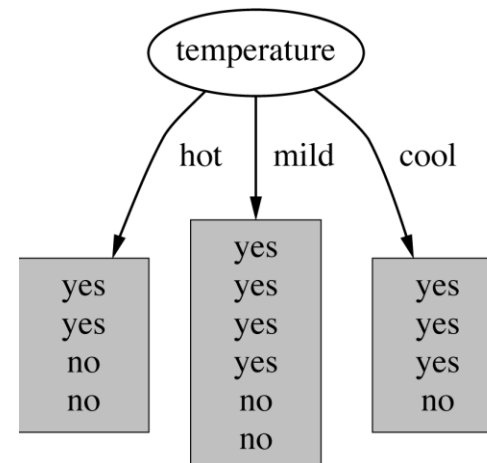
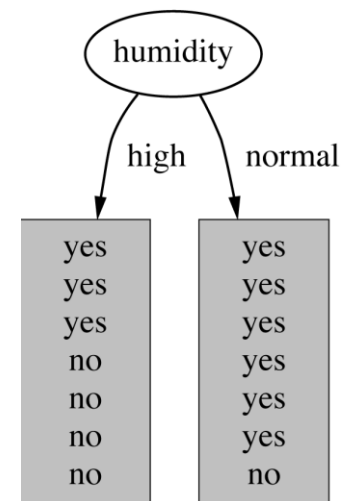
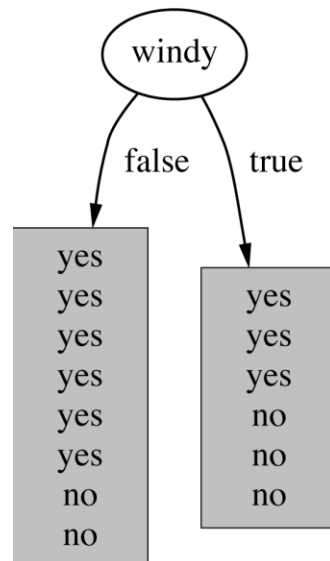
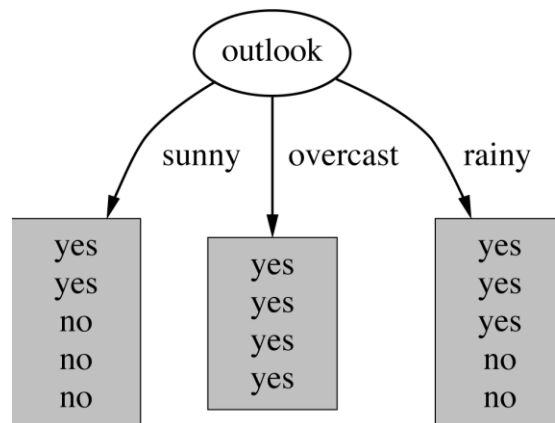
- ❑ Basic algorithm (a greedy algorithm)
  - ▶ Tree is constructed in a top-down recursive divide-and-conquer manner
  - ▶ At start, all the training examples are at the root
  - ▶ Attributes are categorical (if continuous-valued, they are discretized in advance)
  - ▶ Examples are partitioned recursively based on selected attributes
  - ▶ Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- ❑ Conditions for stopping partitioning
  - ▶ All samples for a given node belong to the same class
  - ▶ If there are no remaining attributes for further partitioning, majority voting is employed
  - ▶ There are no samples left



Splitting data

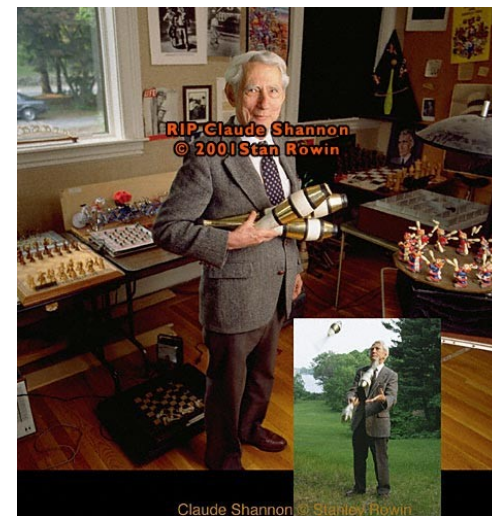


- ❑ At each node, available attributes are evaluated on the basis of separating the classes of the training examples
- ❑ A goodness (purity) function is used for this purpose.
- ❑ Typical goodness functions:
  - ▶ information gain (ID3/C4.5)
  - ▶ information gain ratio
  - ▶ gini index



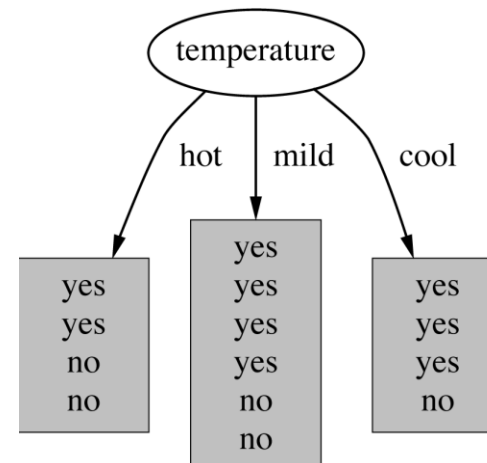
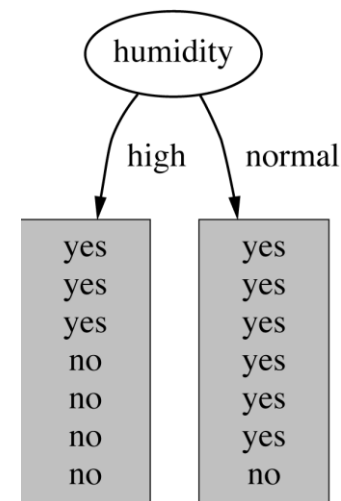
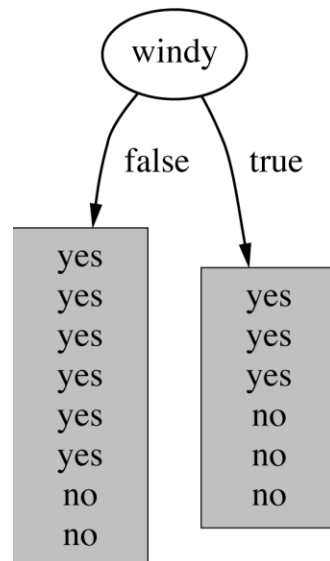
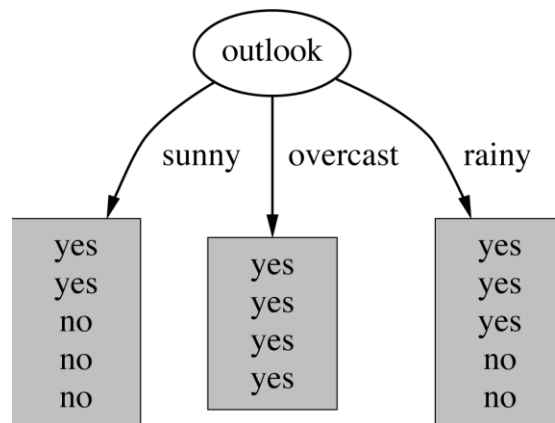
- ❑ Which is the best attribute?
  - ▶ The one which will result in the smallest tree
  - ▶ Heuristic: choose the attribute that produces the “purest” nodes
- ❑ Popular **impurity criterion**: information gain
  - ▶ Information gain increases with the average purity of the subsets that an attribute produces
- ❑ Strategy: choose attribute that results in greatest information gain

- Information is measured in **bits**
  - ▶ Given a probability distribution, the info required to predict an event is the distribution's *entropy*
  - ▶ Entropy gives the information required in bits (this can involve fractions of bits!)



- Formula for computing the entropy:

$$\text{entropy}(p_1, p_2, \dots, p_n) = -p_1 \log p_1 - p_2 \log p_2 \dots - p_n \log p_n$$



- “Outlook” = “Sunny”

$$\text{info}([2, 3]) = \text{entropy}(2/5, 3/5) = 0.971$$

- “Outlook” = “Overcast”

$$\text{info}([4, 0]) = \text{entropy}(1, 0) = 0.000$$

- “Outlook” = “Rainy”

$$\text{info}([3, 2]) = \text{entropy}(3/5, 2/5) = 0.971$$

- Expected information for attribute

$$\begin{aligned} \text{info}([2, 3][4, 0][3, 2]) &= 5/14 \times 0.971 + \\ &\quad 4/14 \times 0 + 5/14 \times 0.971 \end{aligned}$$



- Difference between the information before split and the information after split

$$gain(A) = info(D) - info_A(D)$$

- The information before the split,  $info(D)$ , is the entropy,

$$info(D) = -p_1 \log p_1 - \dots - p_n \log p_n$$

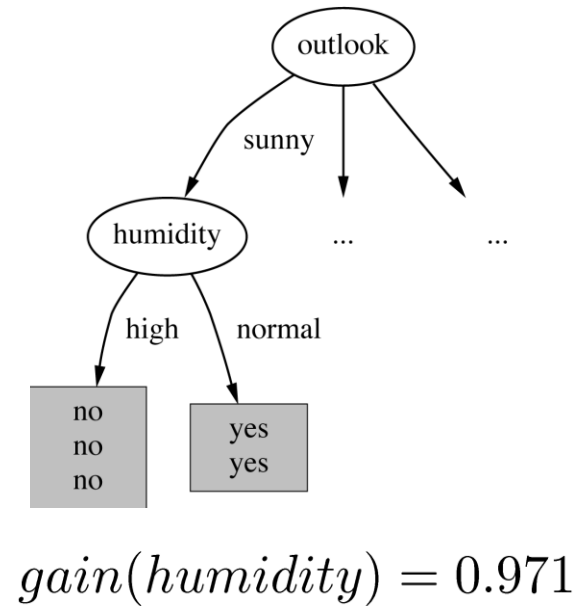
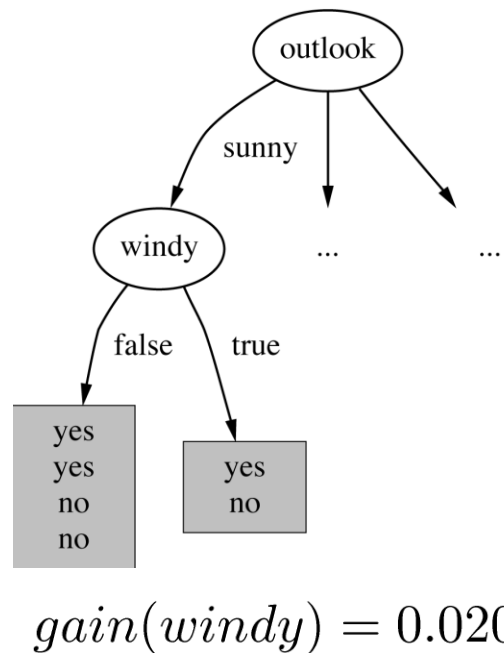
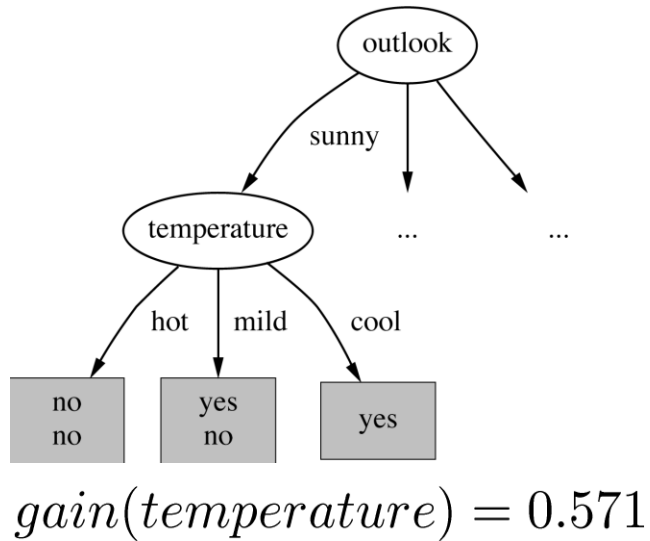
- The information after the split using attribute A is computed as the weighted sum of the entropies on each split, given n splits,

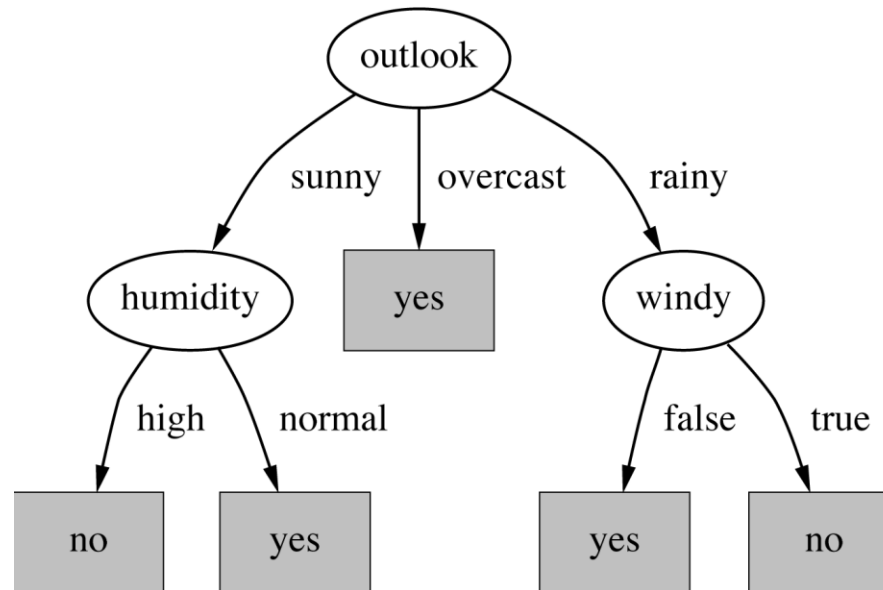
$$info_A(D) = \frac{|D_1|}{|D|} info(D_1) + \dots + \frac{|D_n|}{|D|} info(D_n)$$

- Difference between the information before split and the information after split

$$\begin{aligned} \text{gain}(\text{outlook}) &= \text{info}([9, 5]) - \text{info}([2, 3][4, 0][3, 2]) \\ &= 0.940 - 0.693 \\ &= 0.247 \end{aligned}$$

- Information gain for the attributes from the weather data:
  - ▶  $\text{gain}(\text{"outlook"}) = 0.247$  bits
  - ▶  $\text{gain}(\text{"temperature"}) = 0.029$  bits
  - ▶  $\text{gain}(\text{"humidity"}) = 0.152$  bits
  - ▶  $\text{gain}(\text{"windy"}) = 0.048$  bits





- ❑ Not all leaves need to be pure
- ❑ Splitting stops when data can't be split any further

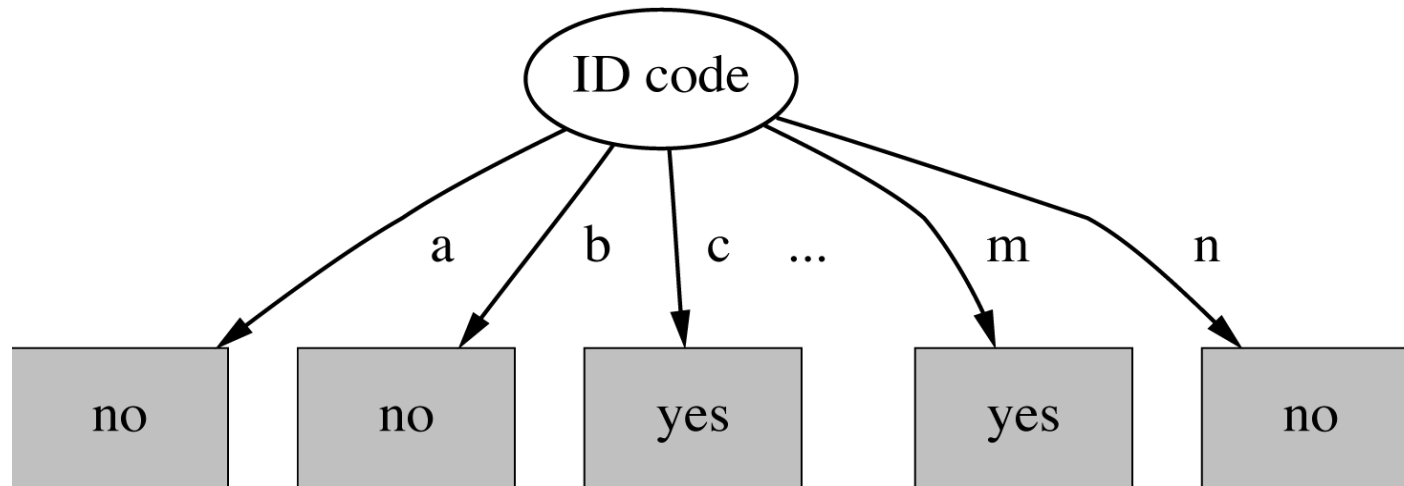
- ❑ Properties we require from a purity measure:
  - ▶ When node is pure, measure should be zero
  - ▶ When impurity is maximal (i.e. all classes equally likely), measure should be maximal
  - ▶ Measure should obey multistage property (i.e. decisions can be made in several stages):

$$\text{measure}([2, 3, 4]) = \text{measure}[2, 7] + 7/9 \times \text{measure}([3, 4])$$

- ❑ Entropy is a function that satisfies all three properties!

- ❑ Problematic: attributes with a large number of values (extreme case: ID code)
- ❑ Subsets are more likely to be pure if there is a large number of values
  - ▶ Information gain is biased towards choosing attributes with a large number of values
  - ▶ This may result in overfitting (selection of an attribute that is non-optimal for prediction)

ID code	Outlook	Temperature	Humidity	Windy	Play
A	Sunny	Hot	High	False	No
B	Sunny	Hot	High	True	No
C	Overcast	Hot	High	False	Yes
D	Rainy	Mild	High	False	Yes
E	Rainy	Cool	Normal	False	Yes
F	Rainy	Cool	Normal	True	No
G	Overcast	Cool	Normal	True	Yes
H	Sunny	Mild	High	False	No
I	Sunny	Cool	Normal	False	Yes
J	Rainy	Mild	Normal	False	Yes
K	Sunny	Mild	Normal	True	Yes
L	Overcast	Mild	High	True	Yes
M	Overcast	Hot	Normal	False	Yes
N	Rainy	Mild	High	True	No



- ❑ Entropy of split = 0 (since each leaf node is “pure”, having only one case.
- ❑ Information gain is maximal for ID code



- ❑ Gain ratio: a modification of the information gain that reduces its bias on high-branch attributes
- ❑ Gain ratio should be
  - ▶ Large when data is evenly spread
  - ▶ Small when all data belong to one branch
- ❑ Gain ratio takes number and size of branches into account when choosing an attribute
- ❑ It corrects the information gain by taking the intrinsic information of a split into account (i.e. how much info do we need to tell which branch an instance belongs to)

## □ Intrinsic information

$$\text{IntrinsicInfo}(S, A) = - \sum \frac{|S_i|}{|S|} \log \frac{|S_i|}{|S|}$$

entropy of distribution of instances into branches

## □ Gain ratio normalizes info gain by

$$\text{GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{\text{IntrinsicInfo}(S, A)}$$

- Example: intrinsic information for ID code

$$\text{info}([1, 1, \dots, 1]) = 14 \times (-1/14 \times \log 1/14) = 3.807$$

- Importance of attribute decreases as intrinsic information gets larger
- The gain ratio of “ID code”,

$$\text{GainRatio}(\text{ID\_code}) = \frac{0.940}{3.807} = 0.246$$

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: $0.940 - 0.693$	0.247	Gain: $0.940 - 0.911$	0.029
Split info: $\text{info}([5,4,5])$	1.577	Split info: $\text{info}([4,6,4])$	1.362
Gain ratio: $0.247 / 1.577$	0.156	Gain ratio: $0.029 / 1.362$	0.021

Humidity		Windy	
Info:	0.788	Info:	0.892
Gain: $0.940 - 0.788$	0.152	Gain: $0.940 - 0.892$	0.048
Split info: $\text{info}([7,7])$	1.000	Split info: $\text{info}([8,6])$	0.985
Gain ratio: $0.152 / 1$	0.152	Gain ratio: $0.048 / 0.985$	0.049

- ❑ “Outlook” still comes out top
- ❑ However “ID code” has greater gain ratio
- ❑ The standard fix is an ad hoc test to prevent splitting on that type of attribute
  
- ❑ Problem: gain ratio may overcompensate
  - ▶ May choose an attribute just because its intrinsic information is very low
  - ▶ Standard fix:
    - First, only consider attributes with greater than average information gain
    - Then, compare them on gain ratio

Example

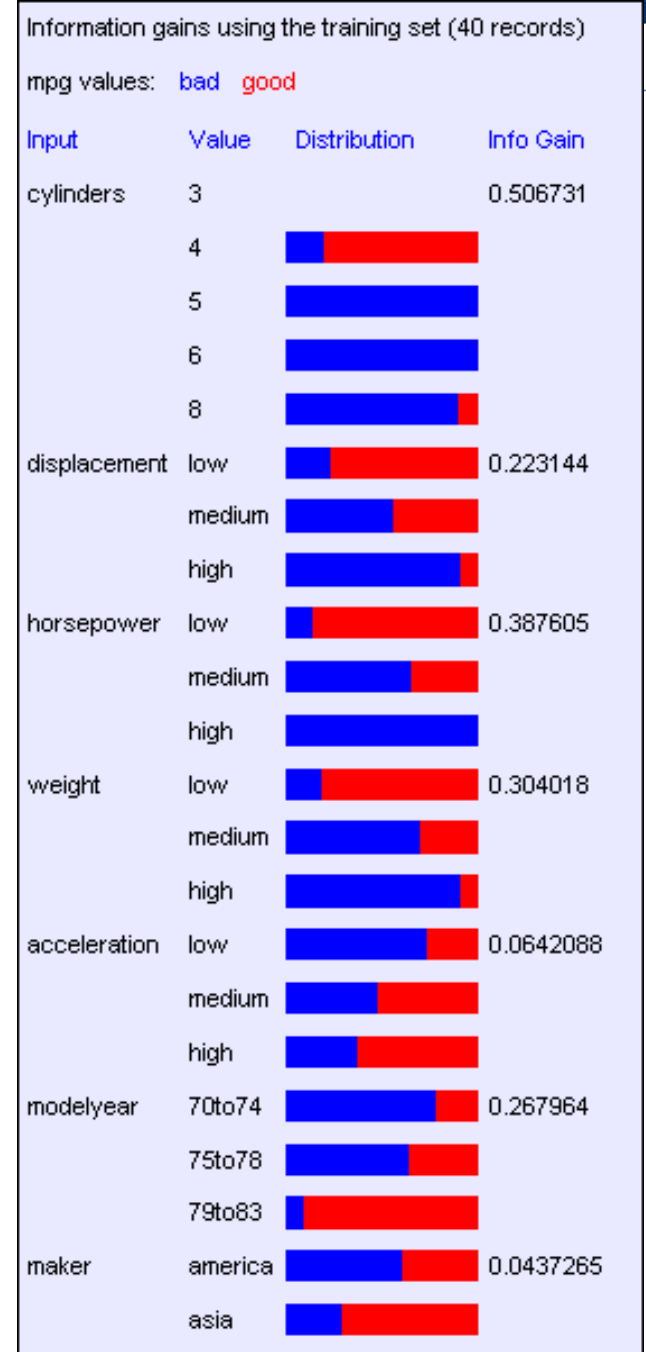
# A small dataset: Miles Per Gallon (40 records)

31

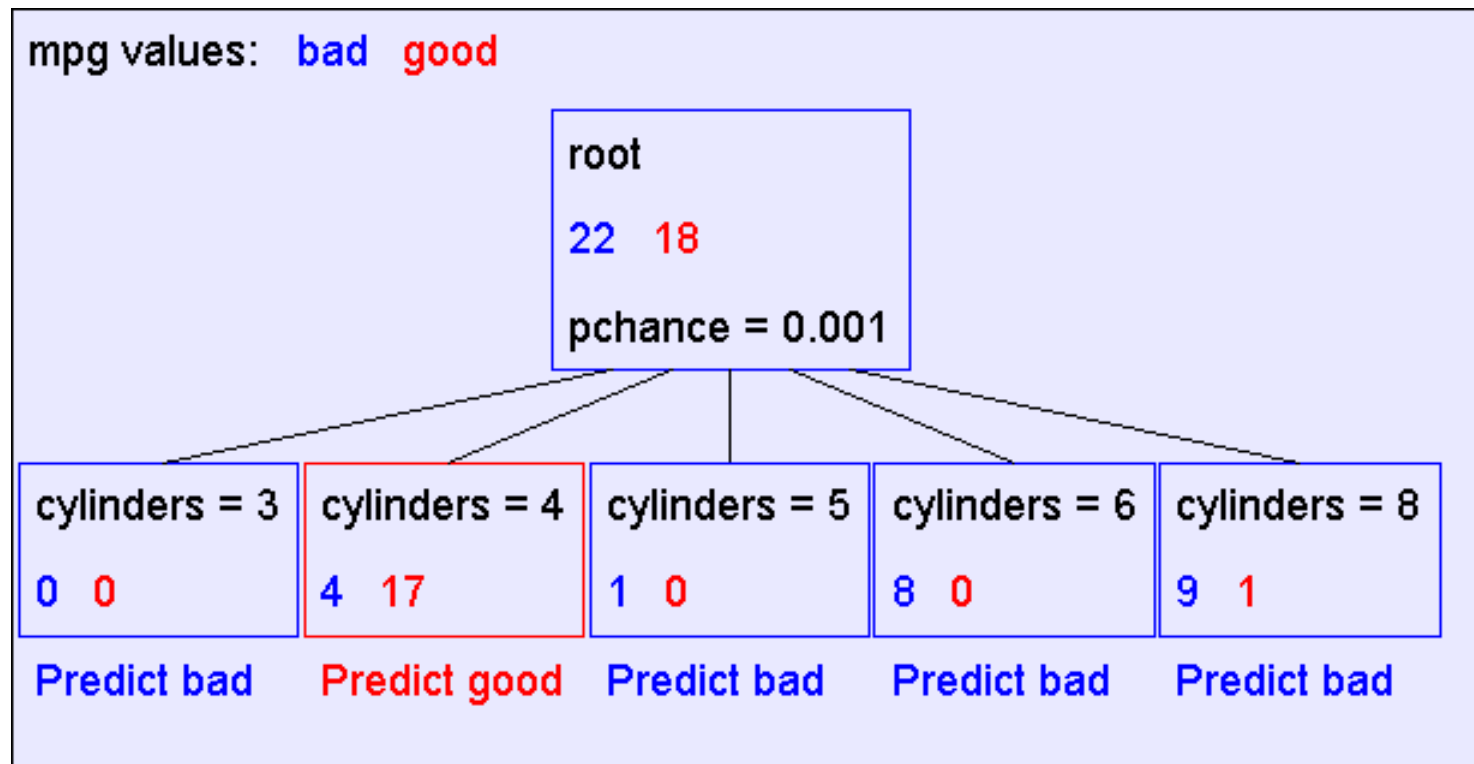
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europa
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low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

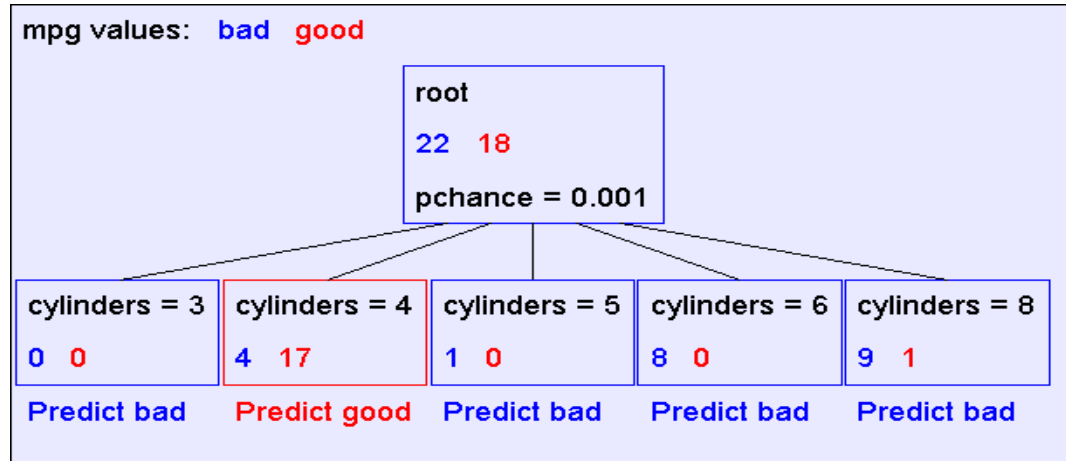
From the UCI repository (from Ross Quinlan)

Look at all the  
information gains...  
Suppose we want to  
predict MPG.

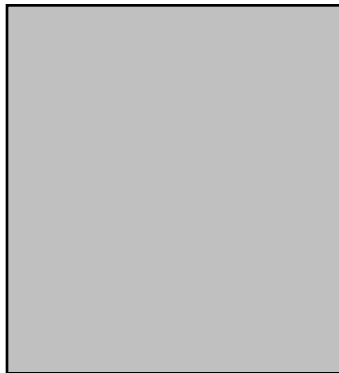




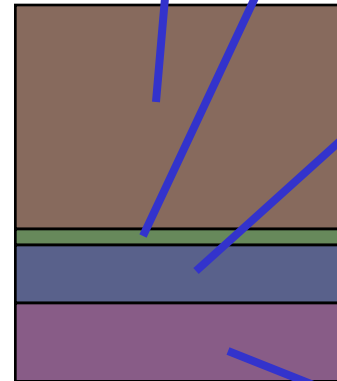




Take the  
Original  
Dataset..



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

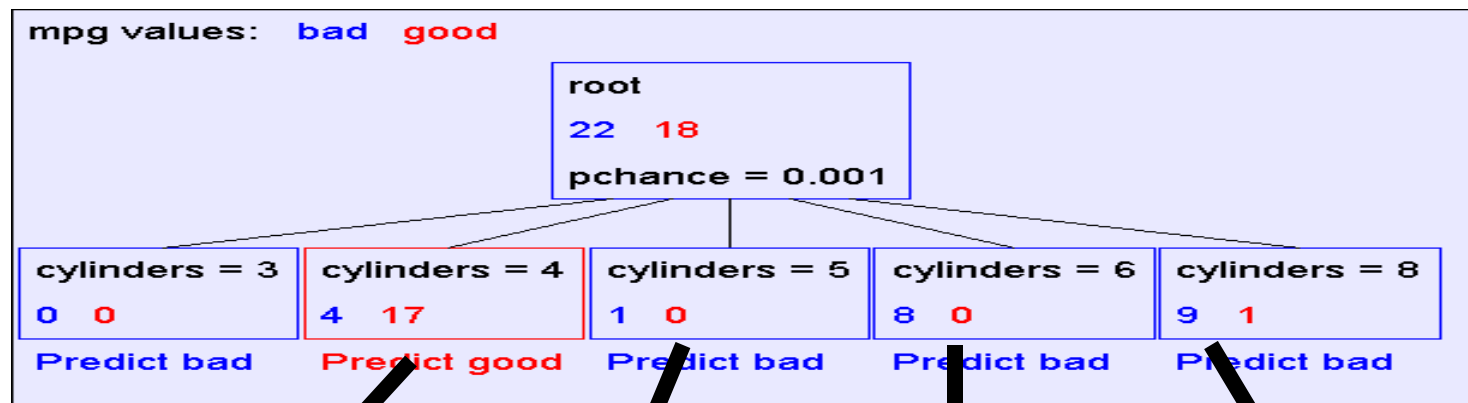


Records  
in which  
cylinders  
= 4

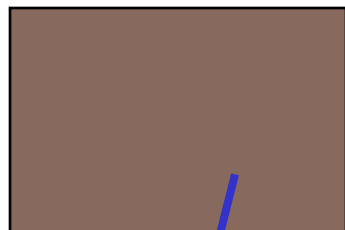
Records  
in which  
cylinders  
= 5

Records  
in which  
cylinders  
= 6

Records  
in which  
cylinders  
= 8



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..

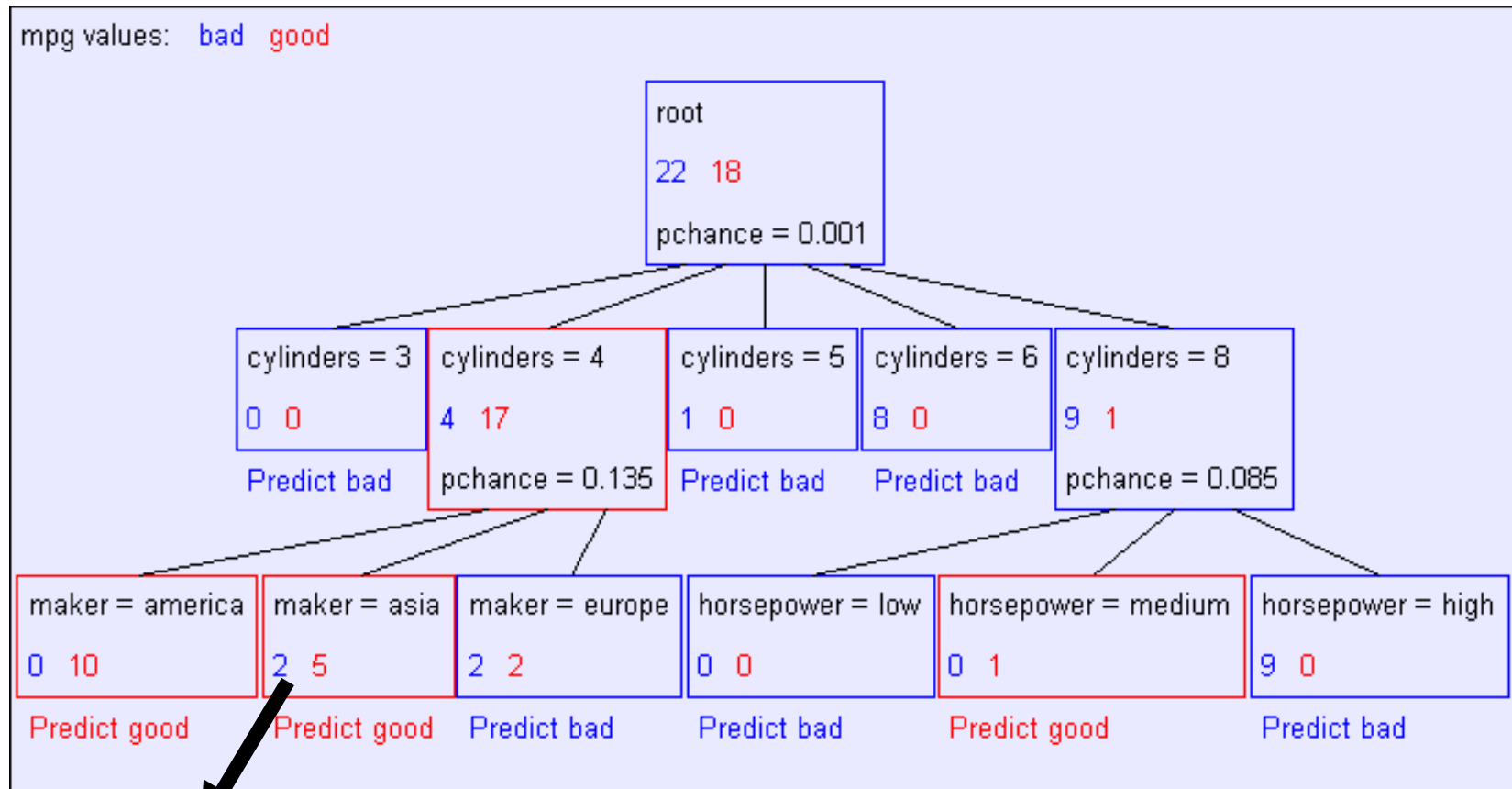


Records in  
which cylinders  
= 6

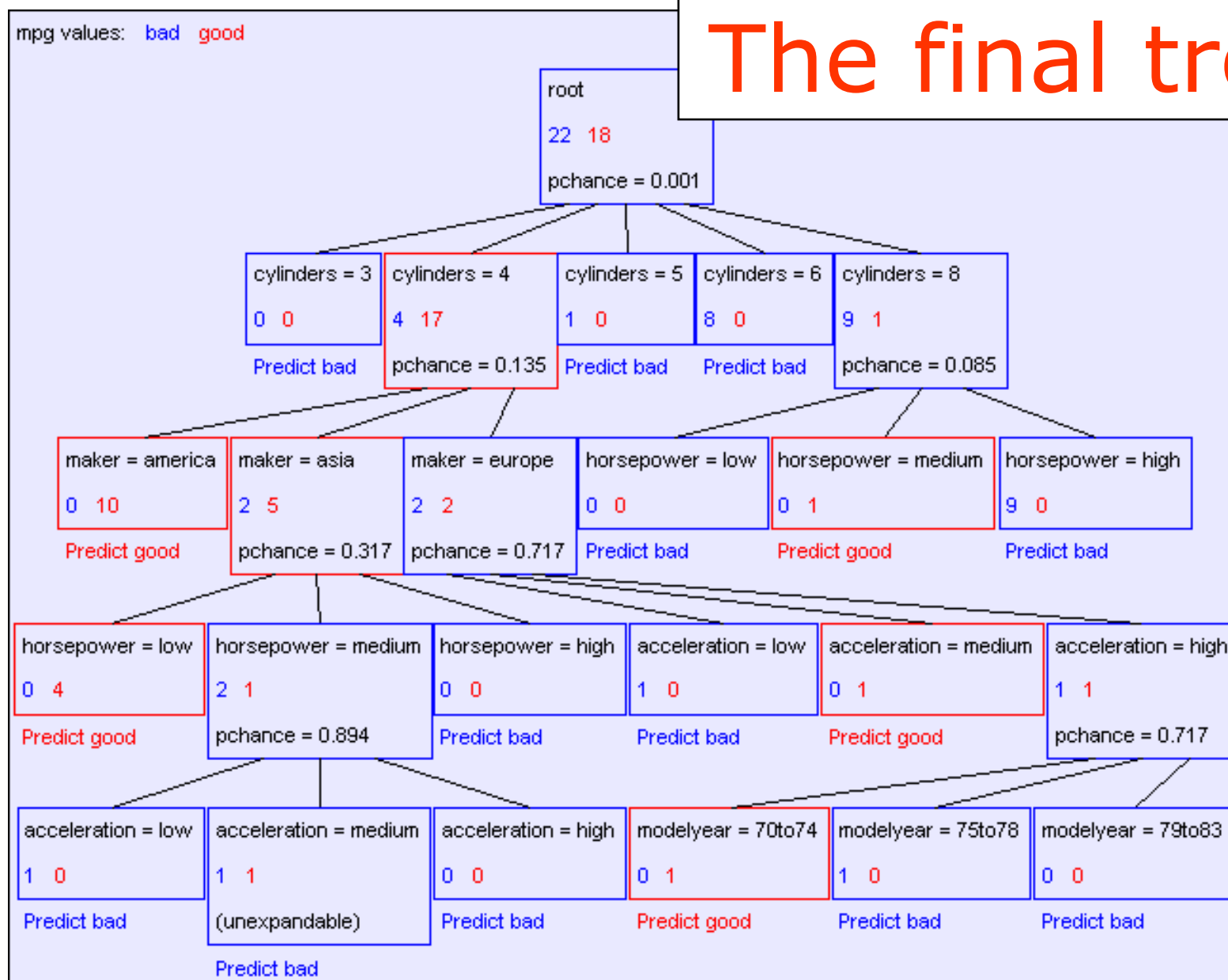
Build tree from  
These records..



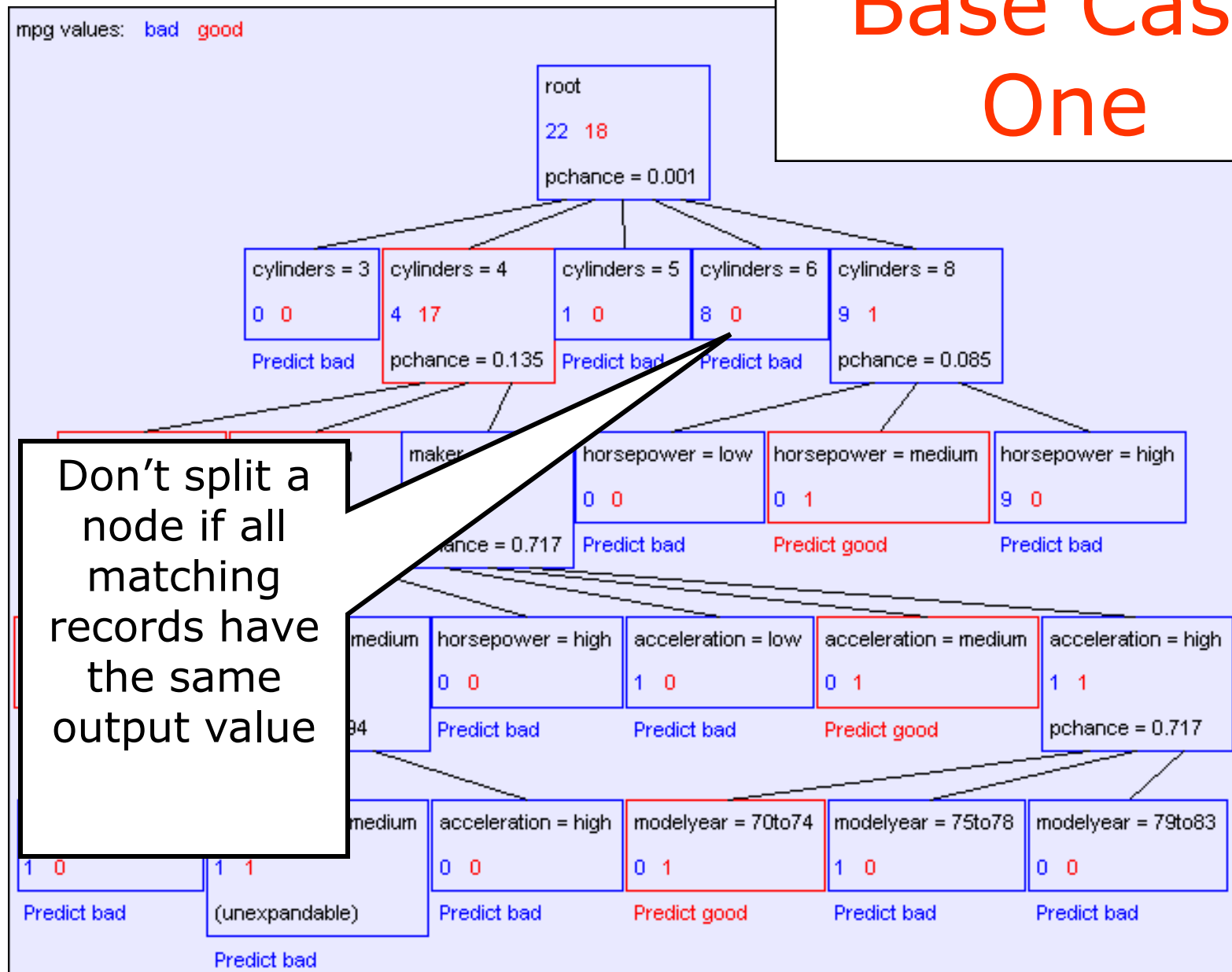
Records in  
which cylinders  
= 8



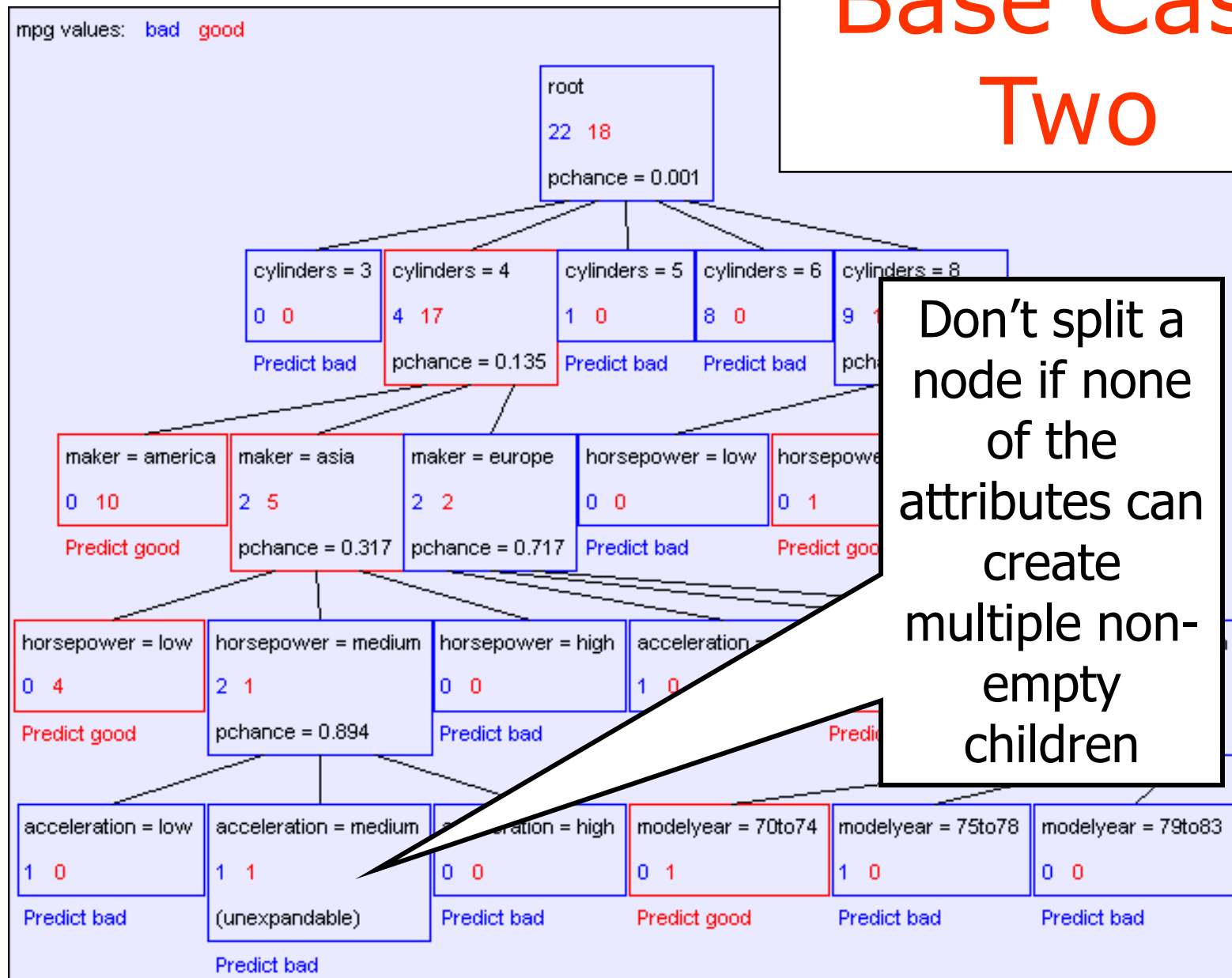
# The final tree



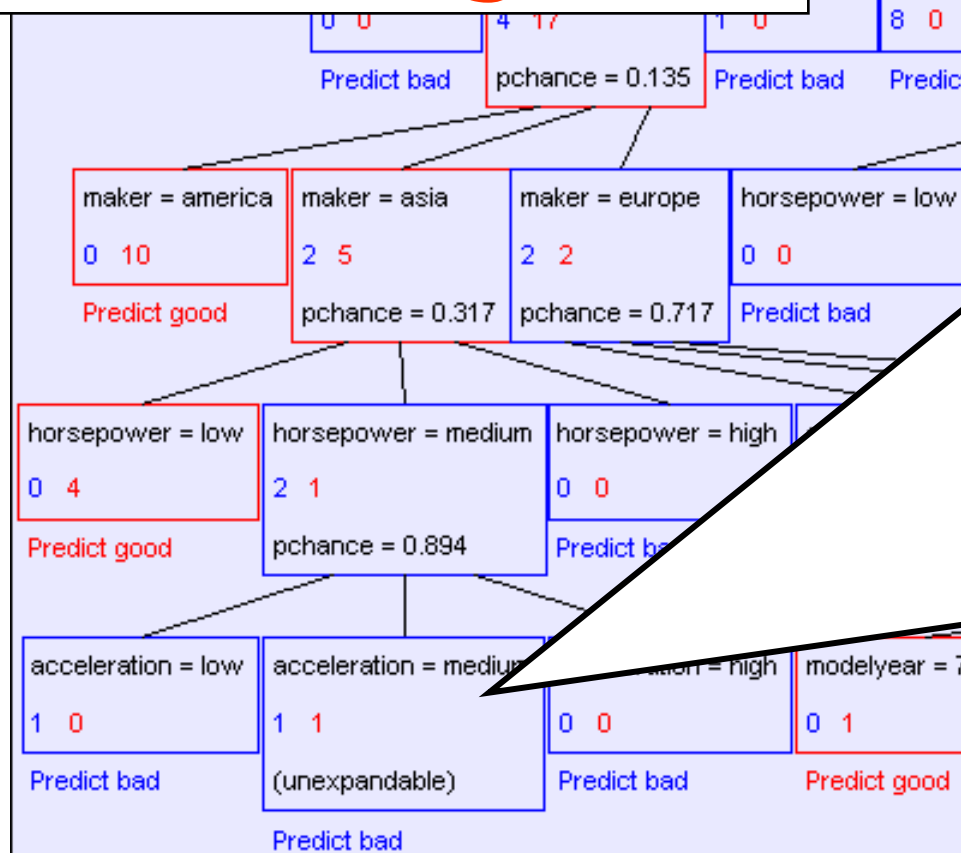
# Base Case One



# Base Case Two



# Base Case Two: No attributes can distinguish



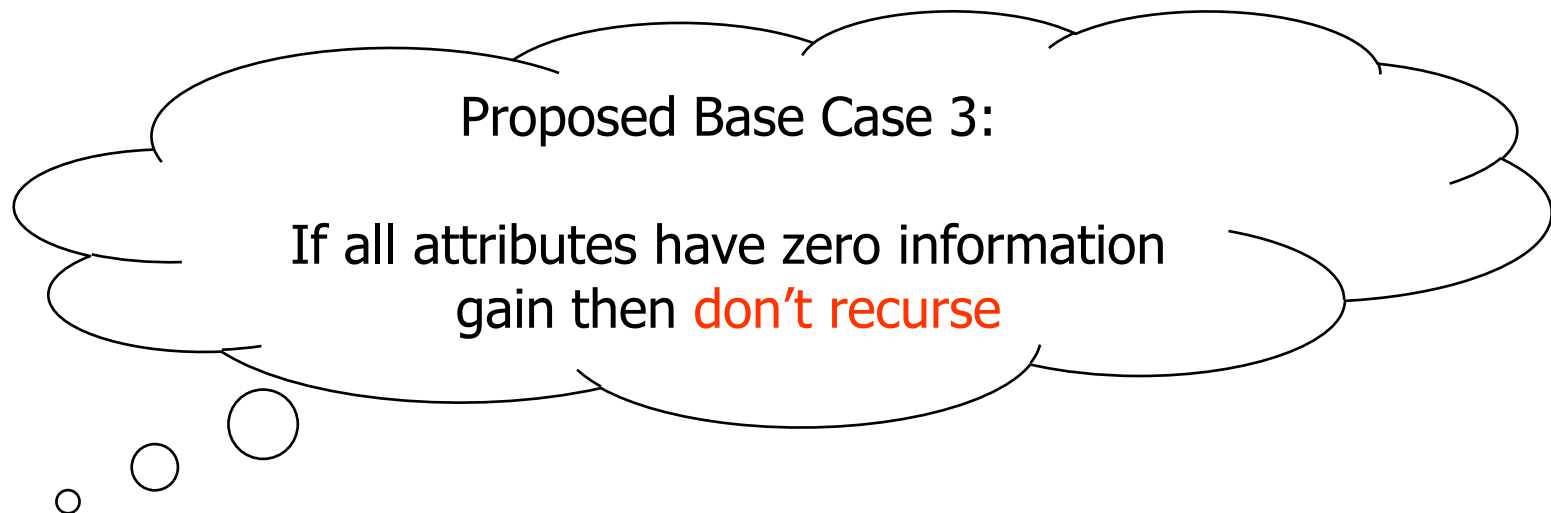
Information gains using the training set (2 records)

mpg values: bad good

Input	Value	Distribution	Info Gain
cylinders	3		0
	4		
	5		
	6		
	8		
displacement	low		0
	medium		
	high		
horsepower	low		0
	medium		
	high		
weight	low		0
	medium		
	high		
acceleration	low		0
	medium		
	high		
modelyear	70to74		0
	75to78		
	79to83		
maker	america		0
	asia		
	europe		







- ❑ 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



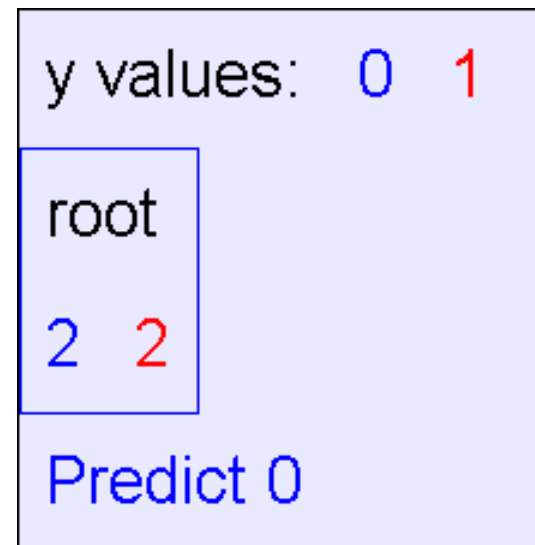
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

$$y = a \text{ XOR } b$$

The information gains:

Information gains using the training set (4 records)				
y values: 0 1				
Input	Value	Distribution	Info Gain	
a	0		0	
	1			
b	0		0	
	1			

The resulting decision tree:



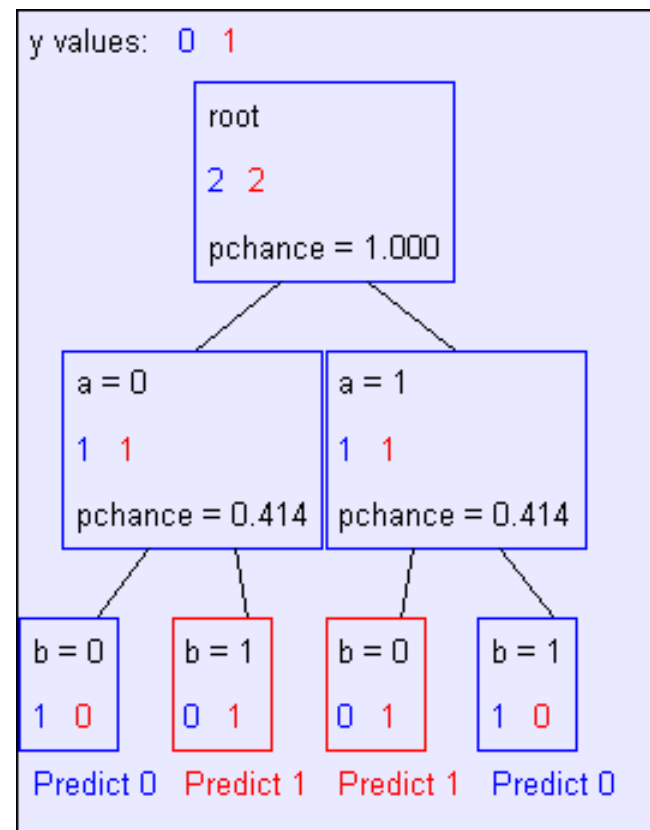
## If we omit Base Case 3:

44

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

$$y = a \text{ XOR } b$$

The resulting decision tree:



Gini index

age	income	student	credit_rating	buys_computer
$\leq 30$	high	no	fair	no
$\leq 30$	high	no	excellent	no
31...40	high	no	fair	yes
$> 40$	medium	no	fair	yes
$> 40$	low	yes	fair	yes
$> 40$	low	yes	excellent	no
31...40	low	yes	excellent	yes
$\leq 30$	medium	no	fair	no
$\leq 30$	low	yes	fair	yes
$> 40$	medium	yes	fair	yes
$\leq 30$	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
$> 40$	medium	no	excellent	no

- If a data set  $T$  contains examples from  $n$  classes, gini index,  $\text{gini}(T)$  is defined as

$$\text{gini}(T) = 1 - \sum_{j=1}^n p_j^2$$

- where  $p_j$  is the relative frequency of class  $j$  in  $T$ .
- $\text{gini}(T)$  is minimized if the classes in  $T$  are skewed.

- If a data set  $D$  contains examples from  $n$  classes, gini index,  $\text{gini}(D)$  is defined as

$$\text{gini}(D) = 1 - \sum_{j=1}^n p_j^2$$

where  $p_j$  is the relative frequency of class  $j$  in  $D$

- If a data set  $D$  is split on  $A$  into two subsets  $D_1$  and  $D_2$ , the gini index  $\text{gini}(D)$  is defined as

$$\text{gini}_A(D) = \frac{|D_1|}{|D|} \text{gini}(D_1) + \frac{|D_2|}{|D|} \text{gini}(D_2)$$

- Reduction in Impurity,

$$\Delta \text{gini}(A) = \text{gini}(D) - \text{gini}_A(D)$$

- The attribute provides the smallest  $\text{gini}_{\text{split}}(D)$  (or the largest reduction in impurity) is chosen to split the node (need to enumerate all the possible splitting points for each attribute)

- D has 9 tuples labeled “yes” and 5 labeled “no”

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

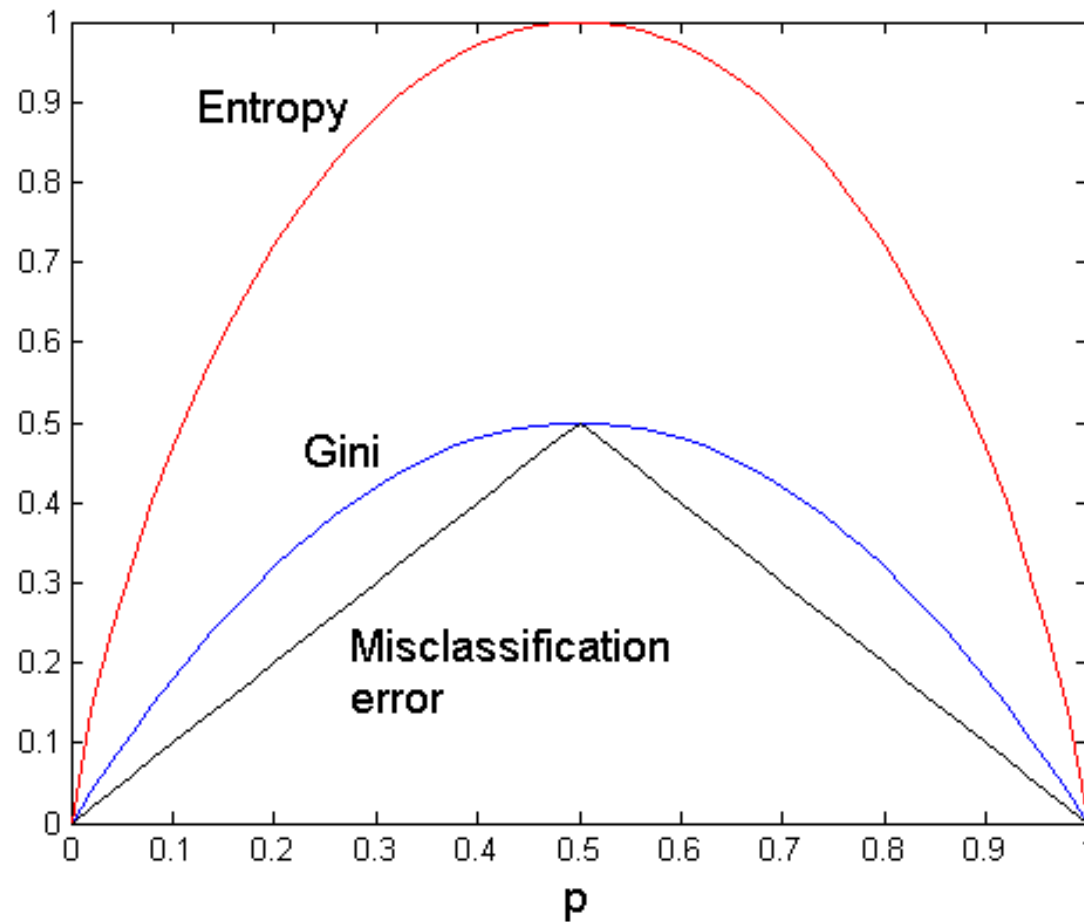
- Suppose the attribute income partitions D into 10 in  $D_1$  branching on low and medium and 4 in  $D_2$

$$\begin{aligned} gini(D)_{\{l,m\}} &= \left(\frac{10}{14}\right) gini(D_1) + \left(\frac{4}{14}\right) gini(D_2) \\ &= \left(\frac{10}{14}\right) \left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \\ &\quad + \left(\frac{4}{14}\right) \left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right) = 0.450 \end{aligned}$$

but  $gini\{\text{medium,high}\}$  is 0.30 and thus the best since it is the lowest



**For a 2-class problem:**



Attribute measures

- ❑ Three measures which generally return good results (but)
- ❑ Information gain:
  - ▶ biased towards multivalued attributes
- ❑ Gain ratio:
  - ▶ tends to prefer unbalanced splits in which one partition is much smaller than the others
- ❑ Gini index:
  - ▶ biased to multivalued attributes
  - ▶ has difficulty when # of classes is large
  - ▶ tends to favor tests that result in equal-sized partitions and purity in both partitions

- ❑ **CHAID**: a popular decision tree algorithm, measure based on  $\chi^2$  test for independence
- ❑ **C-SEP**: performs better than information gain and gini index in certain cases
- ❑ **G-statistics**: has a close approximation to  $\chi^2$  distribution
- ❑ **MDL** (Minimal Description Length) principle, i.e., the simplest solution is preferred
  - ▶ The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- ❑ Multivariate splits (partition based on multiple variable combinations)
  - ▶ **CART**: finds multivariate splits based on a linear combination of attributes
- ❑ Which attribute selection measure is the best?
  - ▶ Most give good results, none is significantly superior than others

Numerical attributes

- ❑ For an algorithm to be useful in a wide range of real-world applications it must:
  - ▶ Permit numeric attributes
  - ▶ Allow missing values
  - ▶ Be robust in the presence of noise
  - ▶ Be able to approximate arbitrary concept descriptions (at least in principle)
- ❑ Basic schemes need to be extended to fulfill these requirements

- ❑ Handling Numeric Attributes
  - ▶ Finding Best Split(s)
- ❑ Dealing with Missing Values
- ❑ Pruning
  - ▶ Pre-pruning, Post-pruning, Error Estimates
- ❑ From Trees to Rules

- ❑ ID3, CHAID – 1960s
- ❑ C4.5 innovations (Quinlan):
  - ▶ permit numeric attributes
  - ▶ deal sensibly with missing values
  - ▶ pruning to deal with for noisy data
- ❑ C4.5 - one of best-known and most widely-used learning algorithms
  - ▶ Last research version: C4.8, implemented in Weka as J4.8 (Java)
  - ▶ Commercial successor: C5.0 (available from Rulequest)



- ❑ The standard method involves binary splits, e.g.,  $\text{temp} < 45$
- ❑ Unlike nominal attributes, every attribute has many possible split points
- ❑ Solution is straightforward extension:
  - ▶ Evaluate info gain (or other measure) for every possible split point of attribute
  - ▶ Choose “best” split point
  - ▶ Info gain for best split point is info gain for attribute
- ❑ Computationally more demanding

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	Normal	False	Yes
...	...	...	...	...

```
If outlook = sunny and humidity = high then play = no
If outlook = rainy and windy = true then play = no
If outlook = overcast then play = yes
If humidity = normal then play = yes
If none of the above then play = yes
```

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
...	...	...	...	...

If outlook = sunny and humidity > 83 then play = no

If outlook = rainy and windy = true then play = no

If outlook = overcast then play = yes

If humidity < 85 then play = yes

If none of the above then play = yes

- ❑ To split on temperature attribute,
  - ▶ check all the cut points
  - ▶ Choose the one with the best information gain

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Y	N	Y	Y	Y	N	N	Y	Y	Y	N	Y	Y	No

- ❑ E.g. temperature < 71.5: yes/4, no/2  
temperature ≥ 71.5: yes/5, no/3
- ❑  $\text{Info}([4,2],[5,3]) = 6/14 \text{ info}([4,2]) + 8/14 \text{ info}([5,3])$   
= 0.939 bits
- ❑ Place split points halfway between values
- ❑ Can evaluate all split points in one pass!

- ❑ Splitting (multi-way) on a nominal attribute exhausts all information in that attribute
  - ▶ Nominal attribute is tested (at most) once on any path in the tree
- ❑ Not so for binary splits on numeric attributes!
  - ▶ Numeric attribute may be tested several times along a path in the tree
- ❑ Disadvantage: tree is hard to read
- ❑ Remedy: pre-discretize numeric attributes, or use multi-way splits instead of binary ones

- ❑ Missing value denoted “?” in C4.X
- ❑ Simple idea: treat missing as a separate value
- ❑ Question: When this is not appropriate?
- ❑ Answer: When values are missing due to different reasons
  - ▶ Example 1: gene expression could be missing when it is very high or very low
  - ▶ Example 2: field IsPregnant=missing for a male patient should be treated differently (no) than for a female patient of age 25 (unknown)

- ❑ Split instances with missing values into pieces
  - ▶ A piece going down a branch receives a weight proportional to the popularity of the branch
  - ▶ weights sum to 1
- ❑ Information gain works with fractional instances
  - ▶ use sums of weights instead of counts
- ❑ During classification, split the instance into pieces in the same way
  - ▶ Merge probability distribution using weights

# Pruning



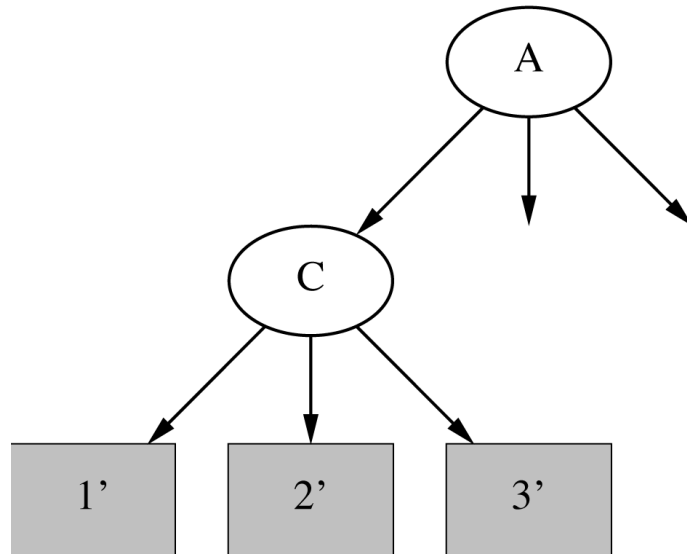
- ❑ The generated tree may overfit the training data
  - ▶ Too many branches, some may reflect anomalies due to noise or outliers
  - ▶ Result is in poor accuracy for unseen samples
- ❑ Two approaches to avoid overfitting: prepruning and postpruning
- ❑ **Prepruning**
  - ▶ Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
  - ▶ Difficult to choose an appropriate threshold
- ❑ **Postpruning**
  - ▶ Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
  - ▶ Use a set of data different from the training data to decide which is the “best pruned tree”

- ❑ Based on statistical significance test
- ❑ Stop growing the tree when there is no statistically significant association between any attribute and the class at a particular node
- ❑ Most popular test: chi-squared test
- ❑ ID3 used chi-squared test in addition to information gain
  - ▶ Only statistically significant attributes were allowed to be selected by information gain procedure

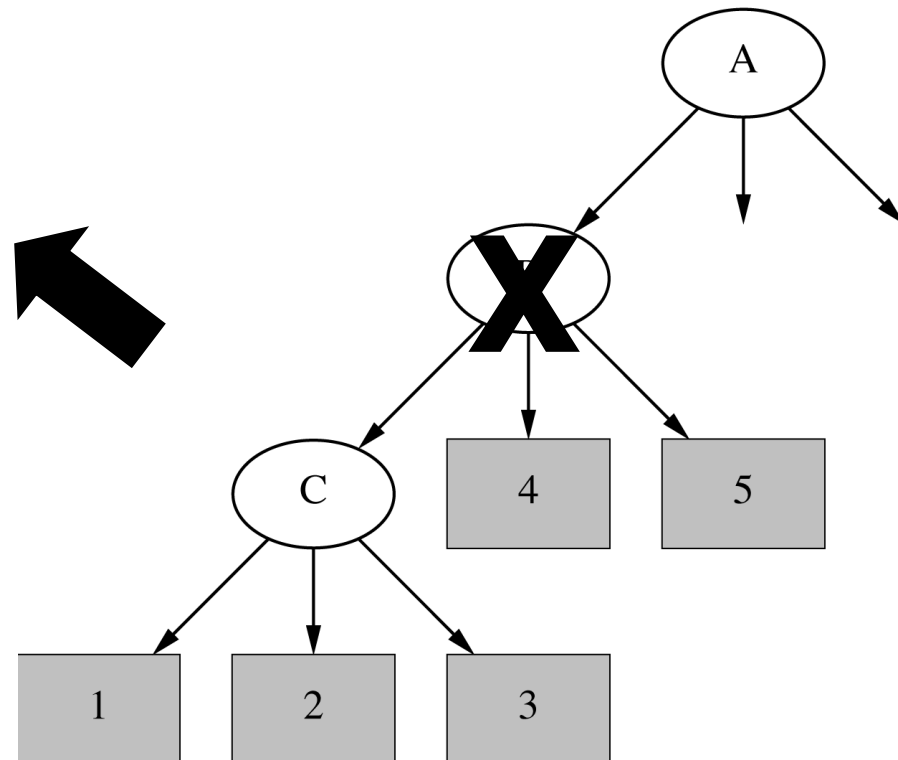
- ❑ Pre-pruning may stop the growth process prematurely:  
early stopping
- ❑ Classic example: XOR/Parity-problem
  - ▶ No individual attribute exhibits any significant association to the class
  - ▶ Structure is only visible in fully expanded tree
  - ▶ Pre-pruning won't expand the root node
- ❑ But: XOR-type problems rare in practice
- ❑ Pre-pruning is faster than post-pruning

	a	b	class
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

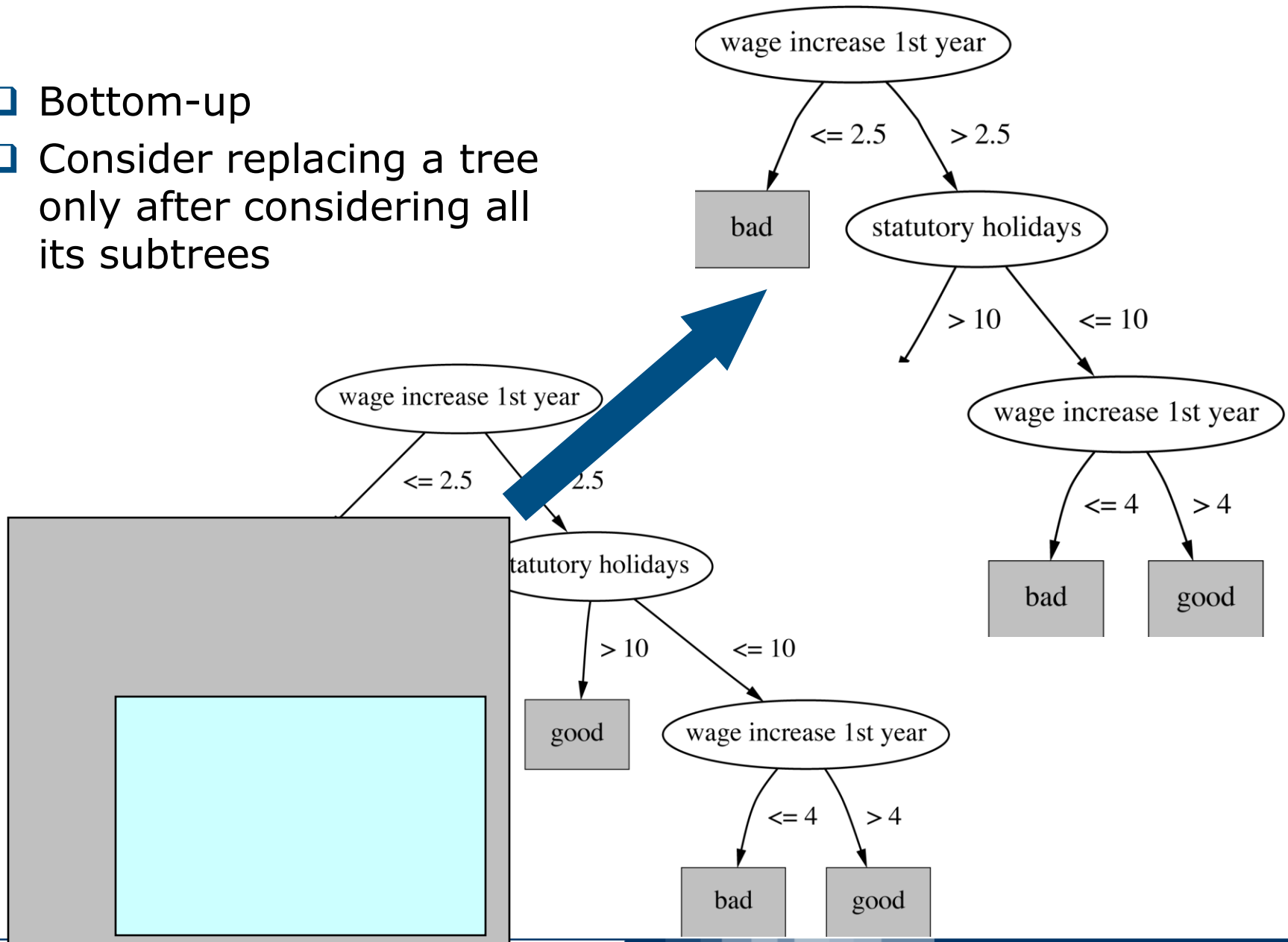
- ❑ First, build full tree, then prune it
- ❑ Fully-grown tree shows all attribute interactions
- ❑ Problem: some subtrees might be due to chance effects
  
- ❑ Two pruning operations
  - ▶ Subtree raising
  - ▶ Subtree replacement
  
- ❑ Possible strategies
  - ▶ Error estimation
  - ▶ Significance testing
  - ▶ MDL principle



- ❑ Delete node
- ❑ Redistribute instances
- ❑ Slower than subtree replacement



- ❑ Bottom-up
- ❑ Consider replacing a tree only after considering all its subtrees



- ❑ Prune only if it reduces the estimated error
- ❑ Error on the training data is NOT a useful estimator  
Q: Why it would result in very little pruning?
- ❑ Use hold-out set for pruning  
("reduced-error pruning")
- ❑ C4.5's method
  - ▶ Derive confidence interval from training data
  - ▶ Use a heuristic limit, derived from this, for pruning
  - ▶ Standard Bernoulli-process-based method
  - ▶ Shaky statistical assumptions (based on training data)

- ❑ Mean and variance for a Bernoulli trial:  $p, p(1-p)$
- ❑ Expected success rate  $f=S/N$
- ❑ Mean and variance for  $f$  :  $p, p(1-p)/N$
- ❑ For large enough  $N$ ,  $f$  follows a Normal distribution
- ❑  $c\%$  confidence interval  $[-z \leq X \leq z]$  for random variable with 0 mean is given by:
- ❑ With a symmetric distribution,

$$\begin{aligned} P[-z \leq X \leq z] &= c \\ &= 1 - 2 \times P[X \geq z] \end{aligned}$$



□ Confidence limits for the normal distribution with 0 mean and a variance of 1,

□ Thus,

$$P[-1.65 \leq X \leq 1.65] = 90\%$$

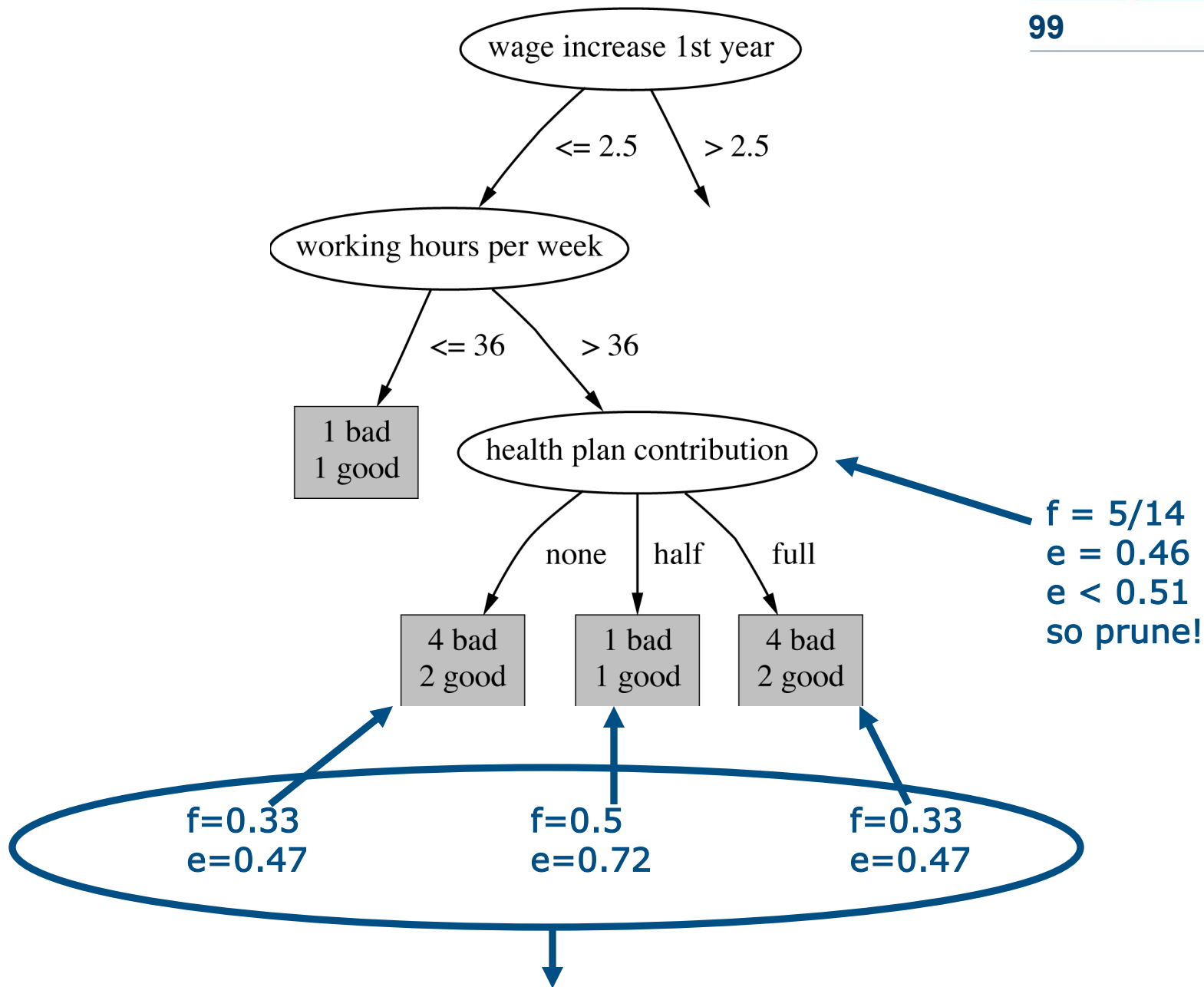
$\Pr[X \geq z]$	$z$
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
25%	0.69
40%	0.25

□ To use this we have to reduce our random variable  $f$  to have 0 mean and unit variance

- ❑ Error estimate for subtree is weighted sum of error estimates for all its leaves
- ❑ Given the error  $f$  on the training data, the error estimate for a node (upper bound):

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

- ❑ If  $c = 25\%$  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



Combined using ratios 6:2:6 gives 0.51

# Summary

- ❑ Classification—a classical problem extensively studied by statisticians and machine learning researchers
- ❑ Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- ❑ Why decision tree induction in data mining?
  - ▶ relatively faster learning speed (than other classification methods)
  - ▶ convertible to simple and easy to understand classification rules
  - ▶ can use SQL queries for accessing databases
  - ▶ comparable classification accuracy with other methods

- ❑ SLIQ (EDBT'96 — Mehta et al.)
  - ▶ builds an index for each attribute and only class list and the current attribute list reside in memory
- ❑ SPRINT (VLDB'96 — J. Shafer et al.)
  - ▶ constructs an attribute list data structure
- ❑ PUBLIC (VLDB'98 — Rastogi & Shim)
  - ▶ integrates tree splitting and tree pruning: stop growing the tree earlier
- ❑ RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - ▶ separates the scalability aspects from the criteria that determine the quality of the tree
  - ▶ builds an AVC-list (attribute, value, class label)

- ❑ Classification is an extensively studied problem (mainly in statistics, machine learning & neural networks)
- ❑ Classification is probably one of the most widely used data mining techniques with a lot of extensions
- ❑ Decision Tree Construction
  - ▶ Choosing the Splitting Attribute
  - ▶ Choosing the purity measure

- ❑ Information Gain biased towards attributes with a large number of values
- ❑ Gain Ratio takes number and size of branches into account when choosing an attribute
- ❑ Scalability is still an important issue for database applications: thus combining classification with database techniques should be a promising topic
- ❑ Research directions: classification of non-relational data, e.g., text, spatial, multimedia, etc.