STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY

CPE/EE 695: Applied Machine Learning

Lecture 4: Decision Tree Learning

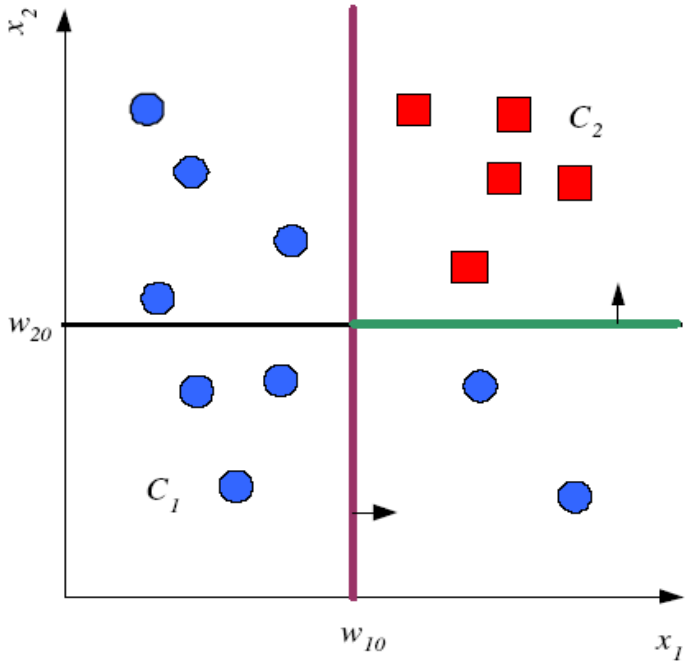
Dr. Shucheng Yu, Associate Professor
Department of Electrical and Computer Engineering
Stevens Institute of Technology

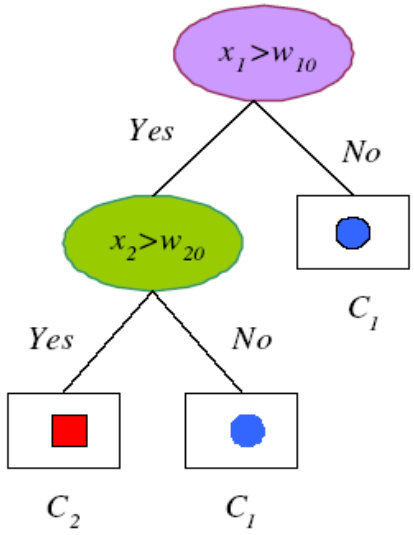
Today's Topics

- Decision tree concept
- Information gain (review of entropy)
- Construct decision tree
- Training and testing Error
- Over-fitting issue
- Pruning a tree

Decision tree *is*

- a hierarchical structure that used to classify classes based on a set of rules.
- called ***classification tree***, if the target values are discrete.
- called ***regression tree***, if the target values are continuous.
- iteratively split nodes at the optimum attribute that produce the *maximum* ***information gain***.
- Learning is *greedy*; find the best split recursively (Breiman et al, 1984; Quinlan, 1986, 1993)





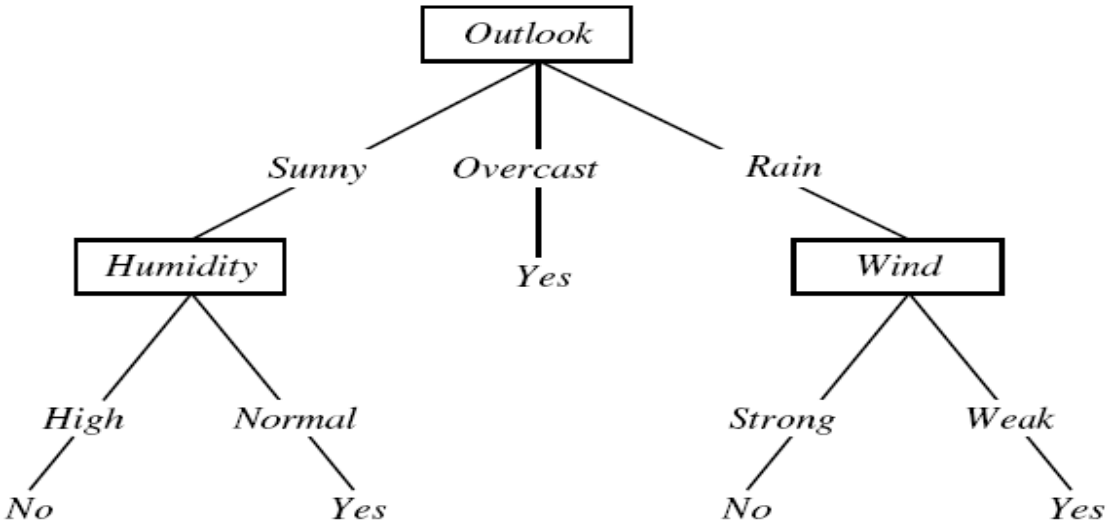
Training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTenı
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Classification tree elements

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

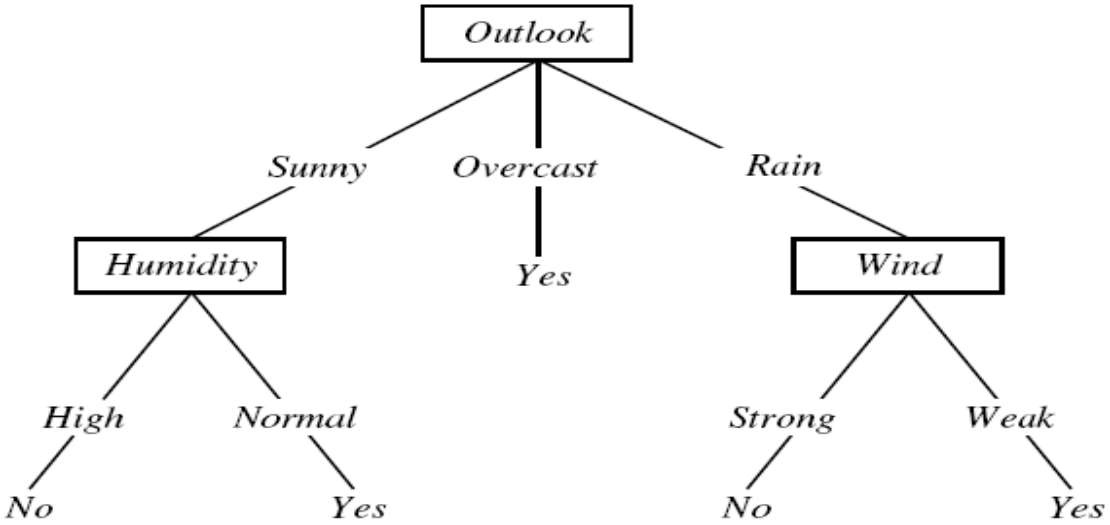
Decision tree for *PlayTennist*



Popular classification tree algorithms: ID3, CART, ASSISTANT, C4.5,…

A decision tree represents a **disjunction of conjunctions** of constraints on the attribute values of instances

Decision tree for *PlayTennist*



(Outlook = Sunny Λ Humidity = Normal) V (Outlook = Overcast) V

(Outlook = Rain Λ Wind = Weak)

All decision trees make a **complete space** of finite discrete-valued functions.

Where decision tree learning is appropriate?

For problems with the following characteristics:

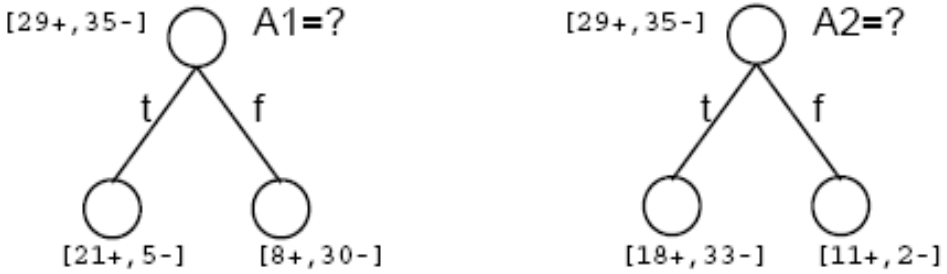
- Instances are represented by attribute-value pairs
  - Real-valued attributes are fine
- Target functions has discrete output values
  - Real-valued outputs are fine
- Disjunctive descriptions may be required
- The training data may contain errors
- The training data may contain missing attribute values

Top-down induction of decision tree

Main loop:

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

Which attribute is best?



Entropy

Entropy $H(X)$ of a random variable X

$$H(X) = -\sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

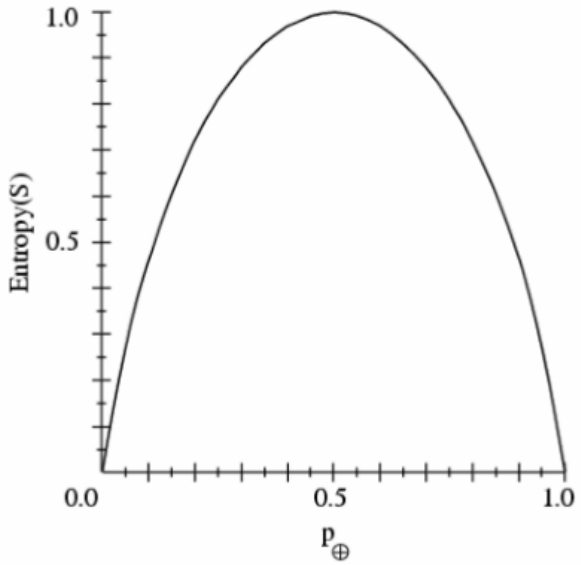
$H(X)$ is the expected number of bits needed to encode a randomly drawn value of X (under most efficient code)

Why? Information theory:

- Most efficient code assigns $-\log_2 P(X=i)$ bits to encode the message $X=i$
- So, expected number of bits is:

$$\sum_{i=1}^{n} P(X = i)(-\log_2 P(X = i))$$

Sample entropy



- S is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in S
- $p_\ominus$ is the proportion of negative examples in S
- Entropy measures the impurity of S

$$Entropy(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

Define $0\log_2 0 = 0$

Information gain

$Gain(S, A)$ = expected reduction in entropy due to sorting on A

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Specific Conditional Entropy:

$H(S \mid A=v_i)$ = The entropy of S among the records in which A has value v_i

Conditional Entropy:

$H(S \mid A)$ = The average specific conditional entropy of S

$= \Sigma_i \, Prob(A=v_i) \, H(S \mid A = v_i)$

Information gain

$Gain(S, A)$ = expected reduction in entropy due to sorting on A

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Specific Conditional Entropy:

$H(S \mid A=v_i)$ = The entropy of S among the records in which A has value v_i

Conditional Entropy:

$H(S \mid A)$ = The average specific conditional entropy of S

$= \Sigma_i \, Prob(A=v_i) \, H(S \mid A = v_i)$

Information gain

$Gain(S, A)$ = expected reduction in entropy due to sorting on A

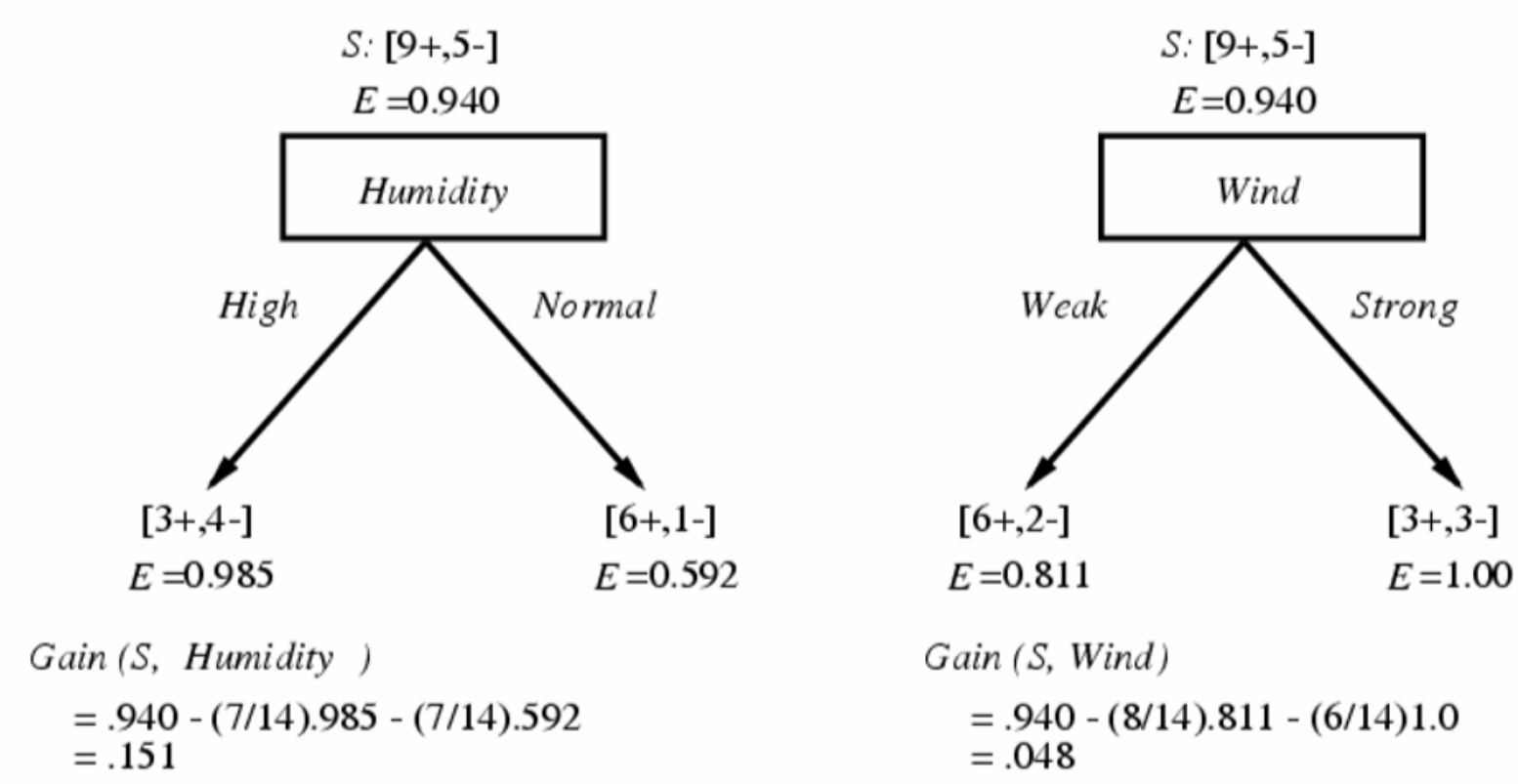
$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Entropy([29+, 35-]) = - 29/64 * $\log_2$29/64 – 35/64 * $\log_2$35/64 = 0.993

Training examples

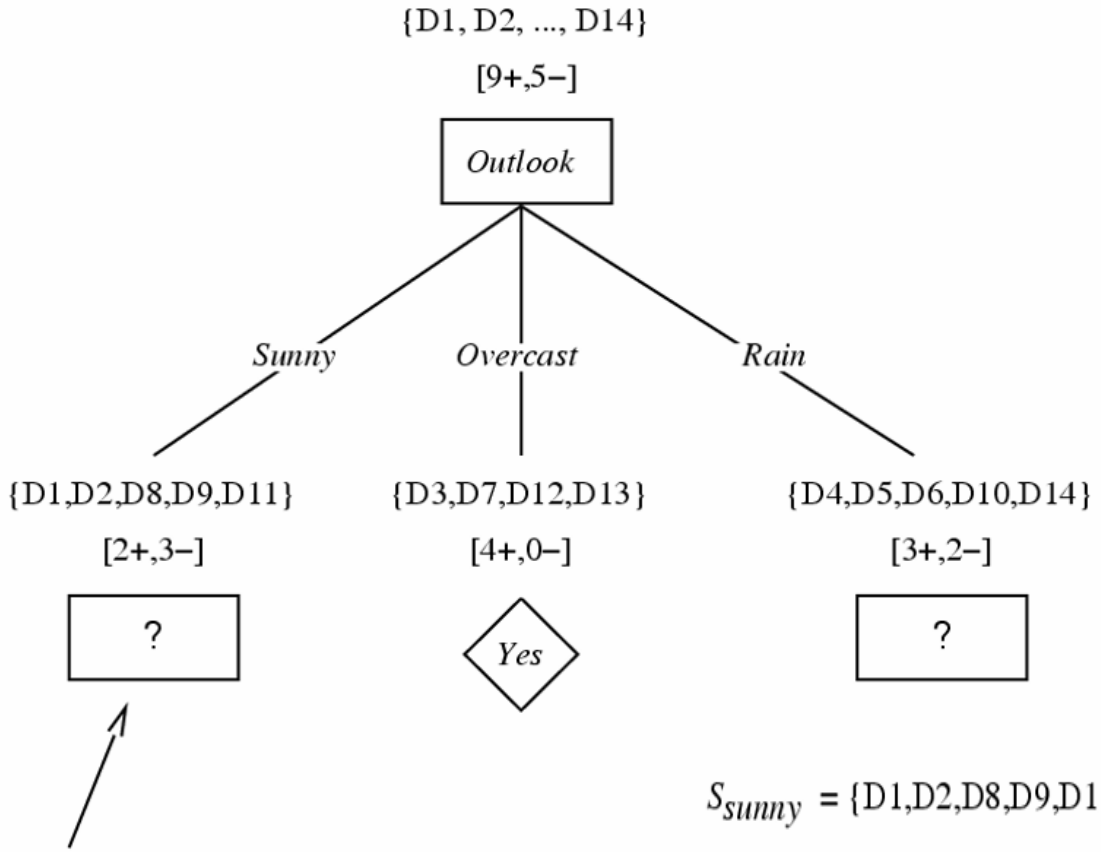
Day	Outlook	Temperature	Humidity	Wind	PlayTenn
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Which attribute is the best classifier?



Gain(S, Outlook) = 0.246
Gain(S, Temperature) = 0.029

Which attribute is the best classifier?



Which attribute should be tested here?

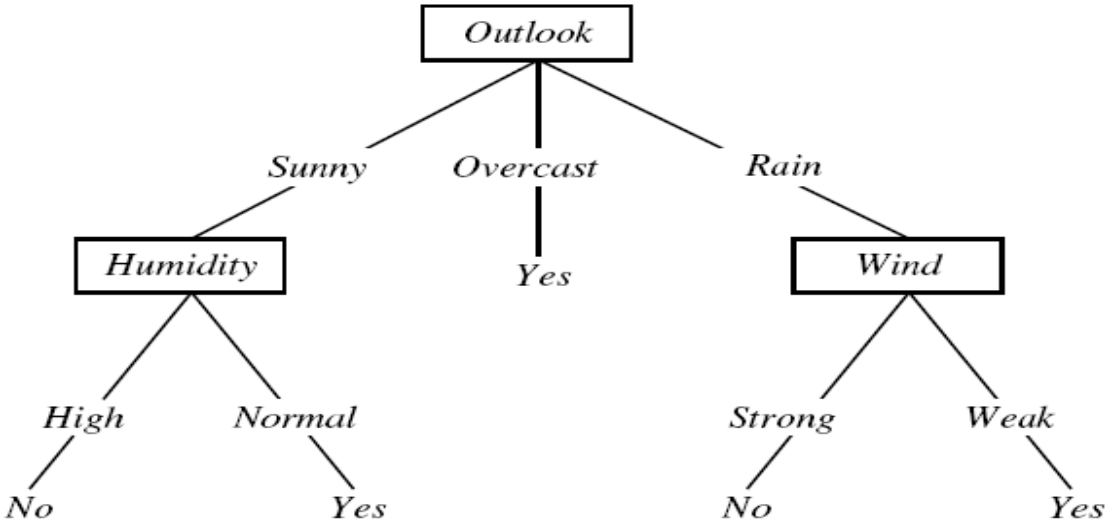
S_{sunny} = {D1,D2,D8,D9,D11}

$Gain\,(S_{sunny}, Humidity) = .970 - (3/5)\,0.0 - (2/5)\,0.0 = .970$

$Gain\,(S_{sunny}, Temperature) = .970 - (2/5)\,0.0 - (2/5)\,1.0 - (1/5)\,0.0 = .570$

$Gain\,(S_{sunny}, Wind) = .970 - (2/5)\,1.0 - (3/5)\,.918 = .019$

When to stop?



1) Every attribute has already been included along the path
2) Training examples associated with this node ALL have the same target attribute value (i.e., their entropy is zero).

Another example

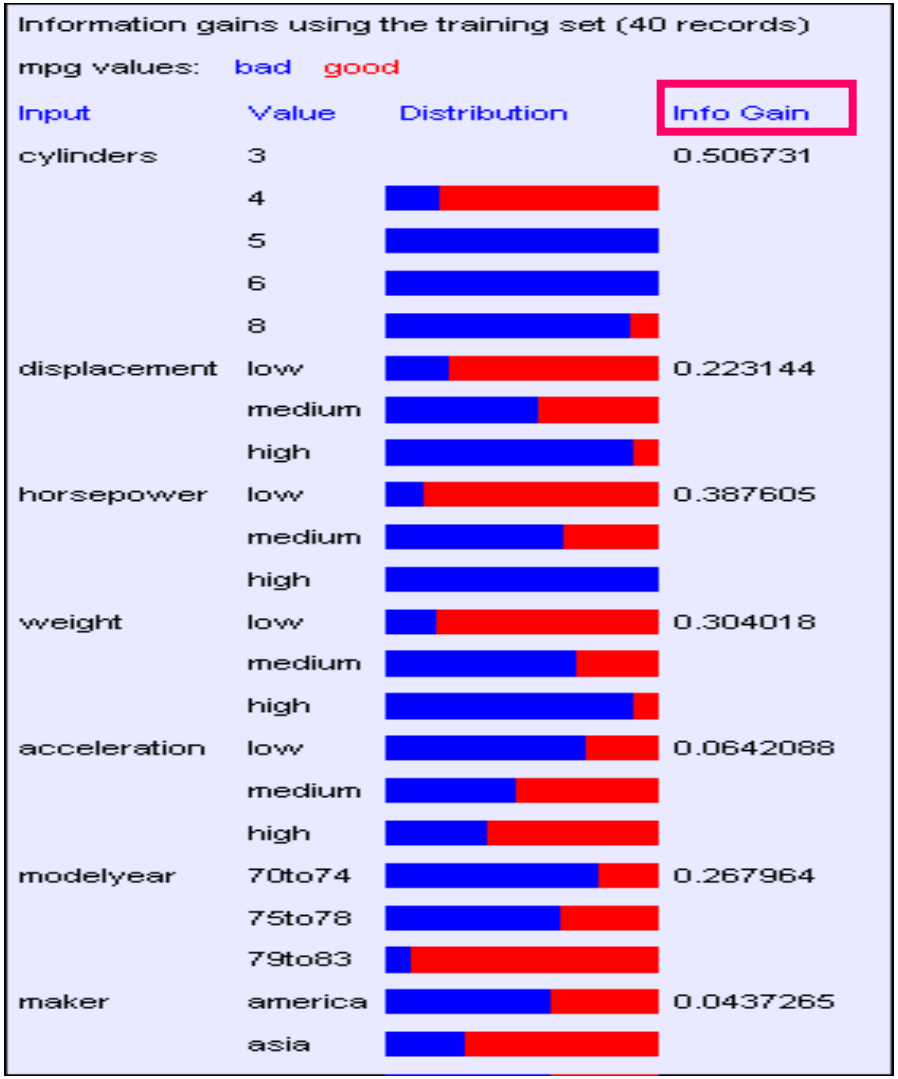
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe

From the UCI repository, 40 Records

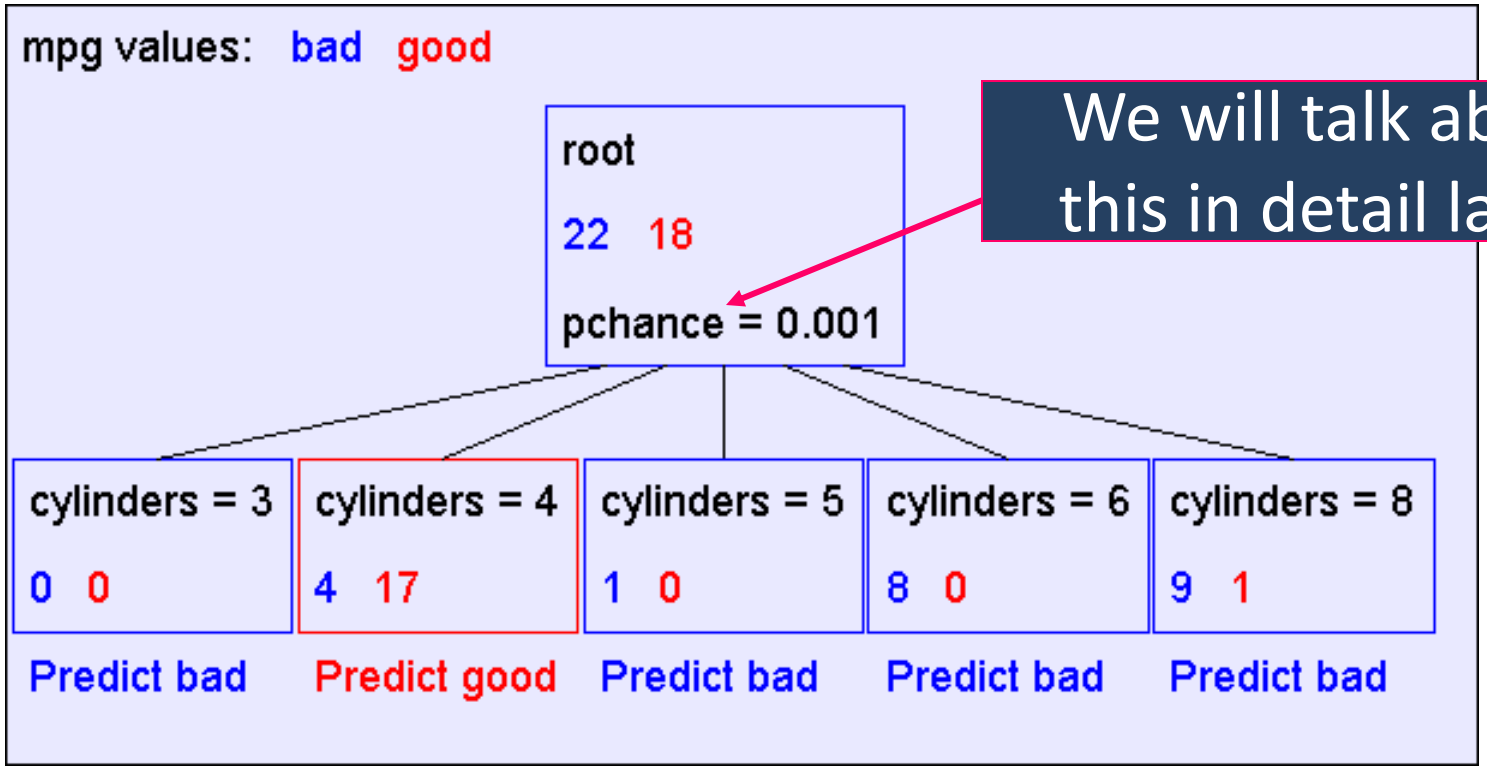
Suppose we want to predict MPG.

Look at all the information gains…

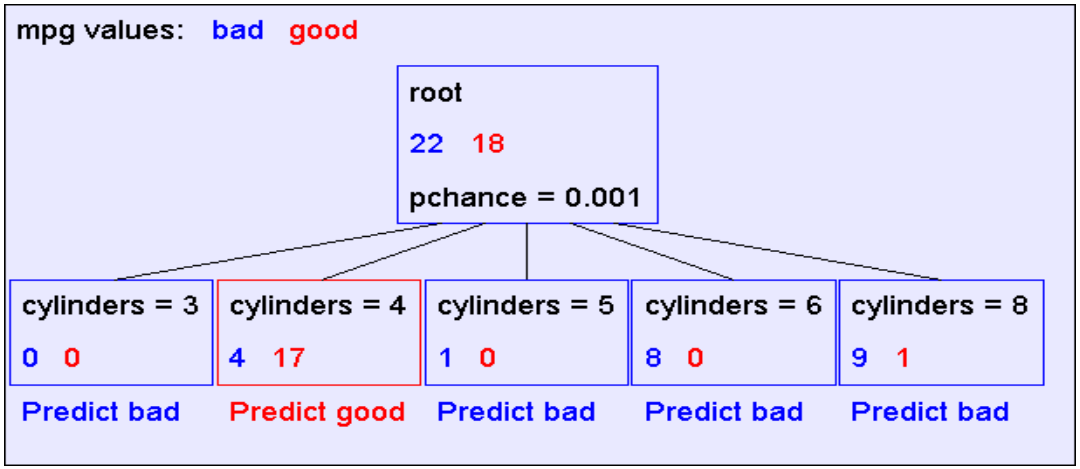
“Cylinders” has the highest Information Gain



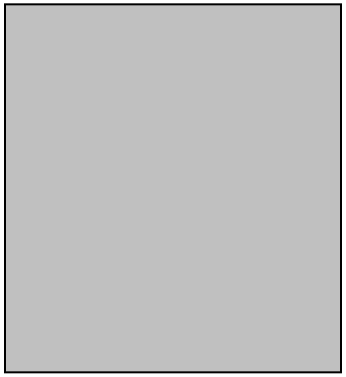
A Decision Stump



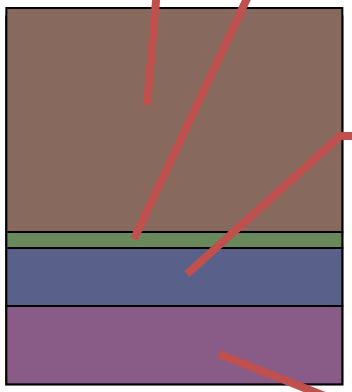
Recursion Step



Take the Original Dataset..



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



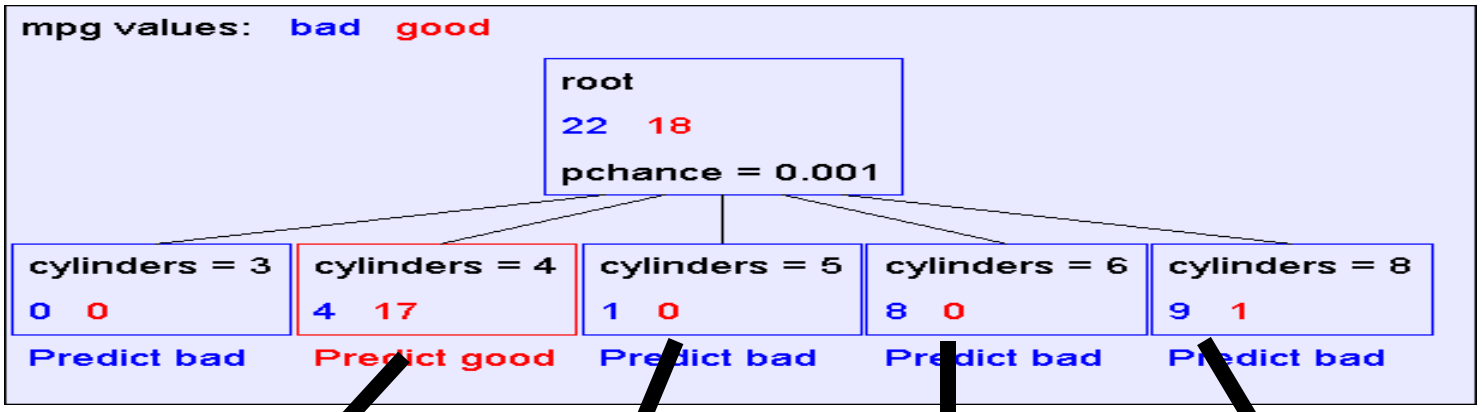
Records in which cylinders = 4

Records in which cylinders = 5

Records in which cylinders = 6

Records in which cylinders = 8

Recursion Step



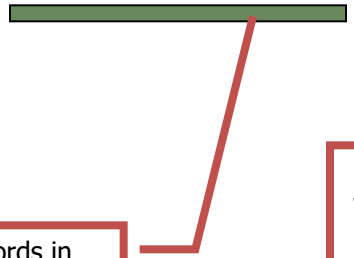
Build tree from These records..

Classification for these records..

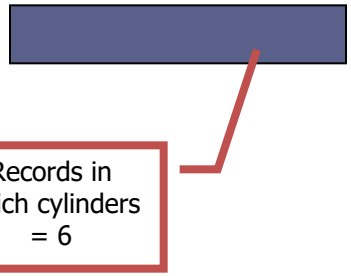
Classification for these records..

Build tree from These records..

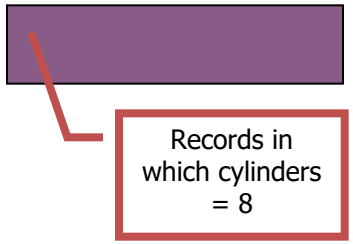
Records in which cylinders = 4



Records in which cylinders = 5

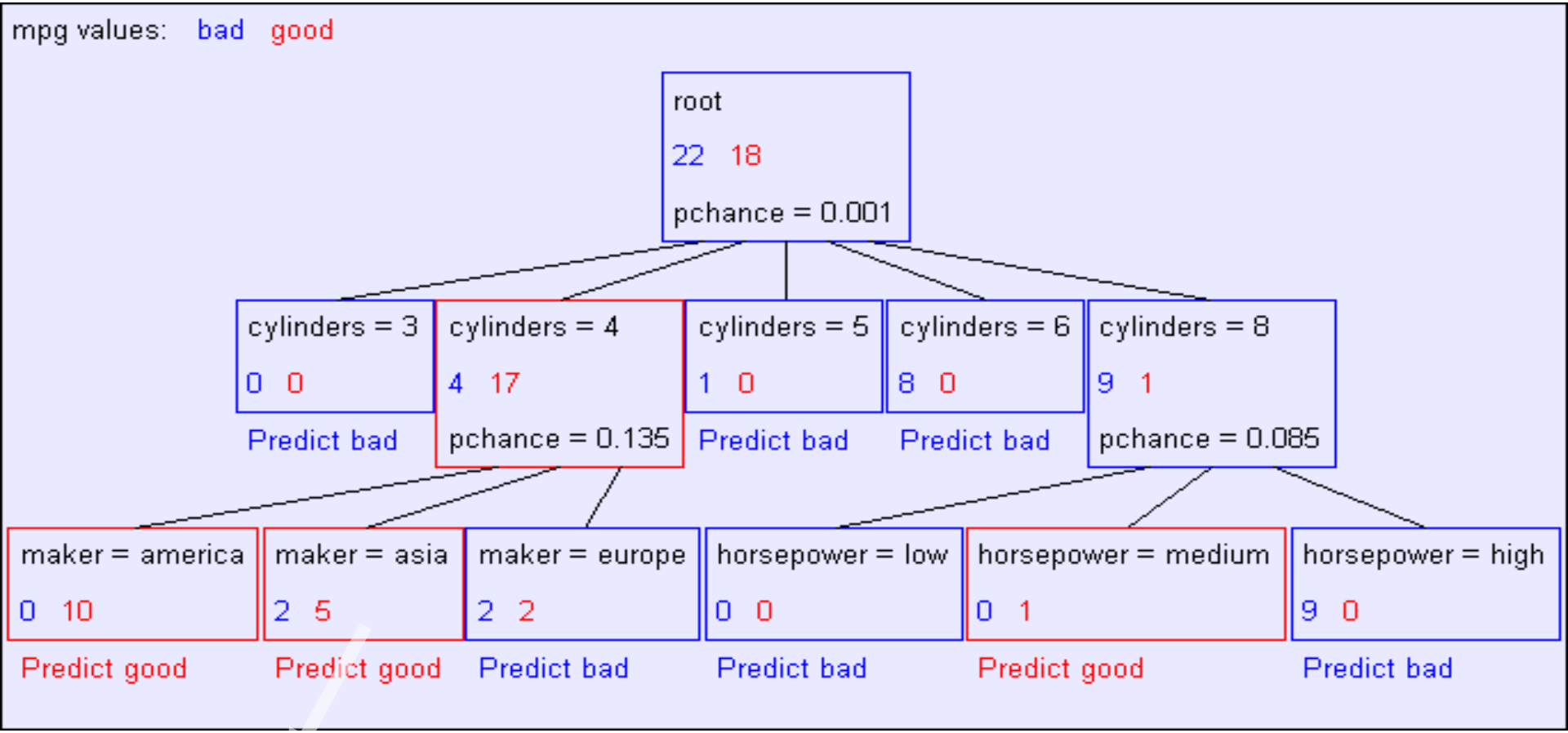


Records in which cylinders = 6



Records in which cylinders = 8

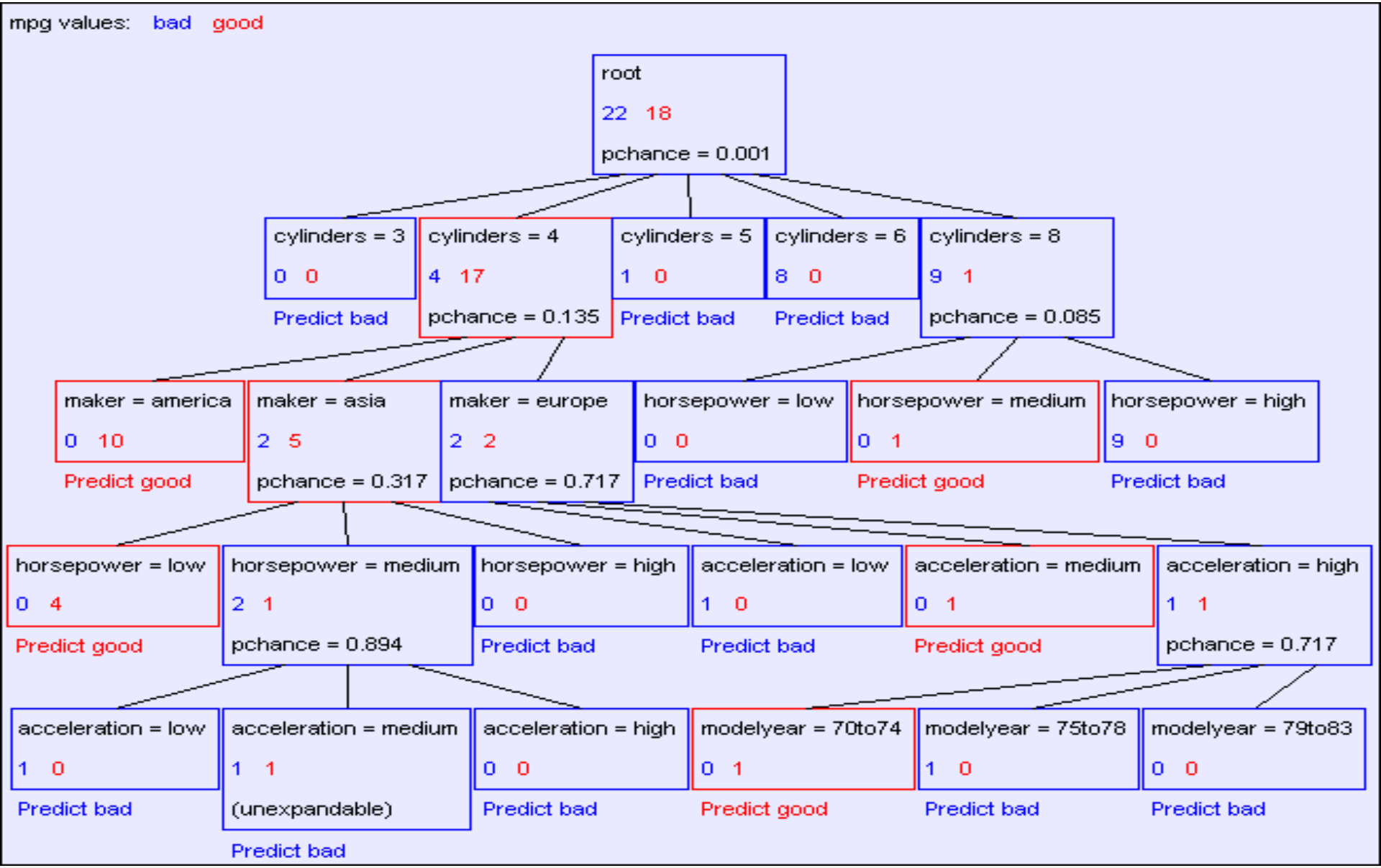
Second level of tree



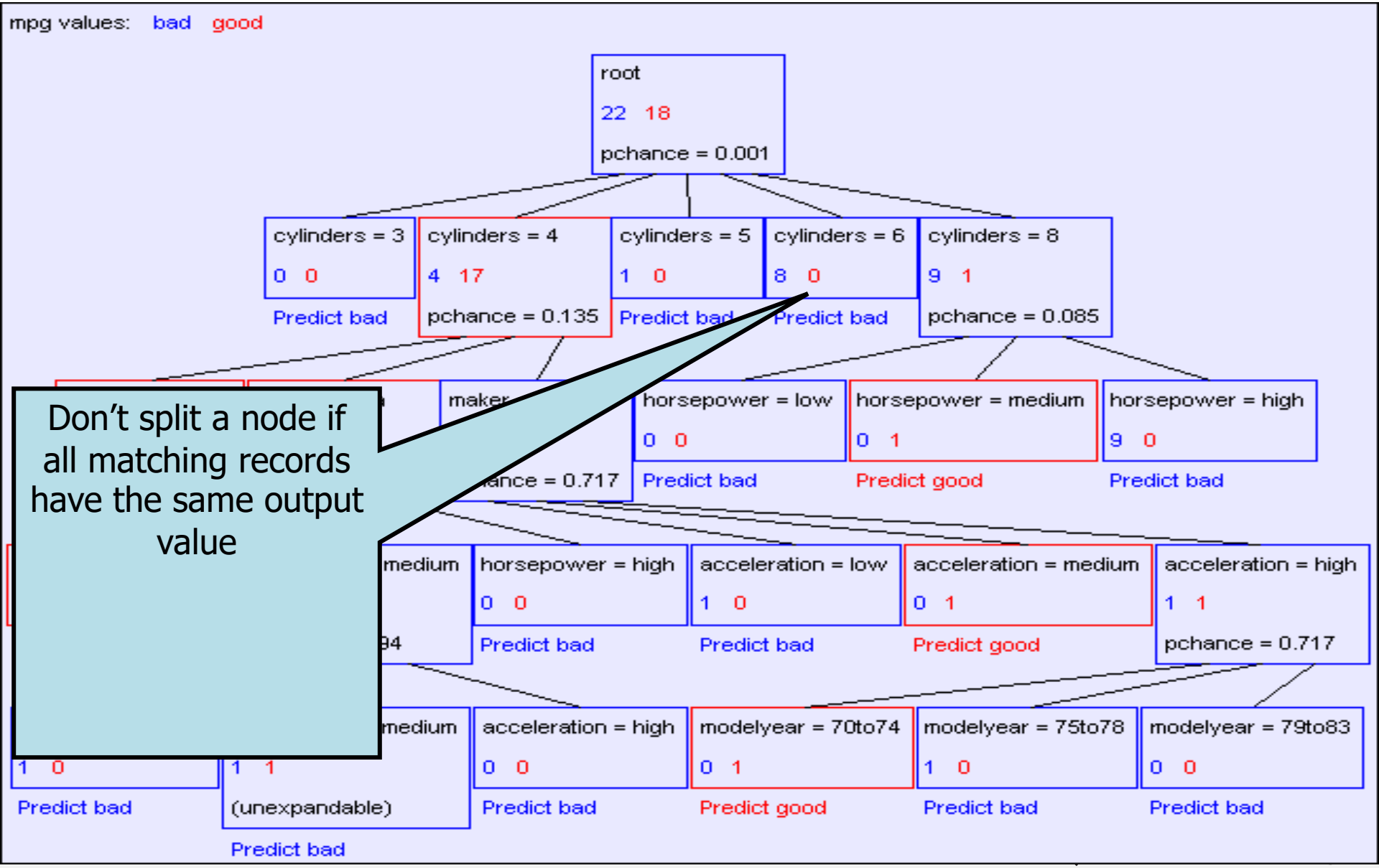
Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

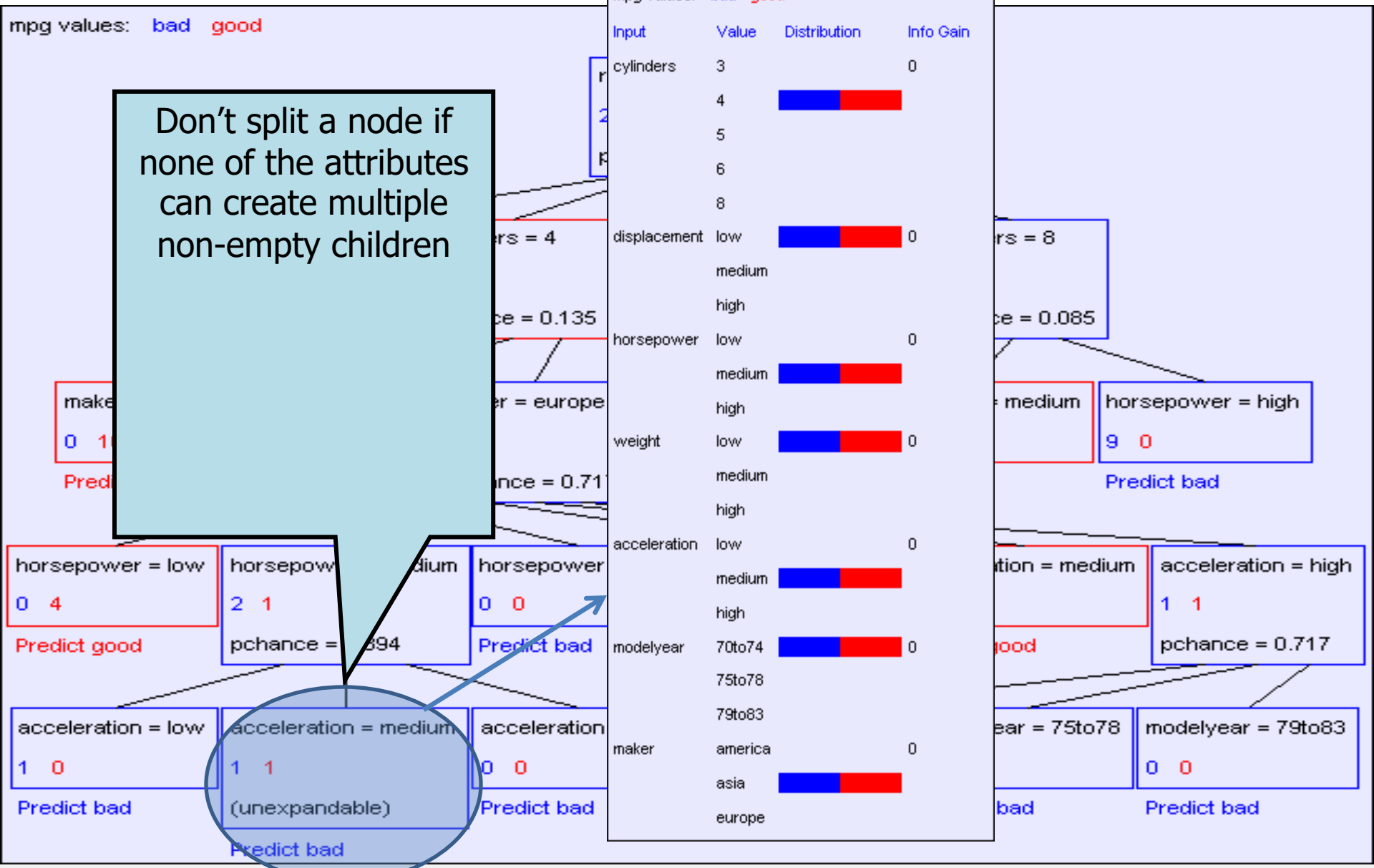
The final tree



Base Case One



Base Case Two



Summary of ID3 algorithm

ID3 (Examples, Target_Attribute, Attributes)

Create a root node for the tree

IF all examples are positive, Return the single-node tree *Root*, with label = +

If all examples are negative, Return the single-node tree *Root*, with label = -

If attributes is empty, then Return the single node tree *Root*, with label = *most common value of the target attribute in the examples*

Otherwise Begin

A ← The Attribute that best classifies examples (according to information gain)

The decision tree attribute for *Root* ← *A*

For each possible value, *vi*, of A,

Add a new tree branch below *Root*, corresponding to the test *A = vi*

Let Examples(vi) be the subset of examples that have the value vi for A

If Examples(vi) is empty

Then below this new branch add a leaf node with label = most common target value in the examples

Else below this new branch add the subtree

ID3 (Examples(vi), Target_Attribute, Attributes – {A})

End

Return Root

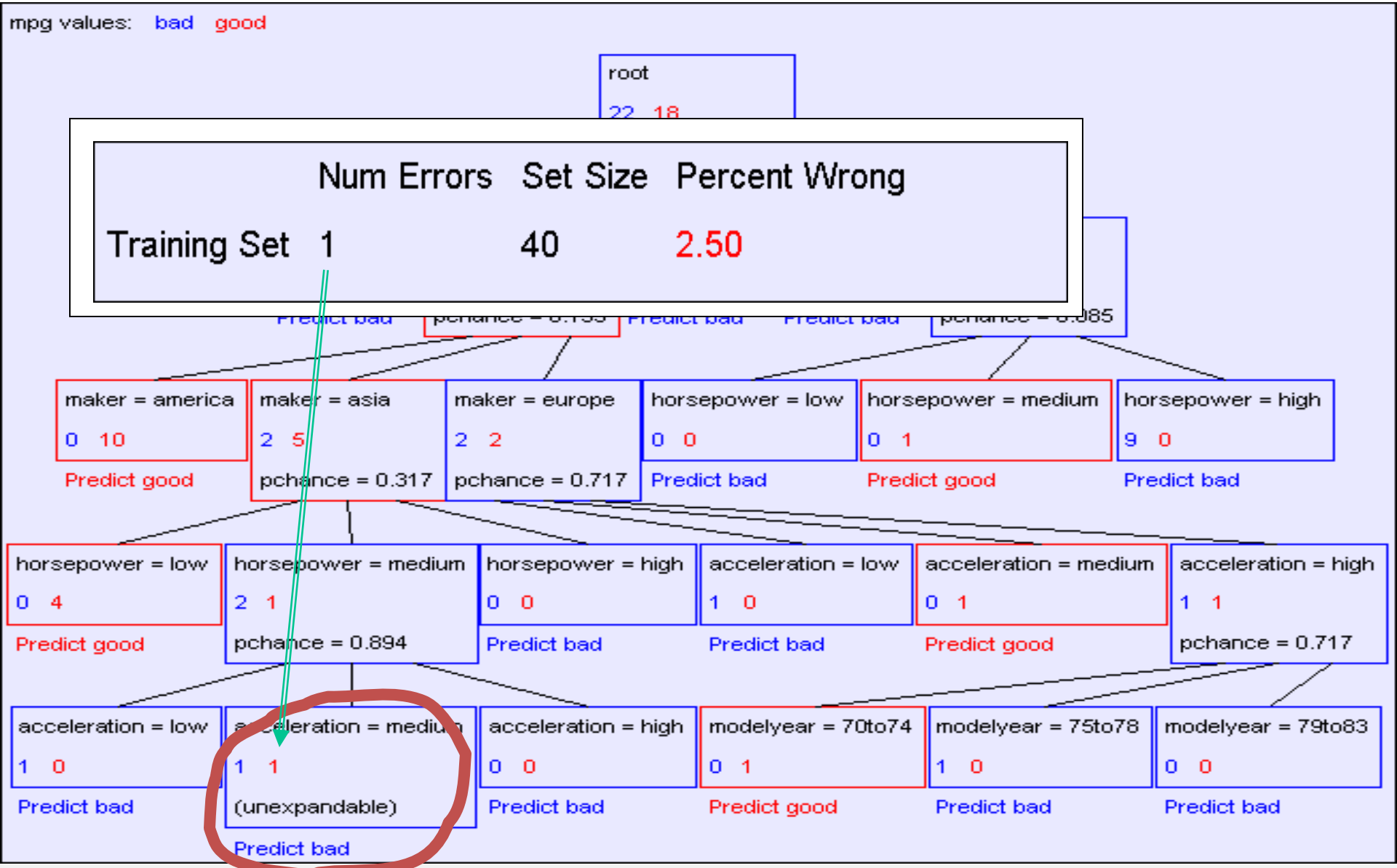
Training Set Error

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

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

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

MPG Training error



Test Set Error

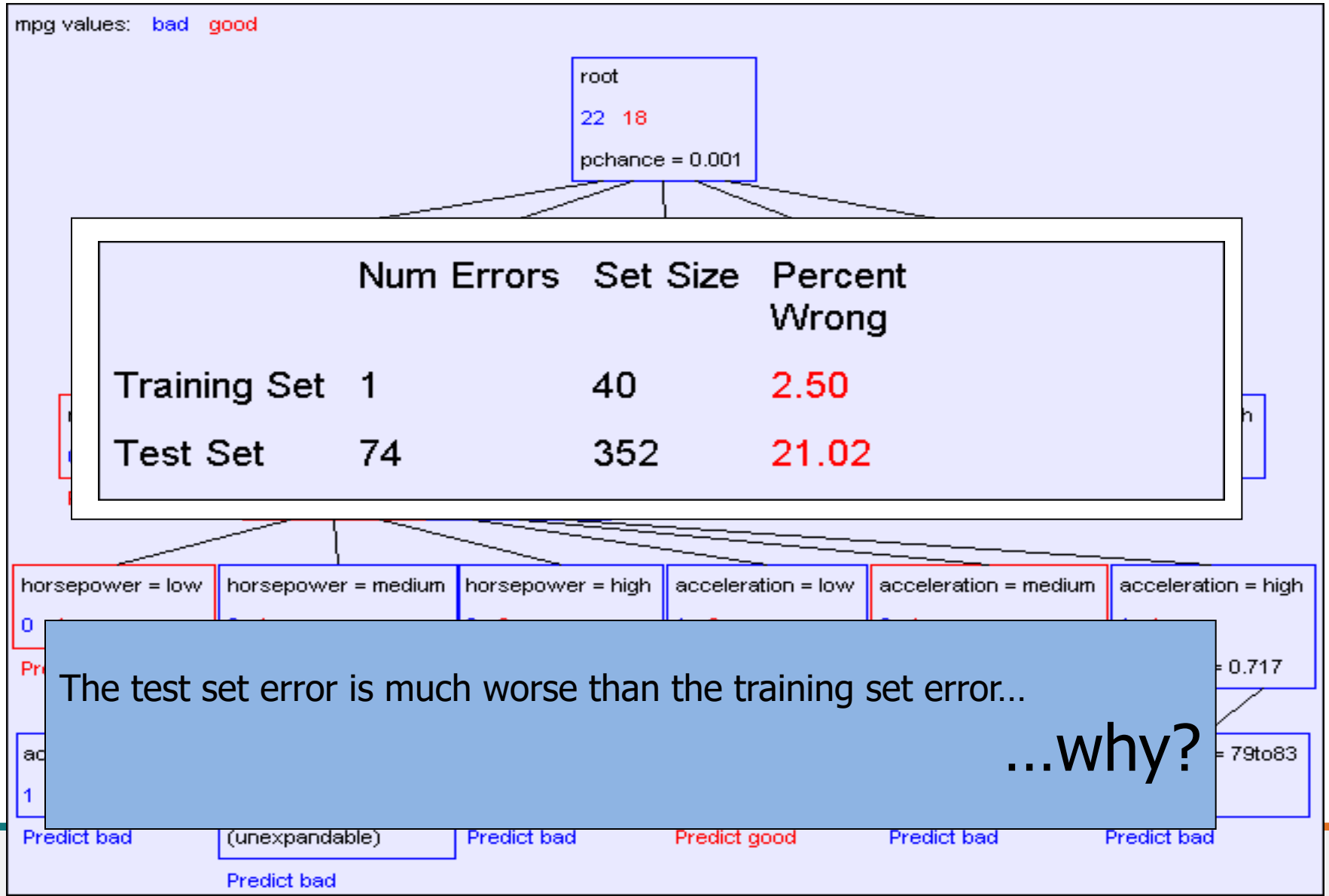
Split data to training and testing

Using training dataset to learn the tree and test how well the tree base on testing data.

This is a good simulation of what happens when we try to predict future data.

And it is called **Test Set Error.**

MPG Test set error



The overfitting issue

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

Consider error of hypothesis h over

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

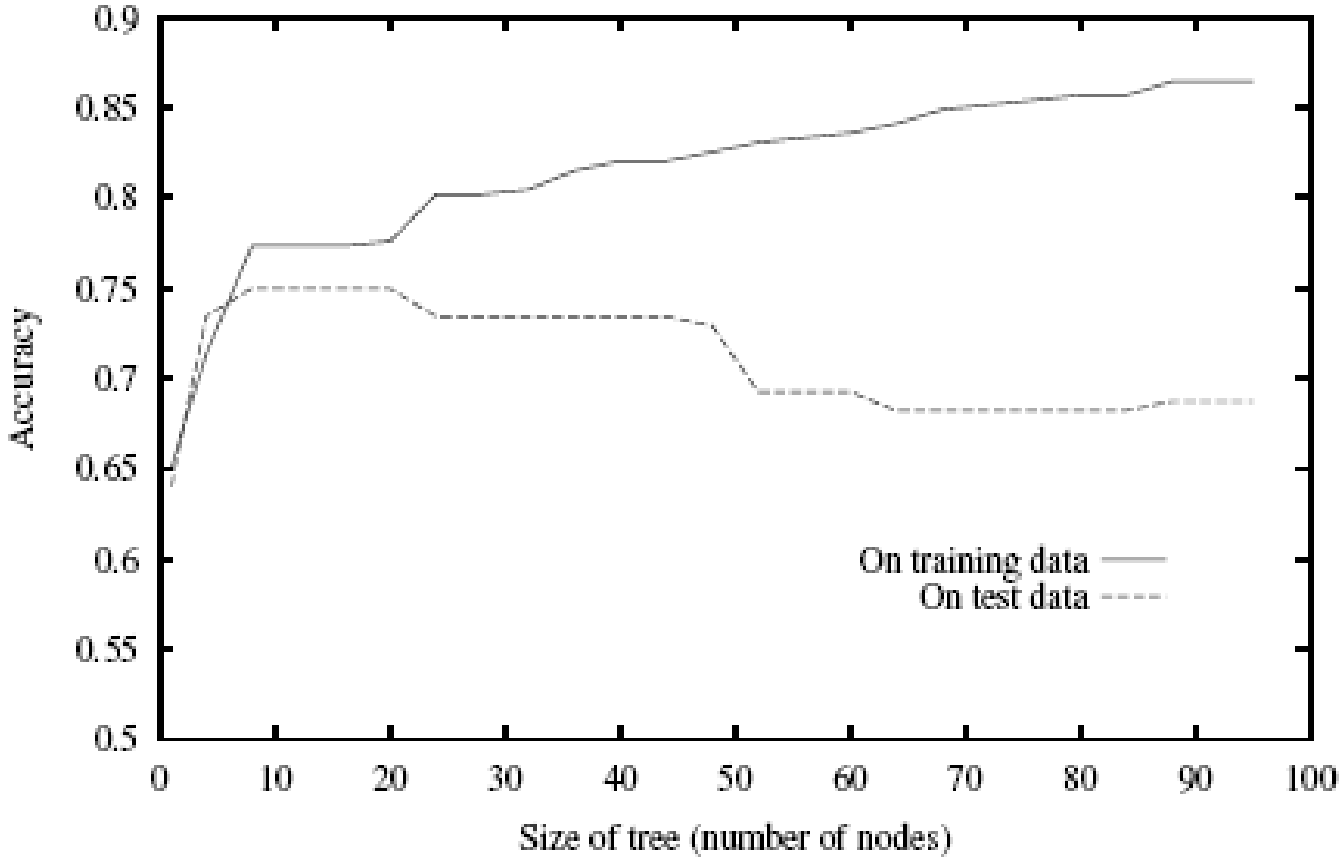
Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

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

and

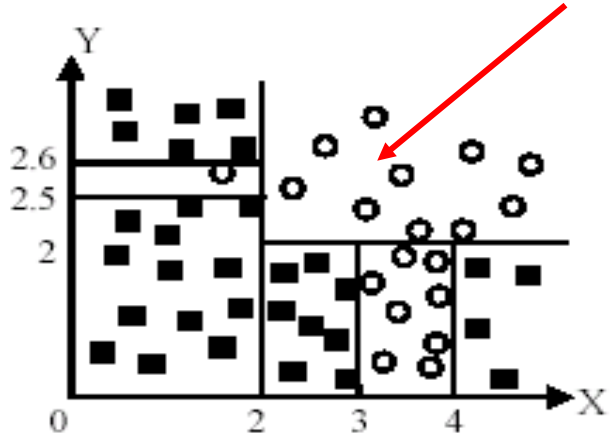
$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Overfitting in decision tree learning (ID3 algorithm)

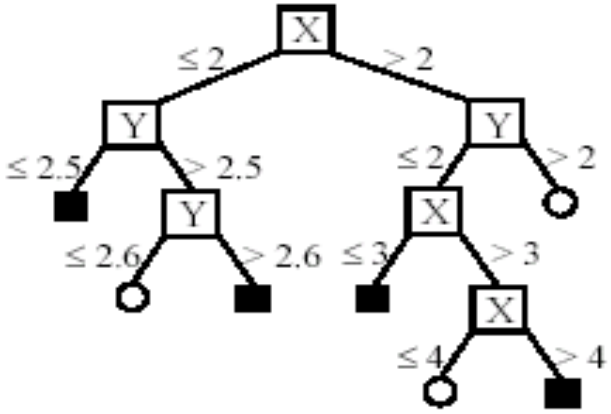


Over-Fitting Example

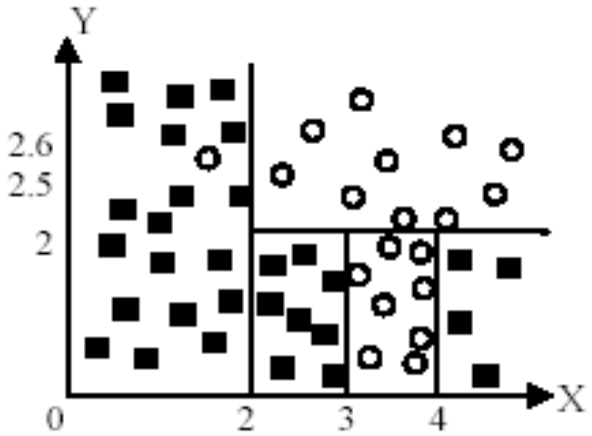
Likely to overfit the data

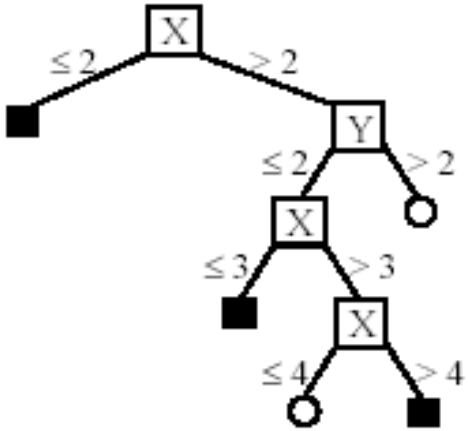


(A) A partition of the data space



(B). The decision tree





Avoiding overfitting

How can we avoid overfitting?

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

How to select "best" tree:

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

Reduced error pruning

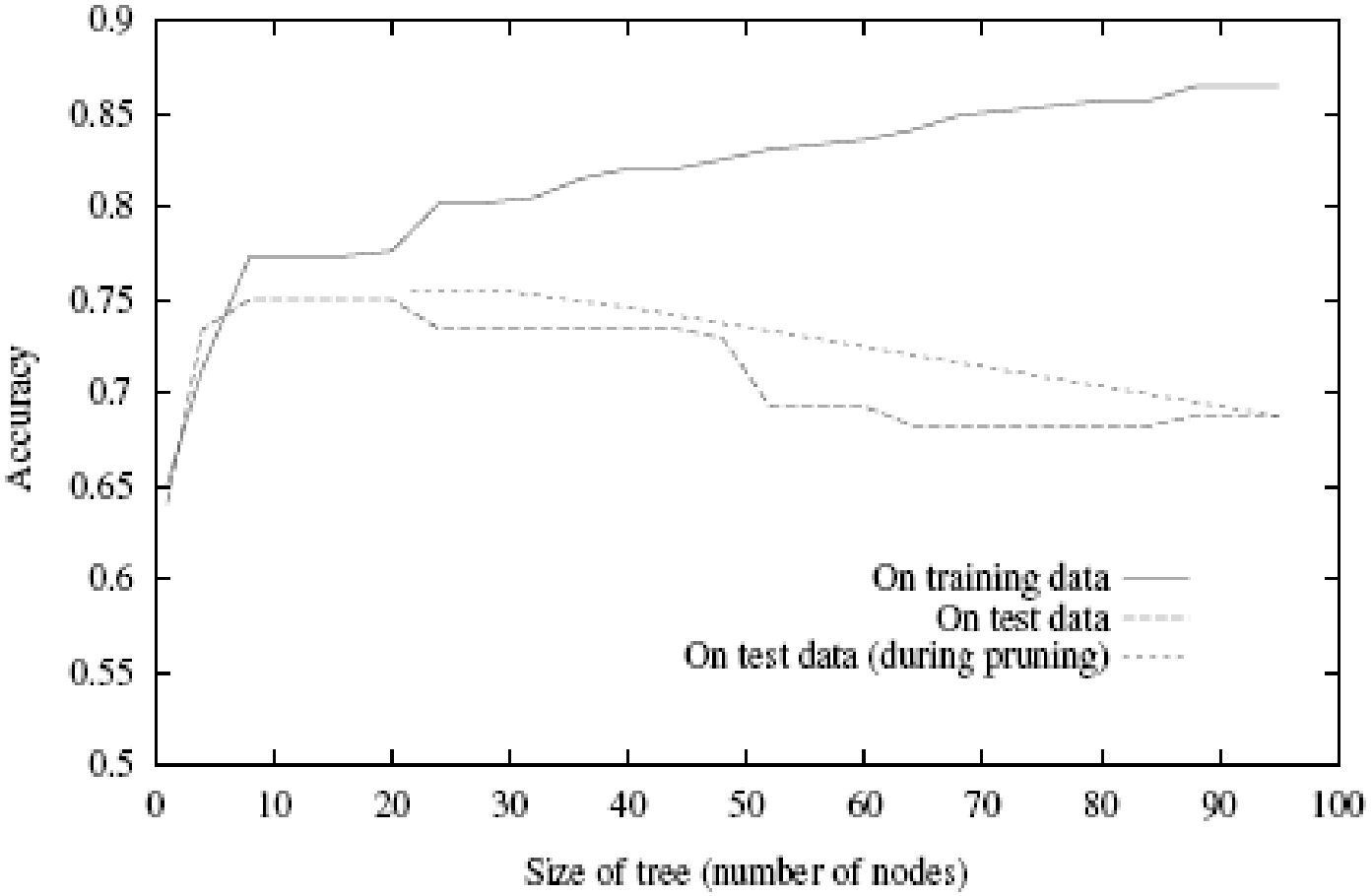
Split data into *training* and *validation* set

Do until further pruning is harmful:

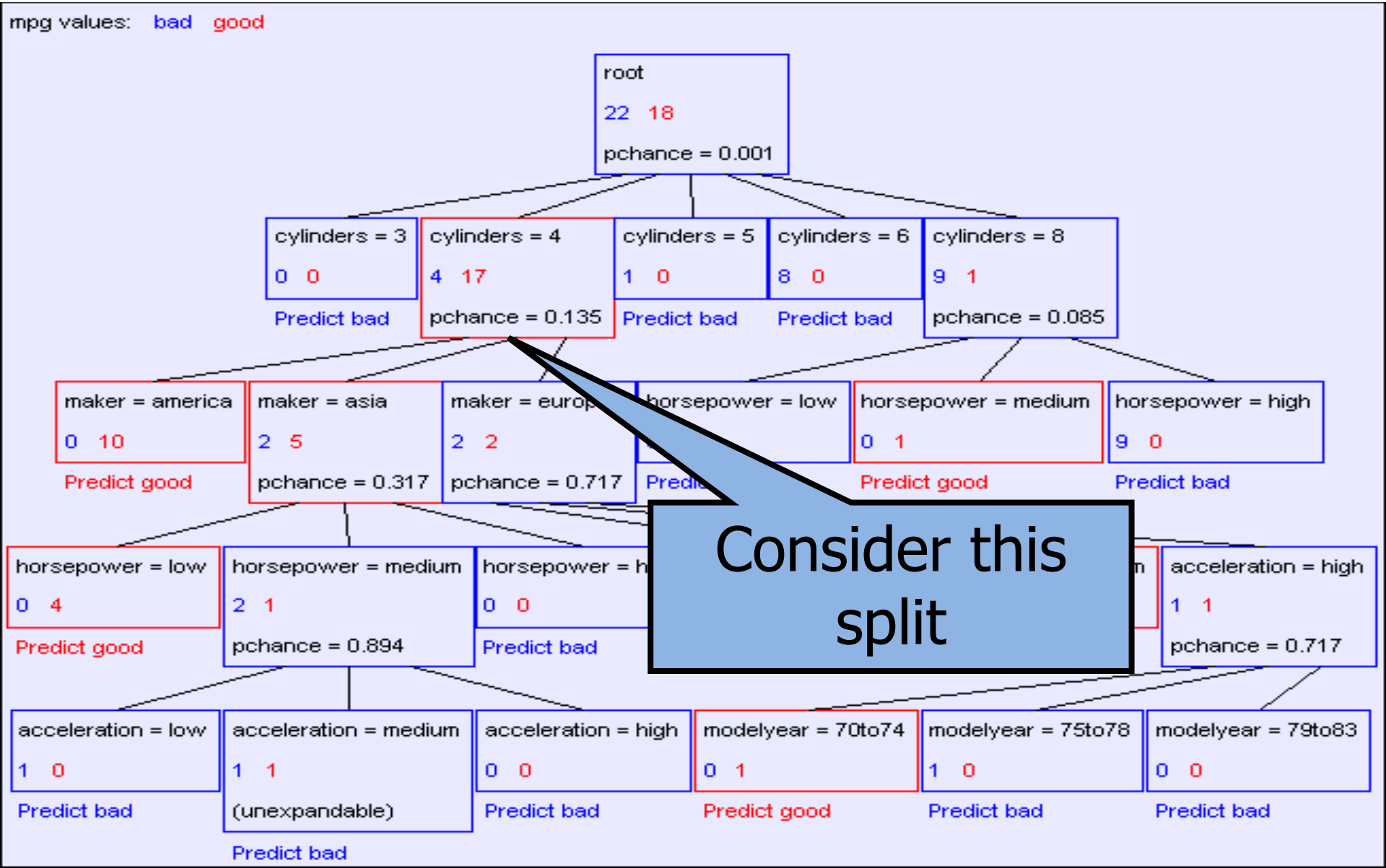
1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

- produces smallest version of most accurate subtree
- What if data is limited?

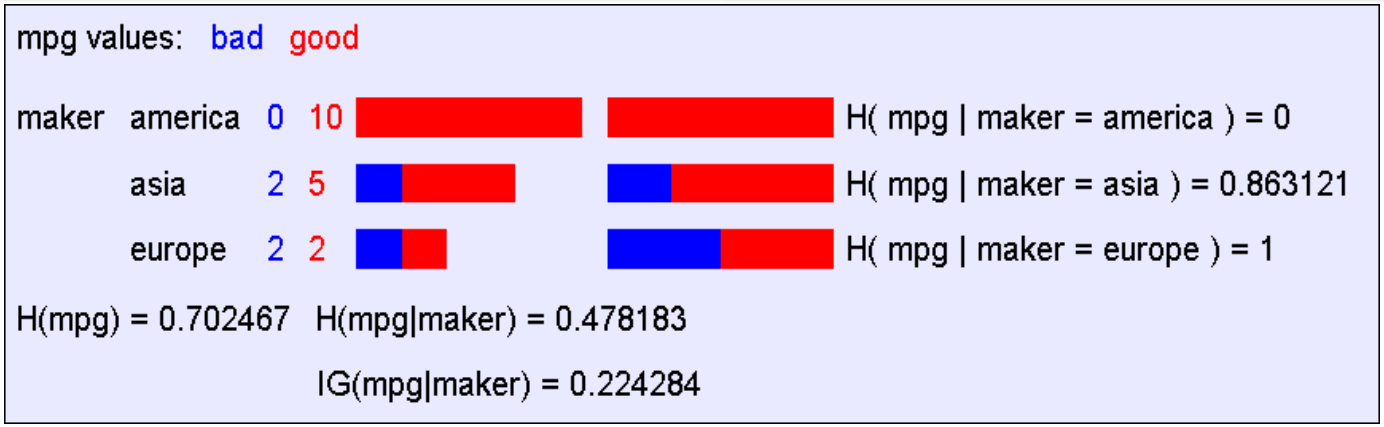
Effects of reduced error pruning



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



A chi-squared test



Suppose that mpg was completely uncorrelated (irrelevant) with maker.

What is the chance that mpg is really not associate with maker?

By using a particular kind of chi-squared test, the answer is 0.135=13.5%.
how to do chi-squared test???

Review of chi-squared test

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

-- Use the statistical significance test to find the irrelevant attribute.

ID3 used Chi-squared test

Review of chi-squared test

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

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

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

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

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

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

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

Review of chi-squared test

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

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

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

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

Reference : Chi-squared pruning, [online], available: http://www.cs.rpi.edu/courses/fall02/ai/assign/assign6/index.html#chi-squared

Review of chi-square distribution

If X_i are k independent, normally distributed random variables with mean 0 and variance 1, then the random variable $Q = \sum_{i=1}^{k} X_i^2$

is distributed according to the chi-square distribution:

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

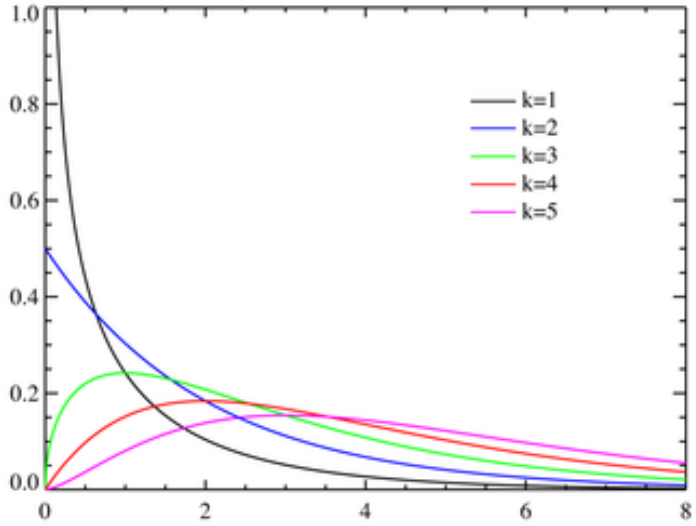
The chi-square distribution has one parameter: k - a positive integer that specifies the number of degrees of freedom.

A probability density function of the chi-square distribution is

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

Where the Gamma function is:

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

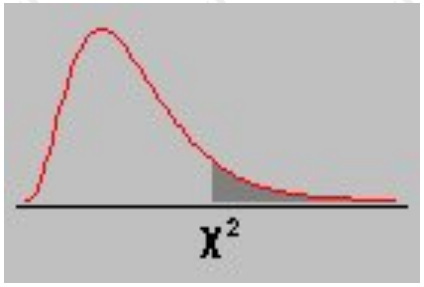


Review of chi-squared test

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

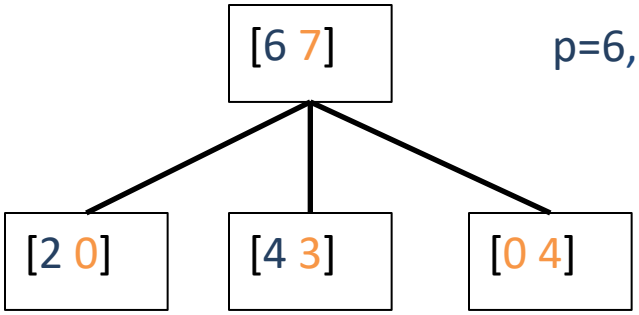
df\area	.995	.990	.975	.950	.900	.750	.500	.250	.100	.050	.025	.010	.005
1	0.00004	0.00016	0.00098	0.00393	0.01579	0.10153	0.45494	1.32330	2.70554	3.84146	5.02389	6.63490	7.87944
2	0.01003	0.02010	0.05064	0.10259	0.21072	0.57536	1.38629	2.77259	4.60517	5.99146	7.37776	9.21034	10.59663
3	0.07172	0.11483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

What you need:
Q and degree of freedom (df)
df = (number of values of this attribute) - 1
= (number of subsets split by this attribute) -1



Prune based on Chi-squared test

Example:



MaxPchance: the worst chance we are willing to accept

$$\hat{p}_1 = \frac{6}{6+7} * 2 = 0.92 \qquad \hat{p}_2 = \frac{6}{6+7} * 7 = 3.23 \qquad \hat{p}_3 = \frac{6}{6+7} * 4 = 1.85$$

$$\hat{n}_1 = \frac{7}{6+7} * 2 = 1.08 \qquad \hat{n}_2 = \frac{7}{6+7} * 7 = 3.77 \qquad \hat{n}_3 = \frac{7}{6+7} * 4 = 2.15$$

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

$$= \frac{(2-0.92)^2}{0.92} + \frac{(0-1.08)^2}{1.08} + \frac{(4-3.23)^2}{3.23} + \frac{(3-3.77)^2}{3.77} + \frac{(0-1.85)^2}{1.85} + \frac{(4-2.15)^2}{2.15}$$

$$= 6.13$$

Degree of freedom (df) = 3-1=2, so p_{chance} = 0.0466 ↔ *Compare this to a MaxPchance*

Using Chi-squared to avoid overfitting

Build the full decision tree as before.

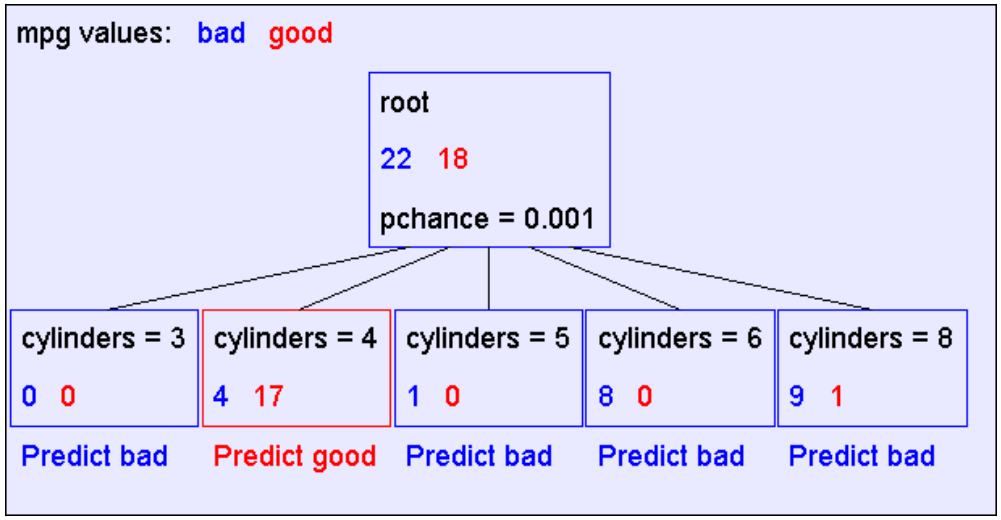
But when you can grow it no more, start to prune:

> 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 = 0.01, you will see the following MPG decision tree:



the improved test set accuracy compared with the unpruned tree

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

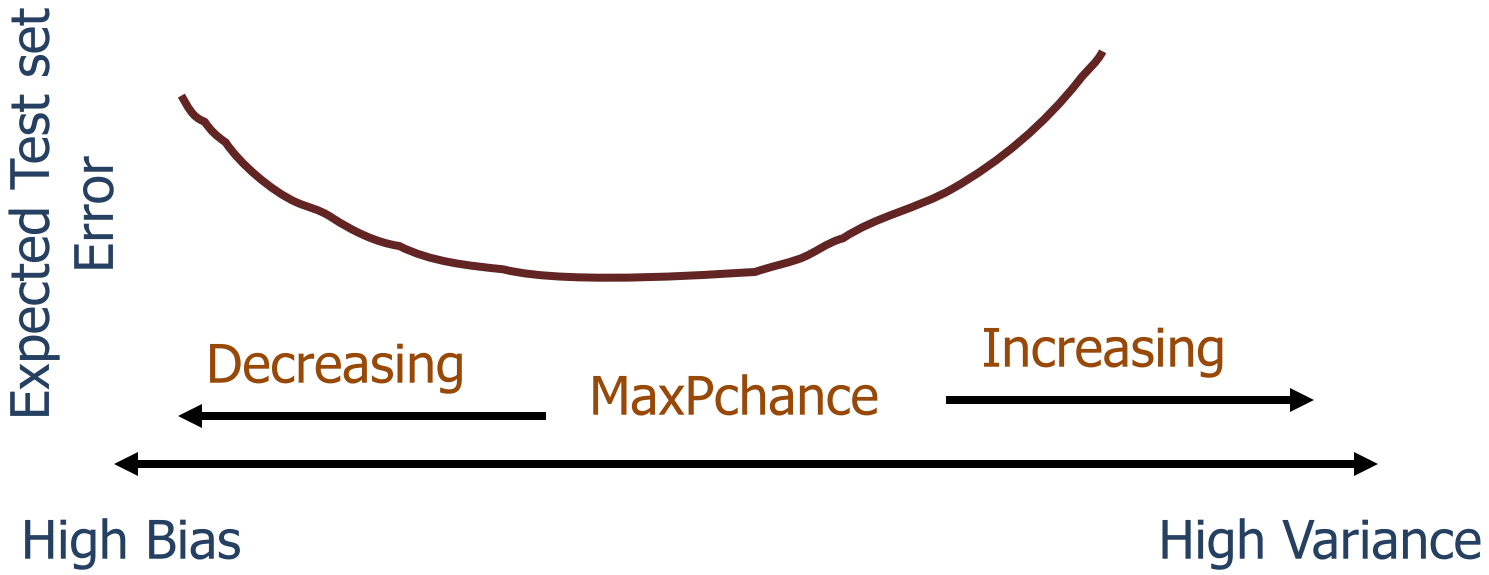
MaxPchance

Good news: The decision tree can automatically adjust its pruning decisions according to the amount of apparent noise and data.

Bad news: The user must come up with a good value of MaxPchance. (for instance, 0.05 as a magic parameter).

Good news: But with extra work, the best MaxPchance value can be estimated automatically by ***cross-validation***.

MaxPchance


Expected Test set Error
Decreasing
MaxPchance
Increasing
High Bias
High Variance

Gini Impurity

- An alternative impurity measurement to entropy:

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

where $p_{i,k}$ is the ratio of class k instances among the training instances in the i-th node.

- Scikit-Learn uses the Classification and Regression Tree (CART) algorithm to train Decision Trees. CART cost function:

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

where G_{left} and G_{right} are gini impurity of left and right subset respectively; m_{left}/ m_{right} is the number of instances in the left/right subset. k, t_k are feature and its threshold respectively.

- Gini impurity vs. Entropy
  - Gini is slightly faster
  - Entropy tends to produce slightly more balanced trees
  - Most of the time they lead to similar trees

Decision Tree Regression

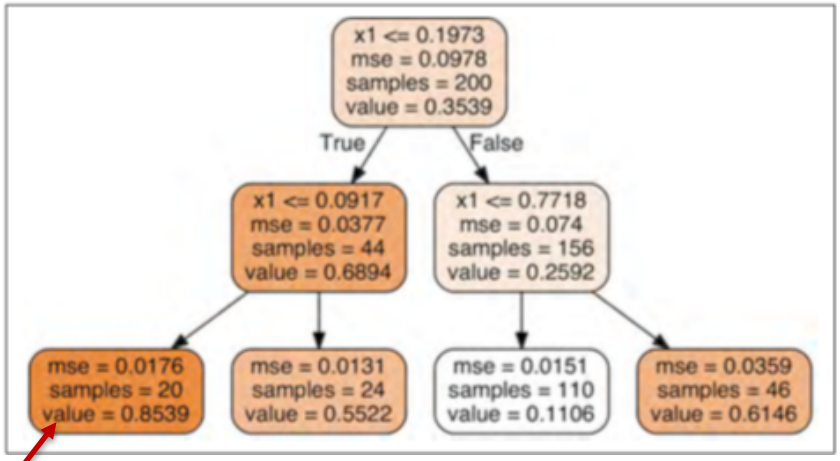
Using MSE as the cost function:

$$J(k, t_k) = \frac{m_{left}}{m} MSE_{left} + \frac{m_{right}}{m} MSE_{right}$$

Decision Tree Regression

Using MSE as the cost function:

$$J(k, t_k) = \frac{m_{left}}{m} MSE_{left} + \frac{m_{right}}{m} MSE_{right}$$



Predicted value = average of target value of all training instances of this leaf node

Acknowledgement

Part of the slide materials were based on Dr. Rong Duan's Fall 2016 course CPE/EE 695A Applied Machine Learning at Stevens Institute of Technology, and Dr. T. M. Mitchell's course slides.

Reference

The lecture notes in this lecture are based on the following textbooks:

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

STEVENS

INSTITUTE *of* TECHNOLOGY

THE INNOVATION UNIVERSITY®

stevens.edu