Chapter 7: Classification and Prediction

Chapter 7. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian Classification
- Classification by backpropagation
- Classification based on concepts from association rule mining
- Other Classification Methods
- Prediction
- Classification accuracy
- Summary

Classification vs. 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

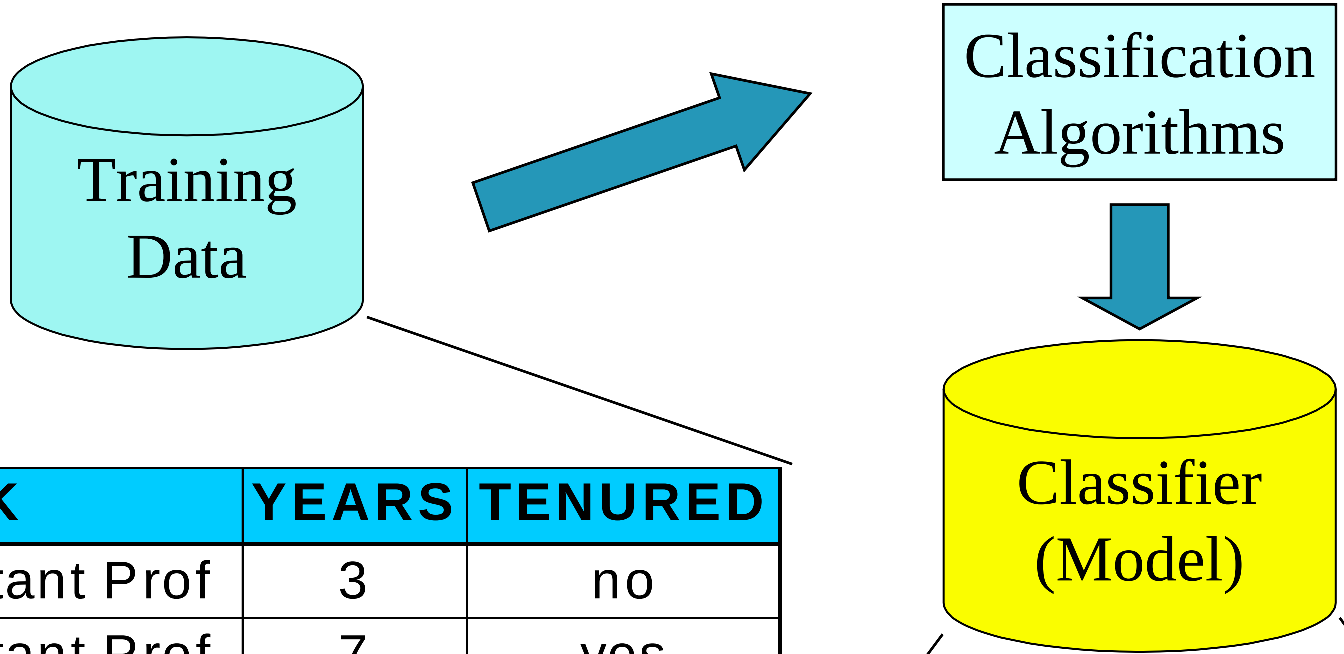
Example Applications (continued)

- 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

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 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

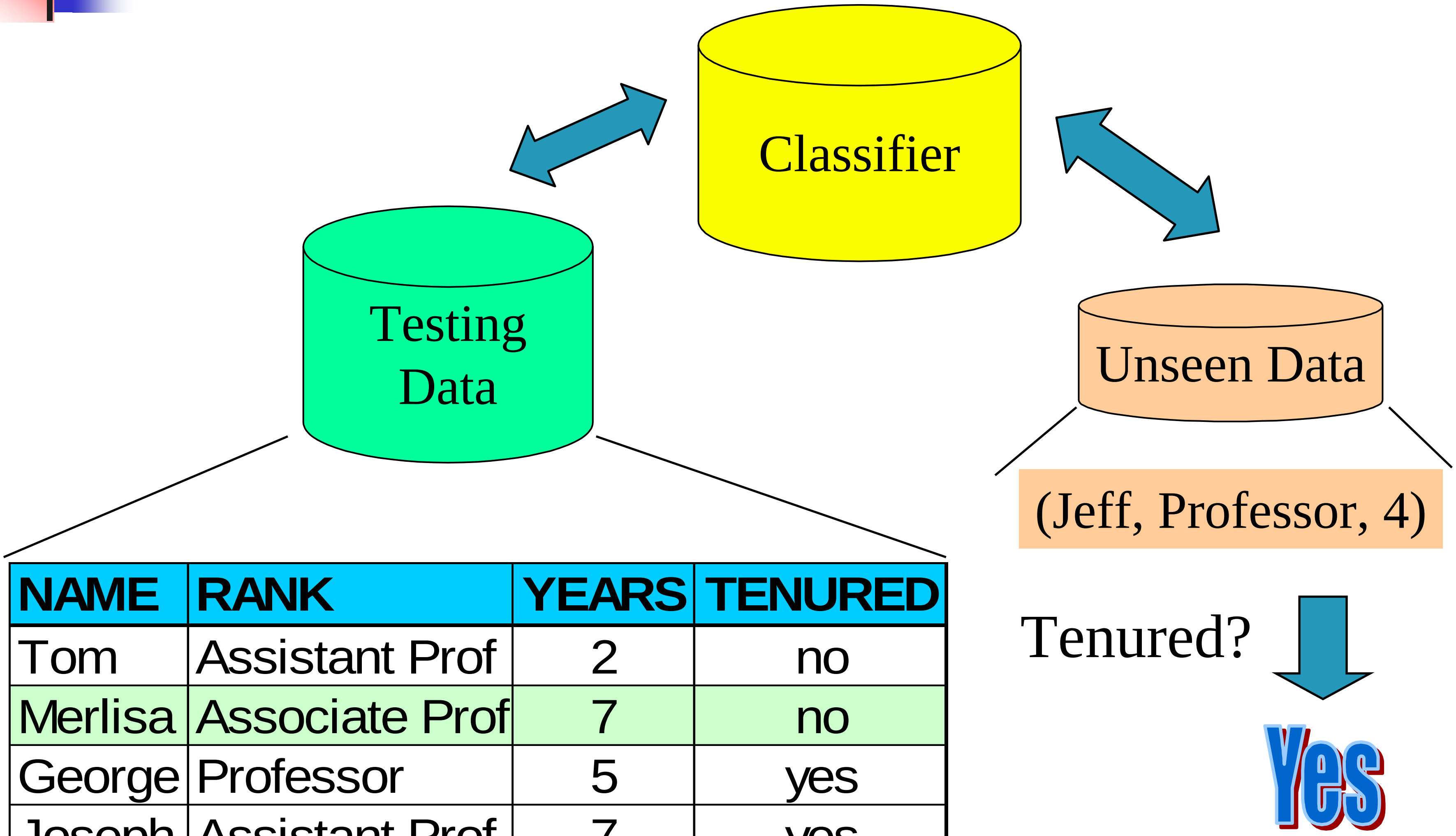
Classification Process (1): Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Classification Process (2): Use the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Chapter 7. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian Classification
- Classification by backpropagation
- Classification based on concepts from association rule mining
- Other Classification Methods
- Prediction
- Classification accuracy
- Summary

Issues regarding classification and prediction (1): Data Preparation

- Data cleaning
  - Preprocess data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes
- Data transformation
  - Generalize and/or normalize data

Issues regarding classification and prediction (2): Evaluating Classification Methods

- Predictive accuracy
- Speed and scalability
  - time to construct the model
  - time to use the model
- Robustness
  - handling noise and missing values
- Scalability
  - efficiency in disk-resident databases
- Interpretability:
  - understanding and insight provided by the model
- Goodness of rules
  - decision tree size
  - compactness of classification rules

Chapter 7. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian Classification
- Classification by backpropagation
- Classification based on concepts from association rule mining
- Other Classification Methods
- Prediction
- Classification accuracy
- Summary

Classification by Decision Tree Induction

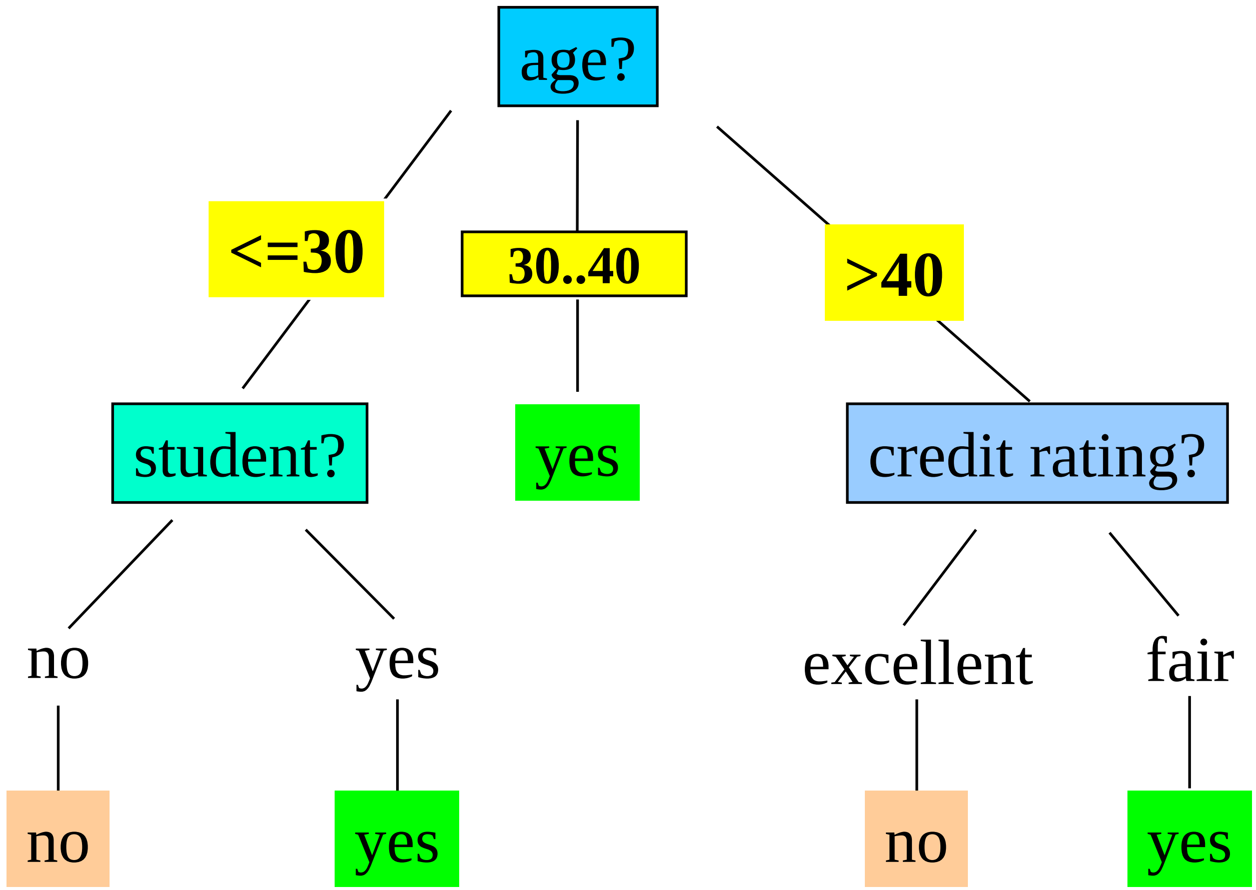
- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution
- Decision tree generation consists of two phases
  - Tree construction
    - At start, all the 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

Training Dataset

This follows an example from Quinlan's ID3

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

Output: A Decision Tree for "*buys_computer*"



Algorithm for Decision Tree Induction(ID3)

- 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

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

Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Assume there are two classes, *P* and *N*
  - Let the set of examples *S* contain *p* elements of class *P* and *n* elements of class *N*
  - The amount of information needed to decide if an arbitrary example in *S* belongs to *P* or *N* is defined as

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Information Gain in Decision Tree Induction

- Assume that using attribute A a set S will be partitioned into sets $\{S_1, S_2, \ldots, S_v\}$
  - If S_i contains p_i examples of P and n_i examples of N, the entropy, or the expected information needed to classify objects in all subtrees S_i is

$$E(A) = \sum_{i=1}^{\nu} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- The encoding information that would be gained by branching on A

$$Gain(A) = I(p, n) - E(A)$$

Attribute Selection by Information Gain Computation

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"
- I(p, n) = I(9, 5) =0.940
- Compute the entropy for *age*:

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
30…40	4	0	0
>40	3	2	0.971

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.69$$

Hence

$$Gain(age) = I(p,n) - E(age)$$

Similarly

$$Gain(income) = 0.029$$

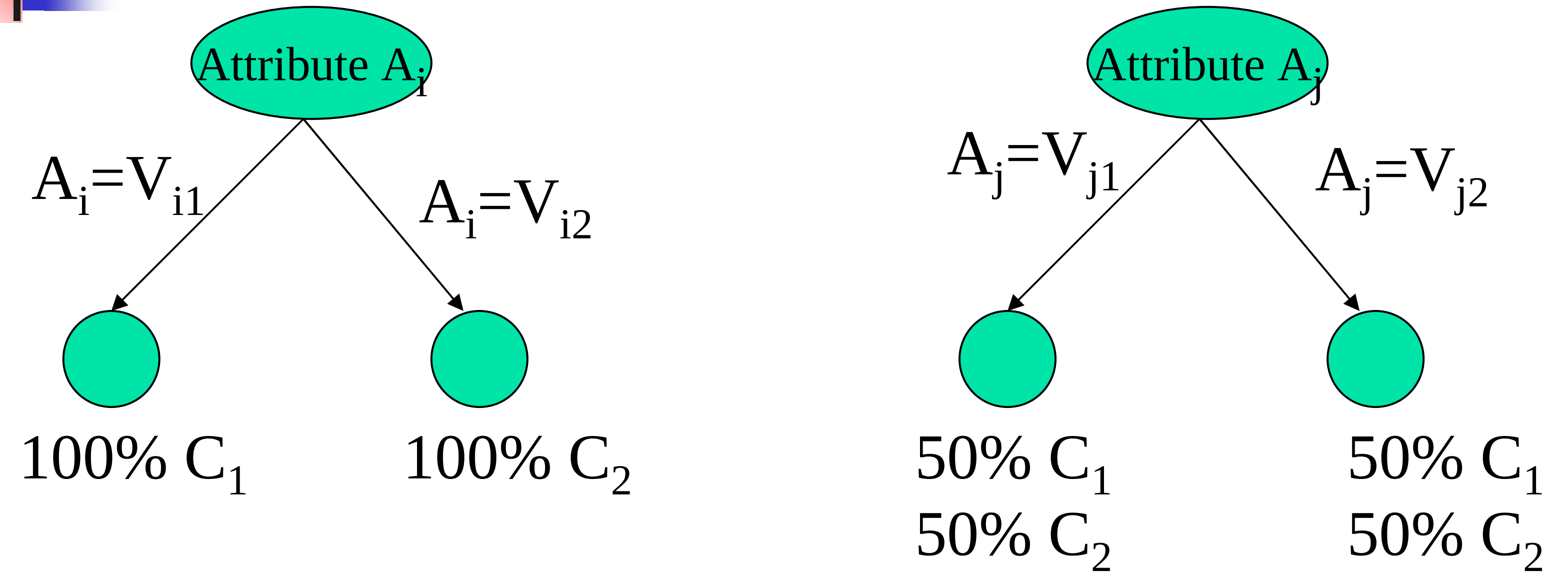
$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

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



Which attribute is better ?

Entropy - General Case

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

$$H(X) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 - \ldots - p_m \log_2 p_m$$
$$= -\sum_{j=1}^{m} p_j \log_2 p_j$$

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

Entropy - General Case

Suppose 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_m

What's the smallest possible number of bits, o[…]ded to transmit a[…]rom X'[…]

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

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

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

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

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)

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

Learning Decision Trees

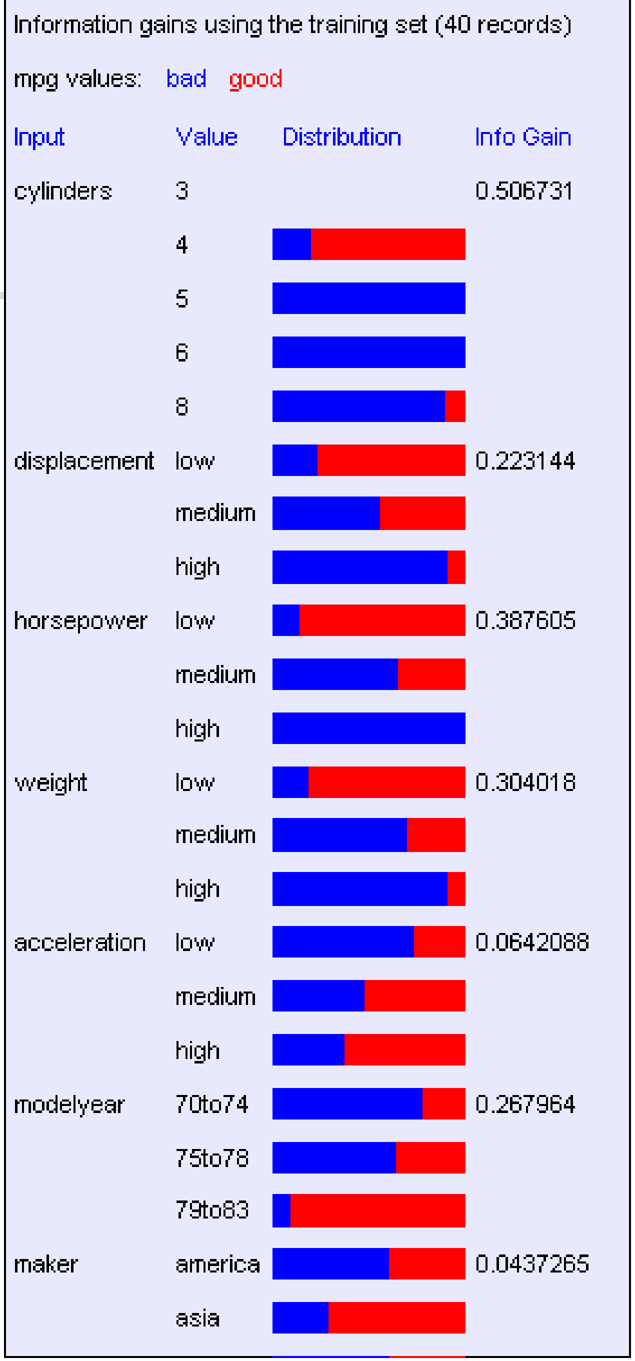
- A Decision Tree is a tree-structured plan of a set of attributes to test in order to predict the output.
- To decide which attribute should be tested first, simply find the one with the highest information gain.
- Then recurse...

A small dataset: Miles Per Gallon

40 Records

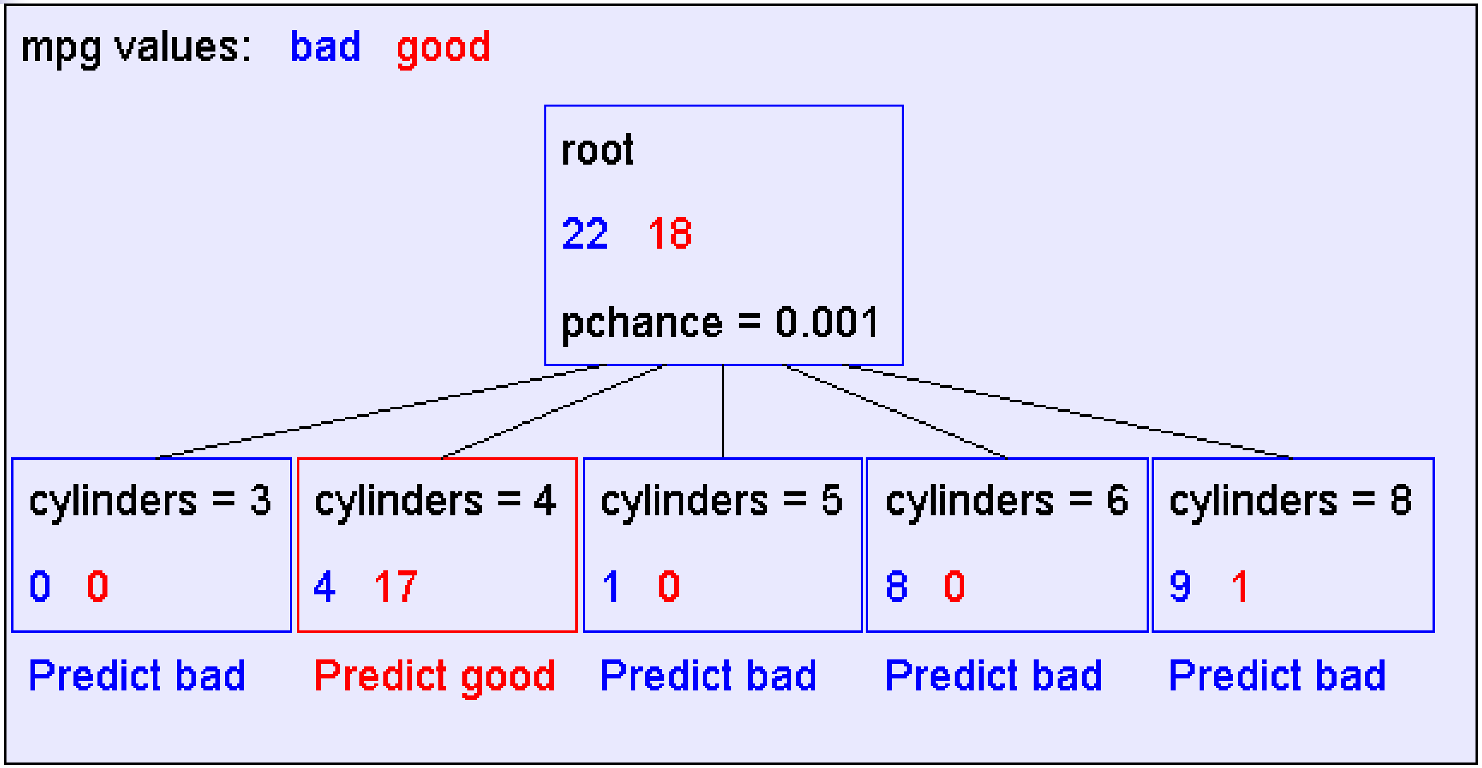
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low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

Suppose we want to predict MPG.

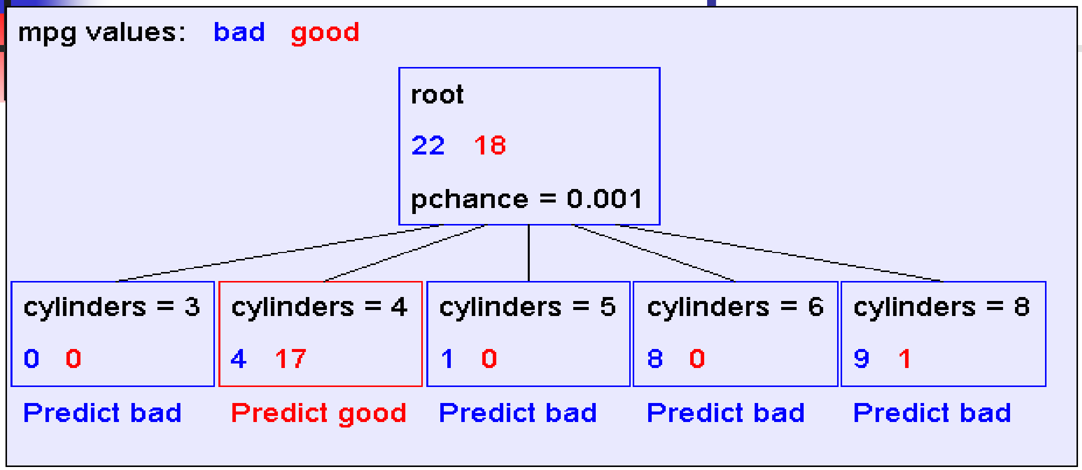


Look at all the information gains...

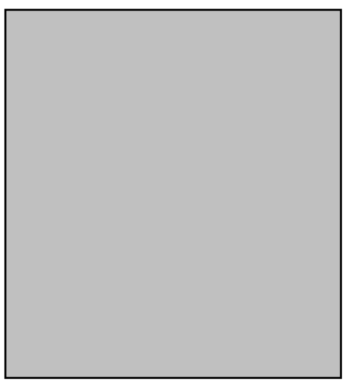
A Decision Stump



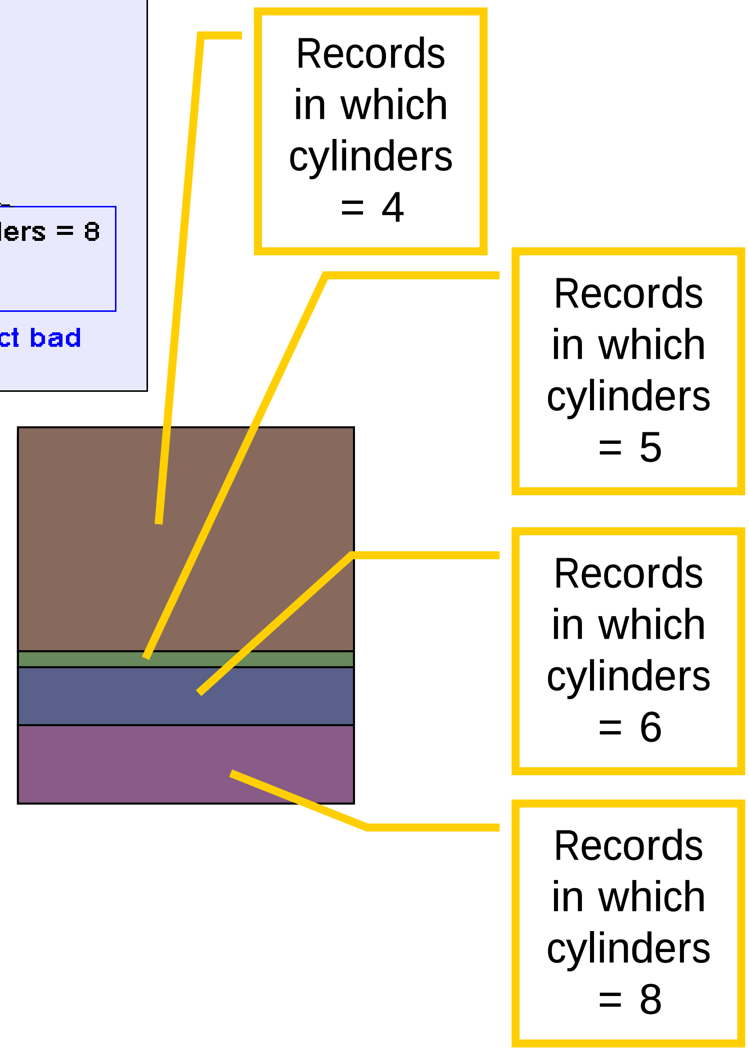
Recursion Step



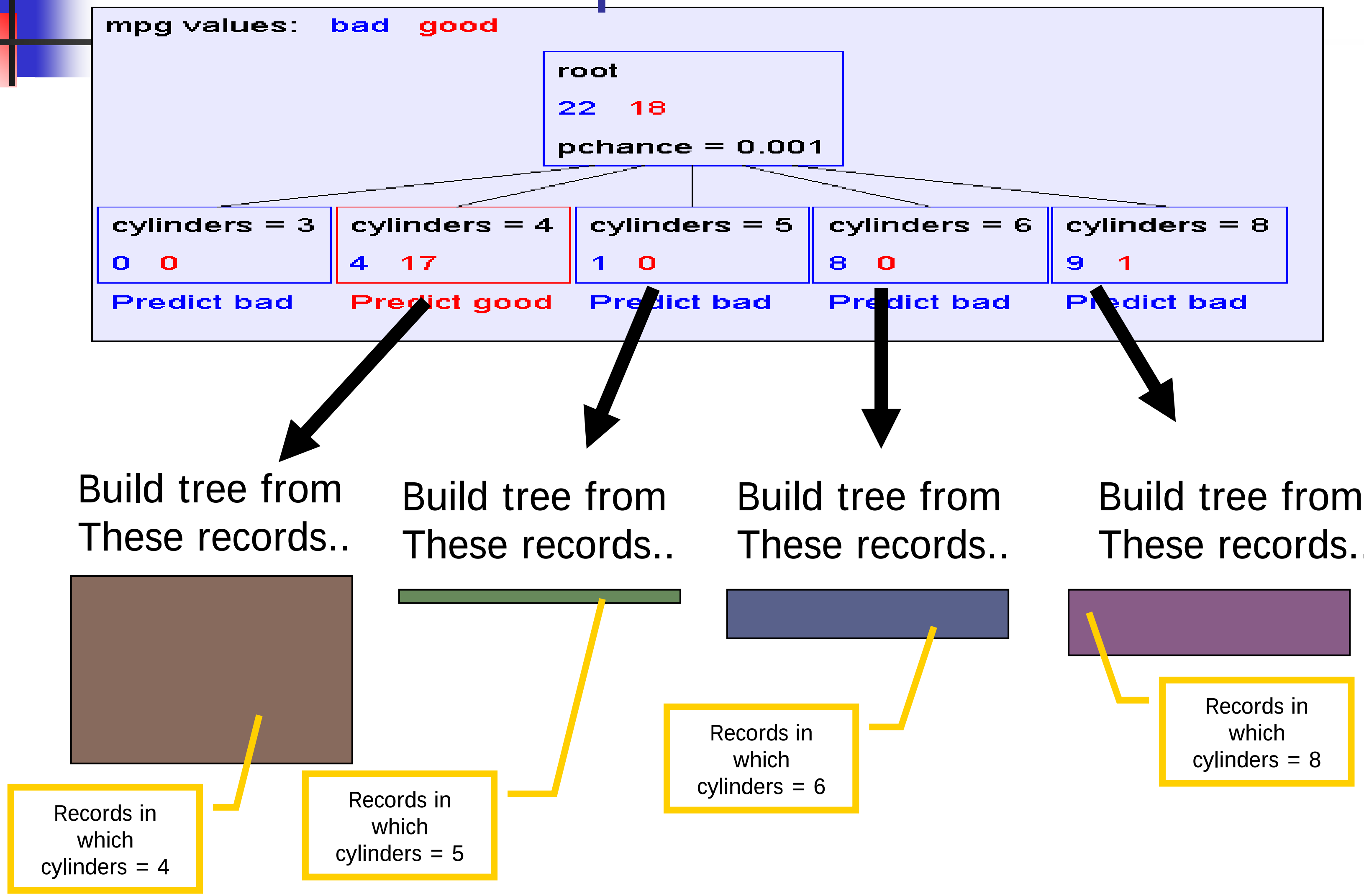
Take the Original Dataset..



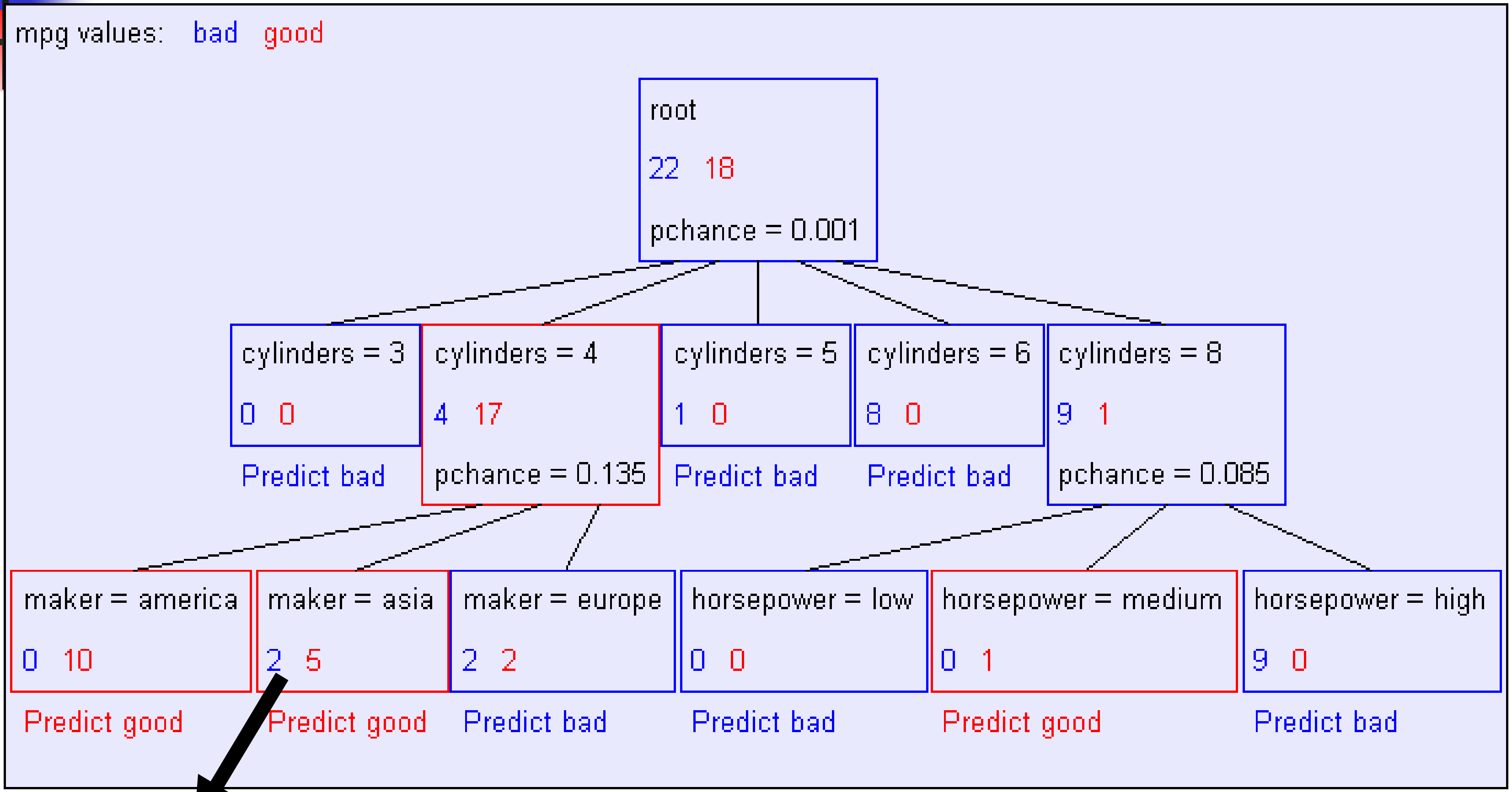
And partition it according to the value of the attribute we split on



Recursion Step



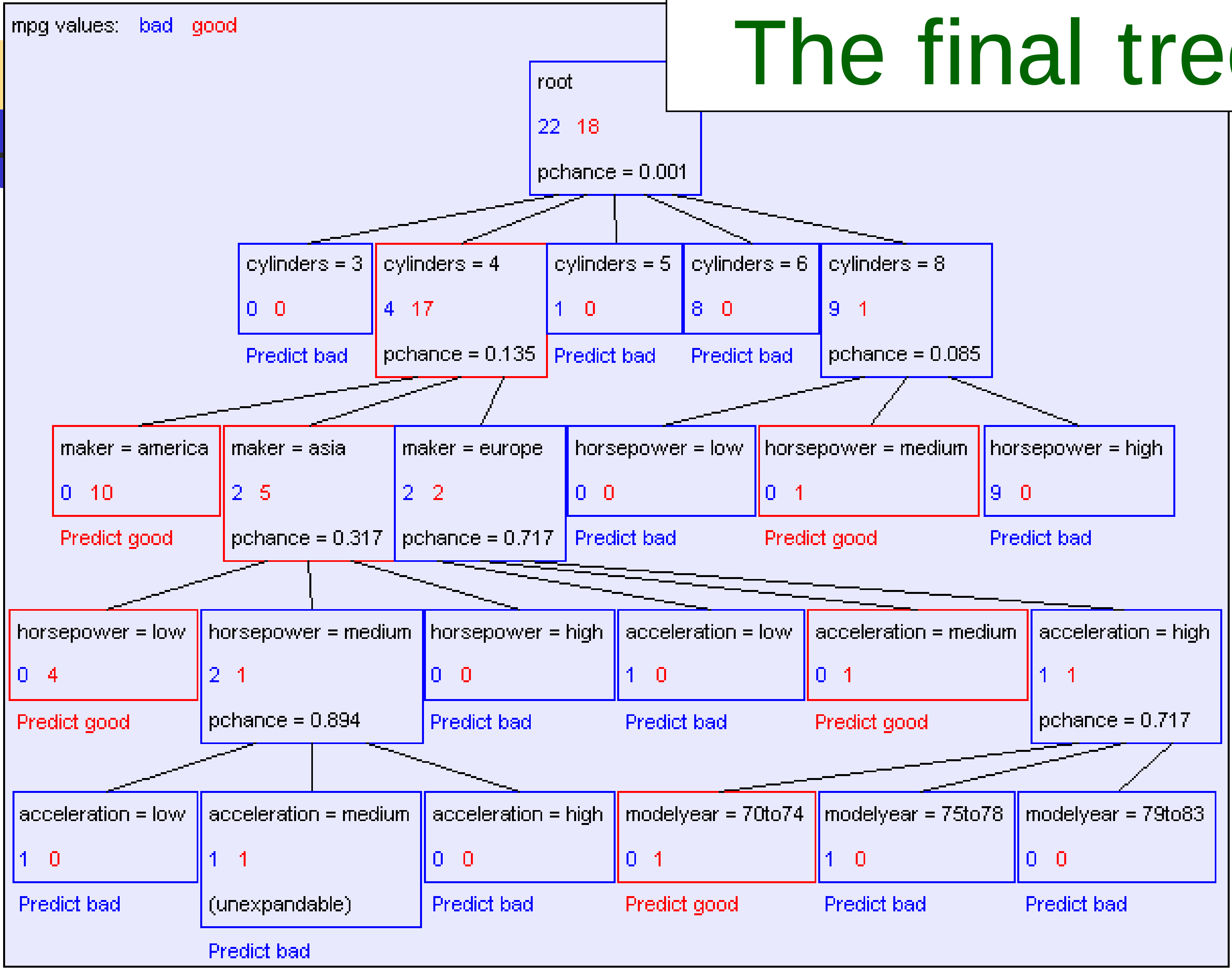
Second level of tree



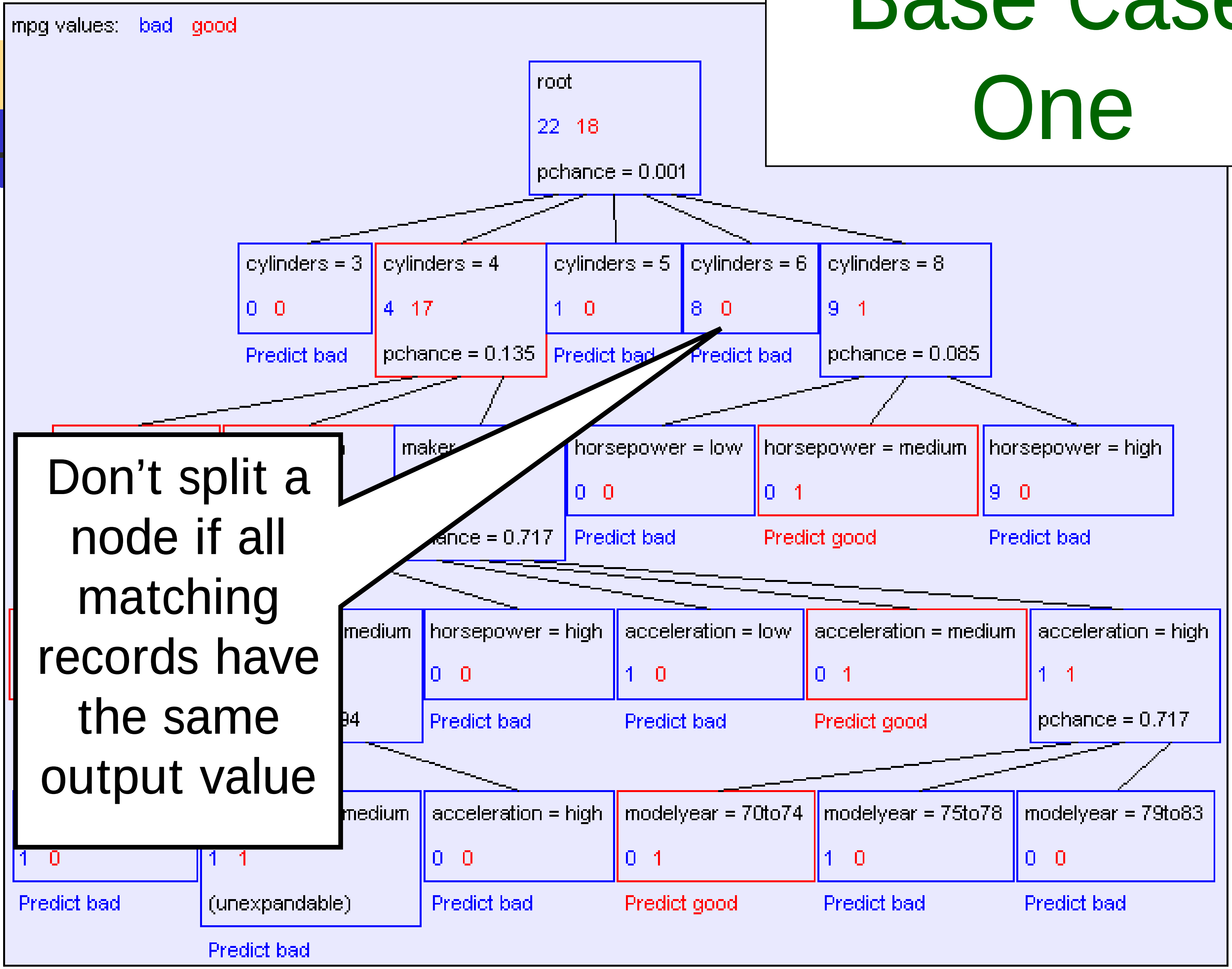
Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

The final tree

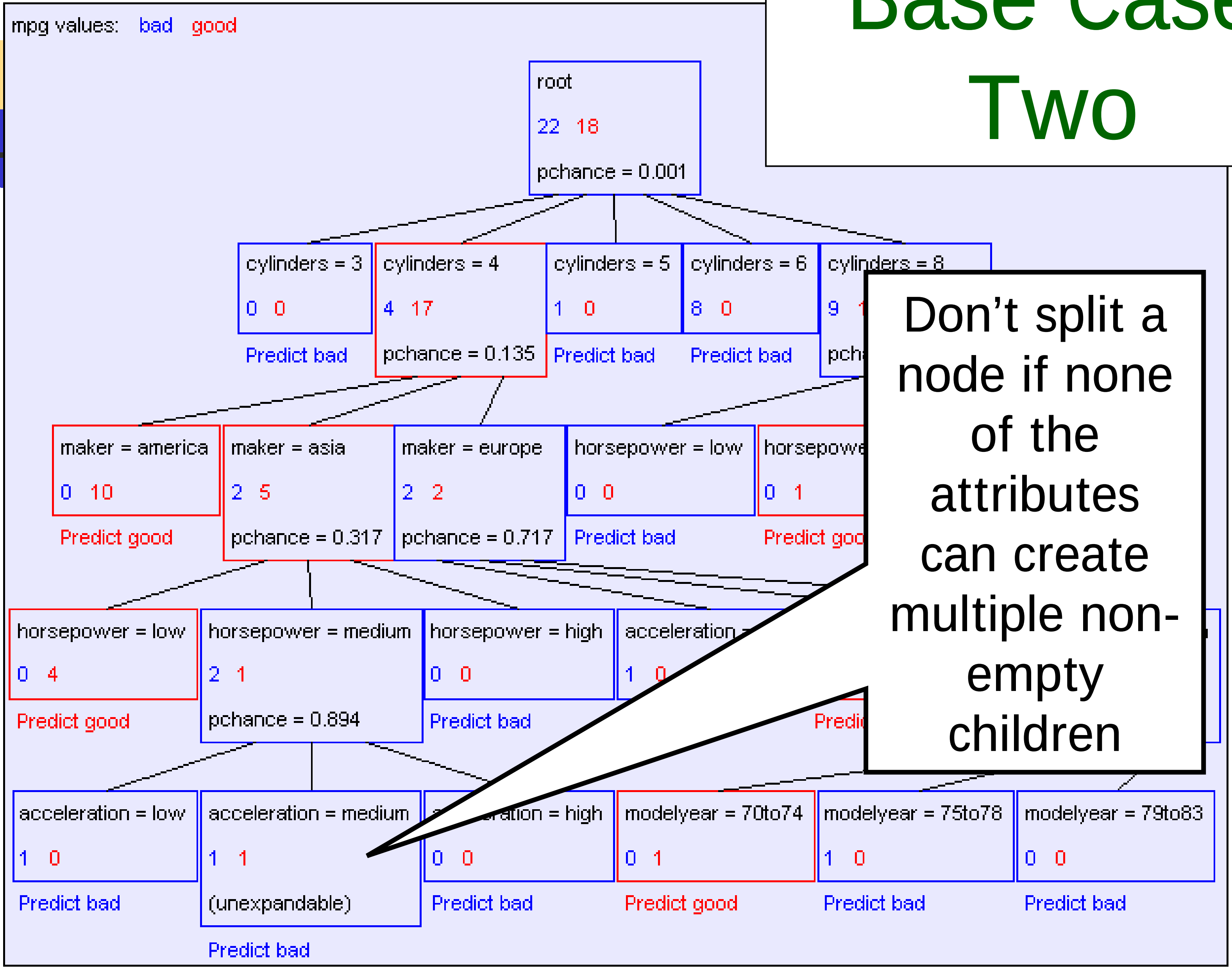


Base Case One



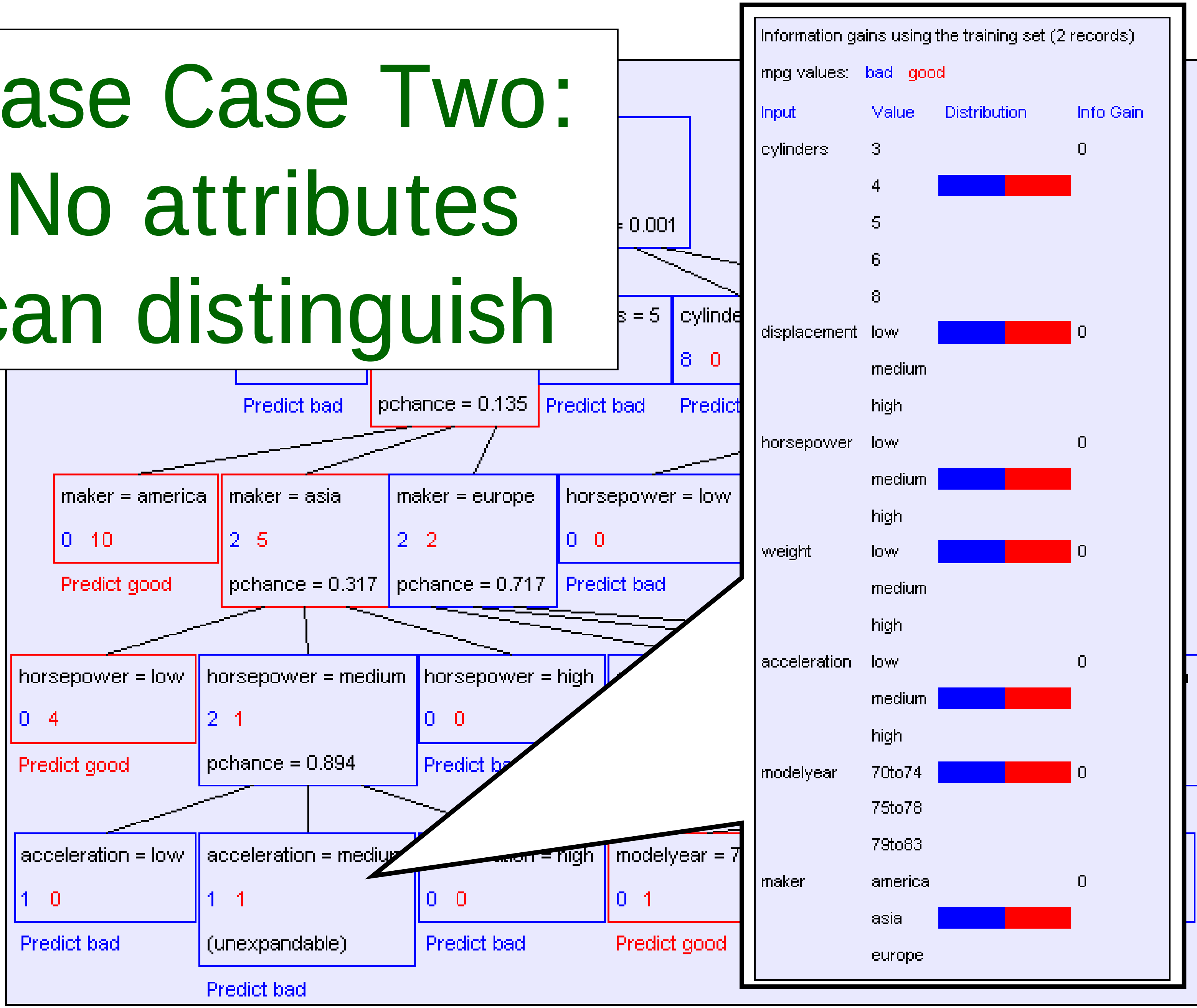
Don't split a node if all matching records have the same output value

Base Case Two



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

Base Case Two: No attributes can distinguish



Base Cases: An idea

- Base Case One: If all records in current data subset have the same output then don’t recurse
- Base Case Two: If all records have exactly the same set of input attributes then don’t recurse

Proposed Base Case 3:

If all attributes have zero information gain then don’t recurse

- *Is this a good idea?*

Practice question

How to build the decision tree for this data?

Data:

Object	color	shape	size	class
1	Red	square	big	like
2	blue	square	big	like
3	red	round	small	don’t like
4	green	square	small	don’t like
5	red	round	big	like
6	green	square	big	don’t like

```
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
```

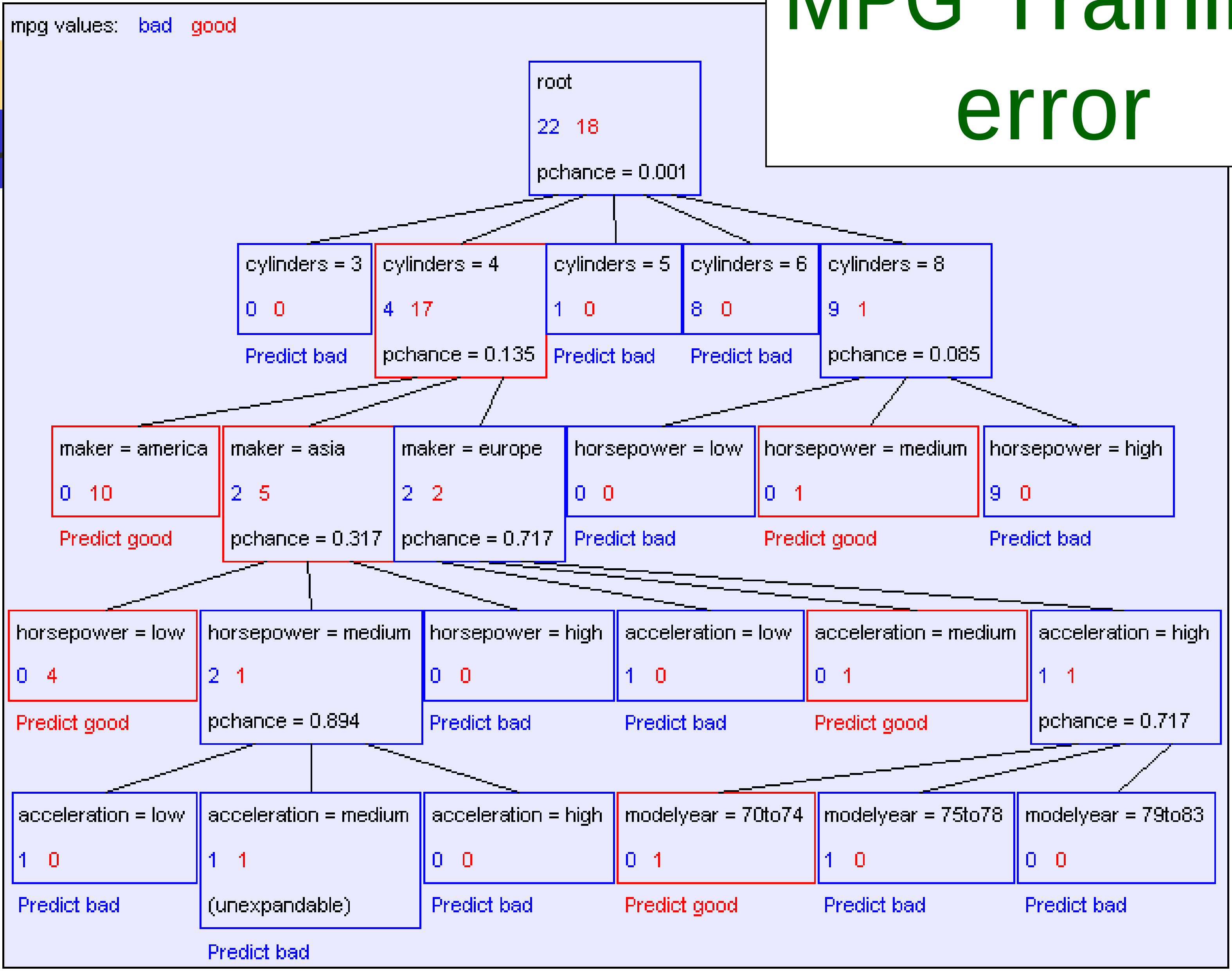
Training Set Error

- For each record, follow the decision tree to see what it would predict

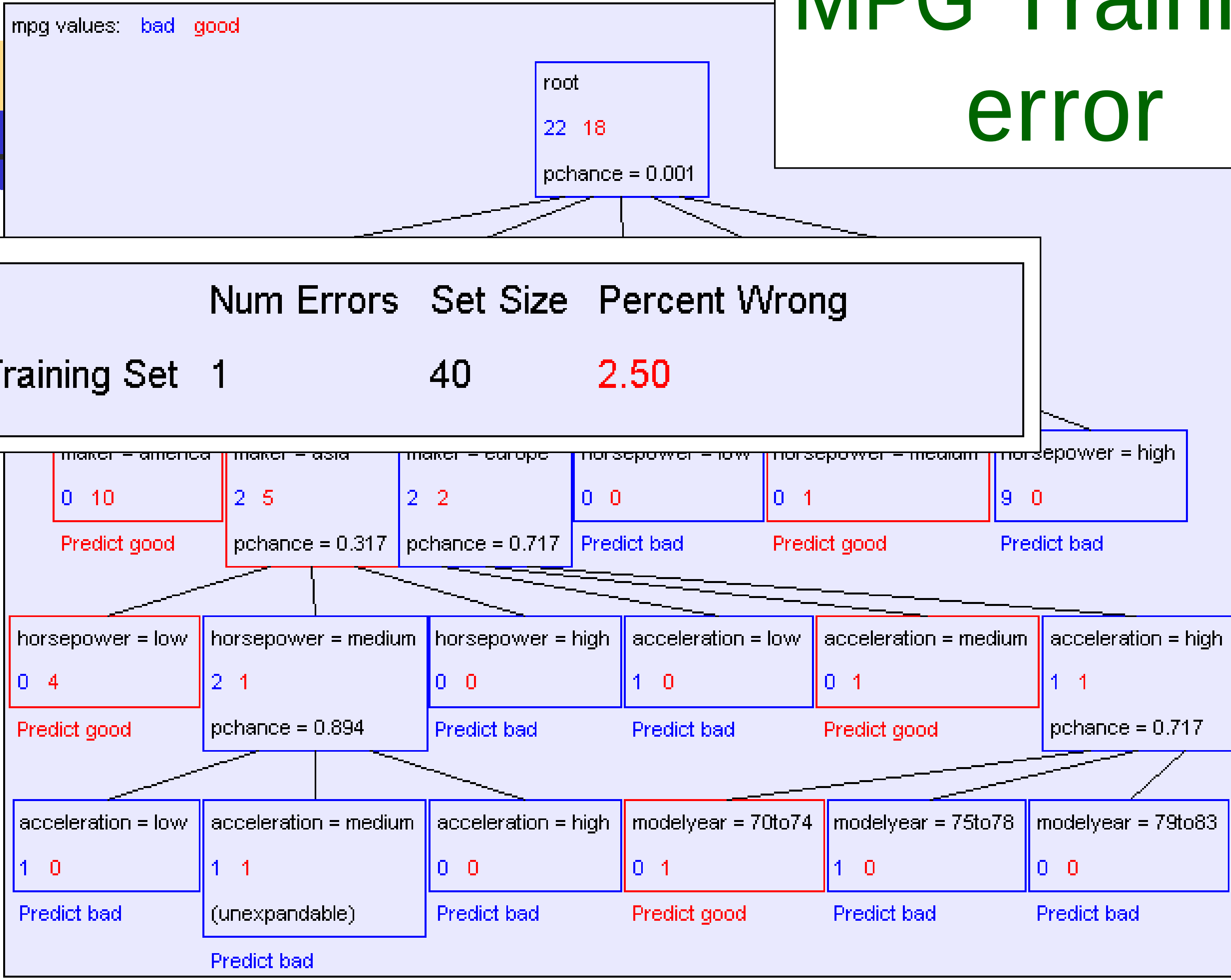
  For what number of records does the decision tree's prediction disagree with the true value in the database?

- This quantity is called the *training set error*. The smaller the better.

MPG Training error



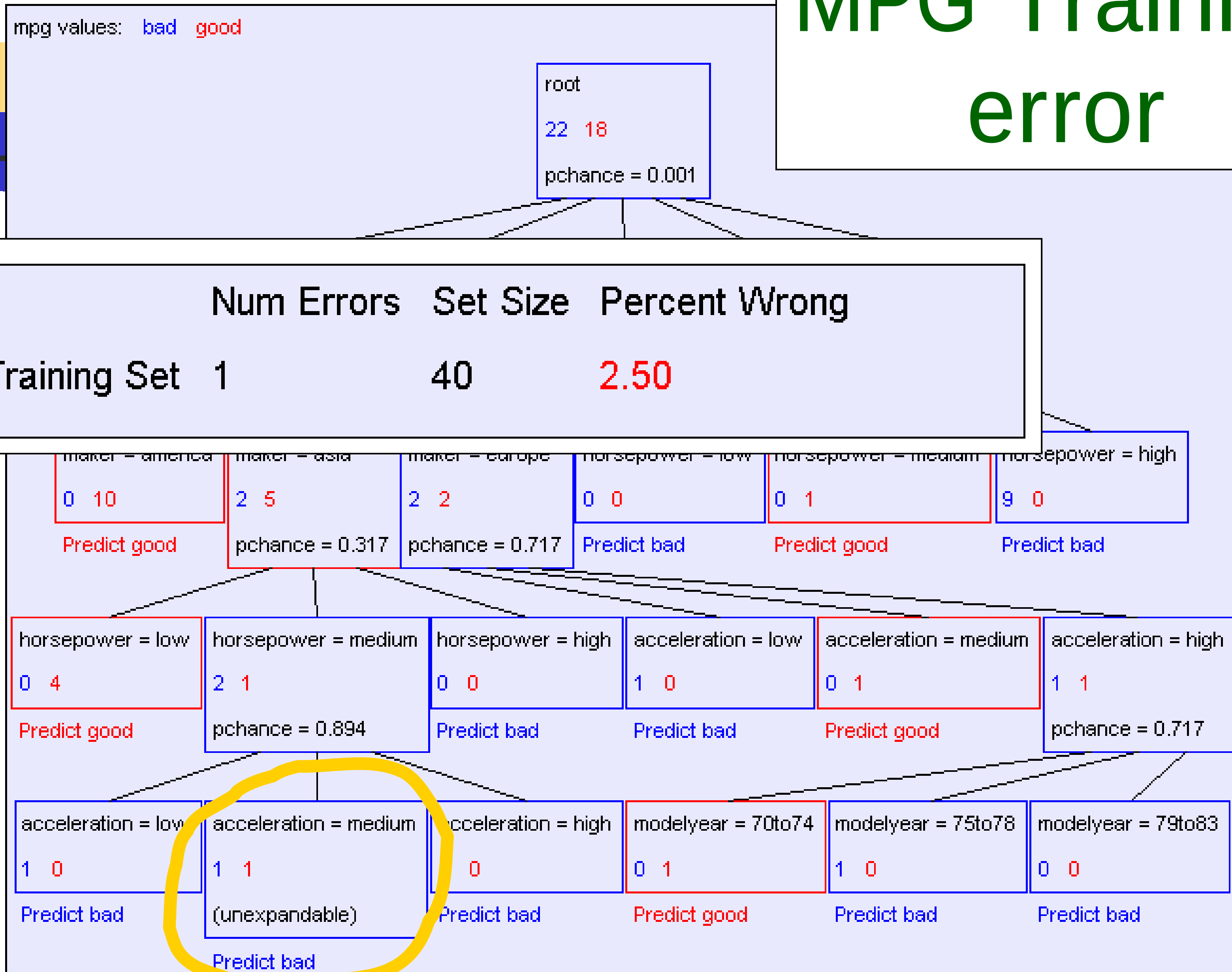
MPG Training error



	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



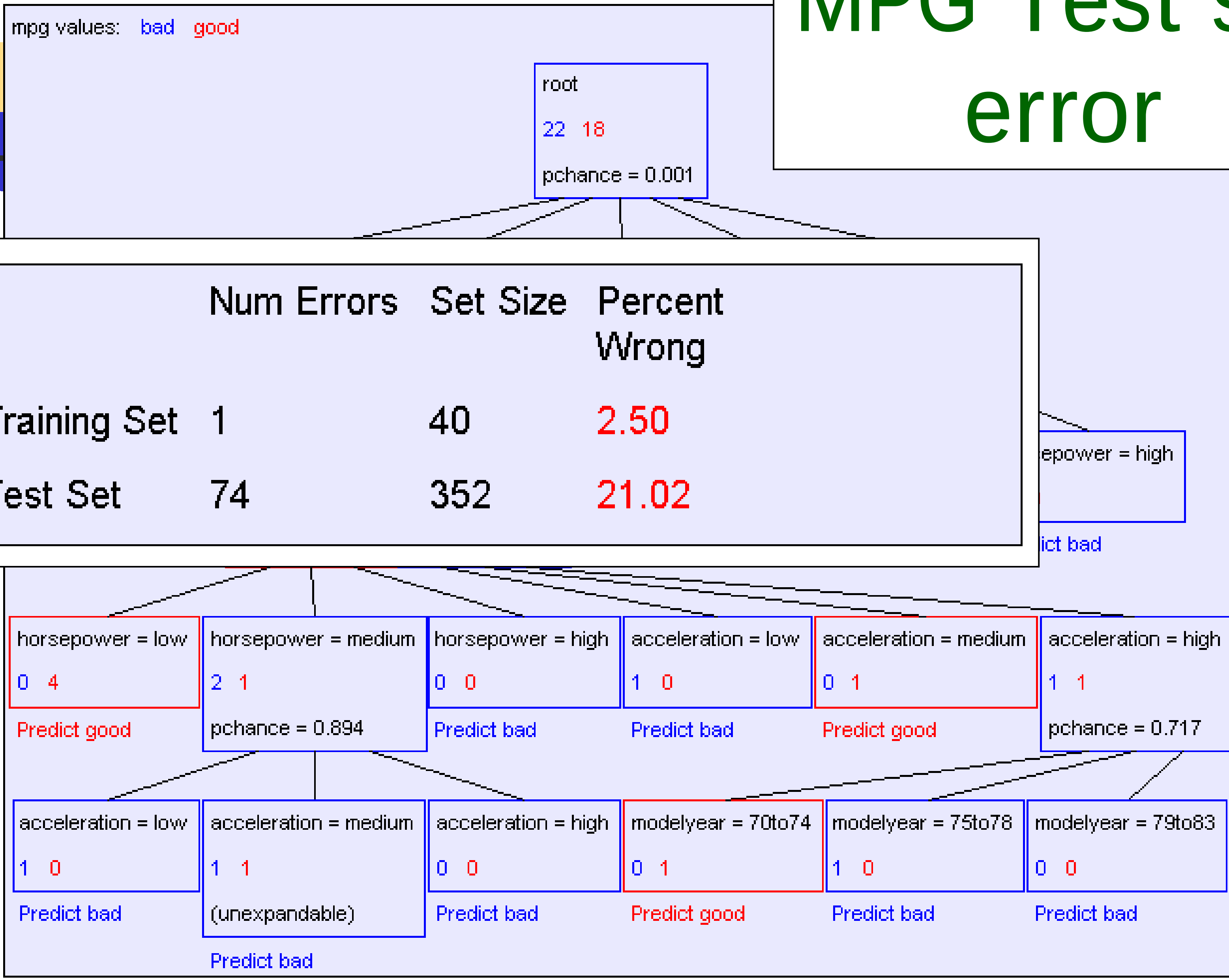
Why are we doing this learning anyway?

- 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.

Test Set Error

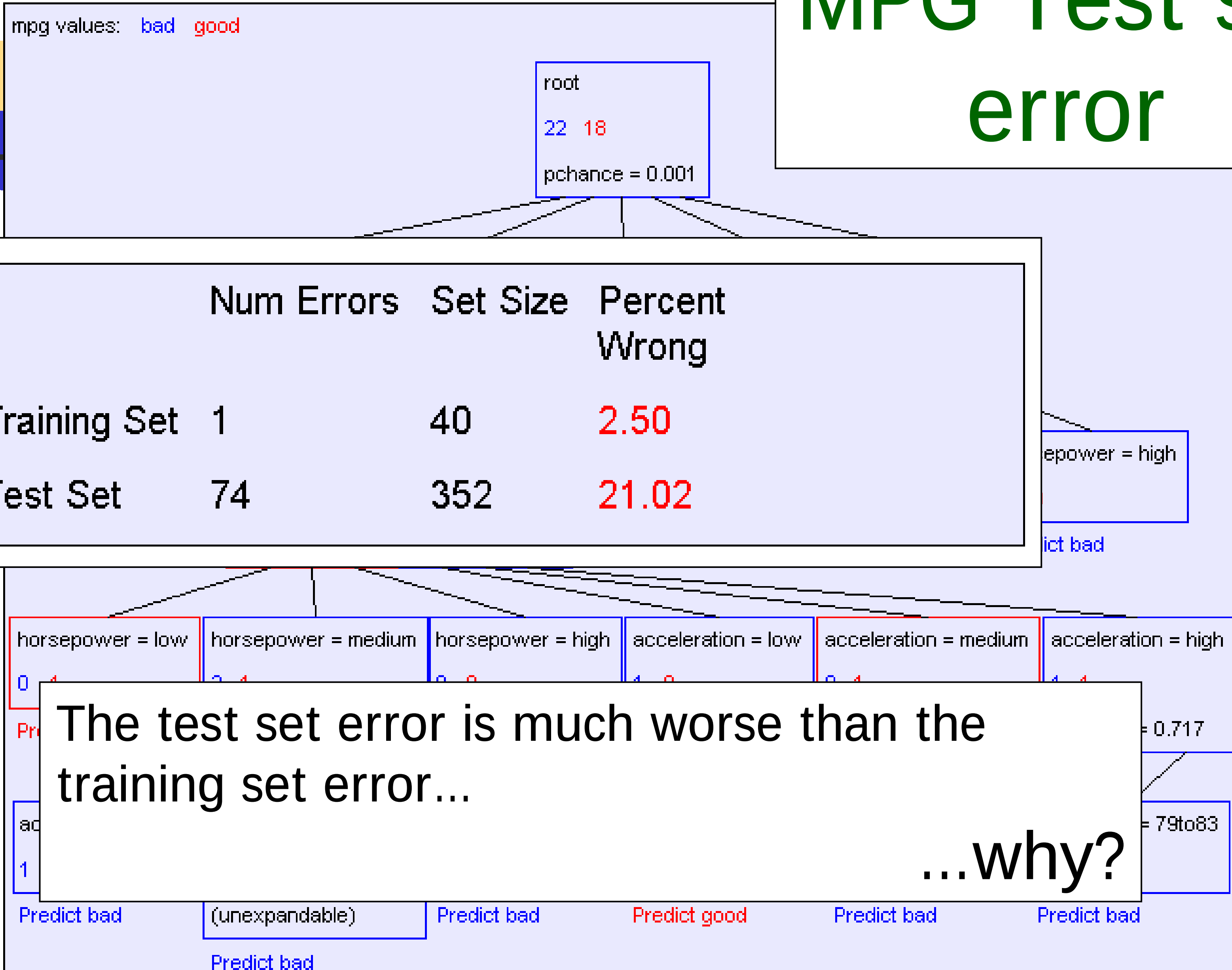
- Suppose we are forward thinking.
- We hide some data away when we learn the decision tree.
- But once learned, we see how well the tree predicts that data.
- This is a good simulation of what happens when we try to predict future data.
- And it is called Test Set Error.

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

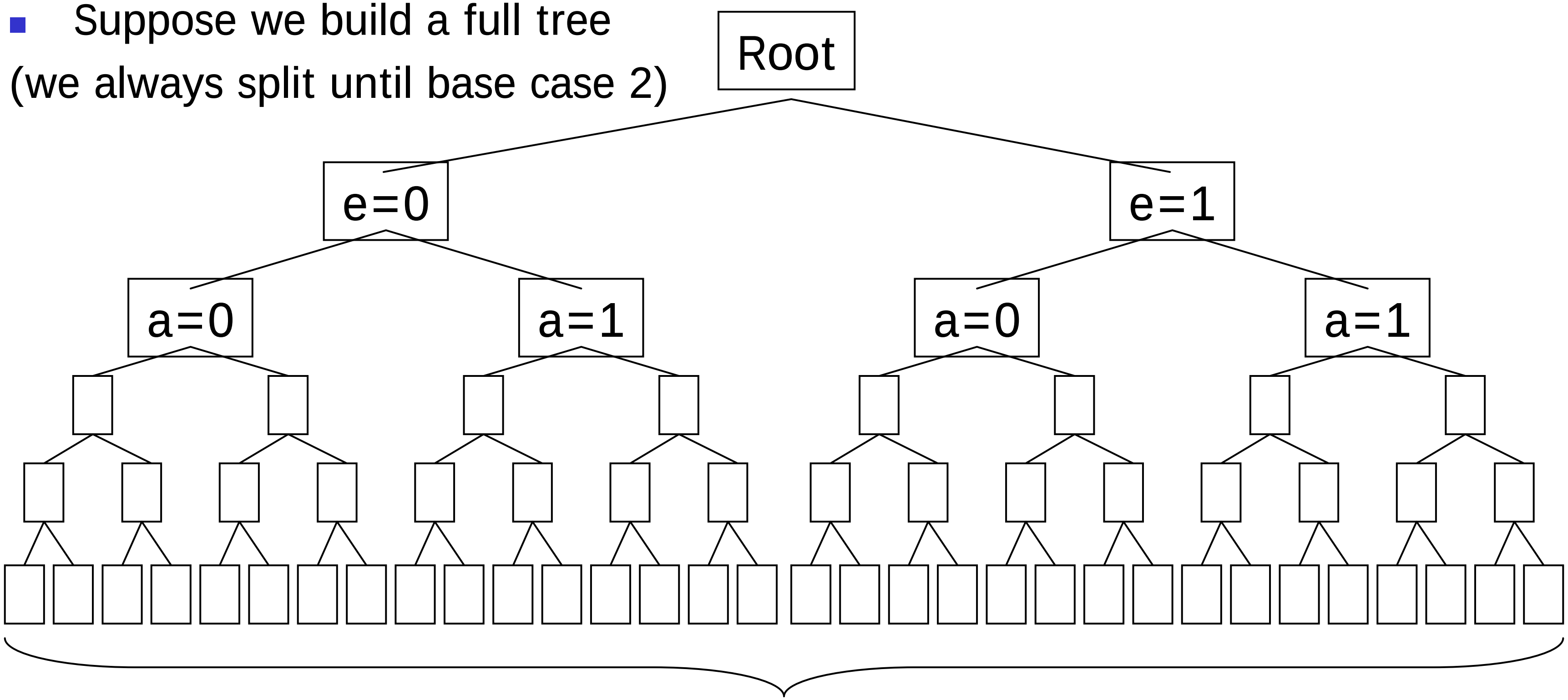
In our artificial example

- 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 testing set.
- Some y's that were uncorrupted in the training set will be corrupted in the test set.

Building a tree with the artificial 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

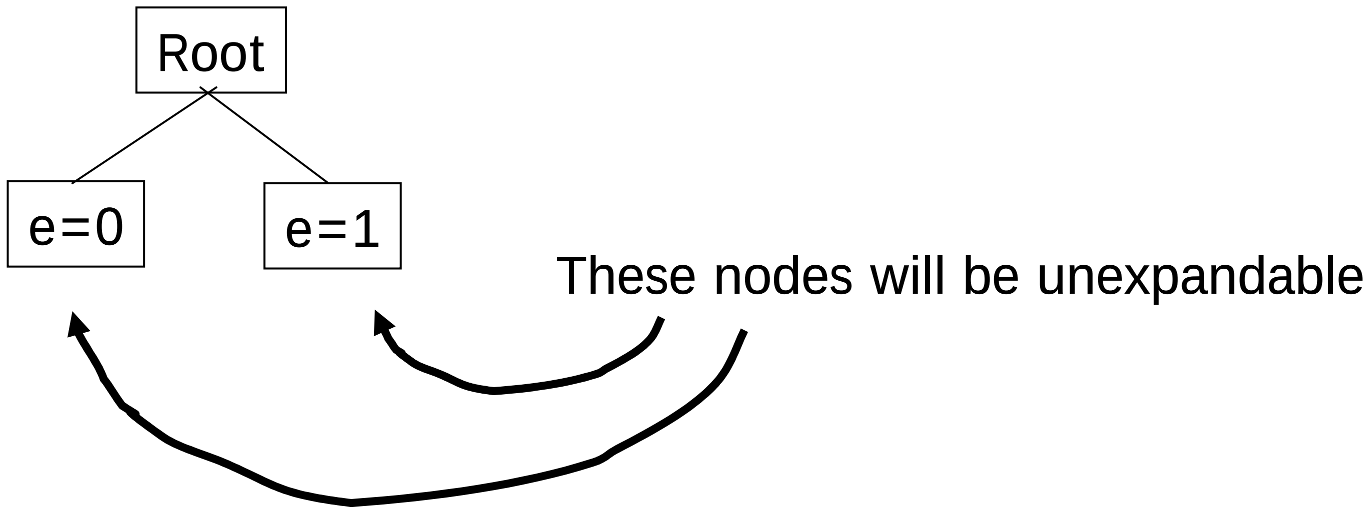
Output y = copy of e, except a random 25% of the records have y set to the opposite of e

32 records

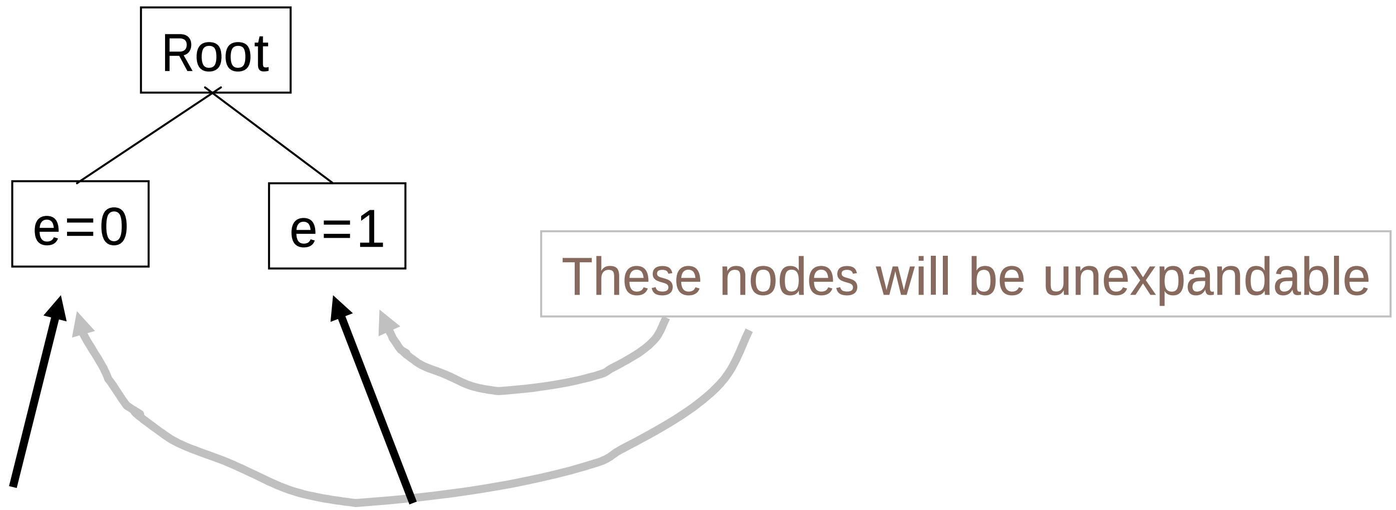
a	b	c	d	e	y
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	1
:	:	:	:	:	:
1	1	1	1	1	1

What decision tree would we learn now?

Without access to the irrelevant bits...



Without access to the irrelevant bits...



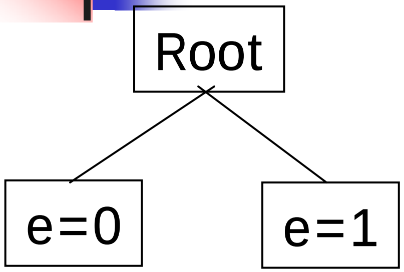
In about 12 of the 16 records in this node the output will be 0

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

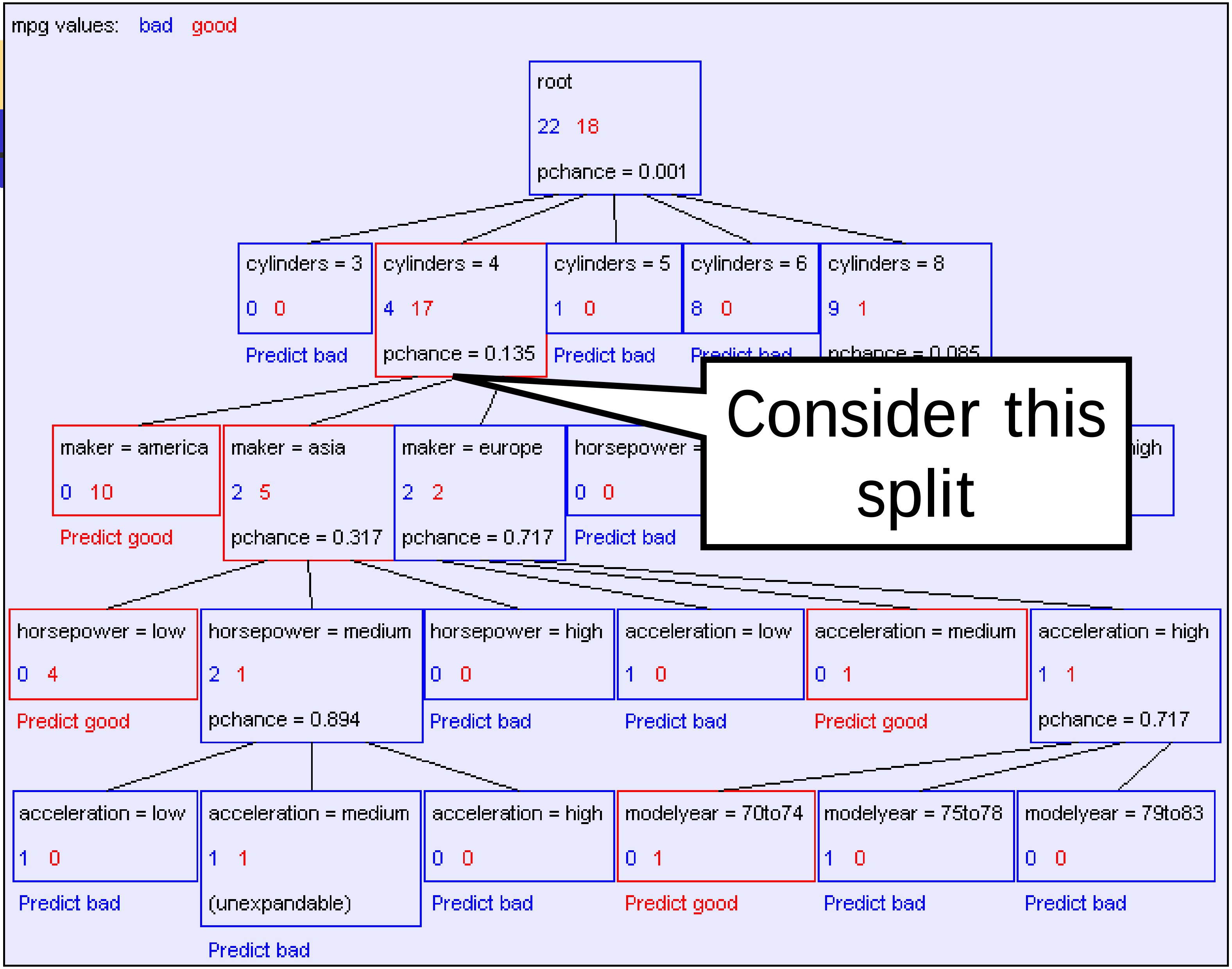
- Definition: If the learning algorithm fits noise (i.e. pays attention to parts of the data that are irrelevant) it is overfitting.
- Fact (theoretical and empirical): If the learning algorithm is overfitting then it may perform less well on test set data.

Avoiding overfitting

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

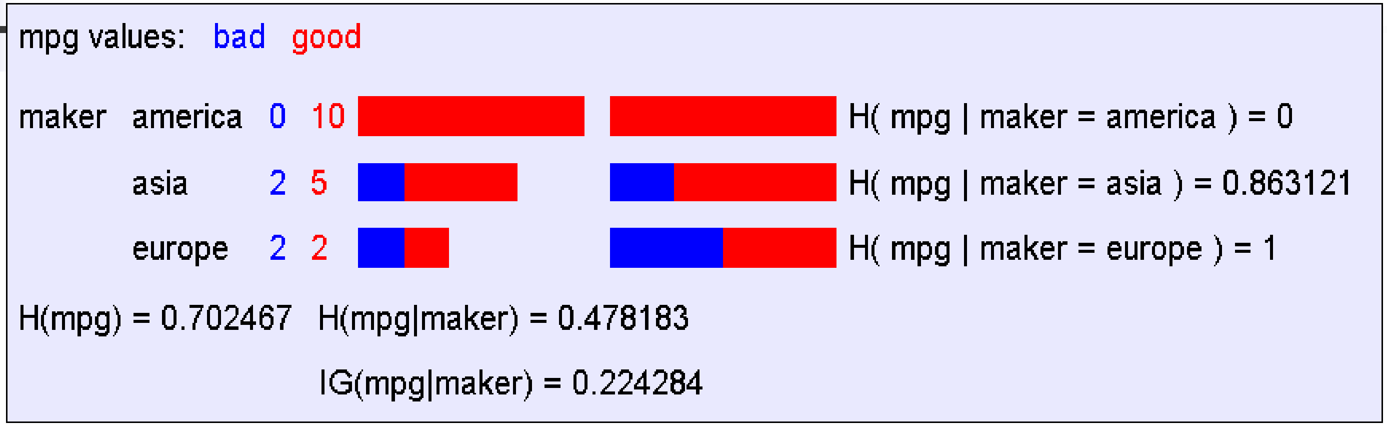
For example, if y = a AND b then b is an irrelevant variable only in the portion of the tree in which $a=0$

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



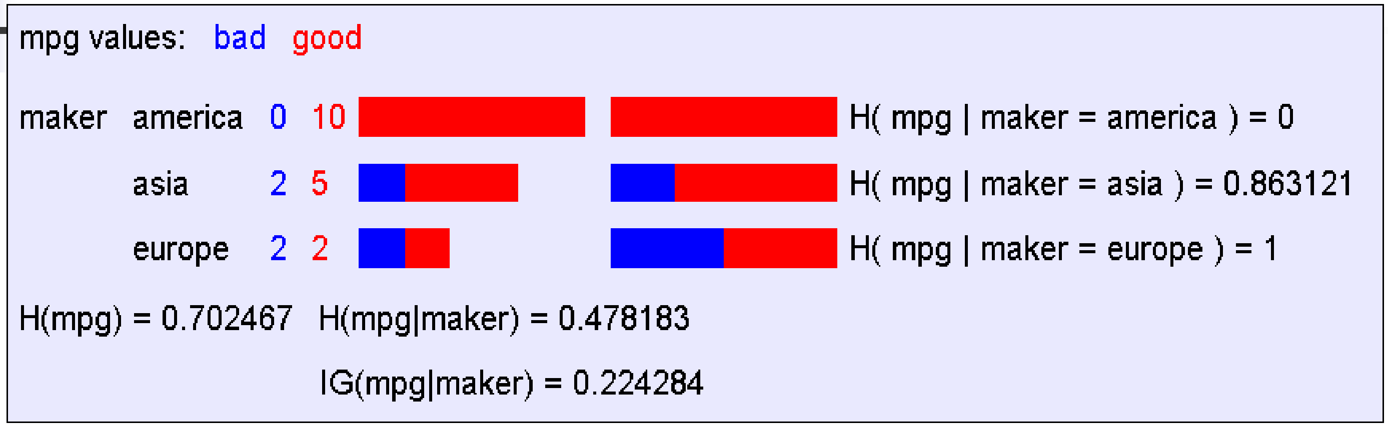
mpg values: bad good
root
22 18
pchance = 0.001
cylinders = 3
0 0
Predict bad
cylinders = 4
4 17
pchance = 0.135
cylinders = 5
1 0
Predict bad
cylinders = 6
8 0
cylinders = 8
9 1
Consider this split
maker = america
0 10
Predict good
maker = asia
2 5
pchance = 0.317
maker = europe
2 2
pchance = 0.717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

A chi-squared test



- Suppose that mpg was completely uncorrelated with maker.
- What is the chance we'd have seen data of at least this apparent level of association anyway?

A chi-squared test



- Suppose that mpg was completely uncorrelated with maker.
- What is the chance we'd have seen data of at least this apparent level of association anyway?

With chi-squared test, the answer is 13.5%.

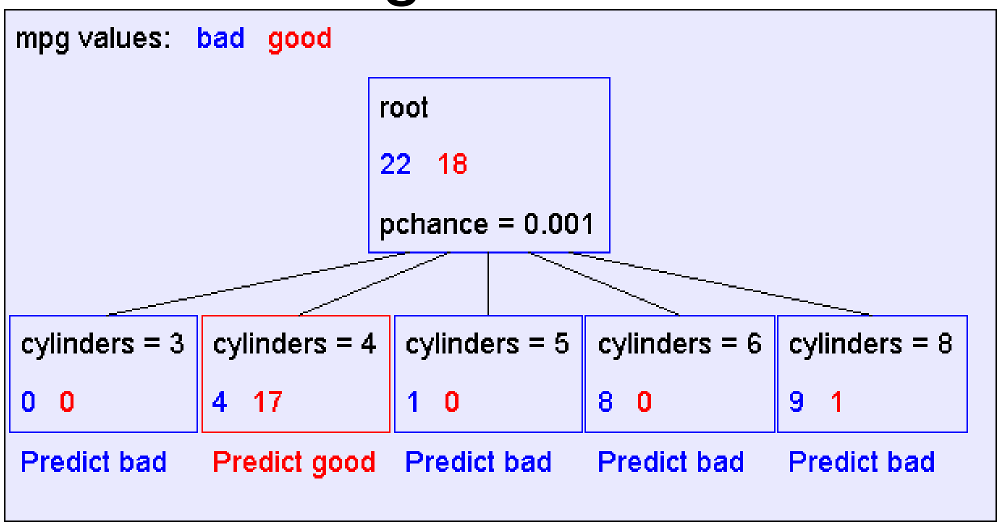
Using Chi-squared to avoid overfitting

- Build the full decision tree as before.
- But when you can grow it no more, start to prune:
  - Beginning at the bottom of the tree, delete splits in which p_{chance} > *MaxPchance.*
  - Continue working you way up until there are no more prunable nodes.

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

Pruning example

- With MaxPchance = 0.1, 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

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 undergrade’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

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

Chi Square Test – Compute Chi Square Value

$(O-E)^2/E$

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

Total = 14.026

	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, conclude: male and female undergraduates of the Univ. differ in their footwear preferences.

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.

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:
  1. Partition data into N disjoint sets S= {S_1, S_2, ... S_N}
  2. 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)

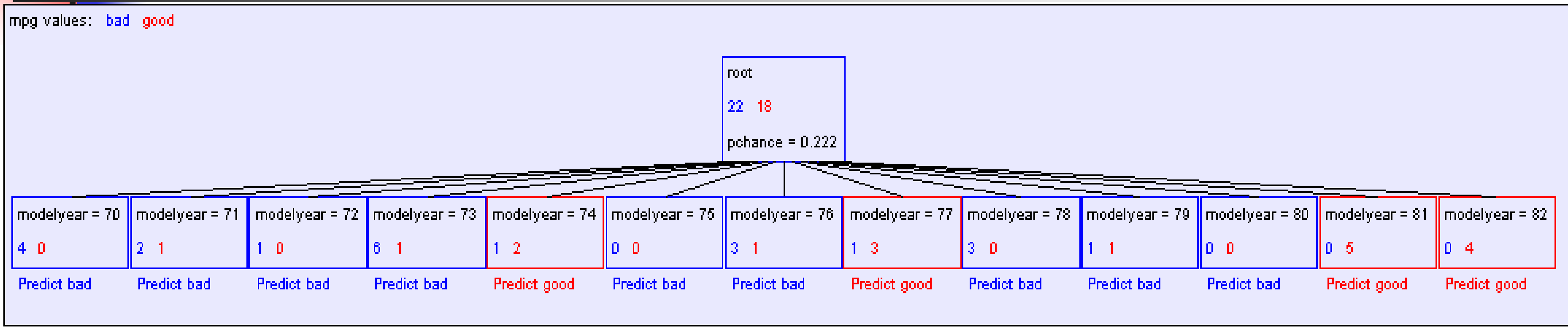
Real-Valued inputs

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

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

Idea One: Branch on each possible real value

“One branch for each numeric value” idea:



Hopeless: with such high branching factor 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.

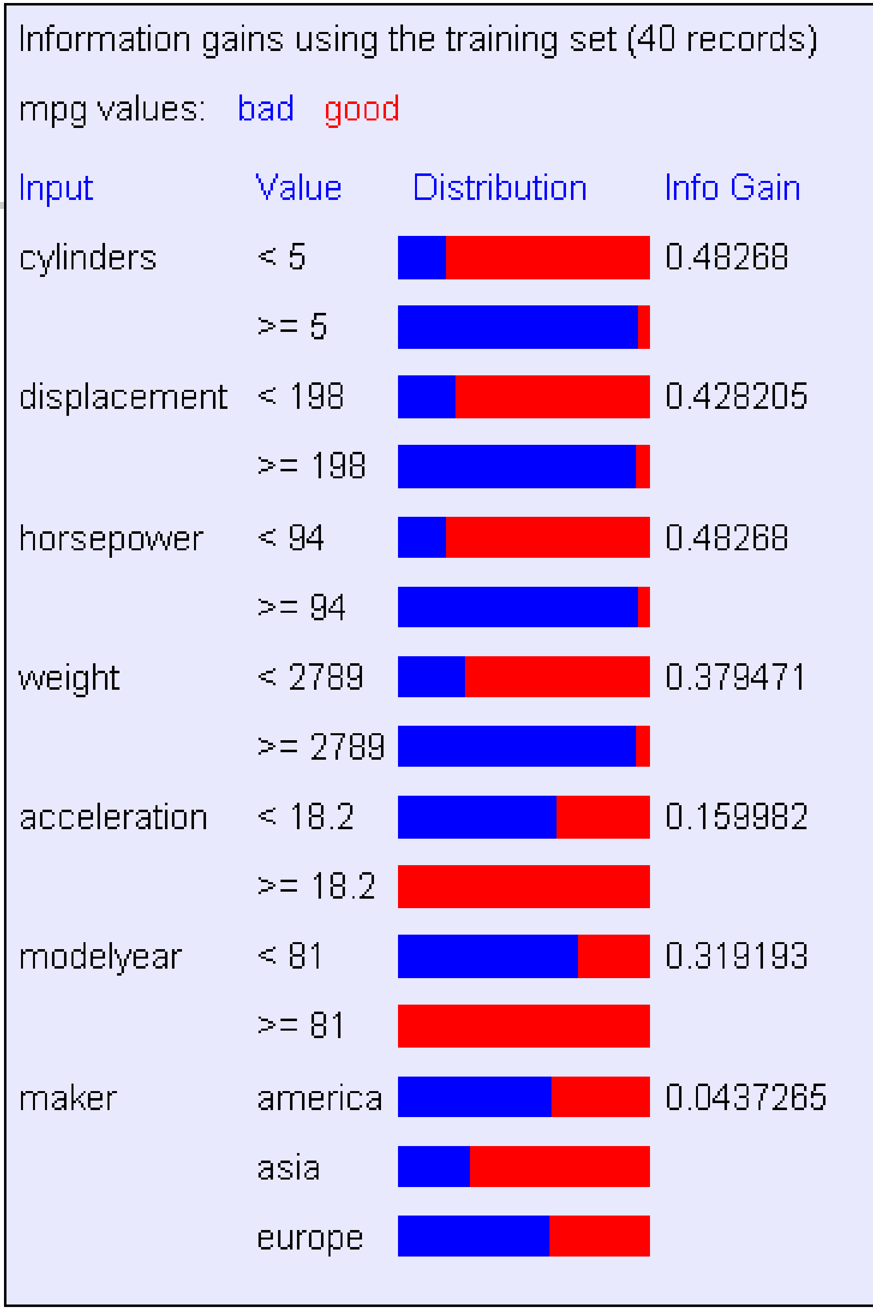
A better idea: thresholded splits

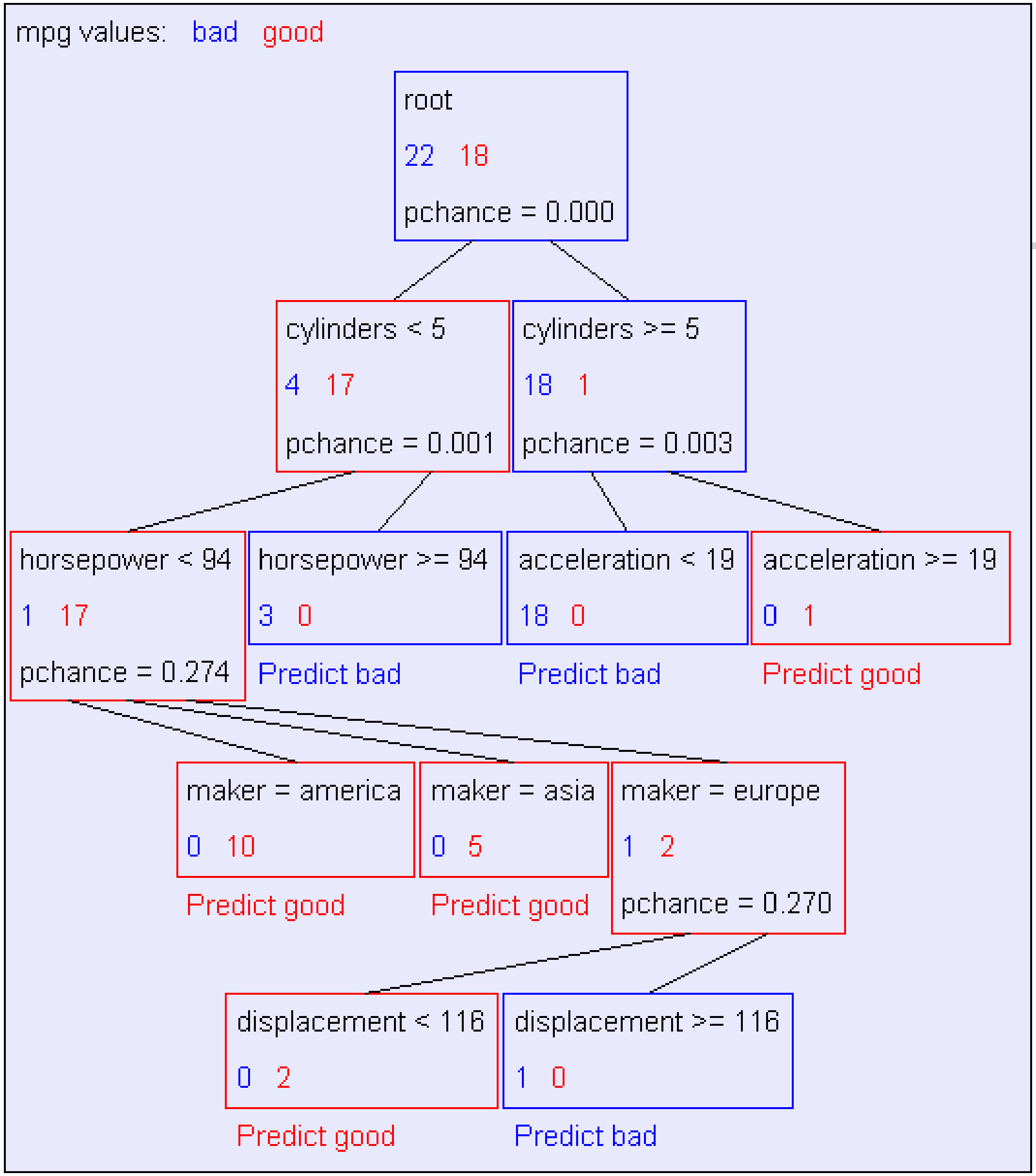
- Suppose X is real valued.
- Define $IG(Y|X:t)$ as $H(Y) - H(Y|X:t)$
- Define $H(Y|X:t) =$

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

  - $IG(Y|X:t)$ 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

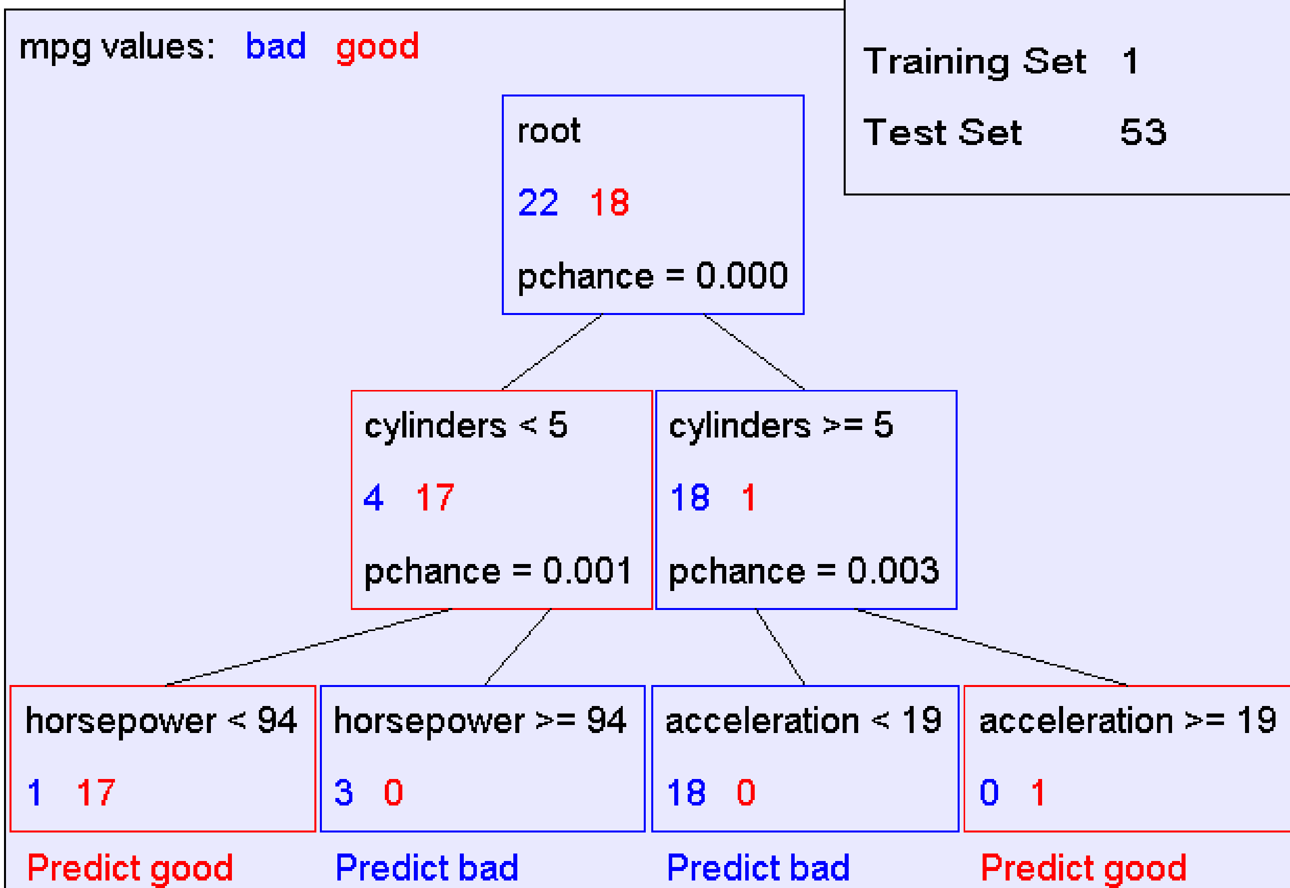




Unpruned tree using reals

Pruned tree using reals

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



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