

Module 2 : Decision Trees

S. M. Jaisakti

Decision Trees Introduction

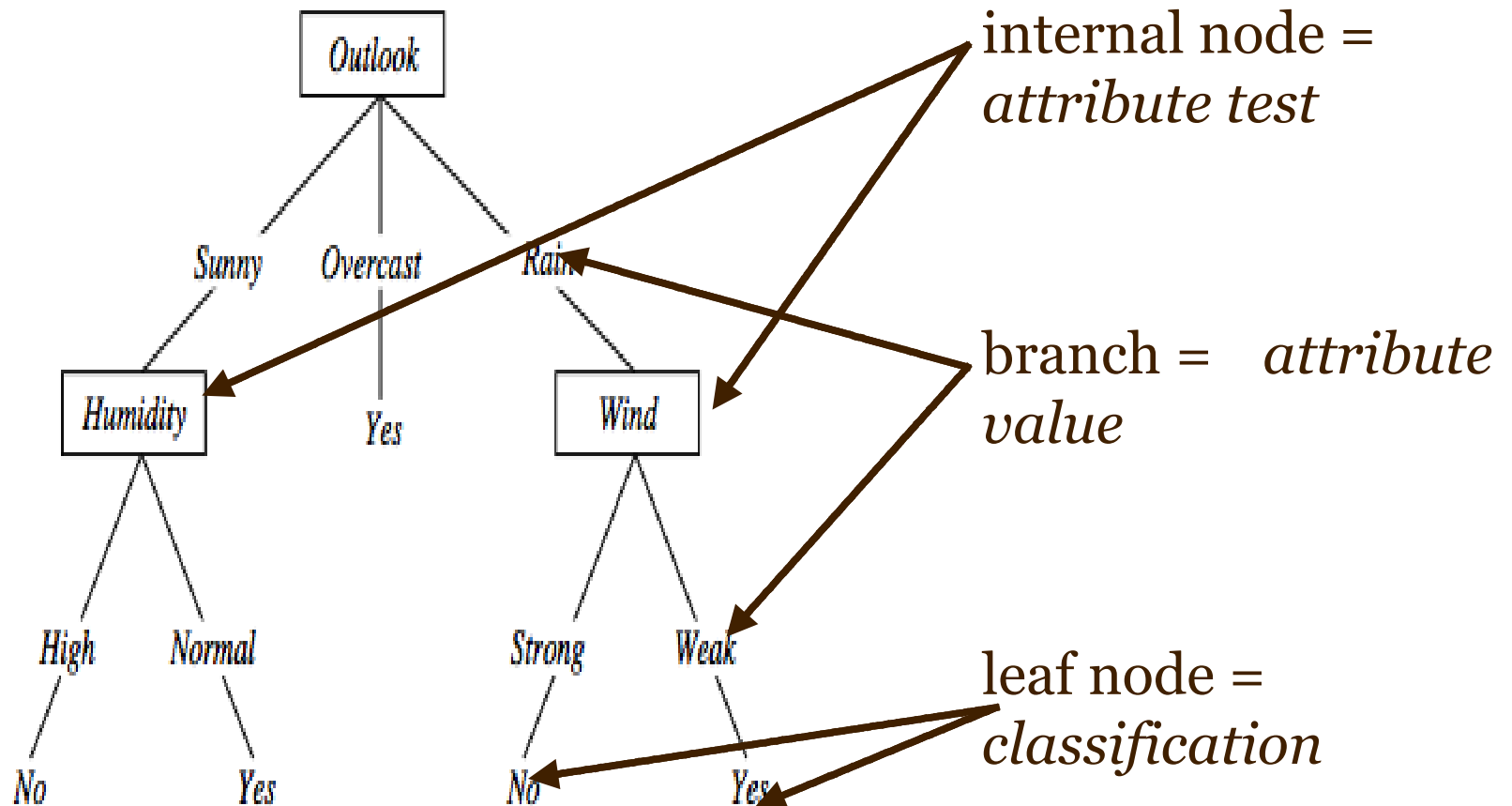
What is a decision tree ?

- ❏ Decision support tool that uses a tree like model of decisions and their consequences.
- ❏ Each internal node represents test on an attribute, each branch represents outcome of the test and each leaf node represents a class label.
- ❏ The path from root to leaf represents the class label.

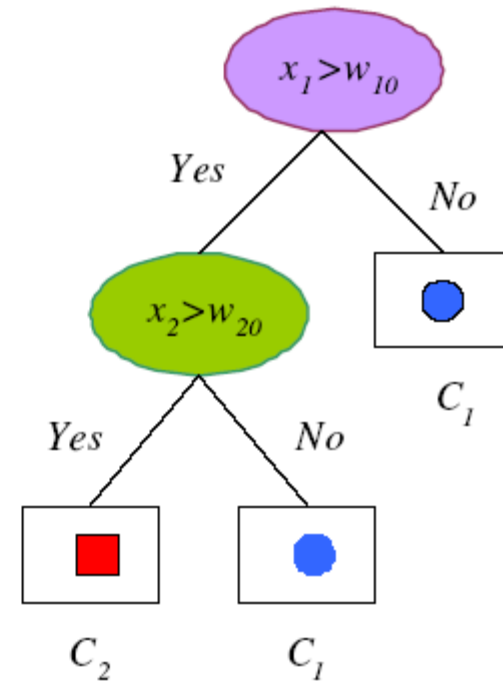
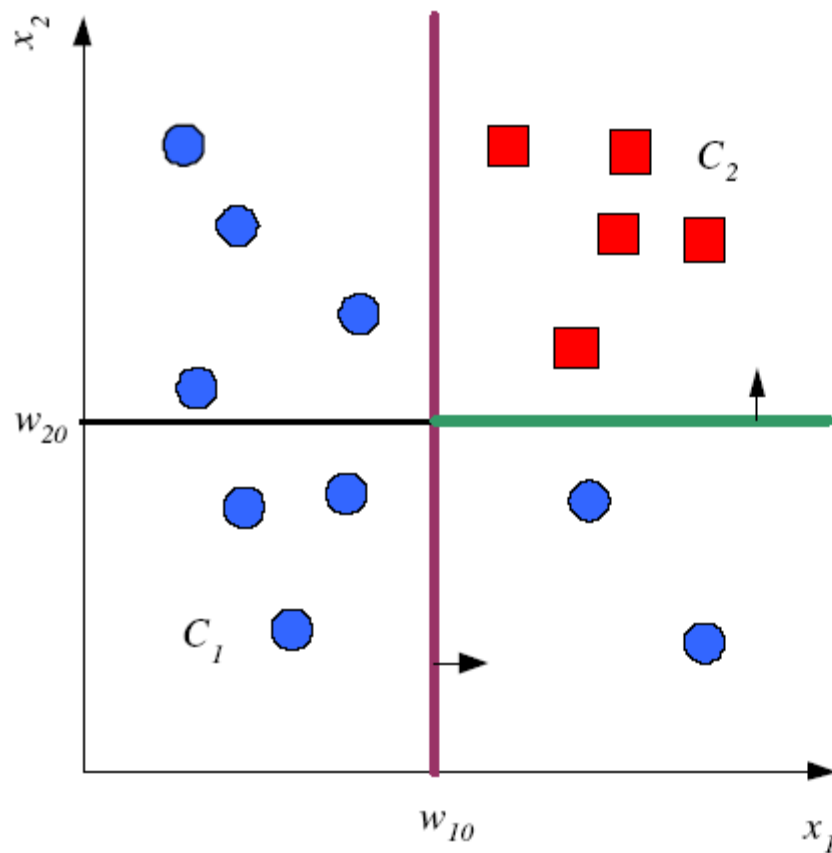
Decision Trees

- ❏ Non Parametric model
- ❏ Decision trees are supervised learning techniques.
- ❏ They can model non linear relationships well and can be used for both classification and regression problems.
- ❏ It is a simple way of inducing rules.
- ❏ Ex:- If (Age is x) and (income is y) and (family size is z) and (credit card spending is p), then he will accept the loan.

Representation



Decision Trees



When to use DT?

❏ Problem characteristics:

- ❏ Instances can be described by attribute value pairs
- ❏ Target function is discrete valued
- ❏ Disjunctive hypothesis may be required
- ❏ Possibly noisy training data samples
 - ❏ Robust to errors in training data
 - ❏ Missing attribute values

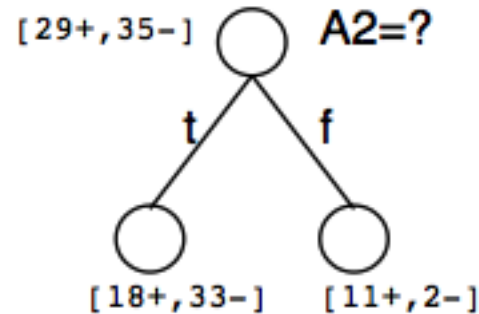
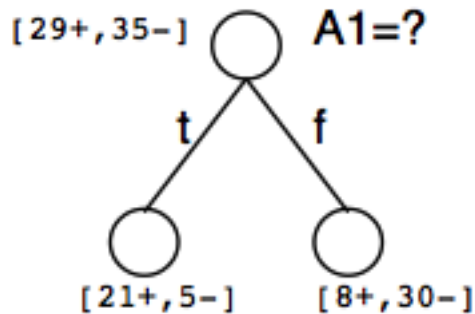
❏ Different **classification problems**:

- ❏ Equipment or medical diagnosis
- ❏ Credit risk analysis
- ❏ Several tasks in natural language processing

Basic DT Algorithm

- ❏ ID3 (Quinlan, 1986) is a basic algorithm for learning DT's
- ❏ Given a training set of examples, the algorithm for building DT performs search in the space of decision trees
- ❏ The construction of the tree is top-down. The algorithm is greedy.
- ❏ The fundamental question is “which attribute should be tested next? Which question gives us more information?”
- ❏ Select the *best* attribute
- ❏ A descendent node is then created for each possible value of this attribute and examples are partitioned according to this value
- ❏ The process is repeated for each successor node until all the examples are classified correctly or there are no attributes left

Which attribute is best?



- ❖ A statistical property called *information gain*, measures how well a given attribute separates the training examples
- ❖ Information gain uses the notion of *entropy*, commonly used in information theory
- ❖ *Information gain = expected reduction of entropy*

Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

Entropy for Binary Classification

Entropy measures the *impurity* of a collection of examples. It depends from the distribution of the random variable p .

S is a collection of training examples

p_+ the proportion of positive examples in S

p_- the proportion of negative examples in S

$$\text{Entropy}(S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_- \quad [0 \log_2 0 = 0]$$

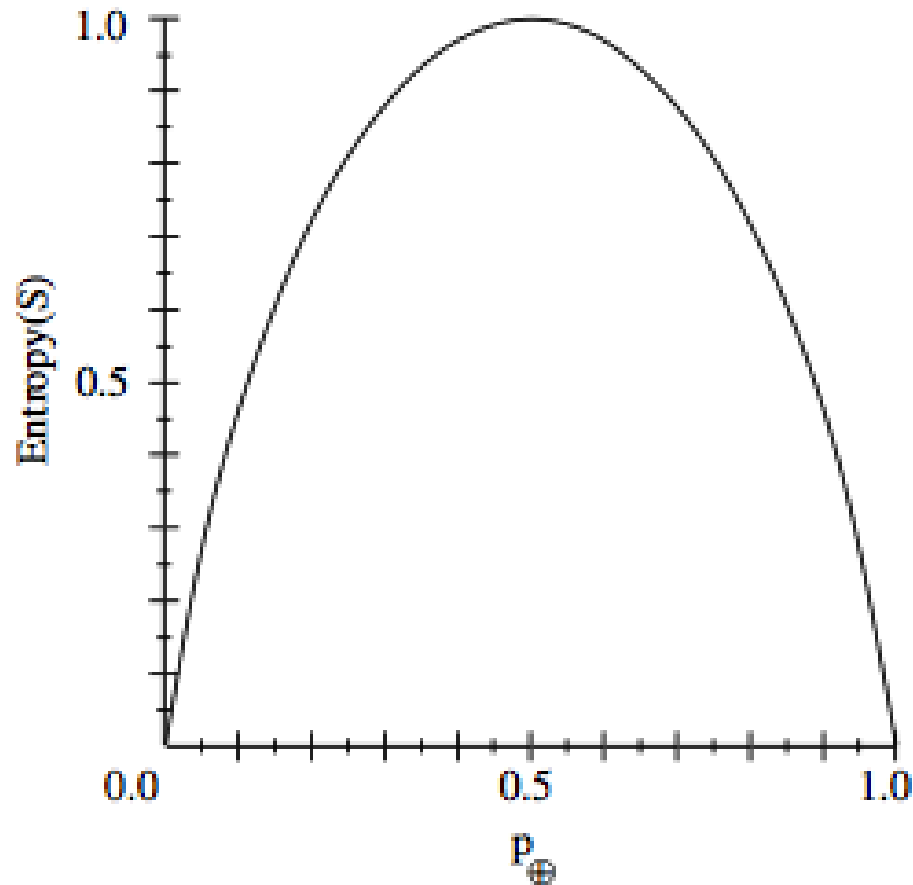
$$\text{Entropy}([14+, 0-]) = -14/14 \log_2 (14/14) - 0 \log_2 (0) = 0$$

$$\text{Entropy}([9+, 5-]) = -9/14 \log_2 (9/14) - 5/14 \log_2 (5/14) = 0.94$$

$$\begin{aligned} \text{Entropy}([7+, 7-]) &= -7/14 \log_2 (7/14) - 7/14 \log_2 (7/14) = \\ &= 1/2 + 1/2 = 1 \quad [\log_2 1/2 = -1] \end{aligned}$$

Note: the log of a number < 1 is negative, $0 \leq p \leq 1$, $0 \leq \text{entropy} \leq 1$

Entropy Function Relative to Binary Classification



Entropy in General

Entropy measures the amount of information in a random variable

$$H(X) = -p_+ \log_2 p_+ - p_- \log_2 p_- \quad X = \{+, -\}$$

for binary classification [two-valued random variable]

$$H(X) = - \sum_{i=1}^c p_i \log_2 p_i = \sum_{i=1}^c p_i \log_2 1/p_i \quad X = \{i, \dots, c\}$$

for classification in c classes

Example: rolling a die with 8, equally probable, sides

$$H(X) = - \sum_{i=1}^8 1/8 \log_2 1/8 = - \log_2 1/8 = \log_2 8 = 3$$

Information Gain

- ❏ *Information gain* is the *expected* reduction in entropy caused by partitioning the examples on an attribute.
- ❏ The higher the information gain the more effective the attribute in classifying training data.
- ❏ Expected reduction in entropy knowing A

$$\text{Entropy} = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

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

$\text{Values}(A)$ possible values for A

S_v subset of S for which A has value v

Information Gain : Example

Let

▣ $Values(Wind) = \{Weak, Strong\}$

▣ $S = [9+, 5-]$

▣ $S_{Weak} = [6+, 2-]$

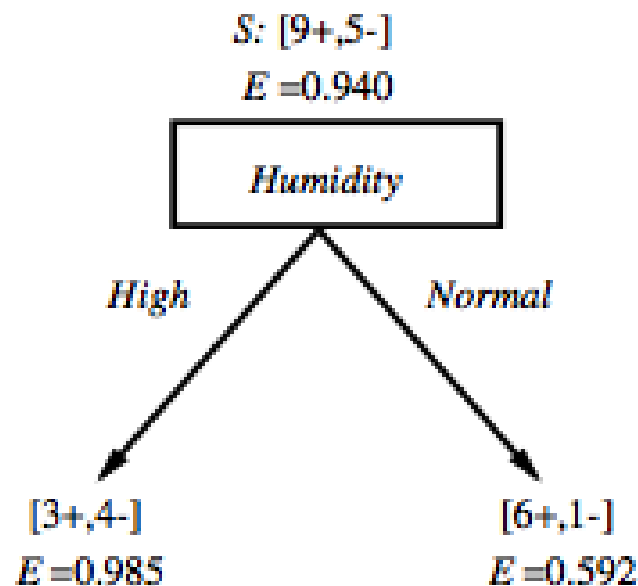
▣ $S_{Strong} = [3+, 3-]$

▣ Information gain due to knowing *Wind*:

$$\begin{aligned} Gain(S, Wind) &= Entropy(S) - 8/14 Entropy(S_{Weak}) - \\ &\quad 6/14 Entropy(S_{Strong}) \\ &= 0.94 - 8/14 \times 0.811 - 6/14 \times 1.00 \\ &= 0.048 \end{aligned}$$

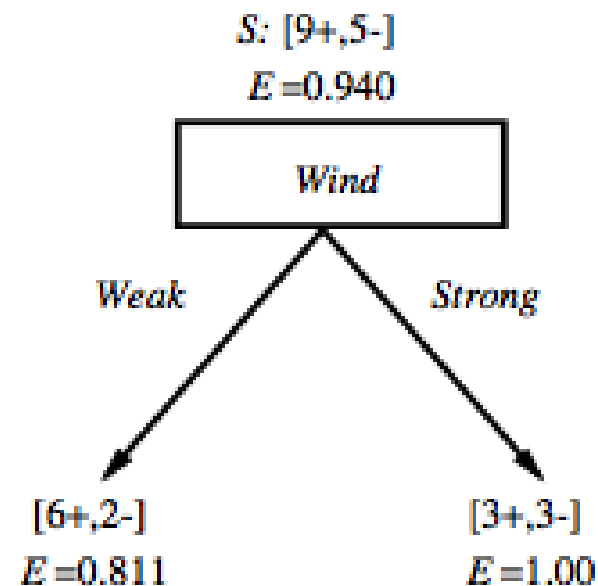
Information Gain : Example

Which attribute is the best classifier?



$Gain(S, Humidity)$

$$\begin{aligned} &= .940 - (7/14).985 - (7/14).592 \\ &= .151 \end{aligned}$$



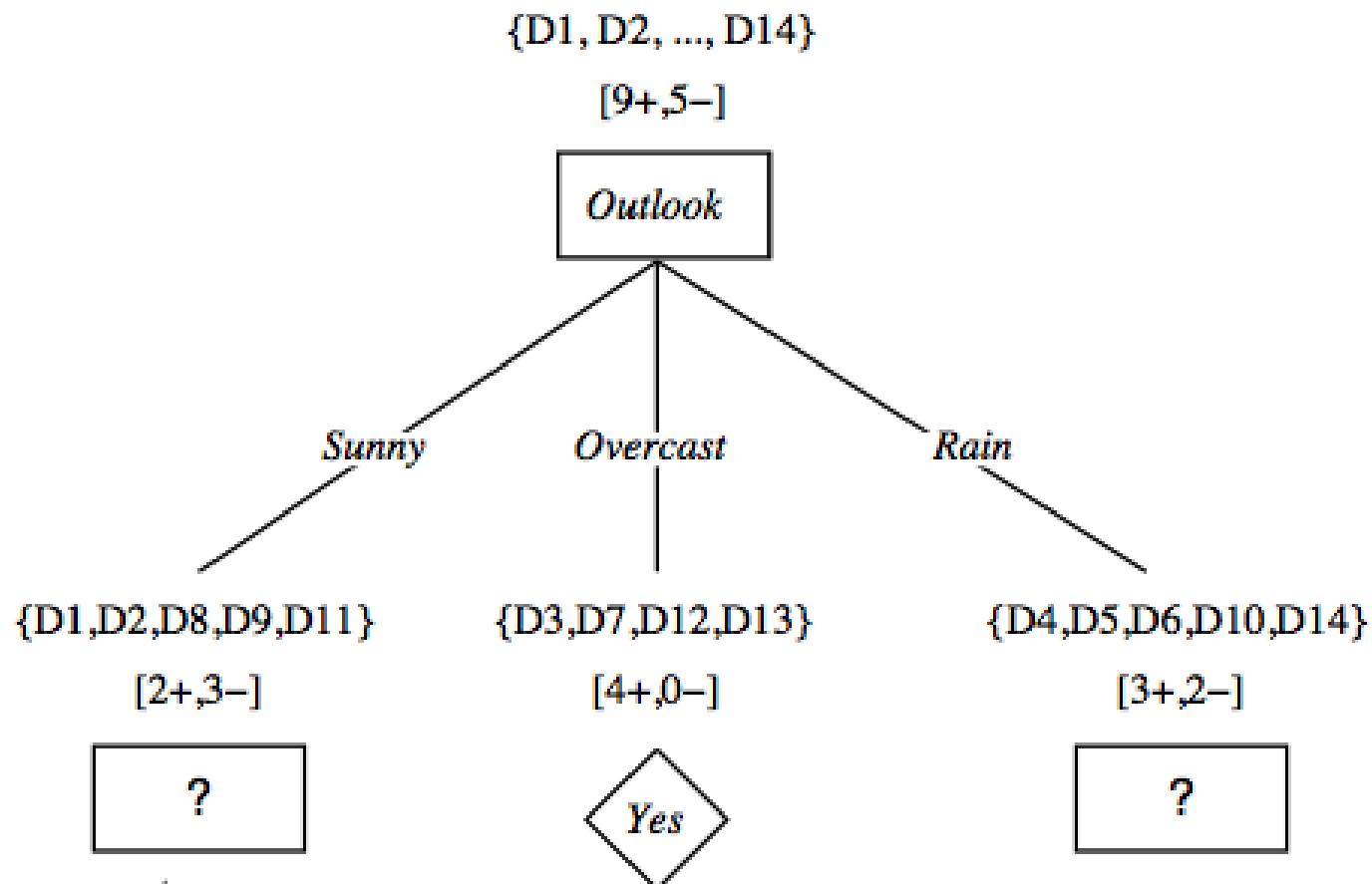
$Gain(S, Wind)$

$$\begin{aligned} &= .940 - (8/14).811 - (6/14)1.0 \\ &= .048 \end{aligned}$$

Building DT : Step 1

- ❏ Which attribute should be tested at the root?
 - ❏ $Gain(S, Outlook) = 0.246$
 - ❏ $Gain(S, Humidity) = 0.151$
 - ❏ $Gain(S, Wind) = 0.084$
 - ❏ $Gain(S, Temperature) = 0.029$
- ❏ *Outlook* provides the best prediction for the target
- ❏ Lets grow the tree:
 - ❏ add to the tree a successor for each possible value of *Outlook*
 - ❏ partition the training samples according to the value of *Outlook*

Building DT : Step -1



Building DT : Step -2

❏ Working on *Outlook=Sunny* node:

$$\begin{aligned} \text{Gain}(S_{\text{Sunny}}, \text{Humidity}) &= 0.970 - 3/5 \times 0.0 - 2/5 \times 0.0 \\ &= 0.970 \end{aligned}$$

$$\begin{aligned} \text{Gain}(S_{\text{Sunny}}, \text{Wind}) &= 0.970 - 2/5 \times 1.0 - 3.5 \times 0.918 = \\ &0.019 \end{aligned}$$

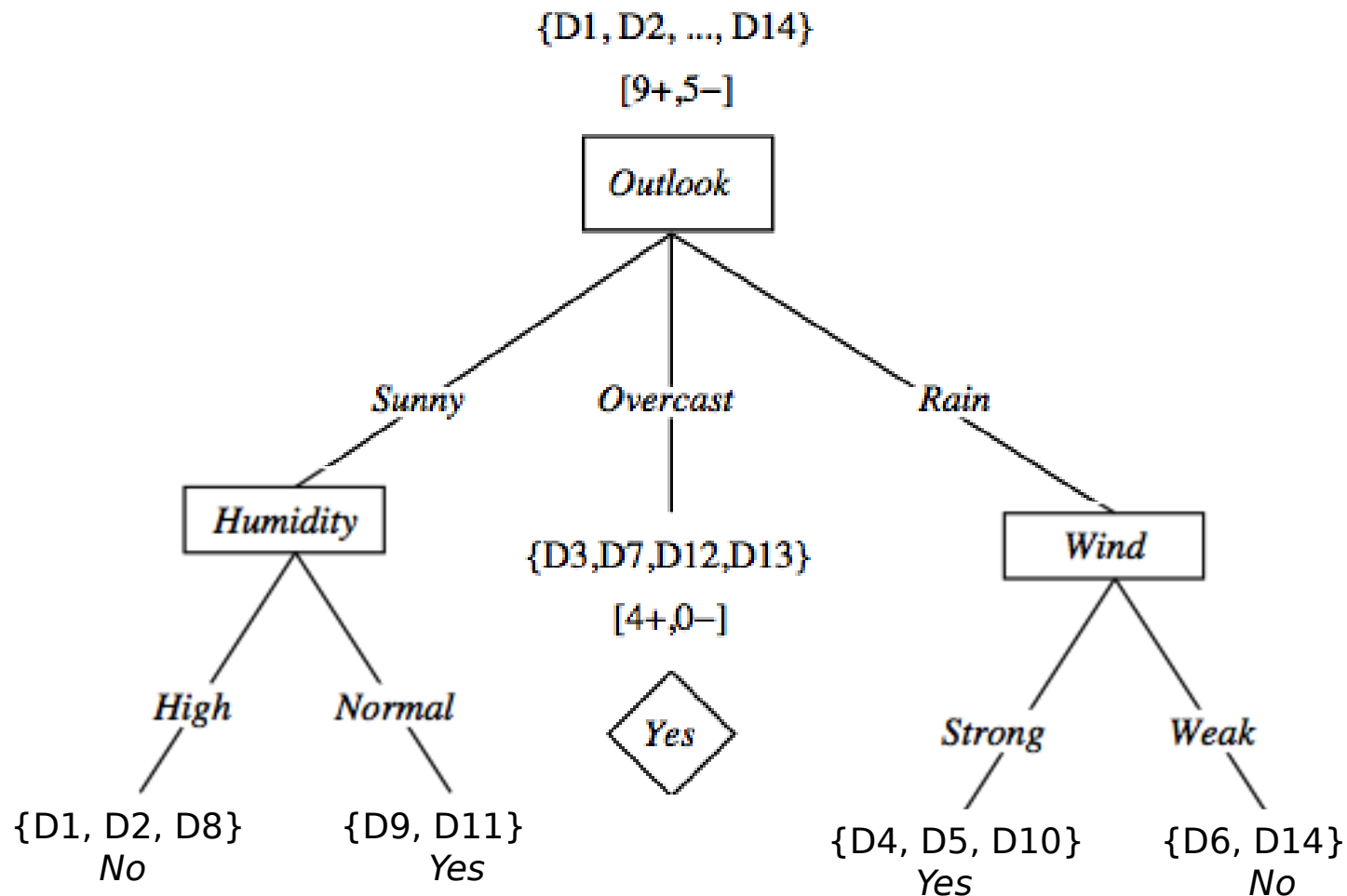
$$\begin{aligned} \text{Gain}(S_{\text{Sunny}}, \text{Temp.}) &= 0.970 - 2/5 \times 0.0 - 2/5 \times 1.0 - \\ &1/5 \times 0.0 = 0.570 \end{aligned}$$

❏ *Humidity* provides the best prediction for the target

❏ Lets grow the tree:

- ❏ add to the tree a successor for each possible value of *Humidity*
- ❏ partition the training samples according to the value of *Humidity*

Building DT : Step -2



Issues in DTs

❏ Overfitting

- ❖ Reduced error pruning
- ❖ Rule post-pruning

❏ Extensions

- ❖ Continuous valued attributes
- ❖ Alternative measures for selecting attributes
- ❖ Handling training examples with missing attribute values
- ❖ Handling attributes with different costs
- ❖ Improving computational efficiency
- ❖ Most of these improvements in C4.5 (Quinlan, 1993)

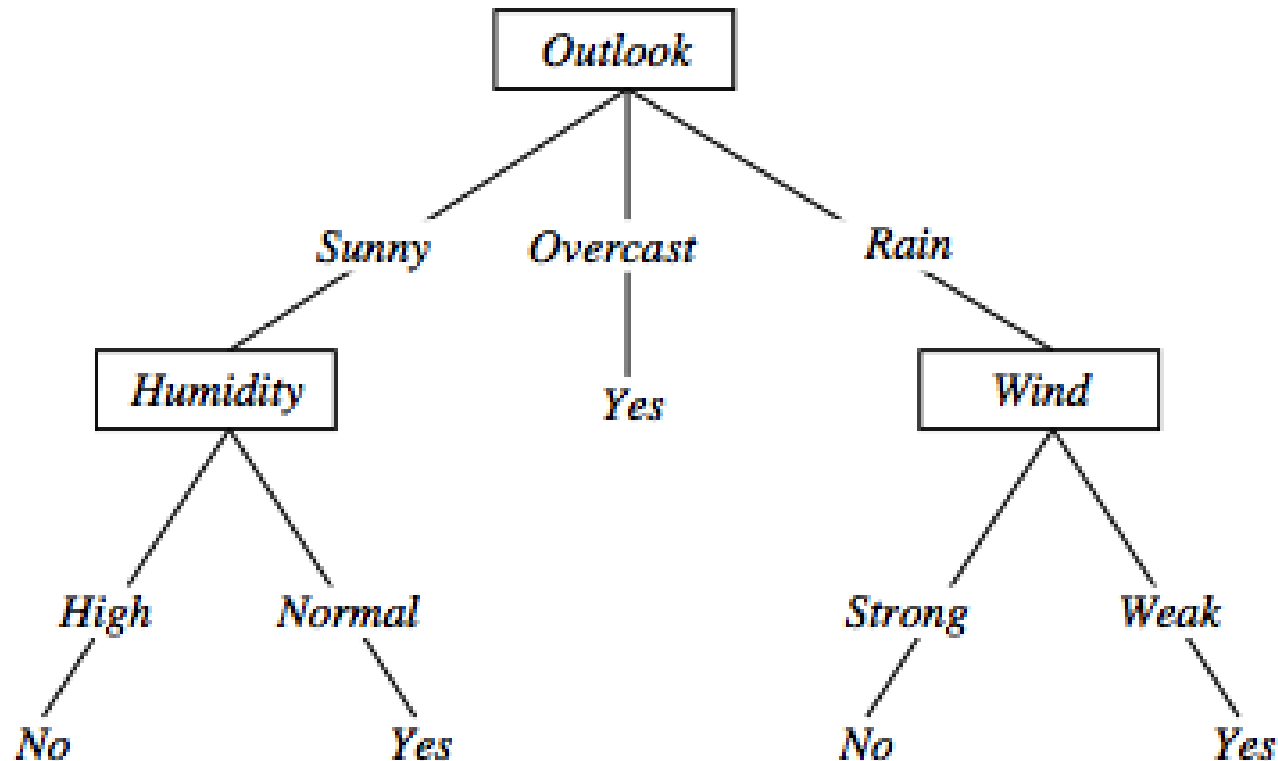
Overfitting

- ❏ Building trees that “adapt too much” to the training examples may lead to “overfitting”.
- ❏ Consider error of hypothesis h over
 - ❖ training data: $error_D(h)$ empirical error
 - ❖ entire distribution X of data: $error_X(h)$ expected error
- ❏ Hypothesis h **overfits** training data if there is an alternative hypothesis $h' \in H$ such that
$$error_D(h) < error_D(h') \quad \text{and}$$
$$error_X(h') < error_X(h)$$
i.e. h' behaves better over unseen data

Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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
D15	Sunny	Hot	Normal	Strong	No

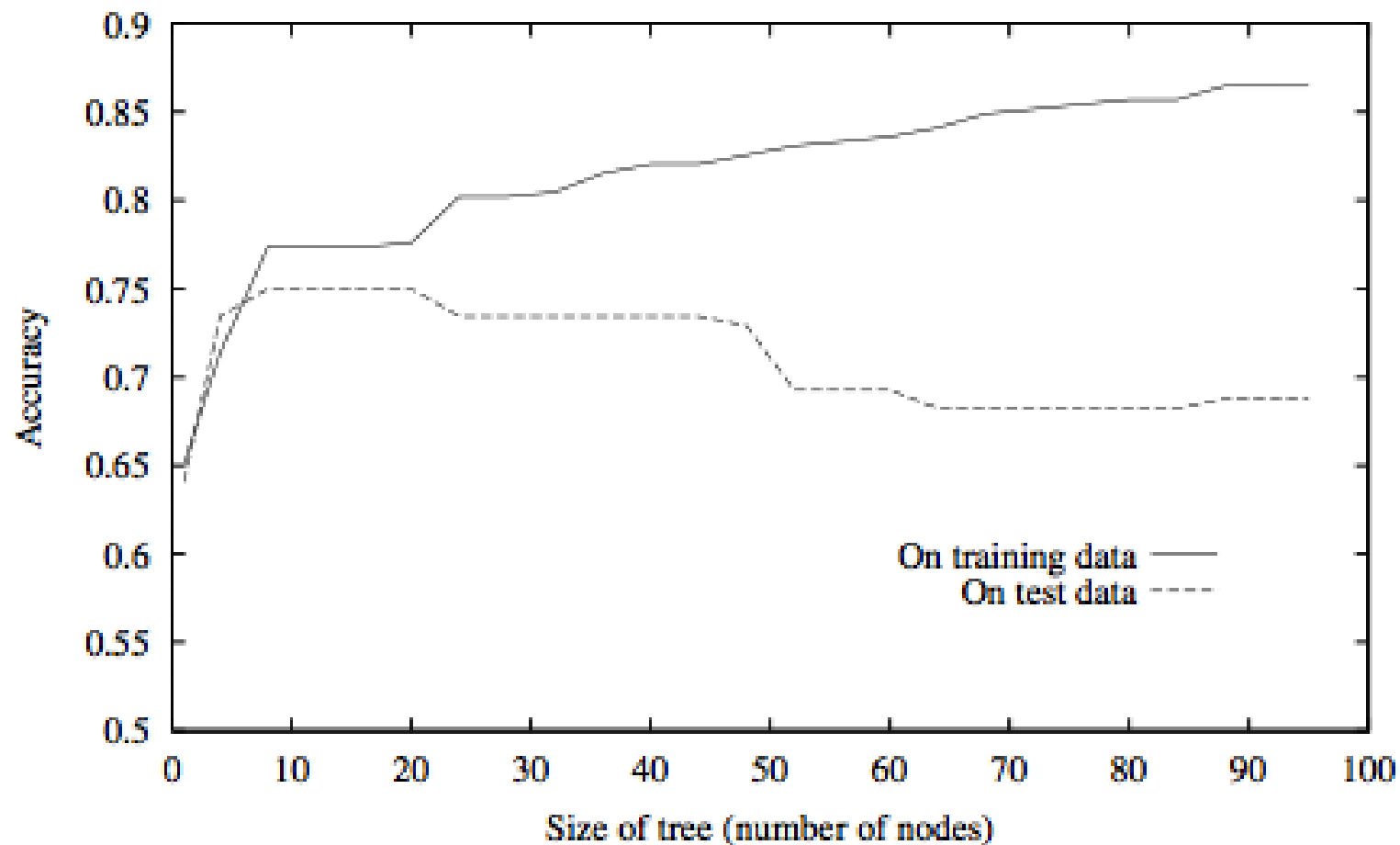
Overfitting in DTs



⟨*Outlook=Sunny, Temp=Hot, Humidity=Normal, Wind=Strong, PlayTennis=No* ⟩

New noisy example causes splitting of second leaf node.

Overfitting in DTs



Avoiding Overfitting

Two strategies:

- Stop growing the tree earlier, before perfect classification
- Allow the tree to *overfit* the data, and then *post-prune* the tree

Training and validation set

- split the training in two parts (training and validation) and use validation to assess the utility of *post-pruning*

- Reduced error pruning*

- Rule pruning*

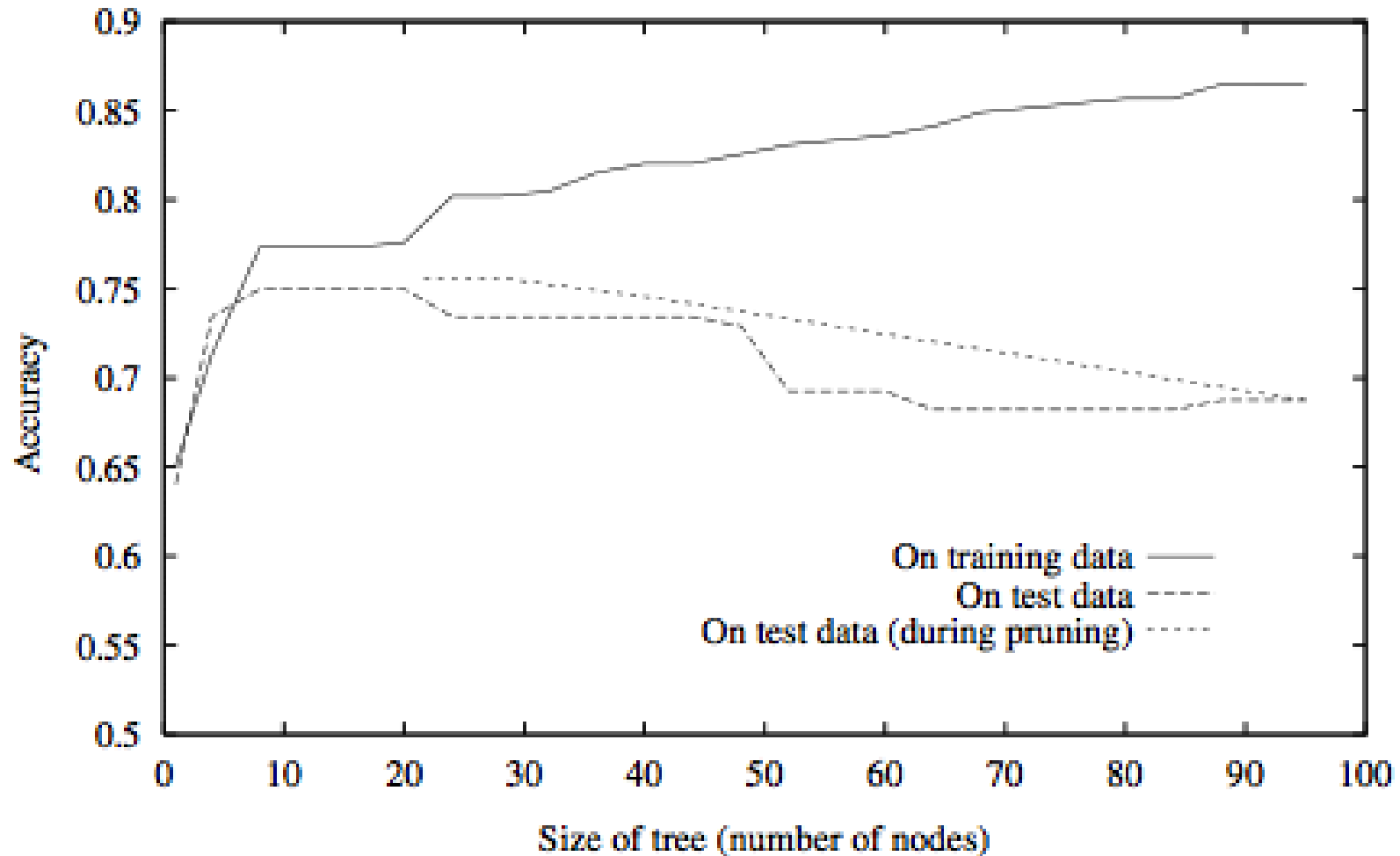
Other approaches

- Use a statistical test to estimate effect of expanding or pruning
- Minimum description length principle*: uses a measure of complexity of encoding the DT and the examples, and halt growing the tree when this encoding size is minimal

Reduced Error Pruning

- ❏ Each node is a candidate for pruning
- ❏ *Pruning* consists in removing a subtree rooted in a node: the node becomes a leaf and is assigned the most common classification
- ❏ Nodes are removed only if the resulting tree performs no worse **on the validation set**.
- ❏ Nodes are pruned iteratively: at each iteration the node whose removal most increases accuracy on the validation set is pruned.
- ❏ Pruning stops when no pruning increases accuracy

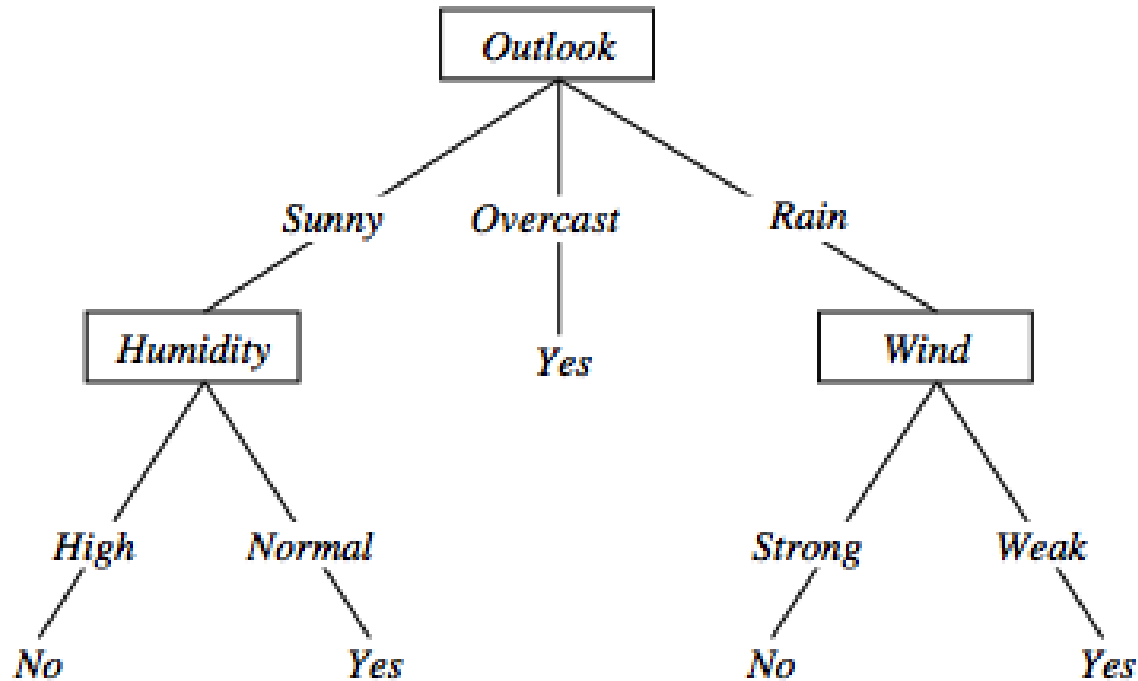
Effect of Reduced Error Pruning



Post Pruning Rules

- ❏ Create the decision tree from the training set
- ❏ Convert the tree into an equivalent set of rules
 - ❖ Each path corresponds to a rule
 - ❖ Each node along a path corresponds to a pre-condition
 - ❖ Each leaf classification to the post-condition
- ❏ Prune (generalize) each rule by removing those preconditions whose removal improves accuracy ...
 - ❖ ... over validation set
 - ❖ ... over training with a pessimistic, statistically inspired, measure
- ❏ Sort the rules in estimated order of accuracy, and consider them in sequence when classifying new instances

Converting to Rules



$(Outlook = Sunny) \wedge (Humidity = High) \Rightarrow (playTennis = No)$

Why Converting Rules?

- ❏ Each distinct path produces a different rule: a condition removal may be based on a local (contextual) criterion. Node pruning is global and affects all the rules
- ❏ In rule form, tests are not ordered and there is no book-keeping involved when conditions (nodes) are removed
- ❏ Converting to rules improves readability for humans

Dealing with Continuous Valued Attributes

- ❏ So far discrete values for attributes and for outcome.
- ❏ Given a continuous-valued attribute A , dynamically create a new attribute A_c

$A_c = \text{True if } A < c, \text{ False otherwise}$

- ❏ How to determine threshold value c ?
- ❏ Example. *Temperature* in the *PlayTennis* example
 - ❏ Sort the examples according to *Temperature*

Temperature 40 48 | 60 72 80 | 90

PlayTennis No No 54 Yes Yes Yes 85 No

- ❏ Determine candidate thresholds by averaging consecutive values where there is a change in classification: $(48+60)/2=54$ and $(80+90)/2=85$
- ❏ Evaluate candidate thresholds (attributes) according to information gain. The best is $\text{Temperature}_{>54}$. The new attribute competes with the other ones

Problems with Information Gain

- ❏ Natural bias of information gain: it favours attributes with many possible values.
- ❏ Consider the attribute *Date* in the *PlayTennis* example.
 - ❏ *Date* would have the highest information gain since it perfectly separates the training data.
 - ❏ It would be selected at the root resulting in a very broad tree
 - ❏ Very good on the training, this tree would perform poorly in predicting unknown instances. Overfitting.
- ❏ The problem is that the partition is too specific, too many small classes are generated.
- ❏ We need to look at alternative measures ...

Alternative Measures for Information Gain

$$\text{SplitInformation}(S, A) \equiv - \sum_{i=1}^c \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

S_i are the sets obtained by partitioning on value i of A

SplitInformation measures the entropy of S with respect to the values of A . The more uniformly dispersed the data the higher it is.

$$\text{Gain Ratio}(S, A) \equiv \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)}$$

GainRatio penalizes attributes that split examples in many small classes such as *Date*. Let $|S|=n$, *Date* splits examples in n classes

- $\text{SplitInformation}(S, \text{Date}) = -[(1/n \log_2 1/n) + \dots + (1/n \log_2 1/n)] = -\log_2 1/n = \log_2 n$

Compare with A , which splits data in two even classes:

- $\text{SplitInformation}(S, A) = -[(1/2 \log_2 1/2) + (1/2 \log_2 1/2)] = -[-1/2 -1/2] = 1$

Adjusting Gain Ratio

- ❏ Problem: $SplitInformation(S, A)$ can be zero or very small when $|S_i| \approx |S|$ for some value i
- ❏ To mitigate this effect, the following heuristics has been used:
 - ❖ apply $GainRatio$ only to attributes with $Gain$ above average
 - ❖ compute $Gain$ for each attribute
- ❏ Other measures have been proposed:
 - ❖ *Distance-based* metric [Lopez-De Mantaras, 1991] on the partitions of data
 - ❖ Each partition (induced by an attribute) is evaluated according to the distance to the partition that perfectly classifies the data.
 - ❖ The partition closest to the *ideal* partition is chosen

Handling Incomplete Training Data

- ❏ How to cope with the problem that the value of some attribute may be missing?
 - ❏ *Example:* Blood-Test-Result in a medical diagnosis problem
- ❏ The strategy: use other examples to guess attribute
 - ❏ Assign the value that is most common among the training examples at the node
 - ❏ Assign a probability to each value, based on frequencies, and assign values to missing attribute, according to this probability distribution
- ❏ Missing values in new instances to be classified are treated accordingly, and the most probable classification is chosen (C4.5)

Handling Attributes with Different Cost

Instance attributes may have an associated cost:
we would prefer decision trees that use low-cost attributes

ID3 can be modified to take into account costs:

Tan and Schlimmer (1990)

$$\frac{Gain^2(S, A)}{Cost(S, A)}$$

Nunez (1988)

$$\frac{2^{Gain(S, A)} - 1}{(cost(A) + 1)^w} \text{ where } w \in [0, 1]$$