Data Mining

Classification and Prediction (Decision Tree)

Outline

- What is classification? What is prediction?
- Classification by decision tree induction
- Bayesian Classification
- Classification by backpropagation
- Other Classification Methods
- Classification accuracy
- Summary

An Example Dataset

This follows an example from Quinlan's ID3

age	income	student	credit_rating	**buys_computer**
<=30	high	no	fair	no
<=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
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Classification and Prediction

- Classification:
  - predicts categorical class labels
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Prediction:
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Example Applications
  - credit approval- classify loan application by their likelihood of defaulting on payments
  - target marketing
  - medical diagnosis
  - treatment effectiveness analysis

Classification Applications

- Example Applications (continued)
  - Image processing : interpretation of digital images in radiology, recognizing 3-D objects, outdoor image segmentation
  - Language processing : text classification
  - Software development : estimate the development effort of a given software module
  - Pharmacology: drug analysis
  - Molecular biology : analyzing amino acid sequences
  - Medicine : cardiology, analyzing sudden infant death syndrome, diagnosing thyroid disorder
  - Manufacturing : classify equipment malfunctions by their cause
- UCI Machine Learning Repository

Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction: training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The set of tuples used for testing the performance of the model: test data
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur

Outline

- What is classification? What is prediction?
- Classification by decision tree induction
- Bayesian Classification
- Classification by backpropagation
- Other Classification Methods
- Classification accuracy
- Summary

Classification by Decision Tree Induction

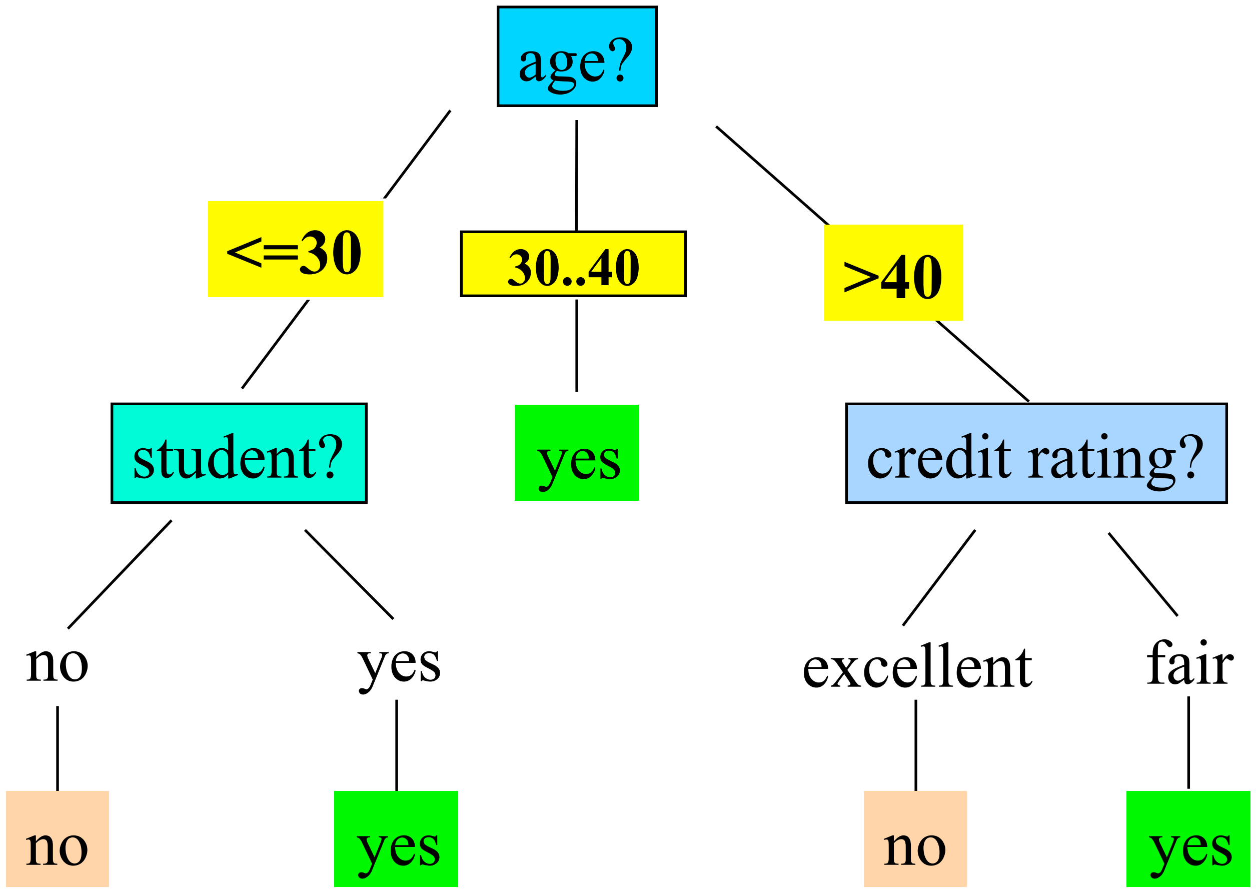
- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution
- Decision tree generation consists of two phases
  - Tree construction
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample
  - Test the attribute values of the sample against the decision tree

An Example Dataset

This follows an example from Quinlan's ID3

age	income	student	credit_rating	**buys_computer**
<=30	high	no	fair	no
<=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
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Output: A Decision Tree for "*buys_computer*"



Algorithm for Decision Tree Induction

- 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
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

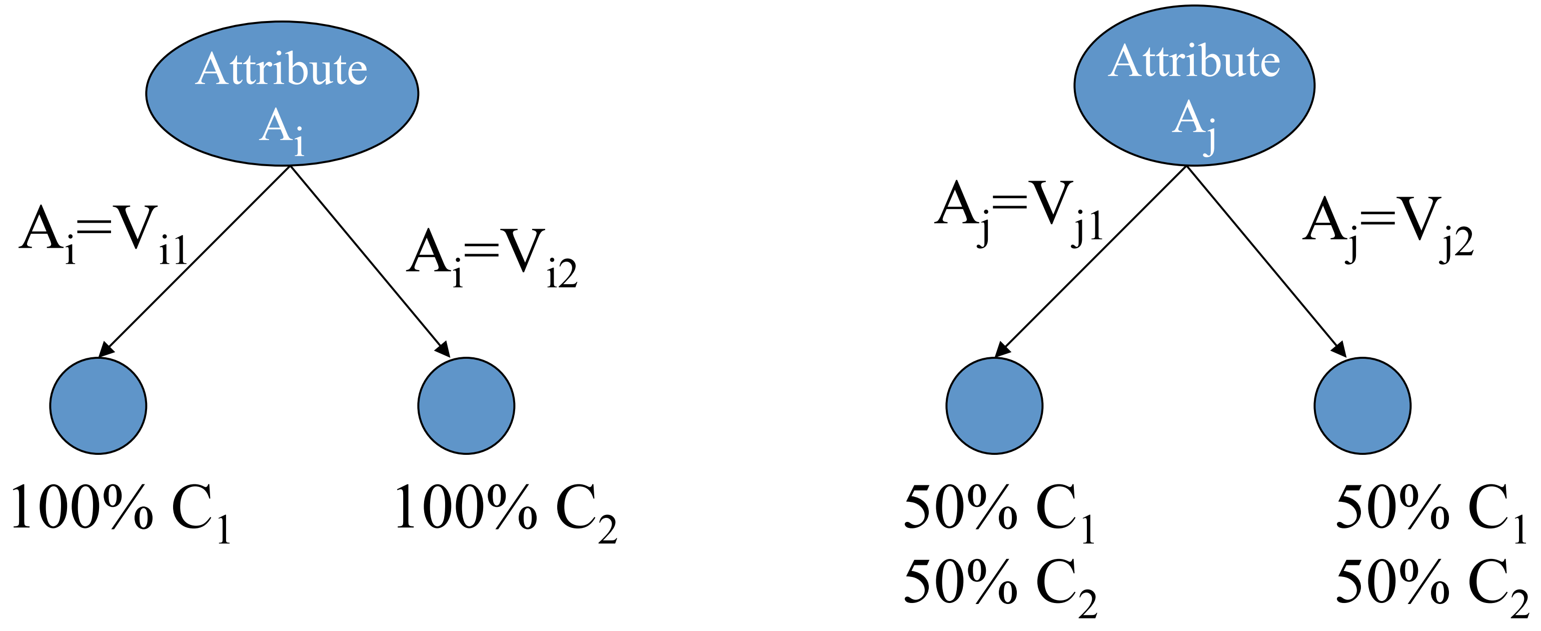
How to select the *best attribute* ?

- Random, or Least values or Most values
- Information gain: choose attribute with largest expected information gain, i.e., choose attribute that will result in the smallest expected size of the sub-tree rooted at its children.
  - ID3 (Quinlan 1987)
  - Occam’s Razor: The simplest explanation that is consistent with all the observations is the best → smallest decision tree that correctly classifies all of the training examples is the best

Attribute Selection Measure

- Information gain (ID3)
  - All attributes are assumed to be categorical
  - Can be modified for continuous-valued attributes
    (C4.5 deals with continuous-valued attributes)
- Gini index (IBM IntelligentMiner)
  - All attributes are assumed continuous-valued
  - Assume there exist several possible split values for each attribute
  - May need other tools, such as clustering, to get the possible split values
  - Can be modified for categorical attributes

Attribute Selection



Which attribute is better at splitting up the data ?

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**?**

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

The more predictable is Y given X,

The smaller H(Y|X), thus the higher

the information gain, IG(Y|X)

Entropy

- **Shannon entropy: quantifies the expected value of the information contained in a message.**
  - Claude E. Shannon in his 1948 paper "A Mathematical Theory of Communication”.
  - The information conveyed by a message depends on its probability and can be measured in bits as minus the logarithm to base 2 of that probability.

$$-\log_2 p_j$$

Entropy

- Suppose we are to convey the following messages, each has a different probability:
  - High (50%), Medium (30%), Low (20%)
- How much information (in bits) is conveyed by selecting to send the message that has "High" probability?
- What is the expected information conveyed if we are to select any of the three messages to send? → Entropy (H)

Entropy used in Classification

Suppose attribute X can have one of *m* values… $V_1, V_2, \ldots V_m$

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

What's the expected number of bits needed to transmit a value randomly drawn from X's distribution?

$$H(X) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 - \ldots - p_m \log_2 p_m$$

$$= -\sum_{j=1}^{m} p_j \log_2 p_j$$

H(X) = The entropy of X

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

Attribute Selection

A message may be coded using 4 characters:

C1:	50%
C2:	25%
C3:	12.5%
C4:	12.5%

Or another set:

C1:	50%
C2:	0%
C3:	50%
C4:	0%

Encoding each character requires $-\log_2 P(C_k)$ bits of information

C1: $0.5 = 2^{-1}$ → 1 bit

C2: $0.25 = 2^{-2}$ → 2 bits

C3: $0.125 = 2^{-3}$ → 3 bits

C4: $0.125 = 2^{-3}$ → 3 bits

$$H(C)=(0.5*1+0.25*2+0.125*3+0.125*3) = 1.75$$

Entropy - General Case

A histogram of the frequency distribution of values of X would be flat

A histogram of the frequency distribution of values of X would have many lows and one or two highs

H(X) = The entropy of X

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

Attribute Selection

- The expected encoding of a class C_k:

$$P(C_k)\cdot(-\log_2 P(C_k))$$

- Expected encoding of one child node containing K classes is

$$H(\text{node with K classes})=\sum_{k=1}^{K} P(C_k)\cdot(-\log_2 P(C_k))$$

- For an attribute A_i, having J different values, the expected encoding of all child nodes resulting from selecting this attribute is:

$$H(\text{J child nodes})=\sum_{j=1}^{J} P(A_i=V_{ij})\cdot\sum_{k=1}^{K} P(C_k \mid A_i=V_{ij})(-\log_2 P(C_k \mid A_i=V_{ij}))$$

Maximize information gain

Information Gain=H(parent node) - H(J child nodes)

$$= \sum_{k=1}^{K} P(C_k)\cdot(-\log_2 P(C_k)) - (\sum_{j=1}^{J} P(A_i = V_{ij})\cdot\sum_{k=1}^{K} P(C_k \mid A_i = V_{ij})(-\log_2 P(C_k \mid A_i = V_{ij}))$$

- Use Information gain for attribute selection:
  - The reduction in the expected number of encoding for all child nodes formed with the current attribute

    The larger the reduction, the higher the gain.
  - Select the best attribute by computing the information gain of all attributes that are currently available, pick the one that generates the highest gain

What is Information Gain used for?

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

- IG(LongLife | HairColor) = 0.01
- IG(LongLife | Smoker) = 0.2
- IG(LongLife | Gender) = 0.25
- IG(LongLife | LastDigitOfSSN) = 0.00001

Practice Question

Given the following data, which attribute should be selected to form the first level of the decision tree?

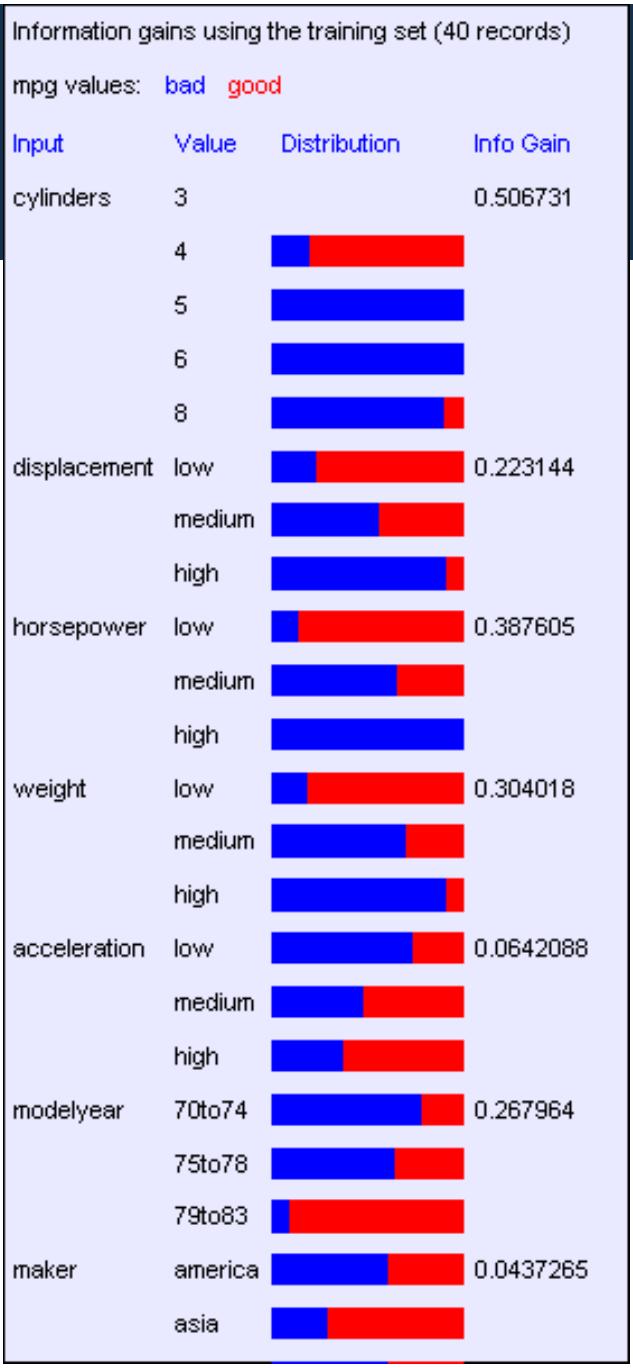
Data:

Object	color	shape	size	class
1	red	square	big	+
2	blue	square	big	+
3	red	round	small	-
4	green	square	small	-
5	red	round	big	+
6	green	square	big	-

A small dataset: Miles Per Gallon

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

- Look at all the information gains…

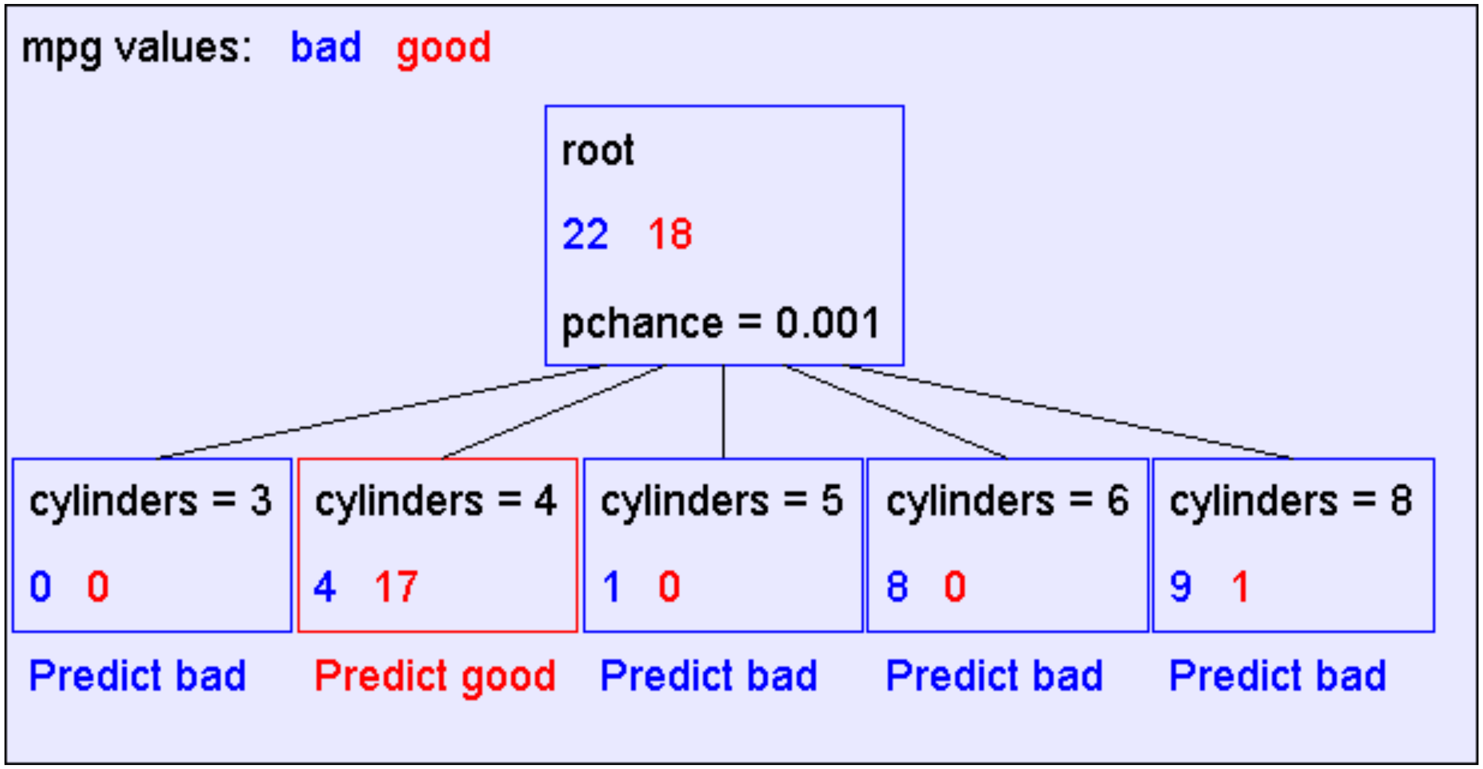


Information gains using the training set (40 records)

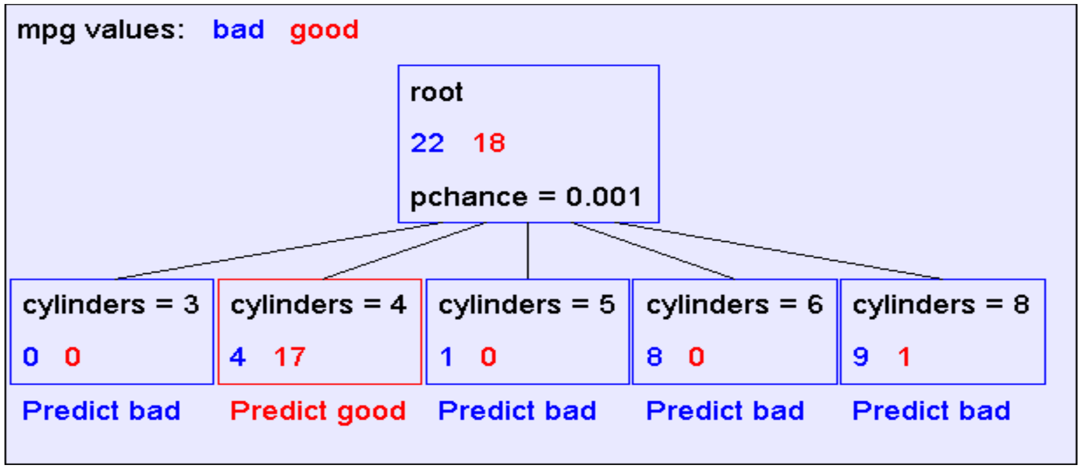
mpg values: bad good

Input	Value	Distribution	Info Gain
cylinders	3		0.506731
	4		
	5		
	6		
	8		
displacement	low		0.223144
	medium		
	high		
horsepower	low		0.387605
	medium		
	high		
weight	low		0.304018
	medium		
	high		
acceleration	low		0.0642088
	medium		
	high		
modelyear	70to74		0.267964
	75to78		
	79to83		
maker	america		0.0437265
	asia		

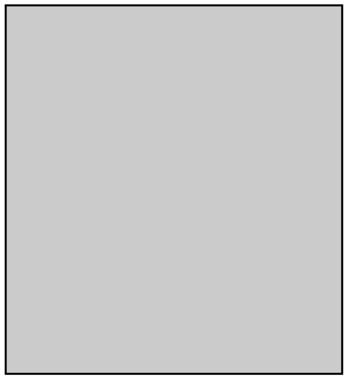
A Decision Stump



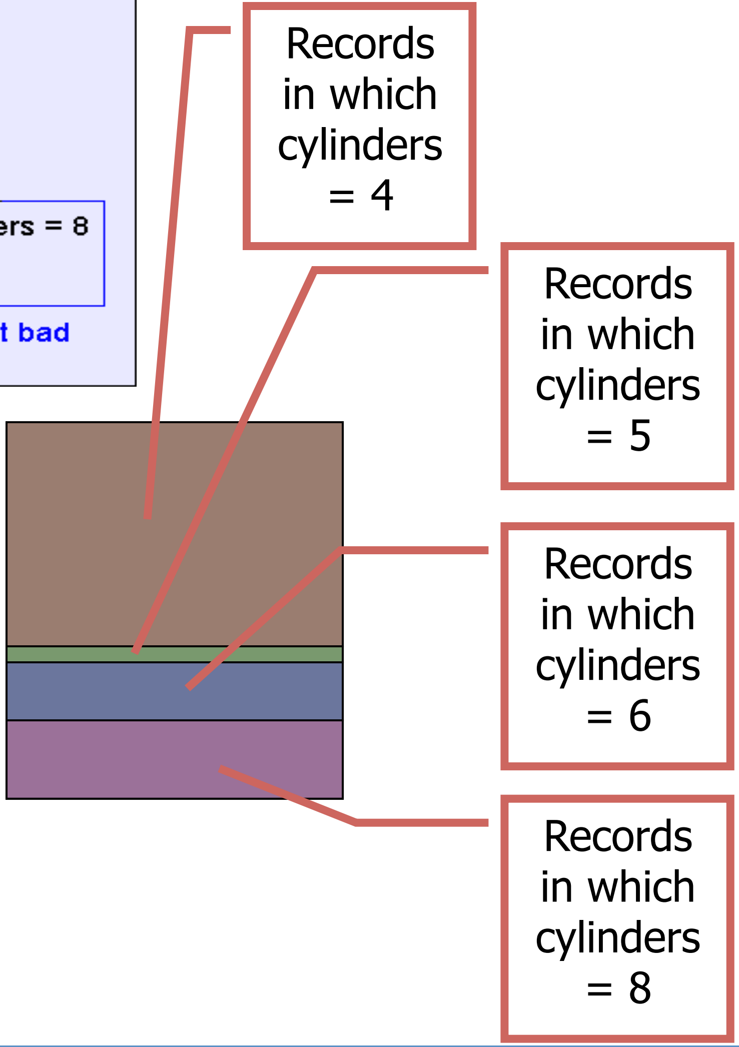
Recursion Step



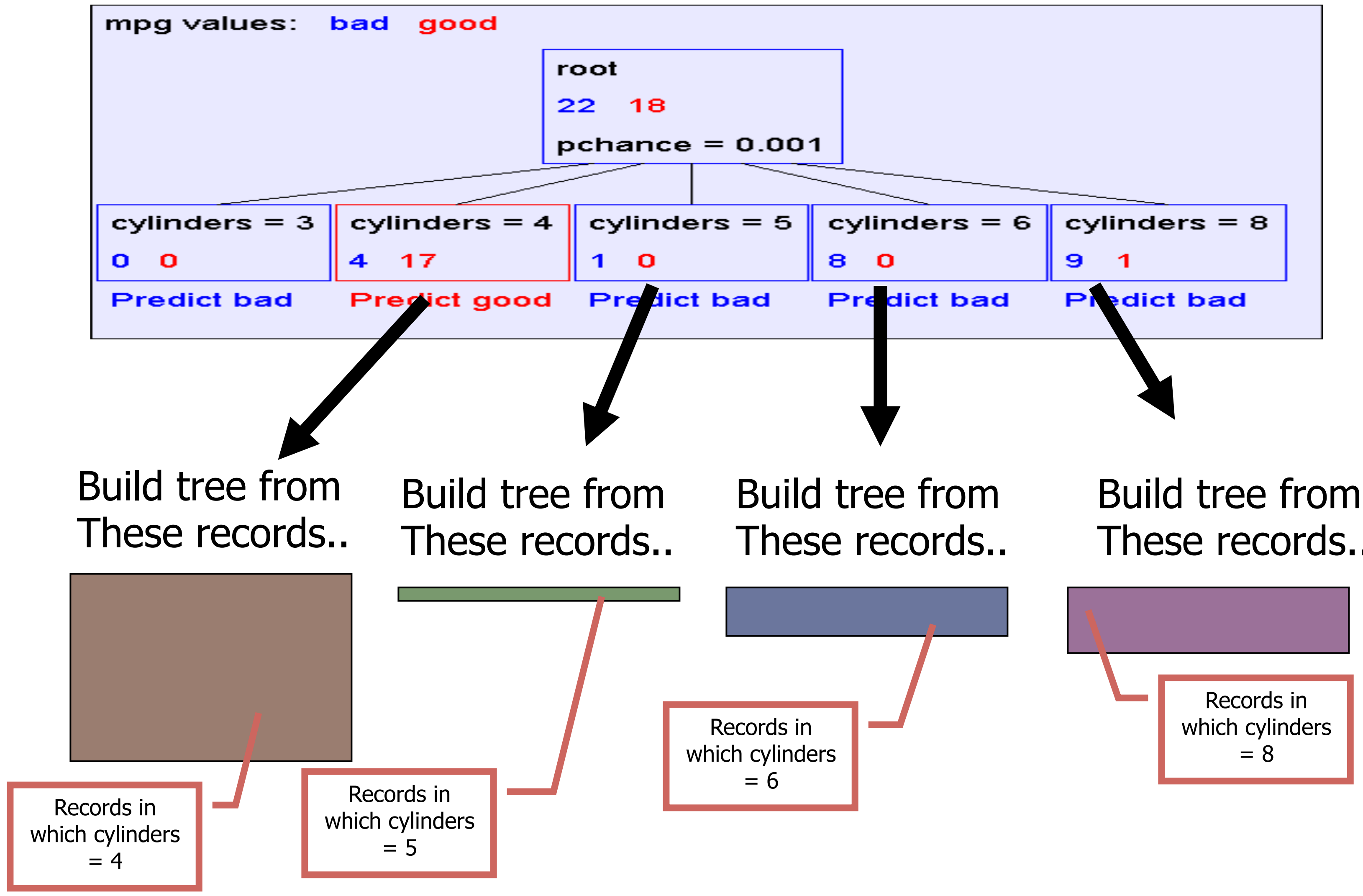
Take the Original Dataset..



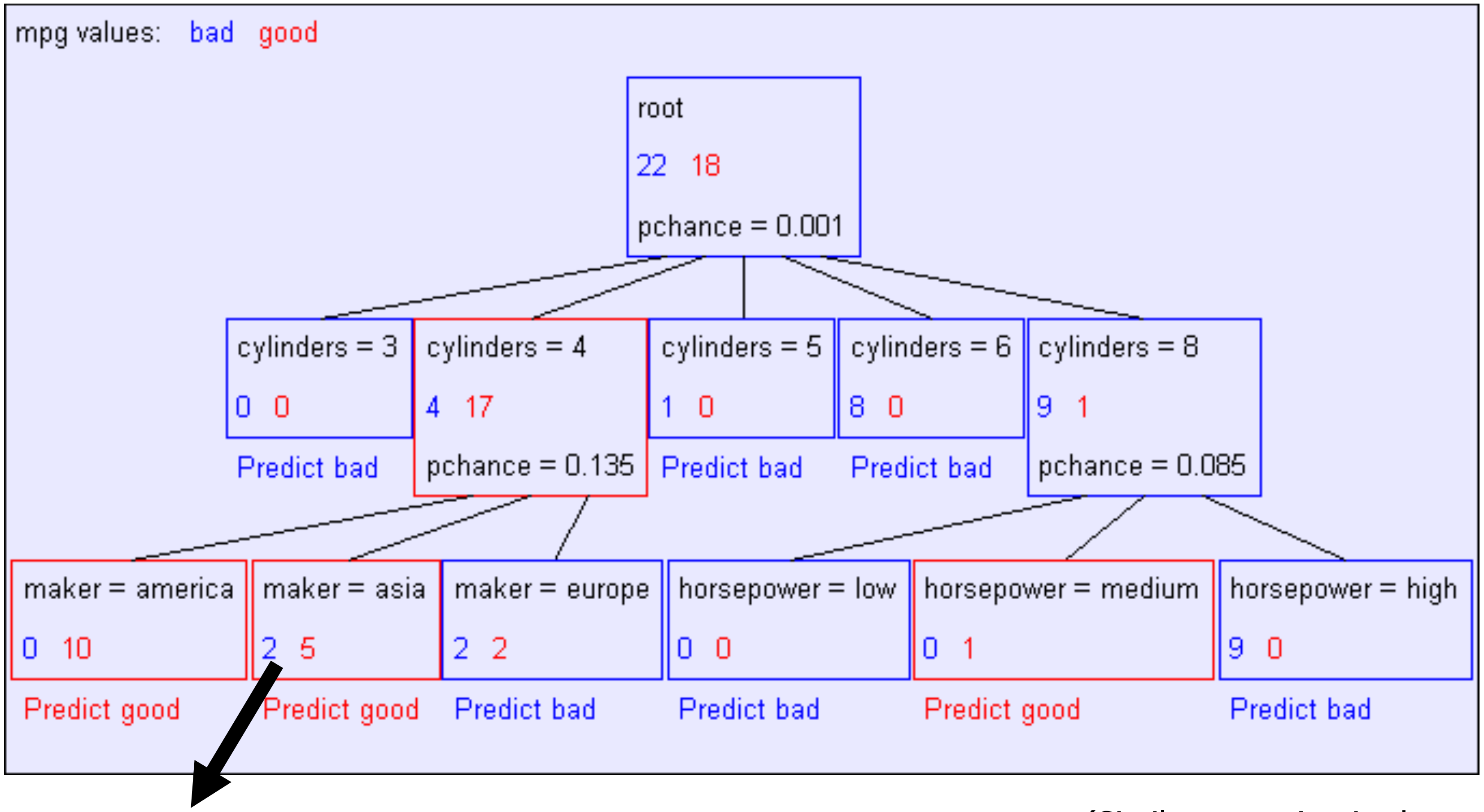
And partition it according to the value of the attribute we split on


Records in which cylinders = 4
Records in which cylinders = 5
Records in which cylinders = 6
Records in which cylinders = 8

Recursion Step


mpg values: bad good
root
22 18
pchance = 0.001
cylinders = 3
0 0
Predict bad
cylinders = 4
4 17
Predict good
cylinders = 5
1 0
Predict bad
cylinders = 6
8 0
Predict bad
cylinders = 8
9 1
Predict bad
Build tree from These records..
Build tree from These records..
Build tree from These records..
Build tree from These records..
Records in which cylinders = 4
Records in which cylinders = 5
Records in which cylinders = 6
Records in which cylinders = 8

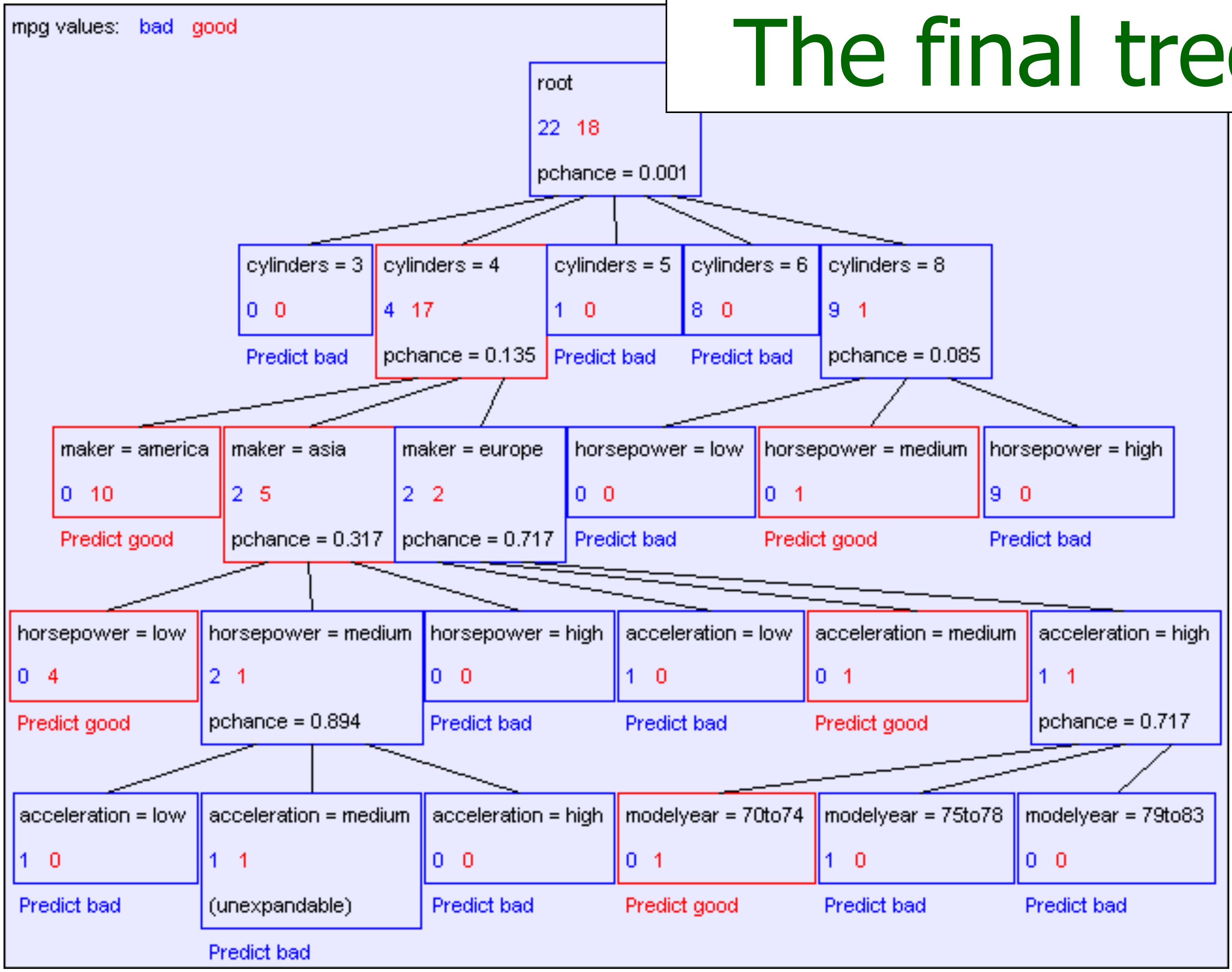
Second level of tree



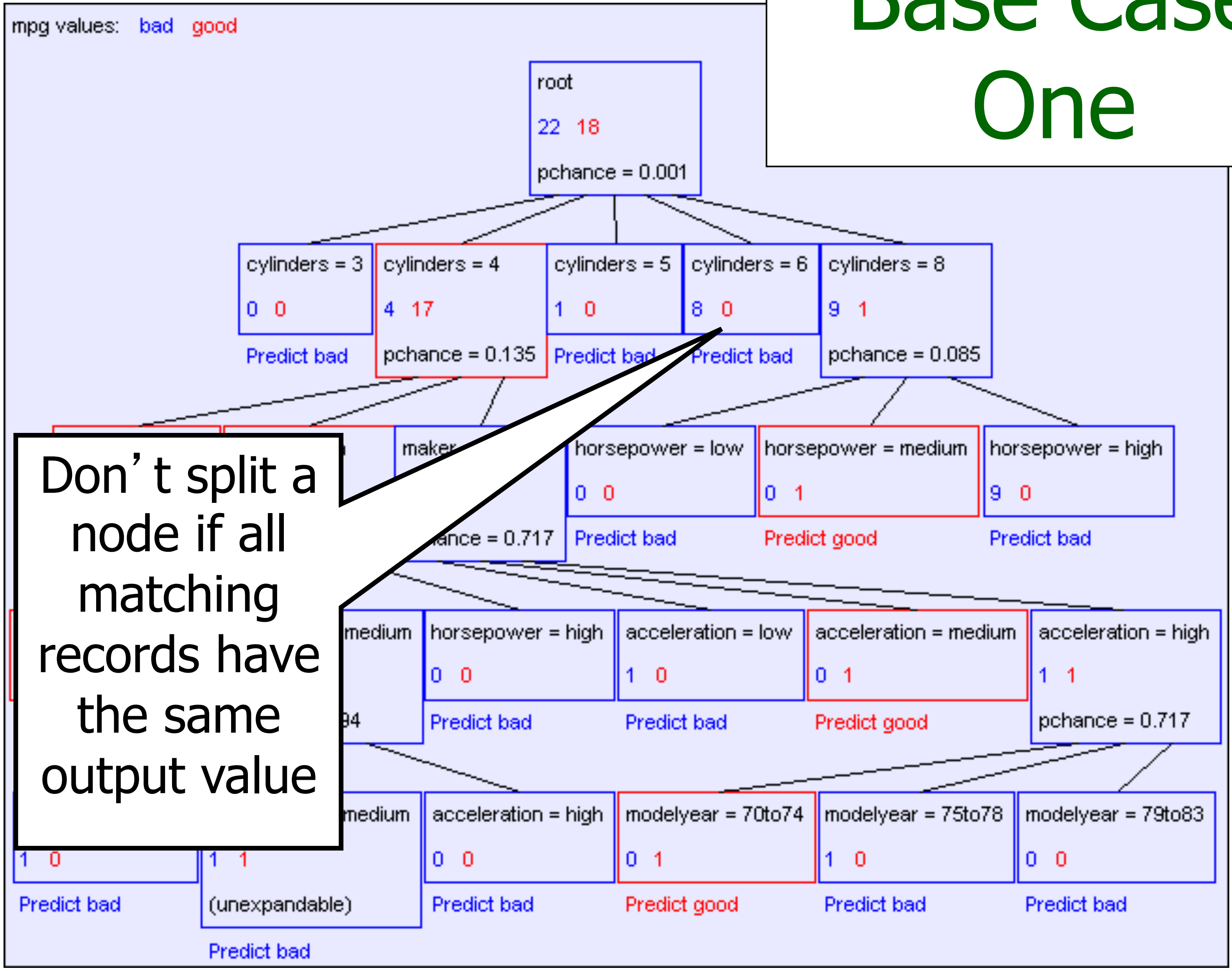
Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

The final tree

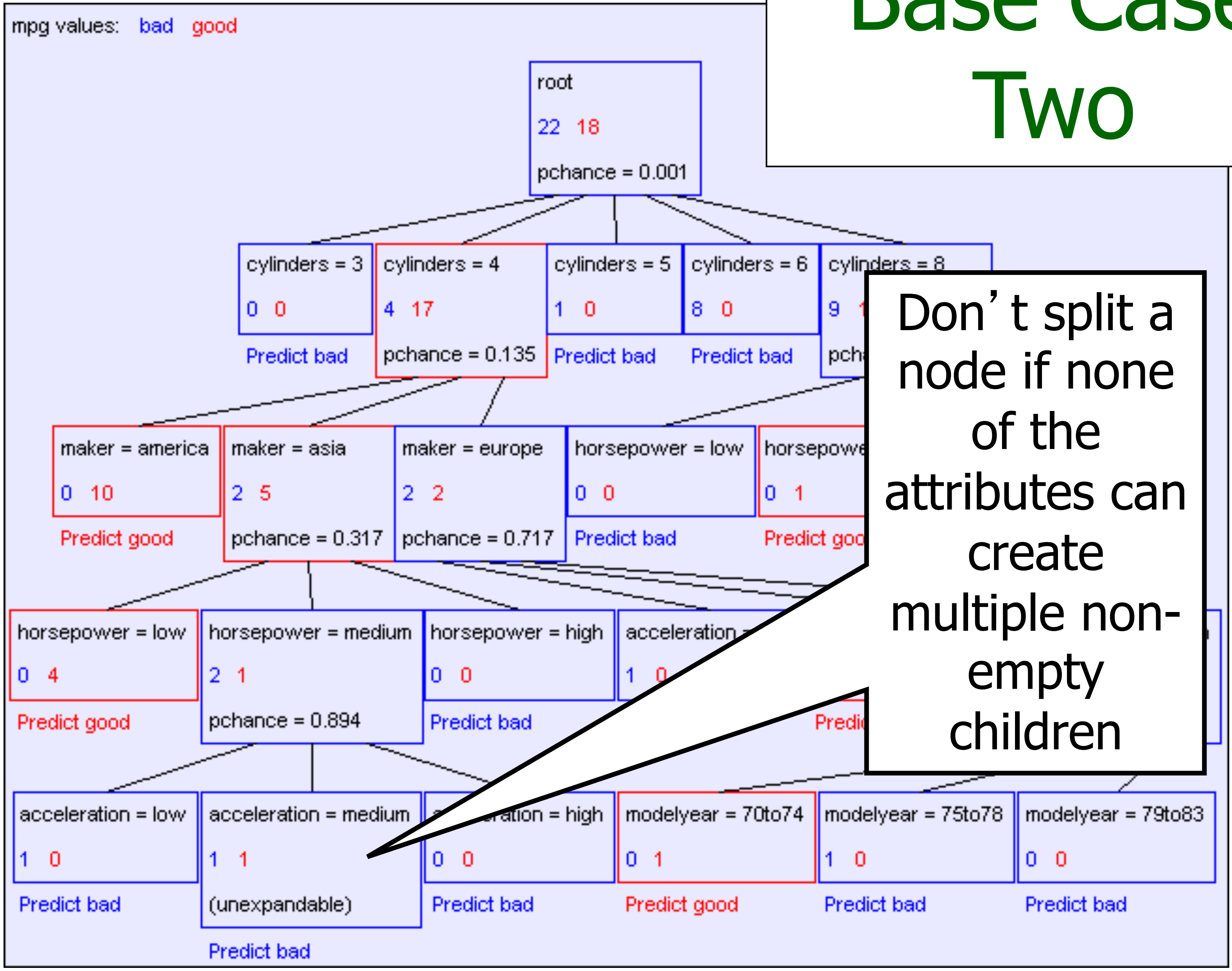


Base Case One



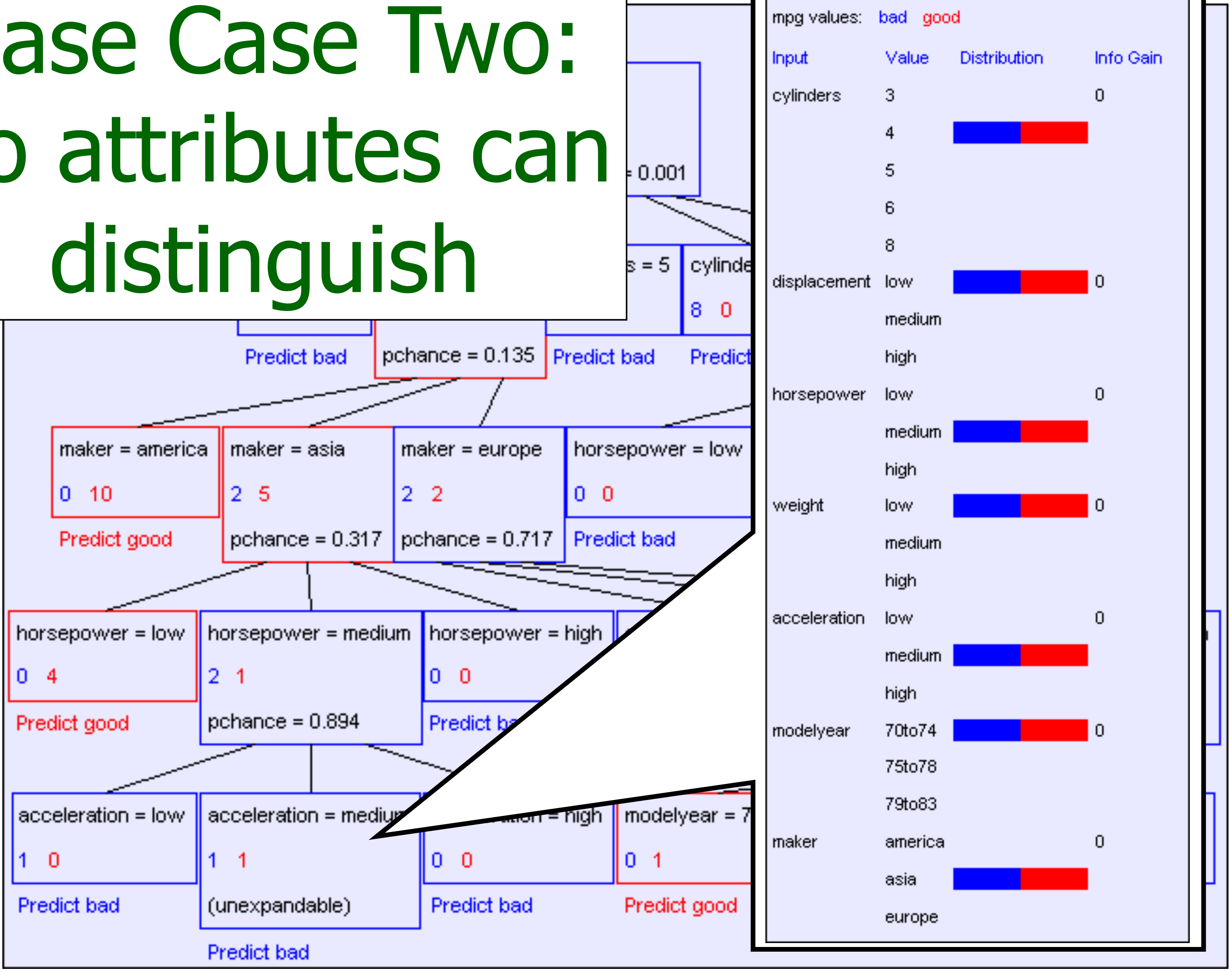
Don't split a node if all matching records have the same output value

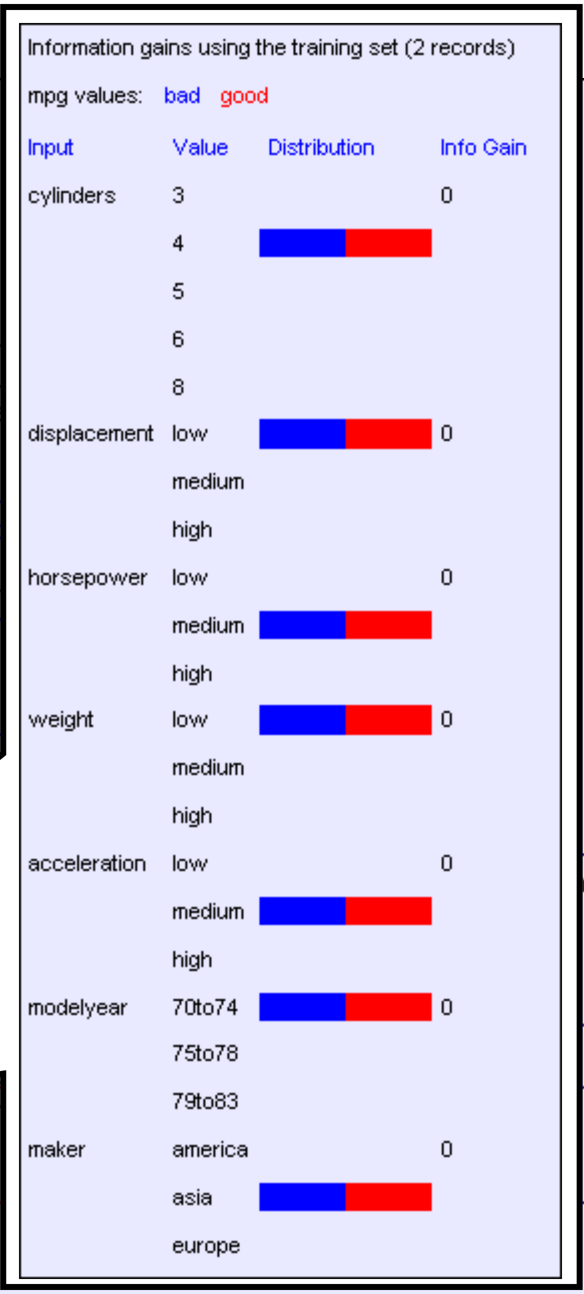
Base Case Two



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

Base Case Two: No attributes can distinguish





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 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?*

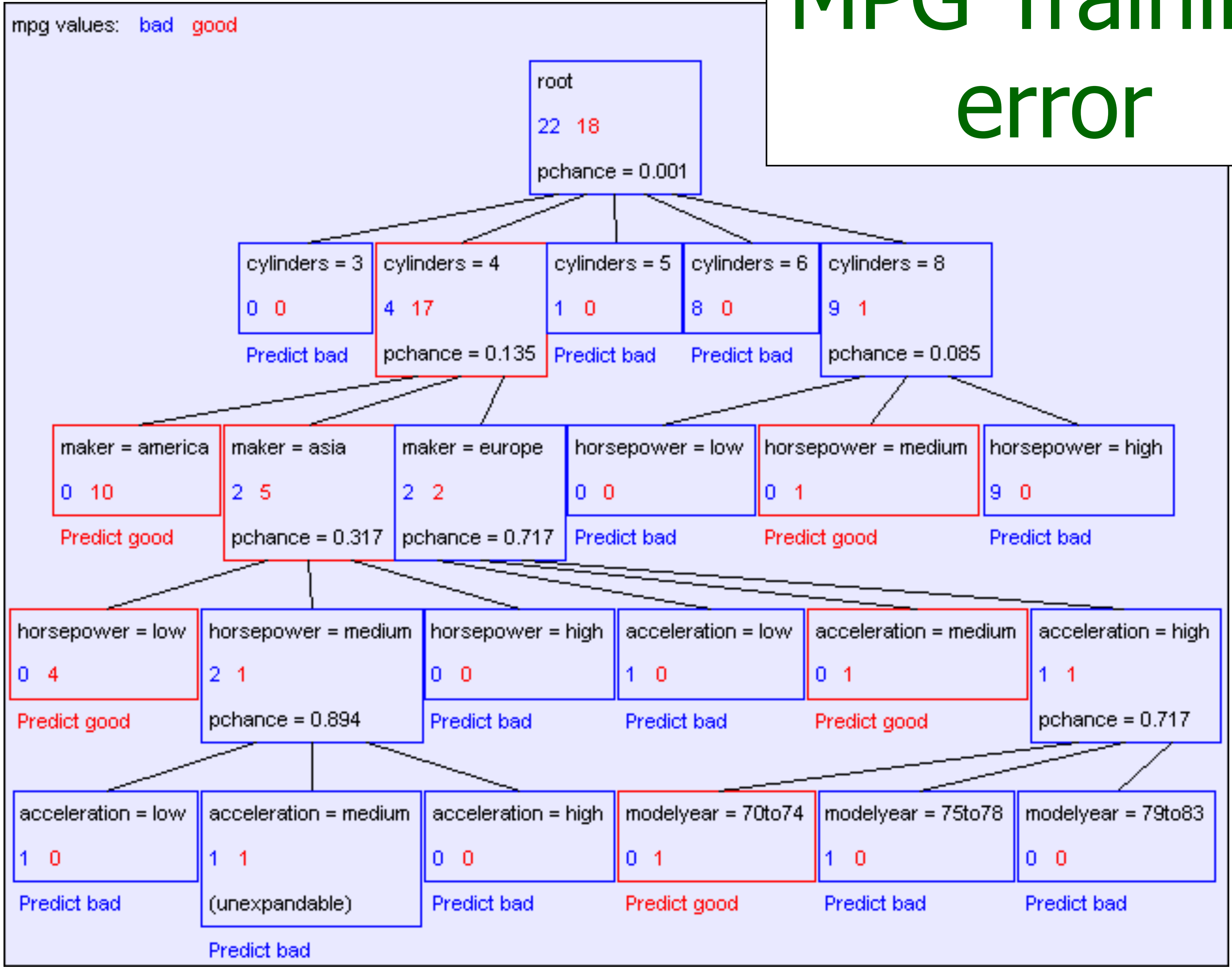
Decision Tree Algorithm

```
function decision-tree-learning (examples, attributes,  default)
begin
    if empty(examples) then return (default)
    else if same-classification(example) then return the classification
    else if can not differentiate examples then return majority-classification(examples)
    else
      best ←choose-attribute(attributes, examples)
           tree ← a new decision tree with root test best
      for each value v of attribute best do
      begin
                v-examples ← subset of examples with best = v
                subtree ← decision-tree-learning (v-examples, attribute-best,
                                                  majority-classification(examples))
                add a branch from tree to subtree with arc labeled v
      end
      return (tree)
end
```

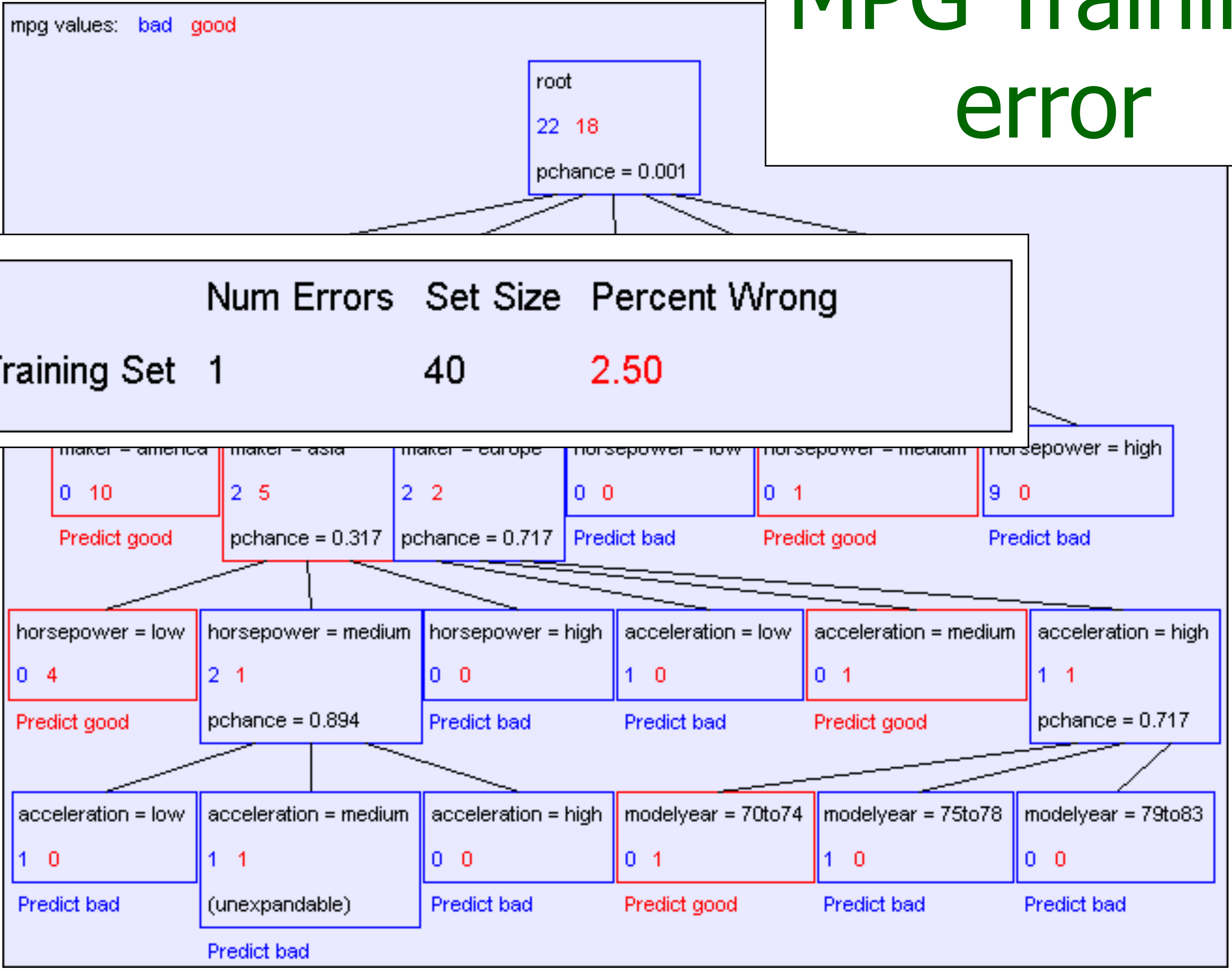
Training Set Error

- 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

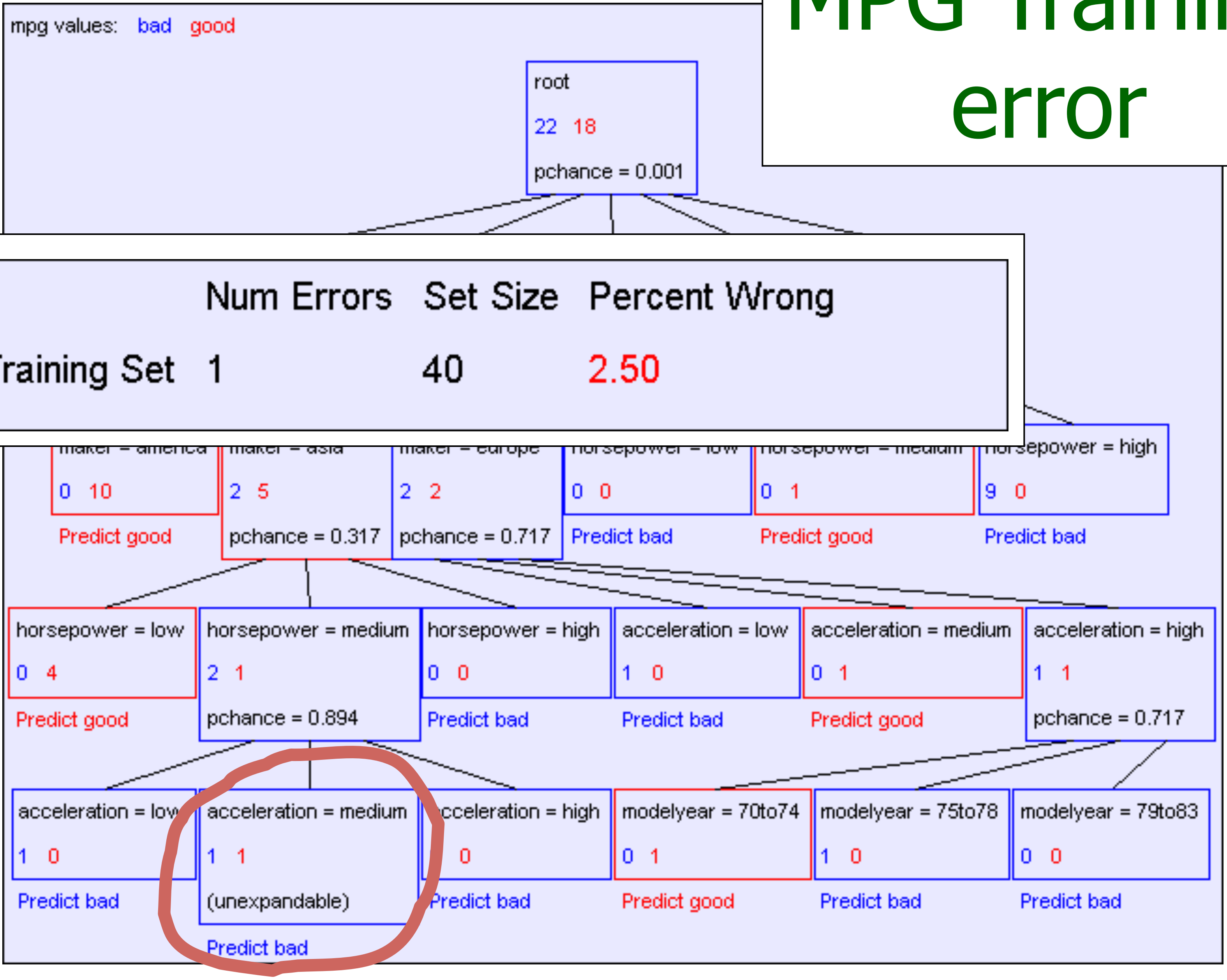


MPG Training error



	Num Errors	Set Size	Percent Wrong
Training Set	1	40	2.50

MPG Training error



	Num Errors	Set Size	Percent Wrong
Training Set	1	40	2.50

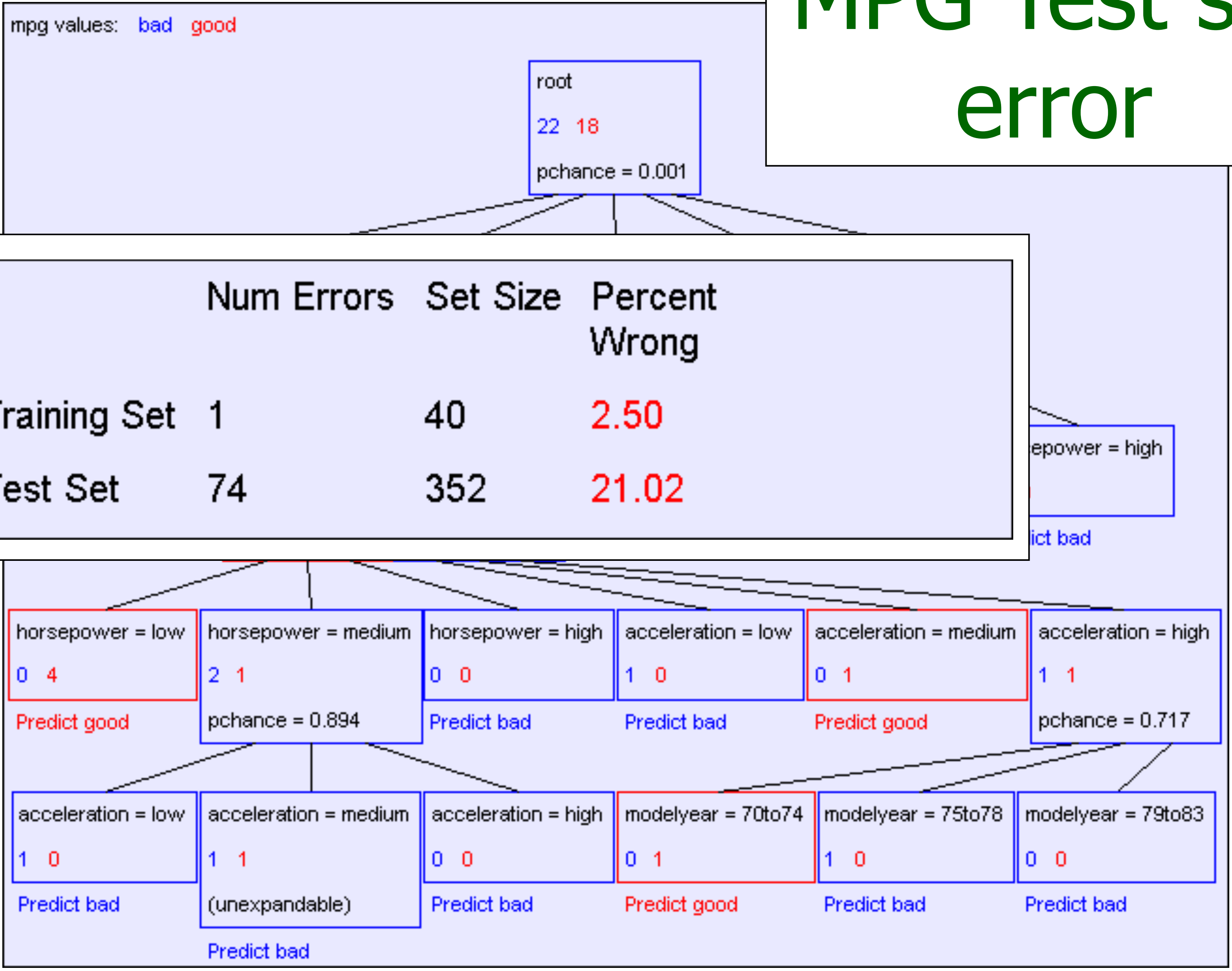
Why are we doing this learning?

- 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.

Test Set Error

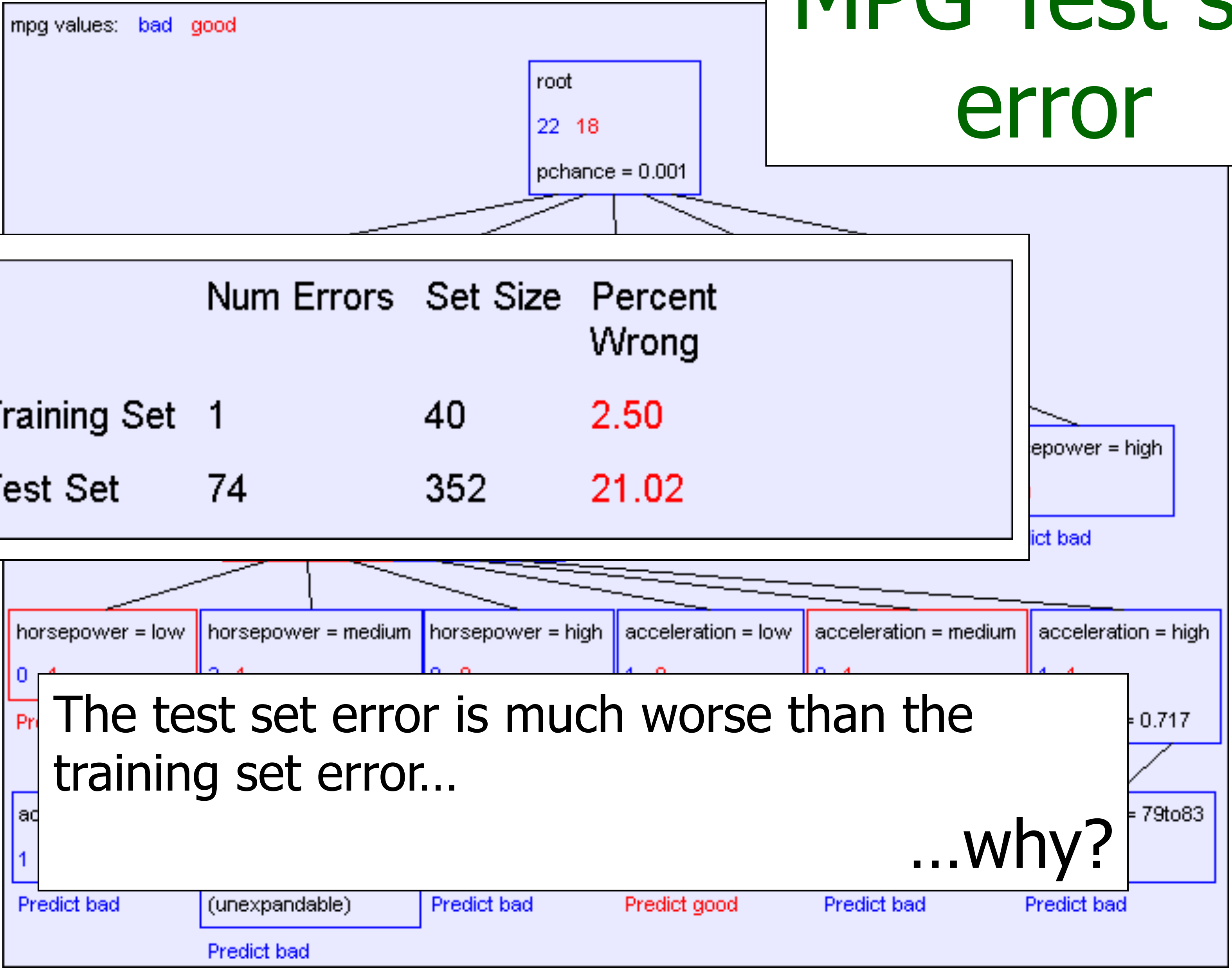
- Suppose we are forward thinking.
- We hide some data away when we learn the decision tree.
- But once learned, we see how well the tree predicts that data.
- This is a good simulation of what happens when we try to predict future data.
- And it is called Test Set Error.

MPG Test set error



	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?

An artificial example

- We'll create a training dataset

Five inputs, all bits, are generated in all 32 possible combinations

Output y = copy of e, Except a random 25% of the records have y set to the opposite of e

32 records

a	b	c	d	e	y
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	1
:	:	:	:	:	:
1	1	1	1	1	1

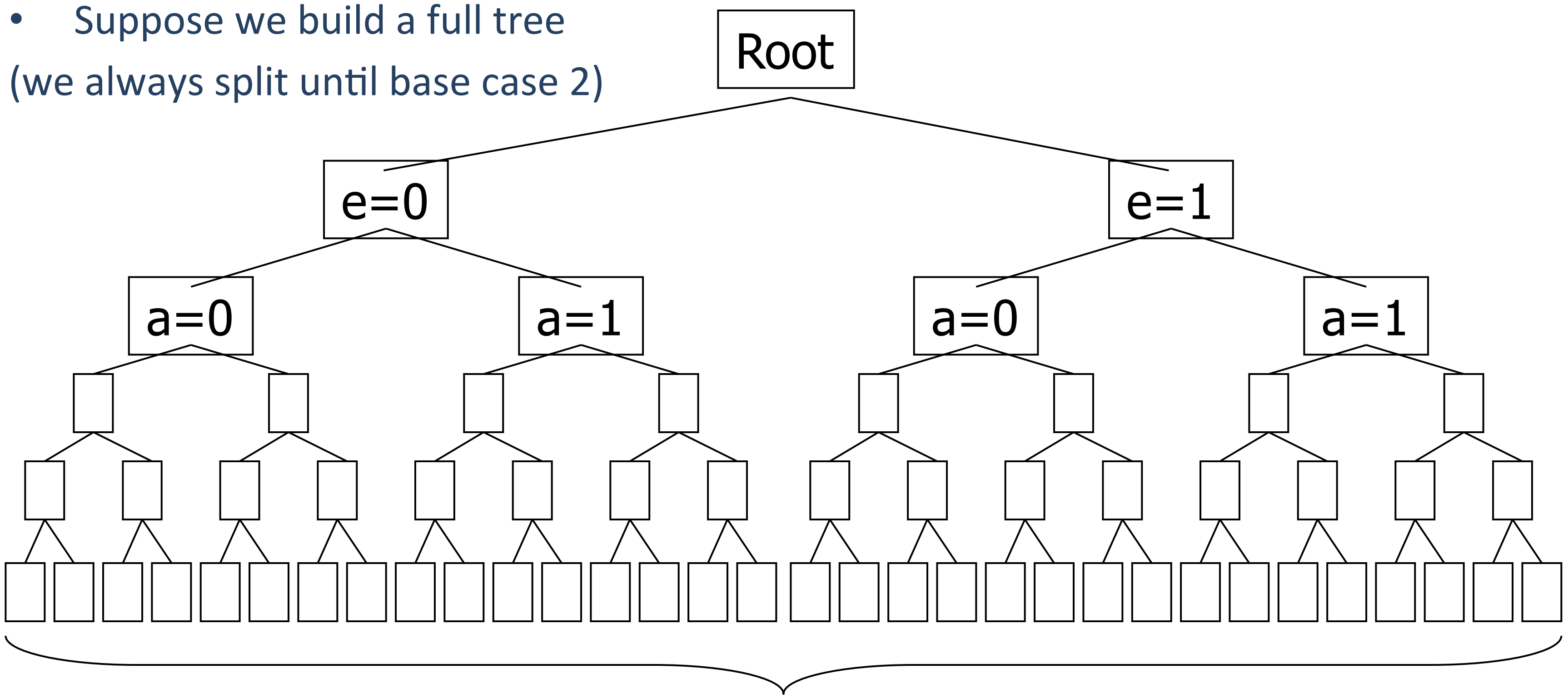
In our artificial example

- Suppose someone generates a test set according to the same method.
- The test set is identical, except that some of the y's will be different.
  - Some y's that were corrupted in the training set will be uncorrupted in the test set.
  - Some y's that were uncorrupted in the training set will be corrupted in the test set.

Building a tree with the training set

- Suppose we build a full tree

(we always split until base case 2)



25% of these leaf node labels will be corrupted

Training set error for our artificial tree

All the leaf nodes contain exactly one record and so…

- We would have a training set error of zero

Testing the tree with the test set

	1/4 of the tree nodes are corrupted	3/4 are fine
1/4 of the test set records are corrupted	1/16 of the test set will be correctly predicted for the wrong reasons	3/16 of the test set will be wrongly predicted because the test record is corrupted
3/4 are fine	3/16 of the test predictions will be wrong because the tree node is corrupted	9/16 of the test predictions will be fine

In total, we expect to be wrong on 3/8 of the test set predictions

What’s this example shown us?

- This explains the discrepancy between training and test set error
- But more importantly… …it indicates there’s something we should do about it if we want to predict well on future data.

Suppose we had less data

- Let's not look at the irrelevant bits

These bits are hidden

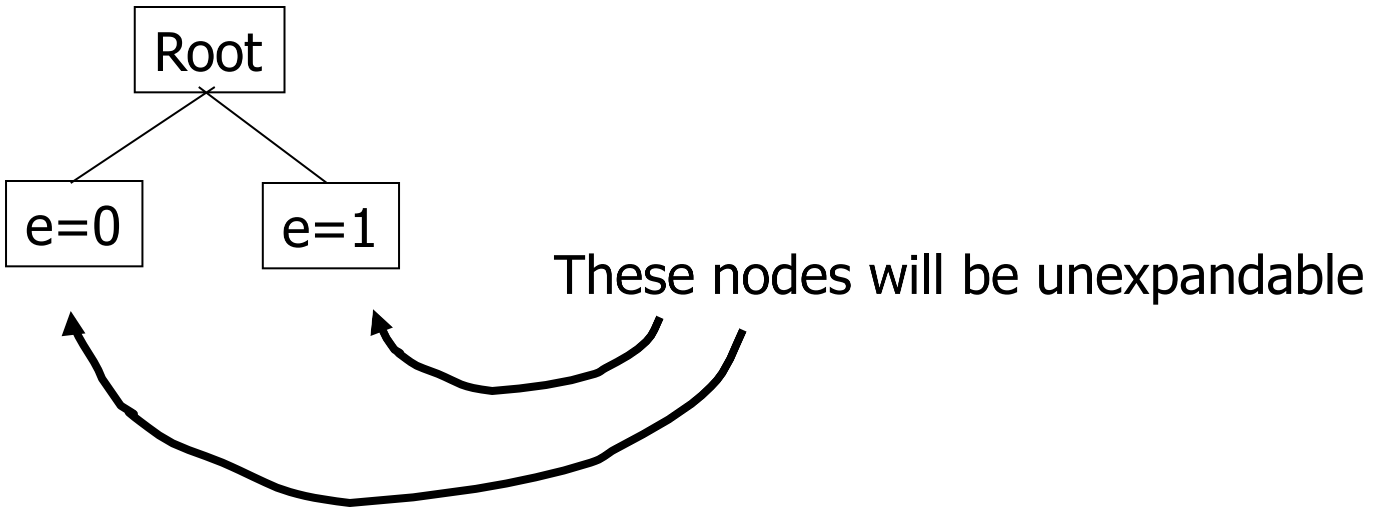
Output y = copy of e, except a random 25% of the records have y set to the opposite of e

32 records

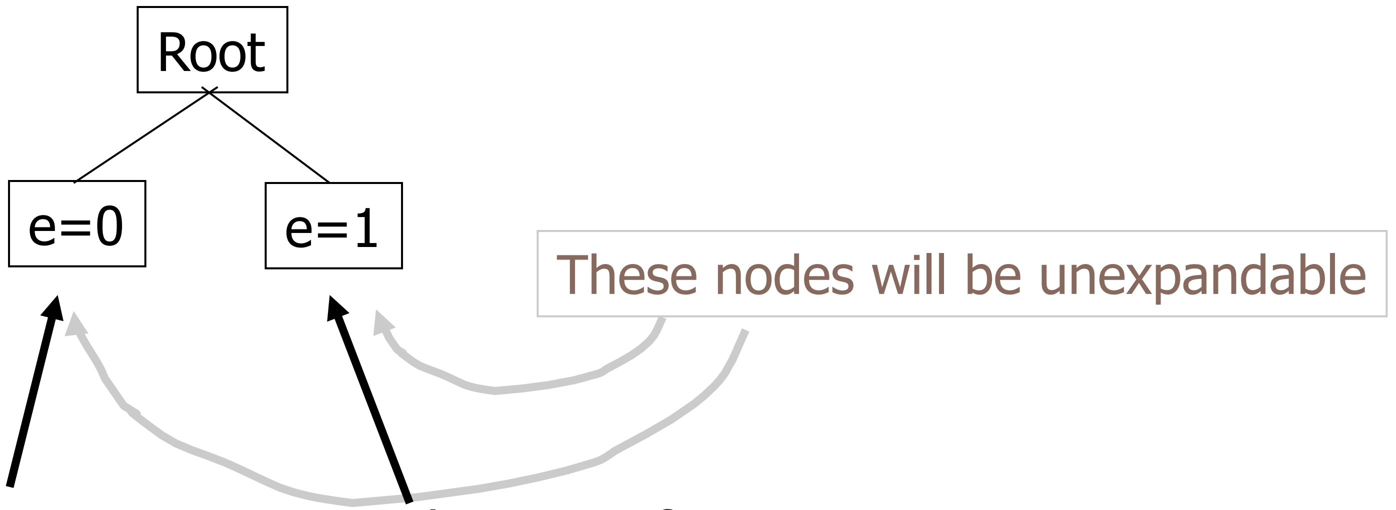
a	b	c	d	e	y
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	1
:	:	:	:	:	:
1	1	1	1	1	1

What decision tree would we learn now?

Without access to the irrelevant bits…



Without access to the irrelevant bits…



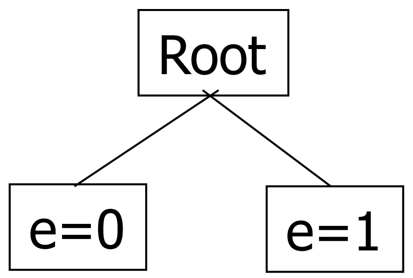
In about 12 of the 16 records in this node the output will be 0

So this will almost certainly predict 0

In about 12 of the 16 records in this node the output will be 1

So this will almost certainly predict 1

Without access to the irrelevant bits...



	almost certainly none of the tree nodes are corrupted	almost certainly all are fine
1/4 of the test set records are corrupted	n/a	1/4 of the test set will be wrongly predicted because the test record is corrupted
3/4 are fine	n/a	3/4 of the test predictions will be fine

In total, we expect to be wrong on only 1/4 of the test set predictions

Overfitting

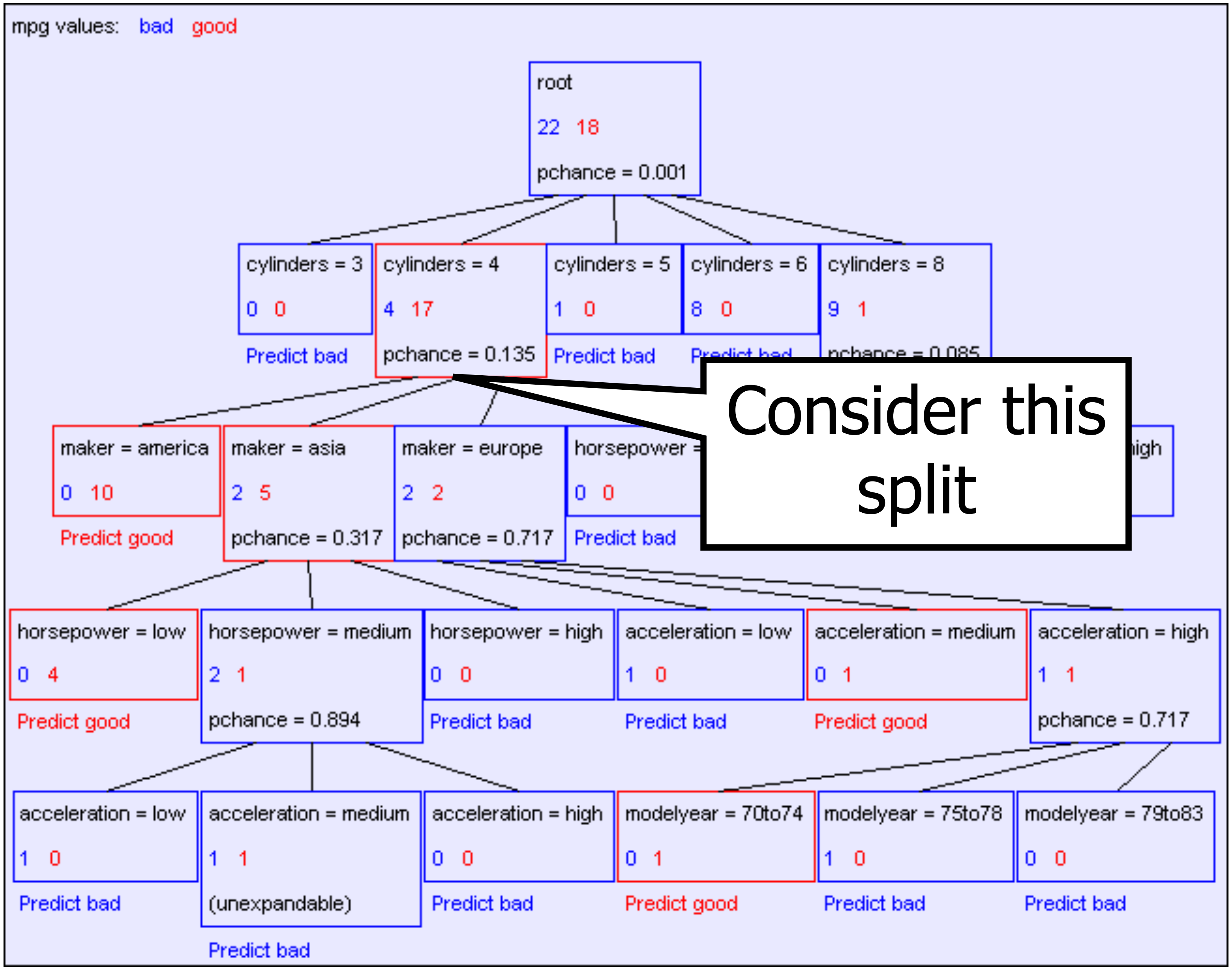
- Definition: If the learning algorithm fits noise (i.e. pays attention to parts of the data that are irrelevant) it is overfitting.
- Fact (theoretical and empirical): If the learning algorithm is overfitting then it may perform less well on test set data.

Avoiding overfitting

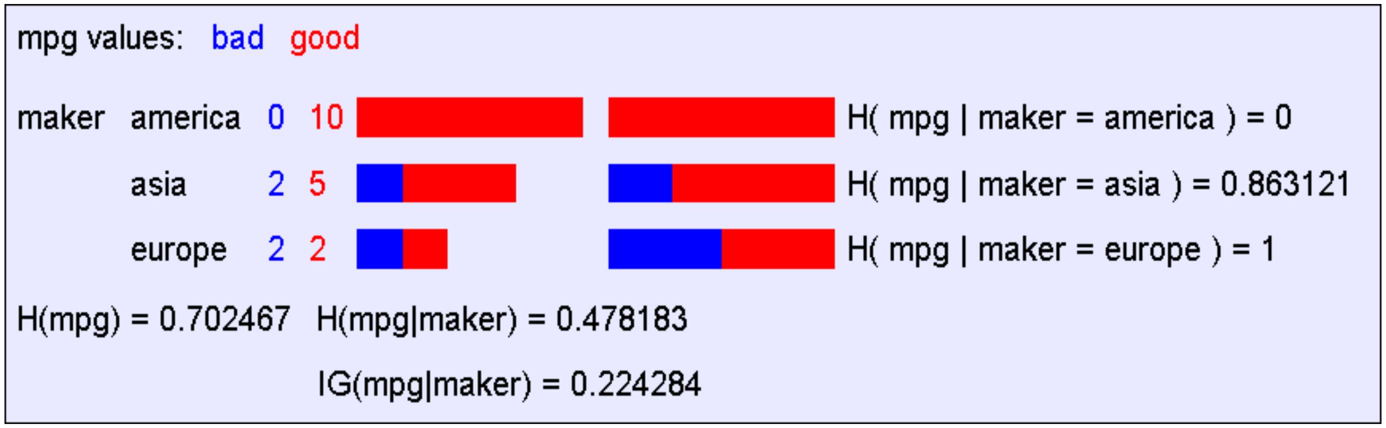
- Usually we do not know in advance which are the irrelevant variables
- …and it may depend on the context

  For example, if y = a AND b then b is an irrelevant variable only in the portion of the tree in which a=0

But we can use simple statistics to warn us that we might be overfitting.


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
Consider this split
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

Test for significance



```
maker  america  0  10   H( mpg | maker = america ) = 0
       asia     2  5    H( mpg | maker = asia ) = 0.863121
       europe   2  2    H( mpg | maker = europe ) = 1
H(mpg) = 0.702467   H(mpg|maker) = 0.478183
                    IG(mpg|maker) = 0.224284
```

- 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?

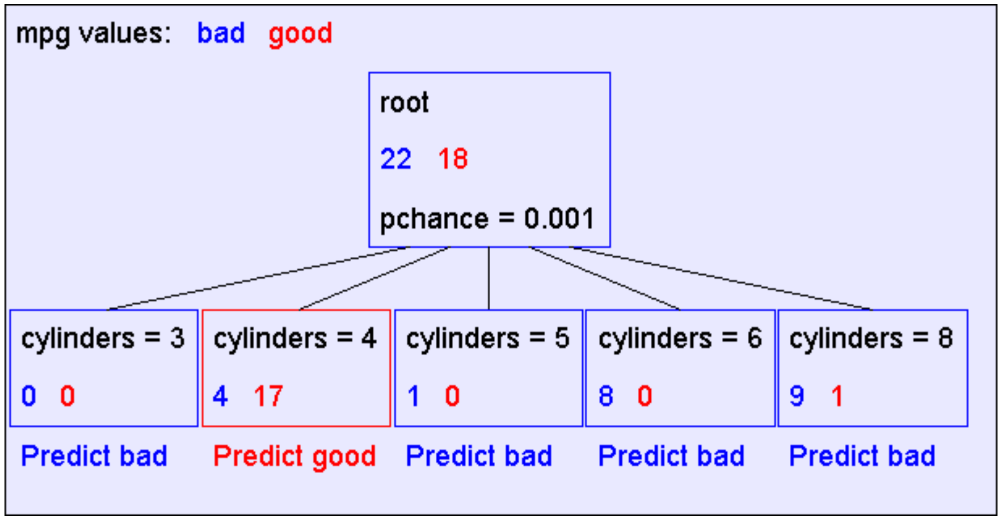
Pruning to avoid overfitting

- 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.

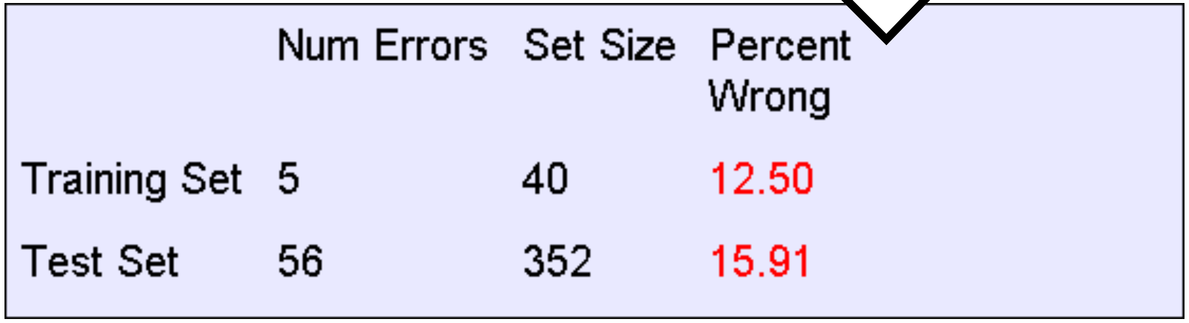
MaxPchance is a magic parameter you must specify to the decision tree, indicating your willingness to risk fitting noise.

Pruning example

- With MaxPchance = 0.1, 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.
- Good news: But with extra work, the best MaxPchance value can be estimated automatically by a technique called cross-validation.

Pruning using Chi Square Test

- Assume: the samples are a good random sample of the population it represents
- Is "Gender" what you can use to predict an undergrad's preference of his/her footwear?
- Null hypothesis "Gender and Footwear Preference have no relationship"

	Sandals	Sneakers	Leather shoes	Boots	Other	Total
Male	6	17	13	9	5	50
Female	13	5	7	16	9	50
Total	19	22	20	25	14	100

Chi Square Test – Compute Expected values

	Sandals	Sneakers	Leather shoes	Boots	Other	Total
Male Observed	6	17	13	9	5	50
Male Expected						
Female Observed	13	5	7	16	9	50
Female Expected						
Total	19	22	20	25	14	100

$$\sum_{i=1}^{rowsize} \sum_{j=1}^{colsize} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

Chi Square Test – Compute Expected values

Male/Sandals: ((19 X 50)/100) = 9.5

Male/Sneakers: ((22 X 50)/100) = 11

Male/Leather Shoes: ((20 X 50)/100) = 10

Male/Boots: ((25 X 50)/100) = 12.5

Male/Other: ((14 X 50)/100) = 7

Female/Sandals: ((19 X 50)/100) = 9.5

Female/Sneakers: ((22 X 50)/100) = 11

Female/Leather Shoes: ((20 X 50)/100) = 10

Female/Boots: ((25 X 50)/100) = 12.5

Female/Other: ((14 X 50)/100) = 7

	Sandals	Sneakers	Leather shoes	Boots	Other	Total
Male Observed	6	17	13	9	5	50
Male Expected	9.5	11	10	12.5	7	
Female Observed	13	5	7	16	9	50
Female Expected	9.5	11	10	12.5	7	
Total	19	22	20	25	14	100

Compute Chi Square Value

$$\sum_{i=1}^{rowsize}\sum_{j=1}^{colsize}\frac{(O_{ij}-E_{ij})^2}{E_{ij}}$$

Total = 14.026

Male/Sandals: $((6 - 9.5)^2/9.5) = 1.289$

Male/Leather Shoes: $((13 - 10)^2/10) = 0.900$

Male/Other: $((5 - 7)^2/7) = 0.571$

Female/Sneakers: $((5 - 11)^2/11) = 3.273$

Female/Boots: $((16 - 12.5)^2/12.5) = 0.980$

Male/Sneakers: $((17 - 11)^2/11) = 3.273$

Male/Boots: $((9-12.5)^2/12.5) = 0.980$

Female/Sandals: $((13-9.5)^2/9.5) = 1.289$

Female/Leather Shoes: $((7-10)^2/10) = 0.900$

Female/Other: $((9 - 7)^2/7) = 0.571$

	Sandals	Sneakers	Leather shoes	Boots	Other	Total
Male Observed	6	17	13	9	5	50
Male Expected	9.5	11	10	12.5	7	
Female Observed	13	5	7	16	9	50
Female Expected	9.5	11	10	12.5	7	
Total	19	22	20	25	14	100

Chi Square Test Computation

- What odds are we willing to accept that we are wrong in generalizing from the results in our sample to the population it represents? ➔ confidence 5%
- Degree of Freedom of this problem

  = (# of rows - 1)(# of cols - 1) = (2-1)(5-1)=4
- From Chi Square table of statistics book, with p=0.05, r=4, critical value is 9.49,
  - if Chi square value is less than 9.49, accept the null hypothesis that there is no statistically significant relationship between gender and shoe preference
- In this case, Chi square value is 14.026 > 9.49, so we can reject the null hypothesis and conclude: male and female undergraduates of the Univ. differ in their footwear preferences.

Cross-validation

- To minimize the effect of dependency on choice of training and test data, measure performance of algorithm using N-fold cross-validation
- Method:

```
Partition data into N disjoint sets S= {$S_1$, $S_2$, … $S_N$}
i=1
    loop N times:
        Let training set be (S – $S_i$), and
            test set be $S_i$,
        Learn the classifier based on the current training set,
        Test the performance of the classifier on the current test set
        Record the predication accuracy
        i = i + 1;
    end  loop
    Compute the average predication accuracy for the N runs
```

- Ten fold cross validation   (N=10)

Gain Ratio

- Problem with Information Gain
- Gain Ratio takes into account the number of branches generated from the split test:

$$\text{split info (X)} = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times \log_2\left(\frac{|T_i|}{|T|}\right)$$

$$Gain\ Ratio(X) = \frac{gain(X)}{\text{split info}(X)}$$

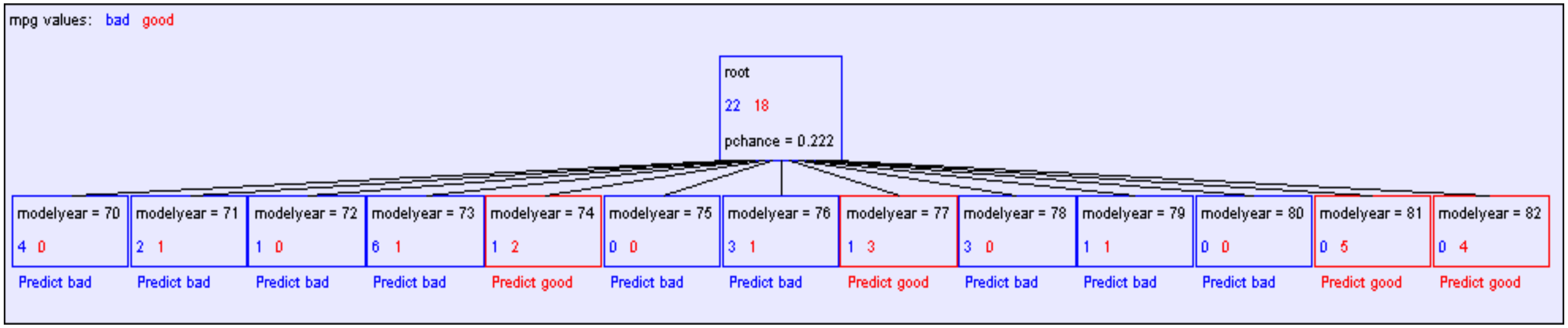
Real-Valued inputs

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

mpg	cylinders	displacement	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”



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.

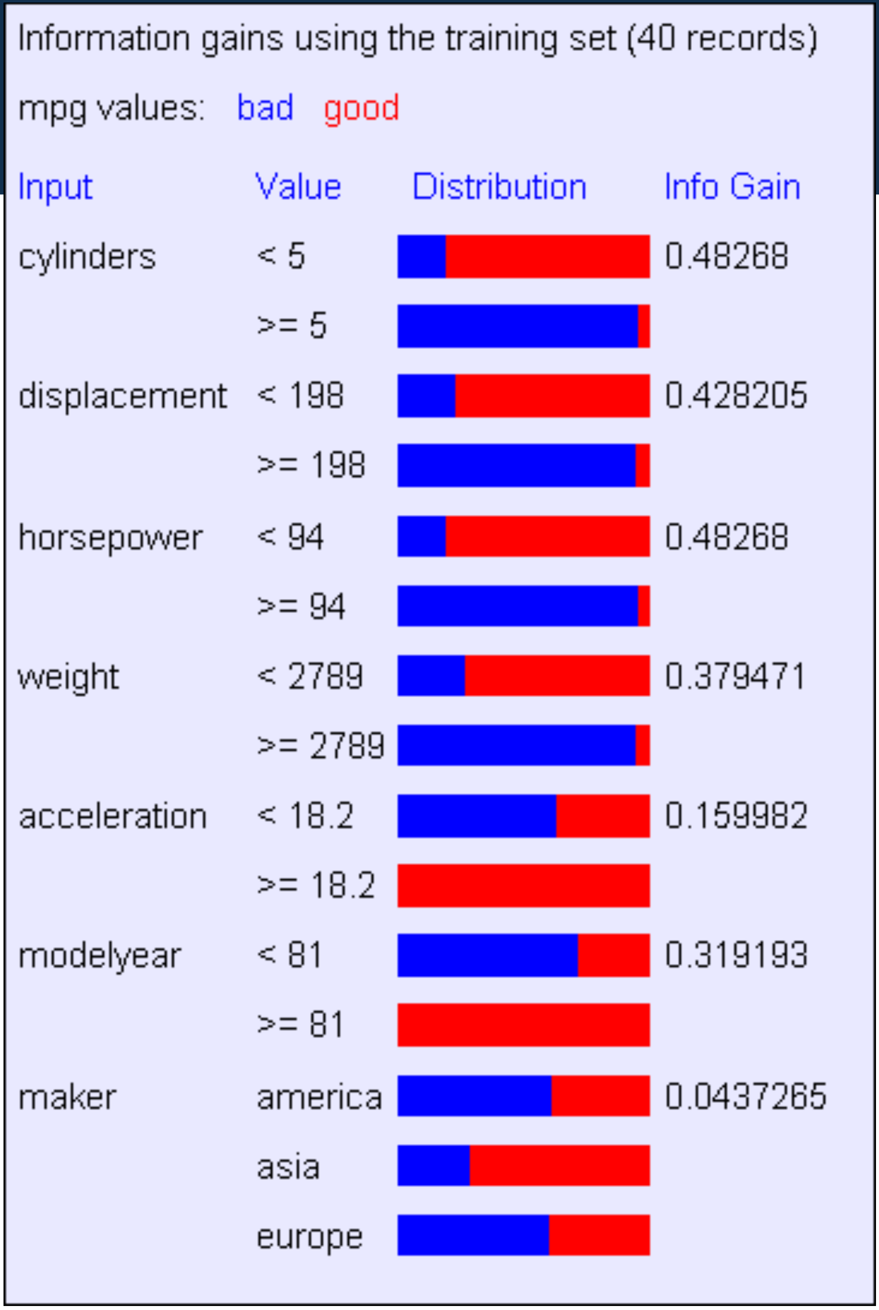
A better idea: thresholded splits

- Suppose X is real valued.
- Define $IG(Y|X{:}t)$ as $H(Y) - H(Y|X{:}t)$
- Define $H(Y|X{:}t) =$

  $$H(Y|X < t)\,P(X < t) + H(Y|X >= t)\,P(X >= t)$$

  - $IG(Y|X{:}t)$ 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

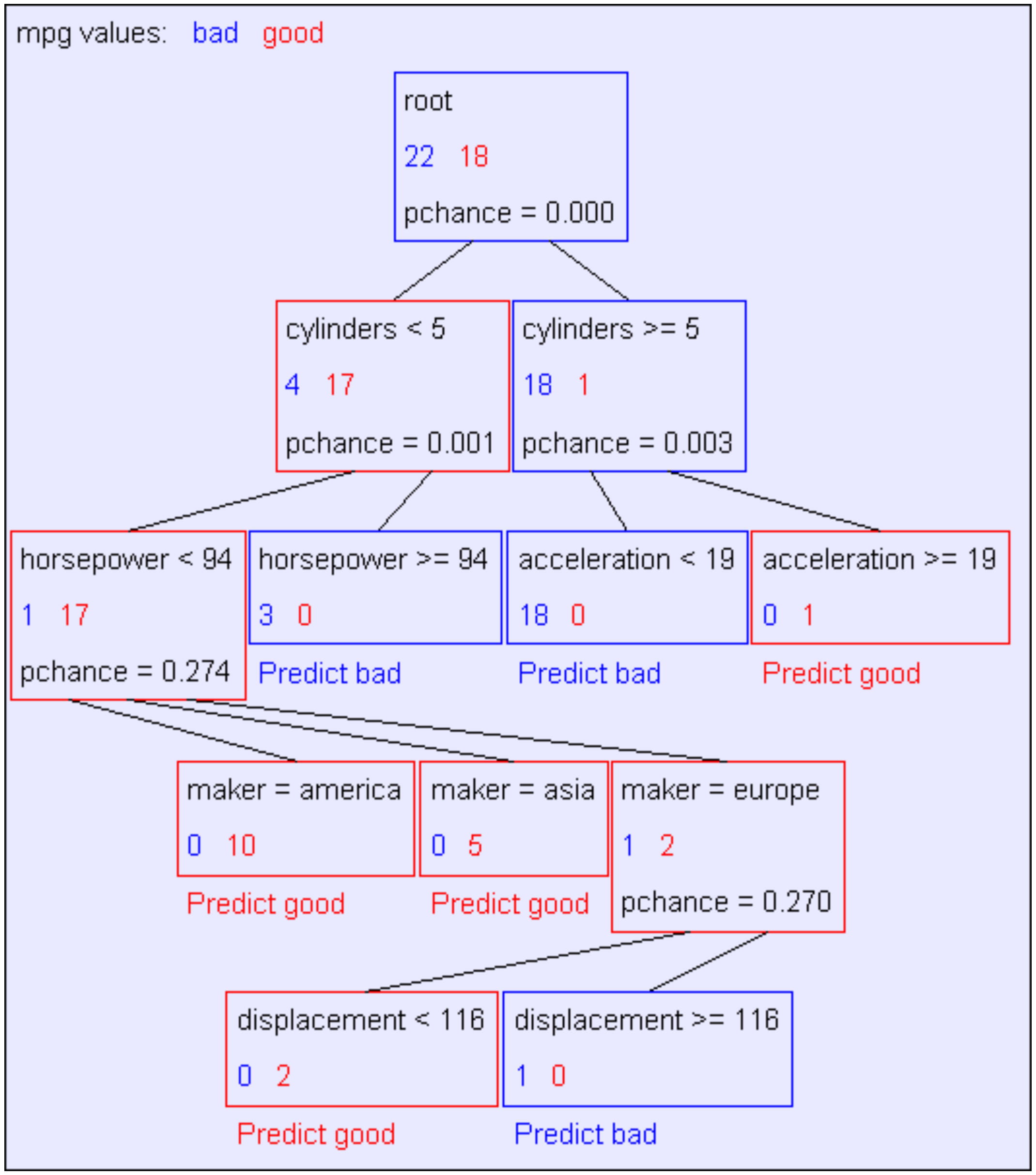
- Example with MPG



Information gains using the training set (40 records)

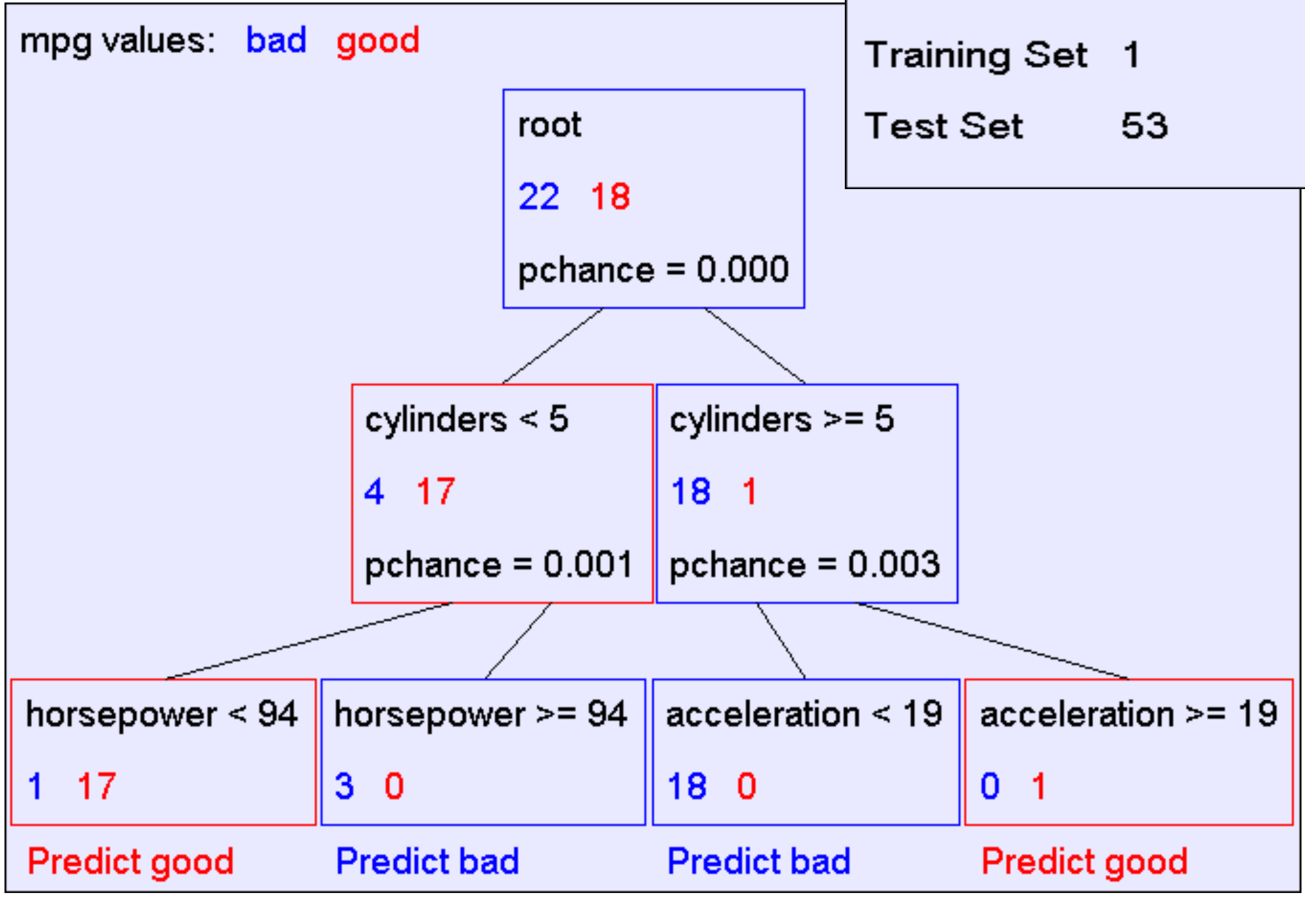
mpg values: bad good

Input	Value	Distribution	Info Gain
cylinders	< 5	🟦🟥	0.48268
	>= 5	🟦🟥	
displacement	< 198	🟦🟥	0.428205
	>= 198	🟦🟥	
horsepower	< 94	🟦🟥	0.48268
	>= 94	🟦🟥	
weight	< 2789	🟦🟥	0.379471
	>= 2789	🟦🟥	
acceleration	< 18.2	🟦🟥	0.159982
	>= 18.2	🟥	
modelyear	< 81	🟦🟥	0.319193
	>= 81	🟥	
maker	america	🟦🟥	0.0437265
	asia	🟦🟥	
	europe	🟦🟥	



Unpruned tree using reals

Pruned tree using reals



	Num Errors	Set Size	Percent Wrong
Training Set	1	40	2.50
Test Set	53	352	15.06

Additional Considerations

- How to deal with Missing Data?
- Other tree pruning methods
  - Cost-complexity pruning
  - Reduced-error pruning

In Summary

- Decision trees are the single most popular data mining tool
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap
- It's possible to get in trouble with overfitting
- They do classification: predict a categorical output from categorical and/or real inputs