Decision Trees

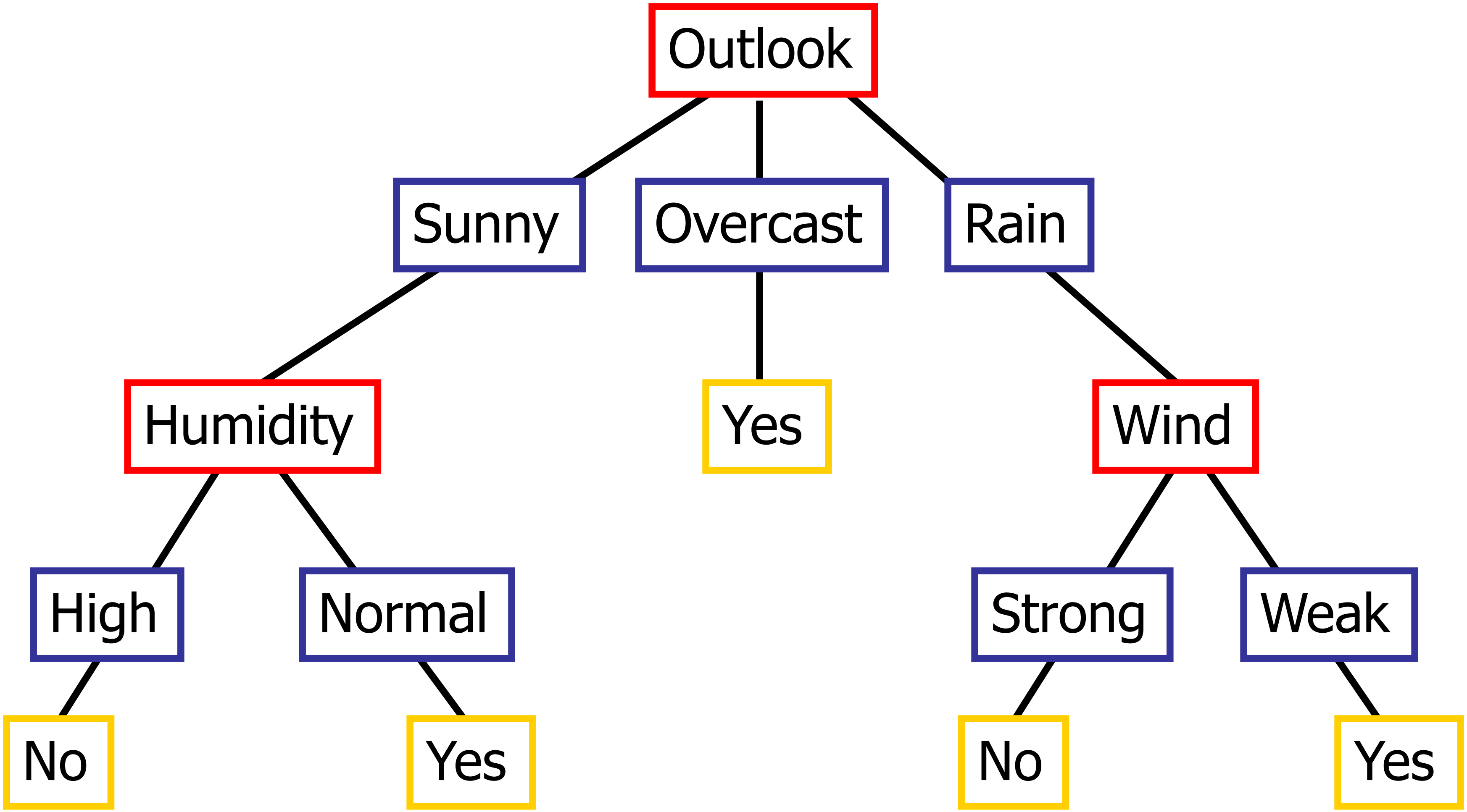
- Decision tree representation
- Iterative Dichotomiser 3 (ID3) learning algorithm
- Entropy, information gain
- Overfitting

Supplementary material

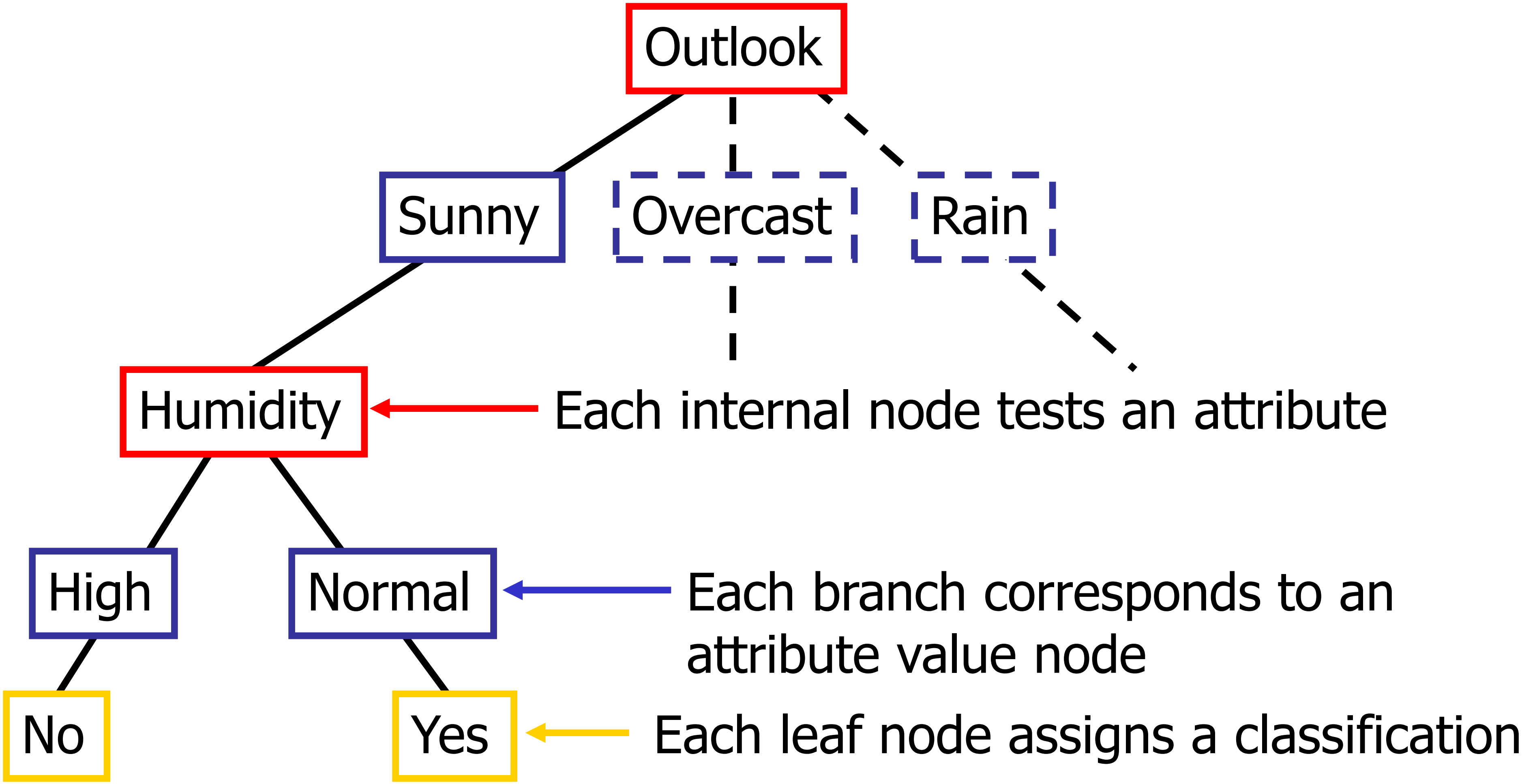
www

- http://dms.irb.hr/tutorial/tut_dtrees.php
- http://www.cs.uregina.ca/~dbd/cs831/notes/ml/dtrees/4_dtrees1.html

Decision Tree for PlayTennis

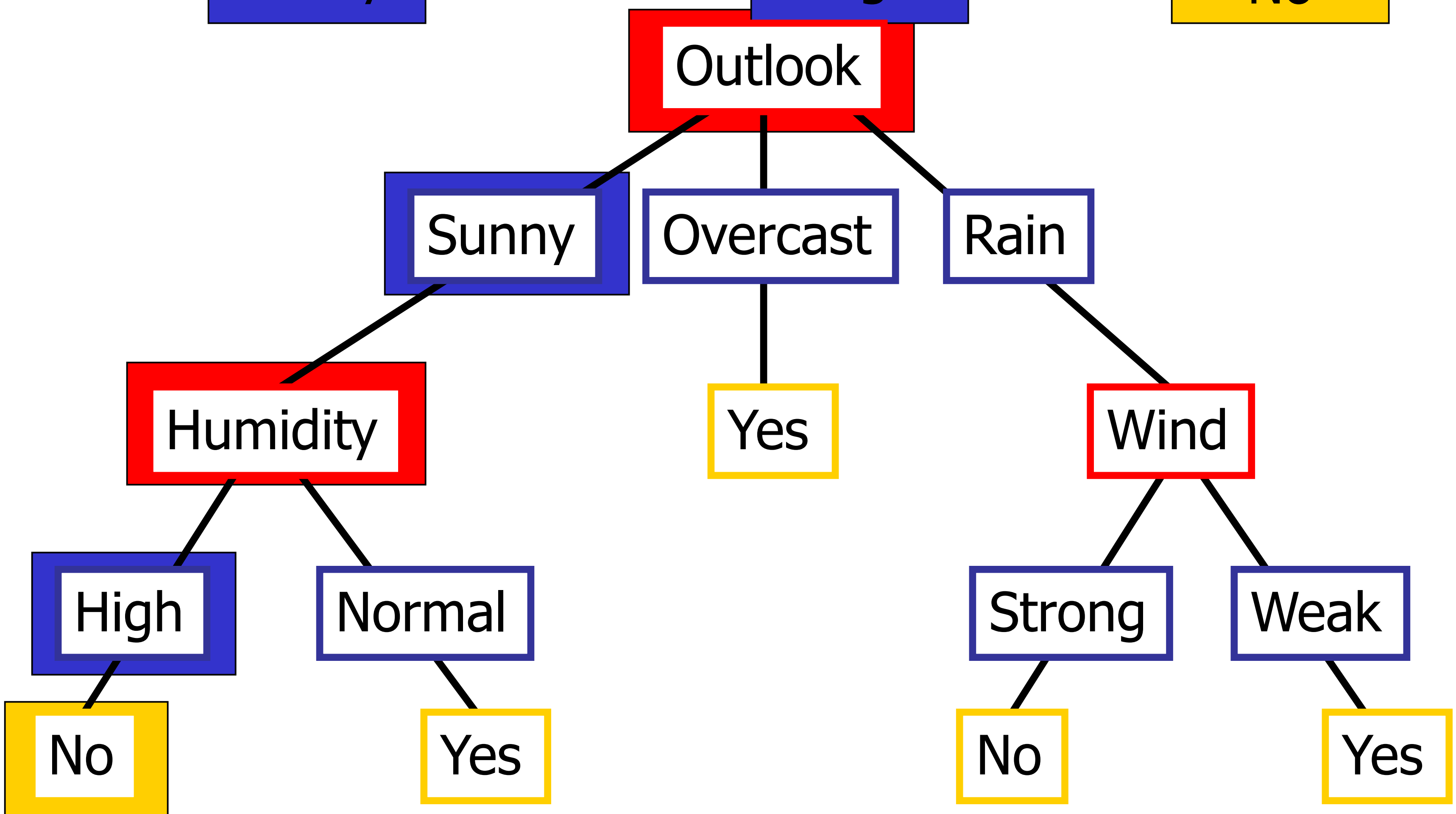


Decision Tree for PlayTennis



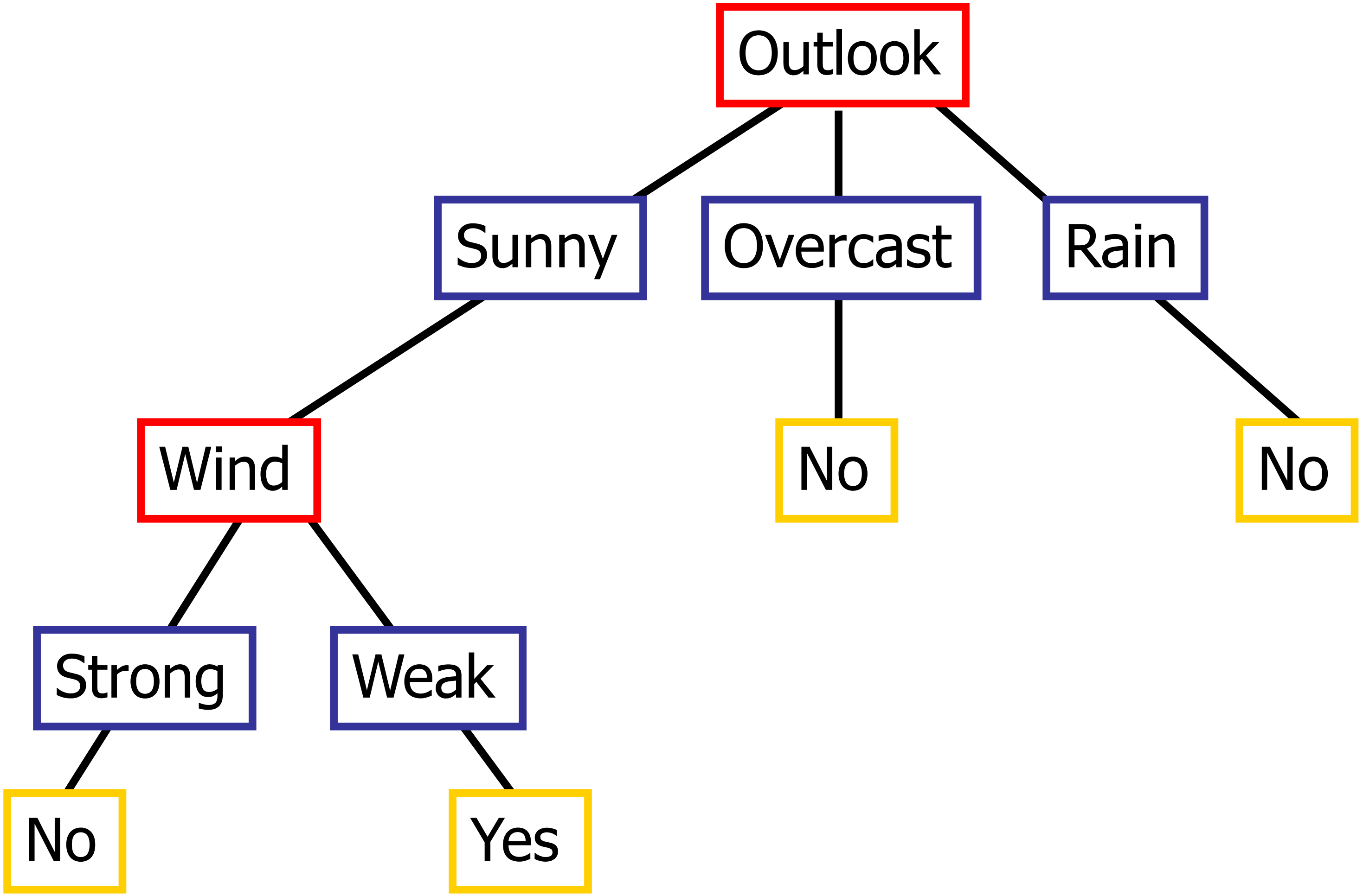
Decision Tree for PlayTennis

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	?No



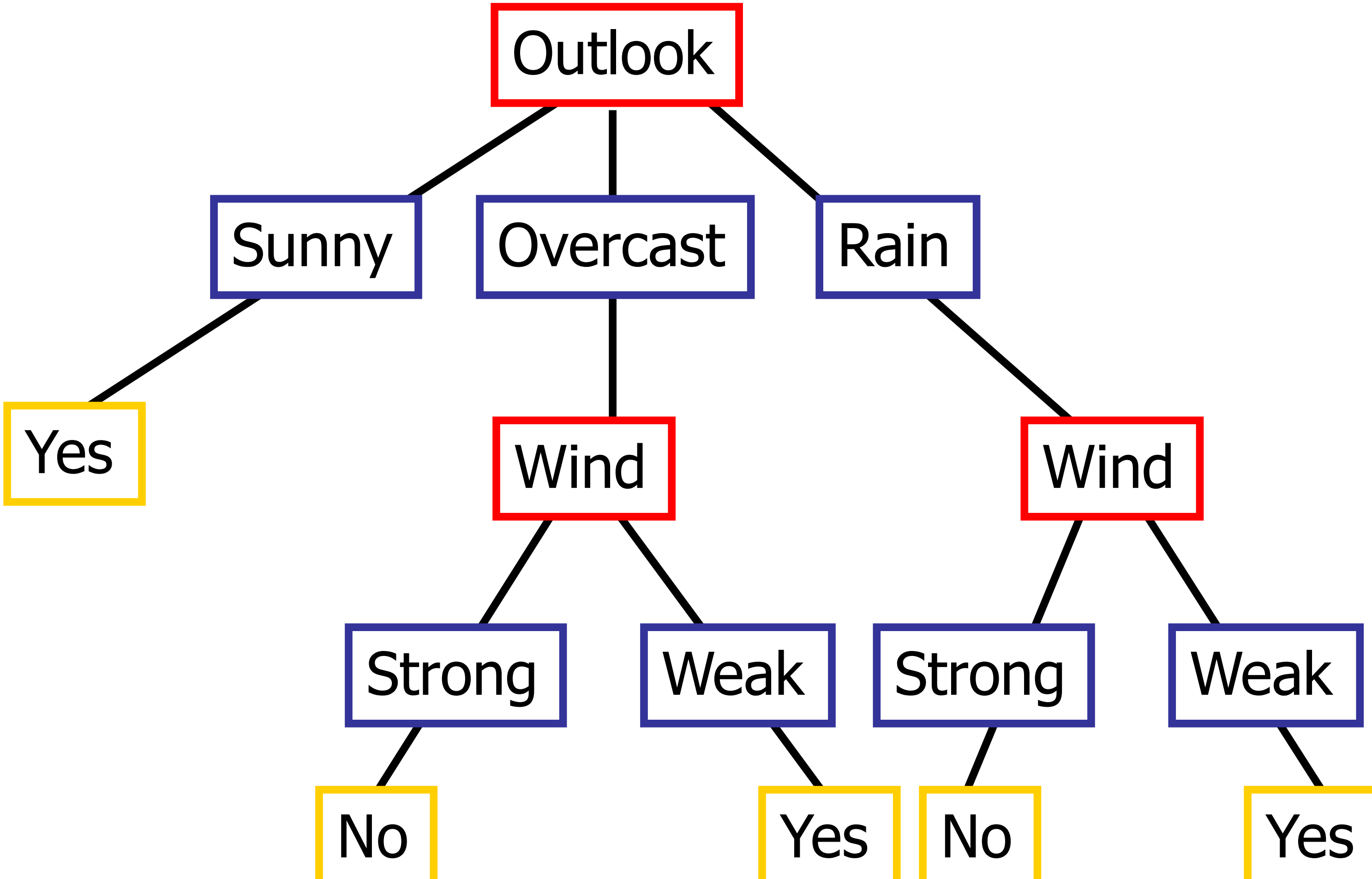
Decision Tree for Conjunction

Outlook=Sunny $\wedge$ Wind=Weak



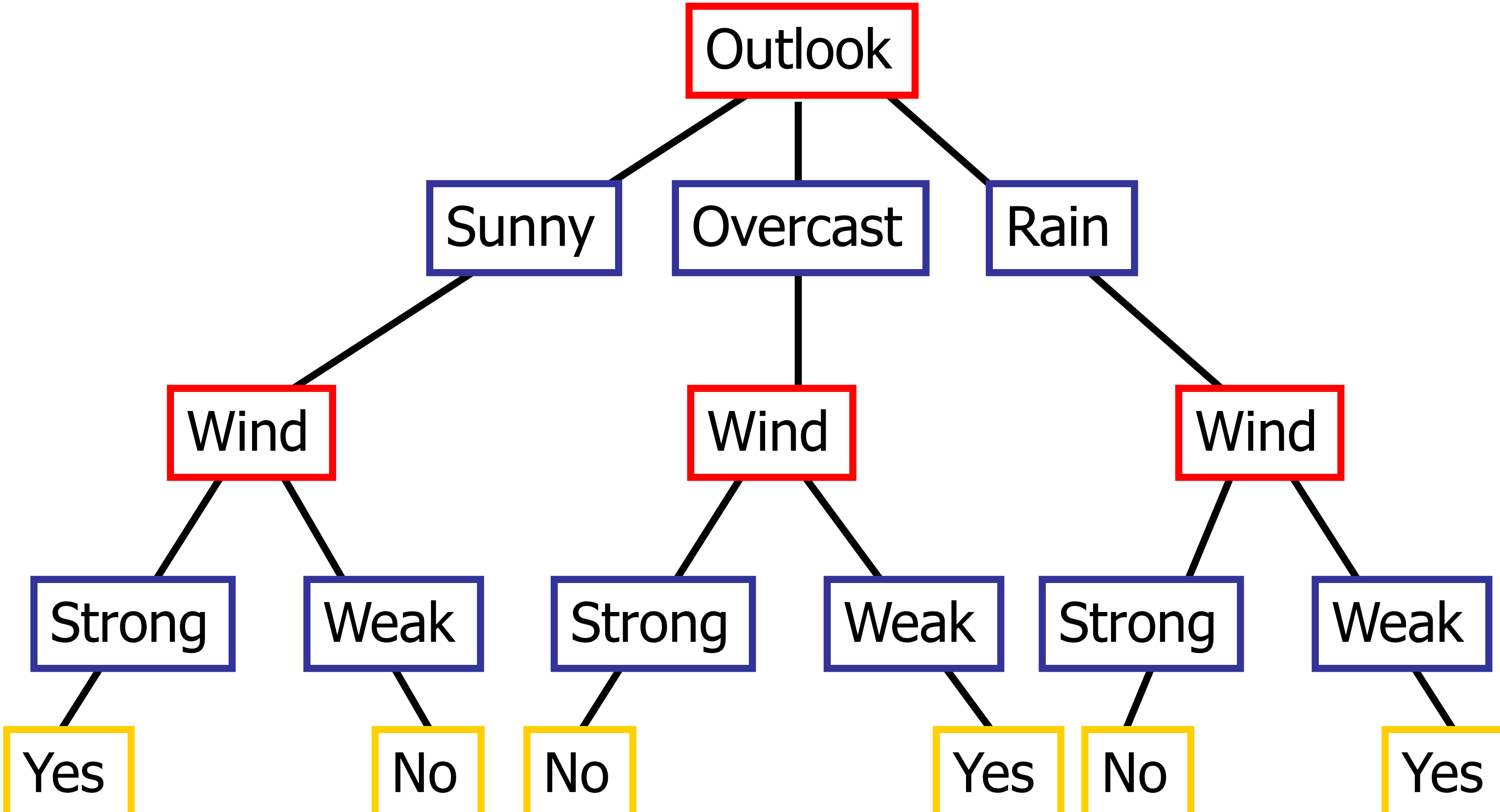
Decision Tree for Disjunction

Outlook=Sunny $\vee$ Wind=Weak



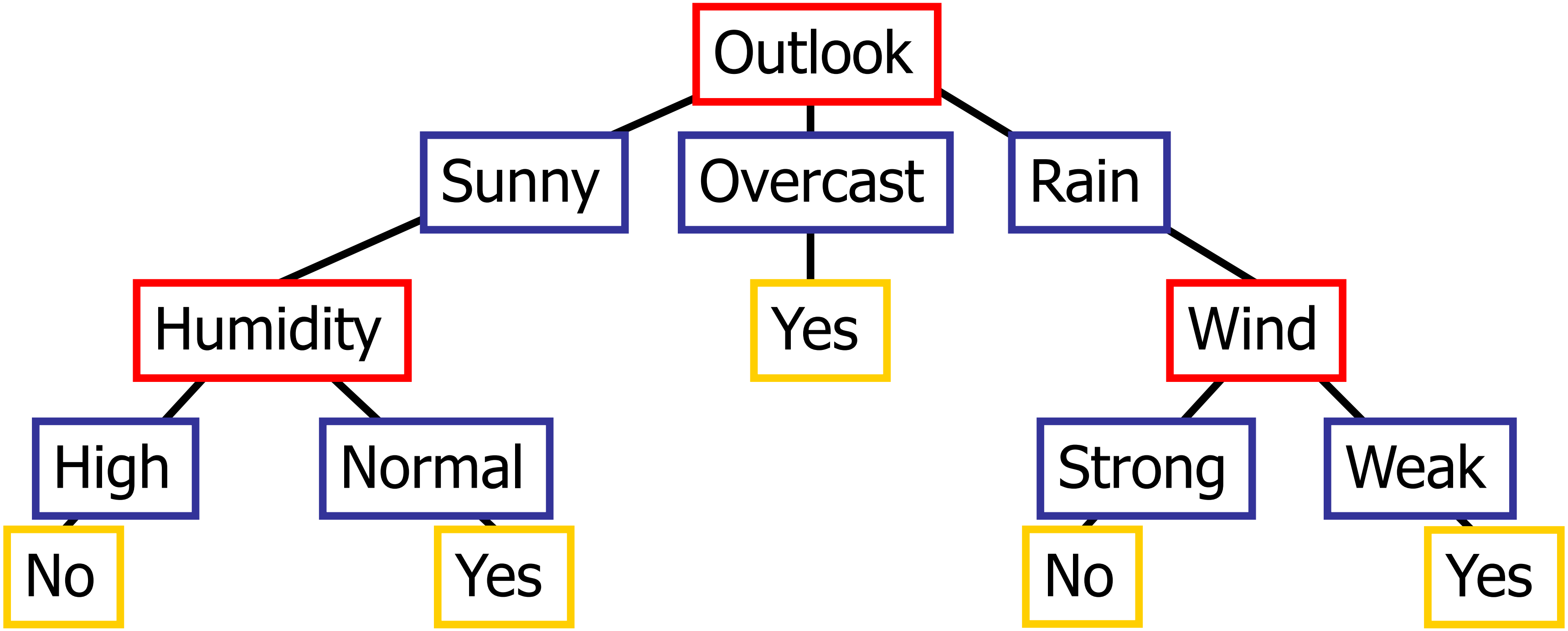
Decision Tree for XOR

Outlook=Sunny XOR Wind=Weak



Decision Tree

- decision trees represent disjunctions of conjunctions



(Outlook=Sunny $\wedge$ Humidity=Normal)
$\vee$ (Outlook=Overcast)
$\vee$ (Outlook=Rain $\wedge$ Wind=Weak)

When to consider Decision Trees

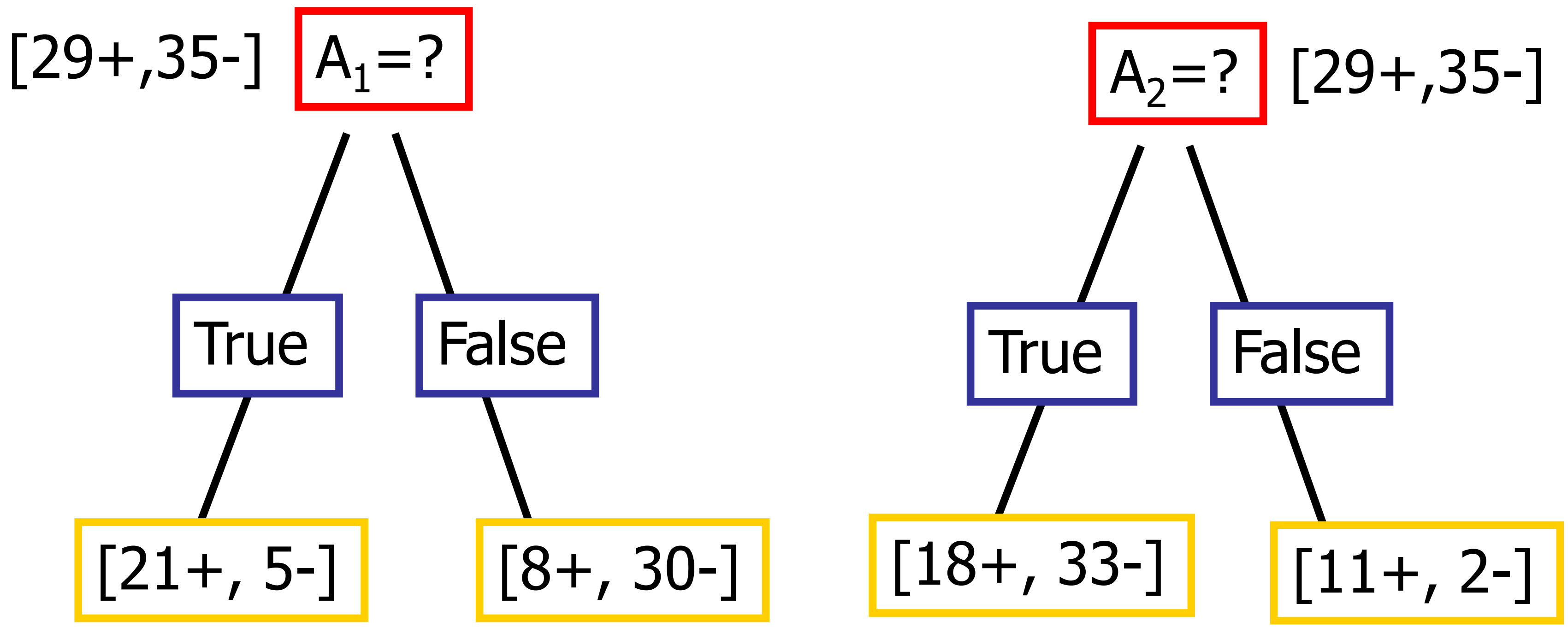
- Instances describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data
- Missing attribute values
- Examples:
  - Medical diagnosis
  - Credit risk analysis
  - Object classification for robot manipulator (Tan 1993)

Top-Down Induction of Decision Trees ID3

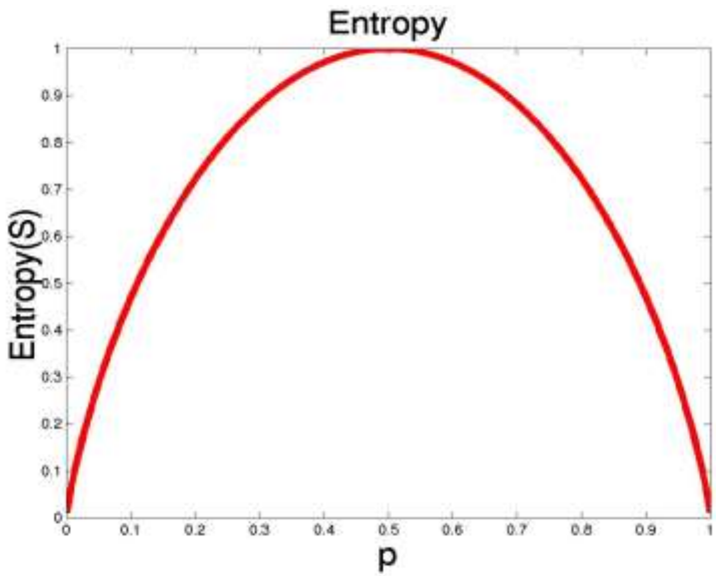
In decision tree learning, **ID3** (**Iterative Dichotomiser 3**) is an algorithm used to generate a decision tree invented by Ross Quinlan

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
4. Sort training examples to leaf node according to the attribute value of the branch
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.

Which Attribute is "best"?



Entropy



- S is a sample of training examples
- p_+ is the proportion of positive examples
- p_- is the proportion of negative examples
- Entropy measures the impurity of S

$$\text{Entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Entropy

- Entropy(S)= expected number of bits needed to encode class (+ or -) of randomly drawn members of S (under the optimal, shortest length-code)

Why?

- Information theory optimal length code assign

  $-\log_2 p$ bits to messages having probability p.

- So the expected number of bits to encode

  (+ or -) of random member of S:

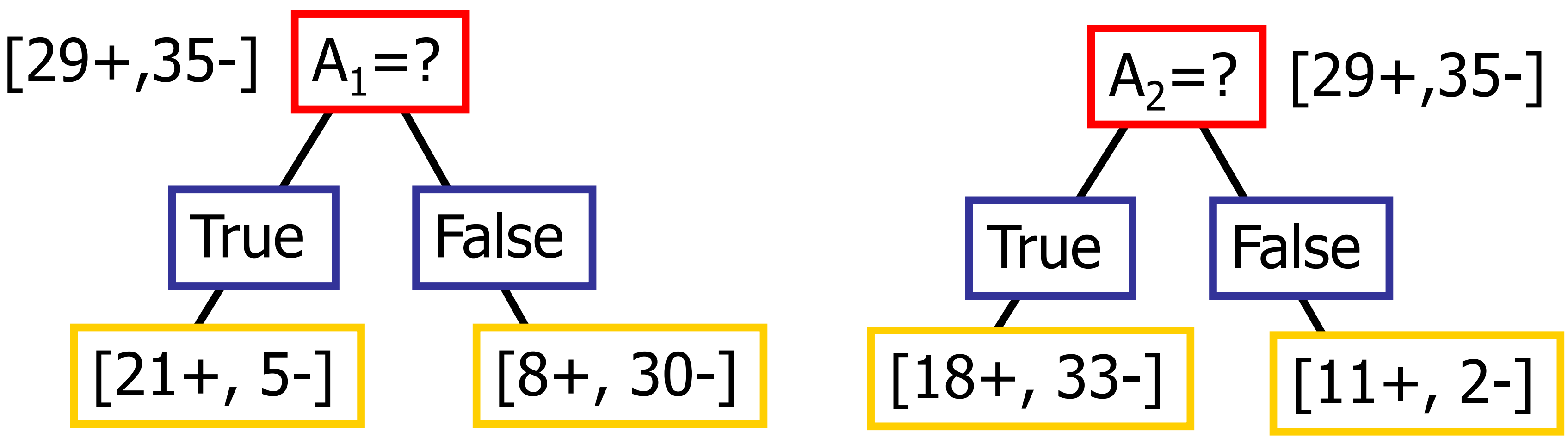
$$-p_+ \log_2 p_+ - p_- \log_2 p_-$$

Information Gain

- Gain(S,A): expected reduction in entropy due to sorting S on attribute A

$$\text{Gain}(S,A) = \text{Entropy}(S) - \sum_{v \in values(A)} |S_v|/|S| \; \text{Entropy}(S_v)$$

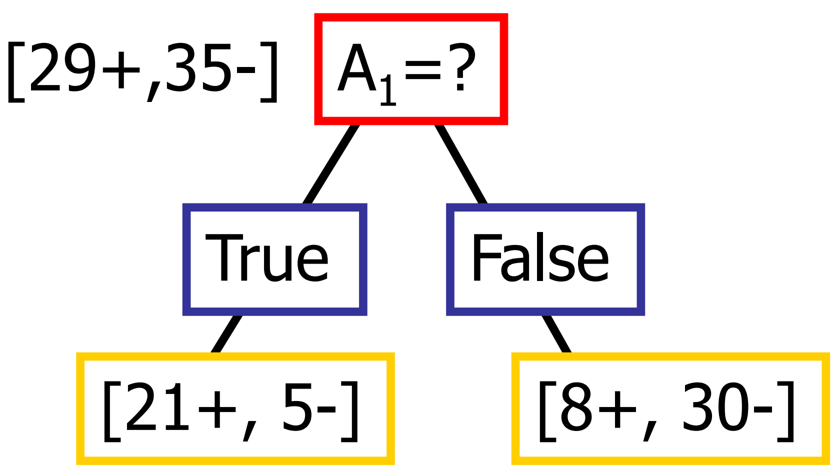
$$\text{Entropy}([29+,35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64$$
$$= 0.99$$

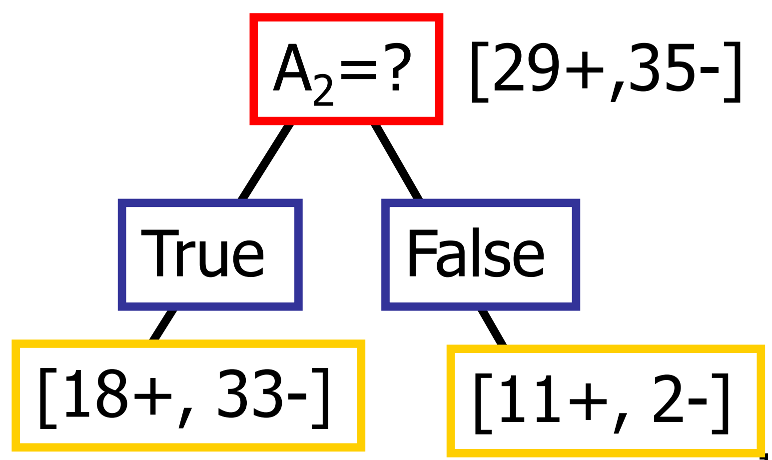


Information Gain

Entropy([21+,5-]) = 0.71
Entropy([8+,30-]) = 0.74
Gain(S,A_1)=Entropy(S)
-26/64*Entropy([21+,5-])
-38/64*Entropy([8+,30-])
=0.27

Entropy([18+,33-]) = 0.94
Entropy([8+,30-]) = 0.62
Gain(S,A_2)=Entropy(S)
-51/64*Entropy([18+,33-])
-13/64*Entropy([11+,2-])
=0.12

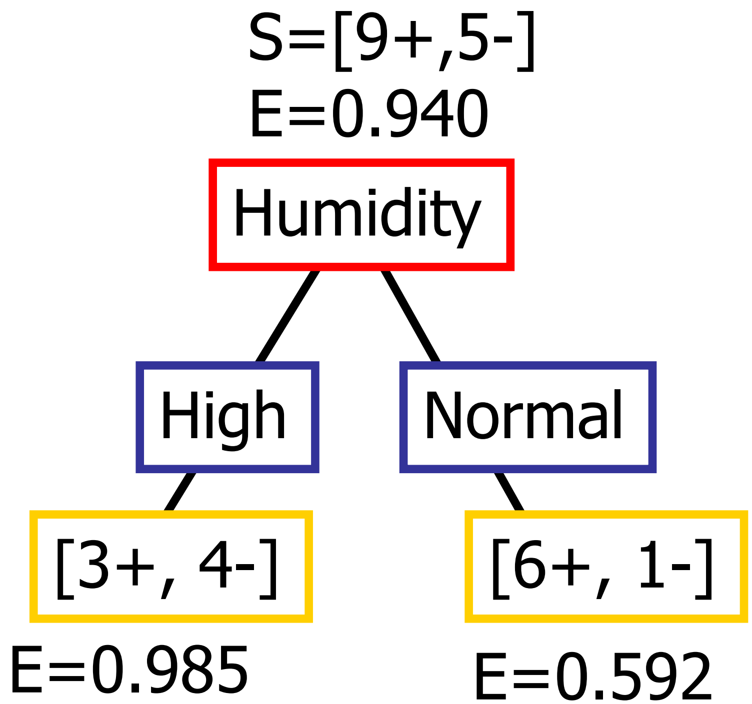


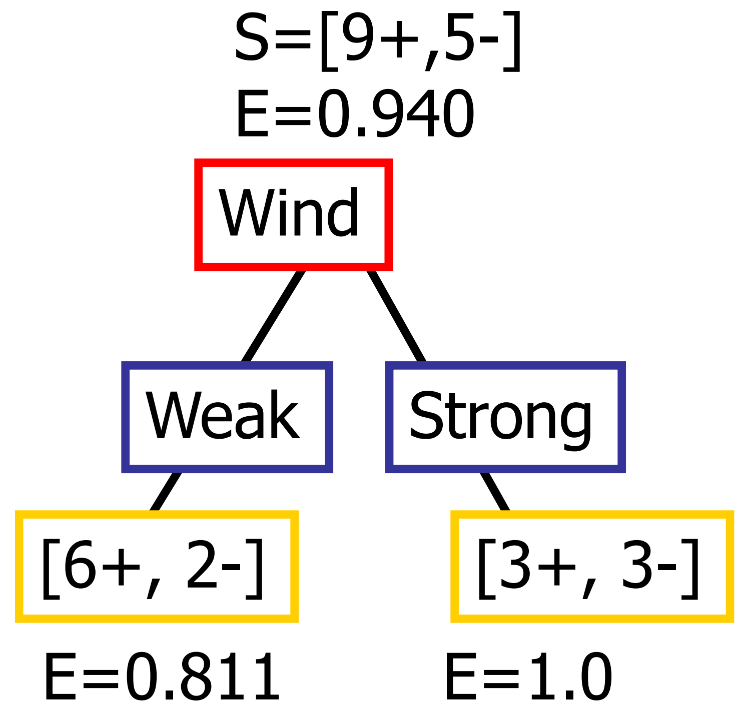


Training Examples

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
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	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	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

Selecting the Next Attribute





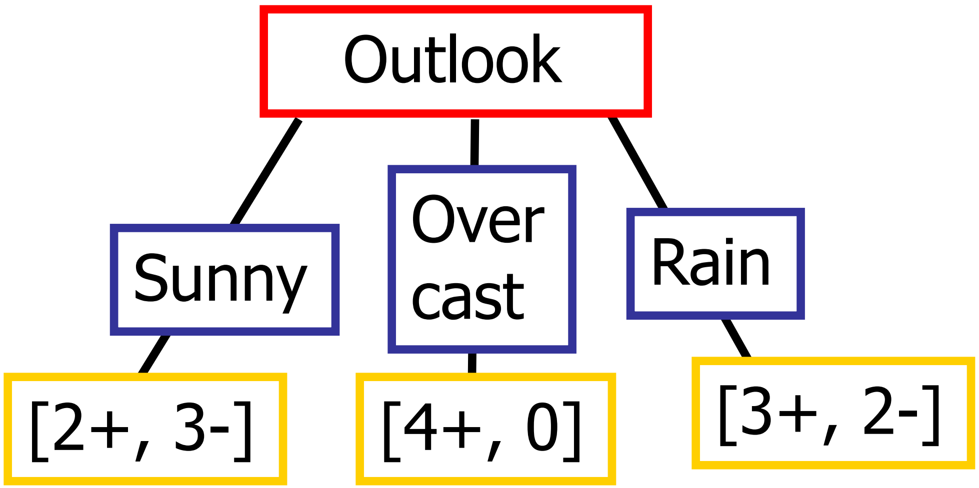
Gain(S,Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151

Gain(S,Wind)
=0.940-(8/14)*0.811
– (6/14)*1.0
=0.048

Humidity provides greater info. gain than Wind, w.r.t target classification.

Selecting the Next Attribute

S=[9+,5-]
E=0.940



E=0.971 E=0.0 E=0.971

Gain(S,Outlook)
=0.940-(5/14)*0.971
-(4/14)*0.0 – (5/14)*0.0971
=0.247

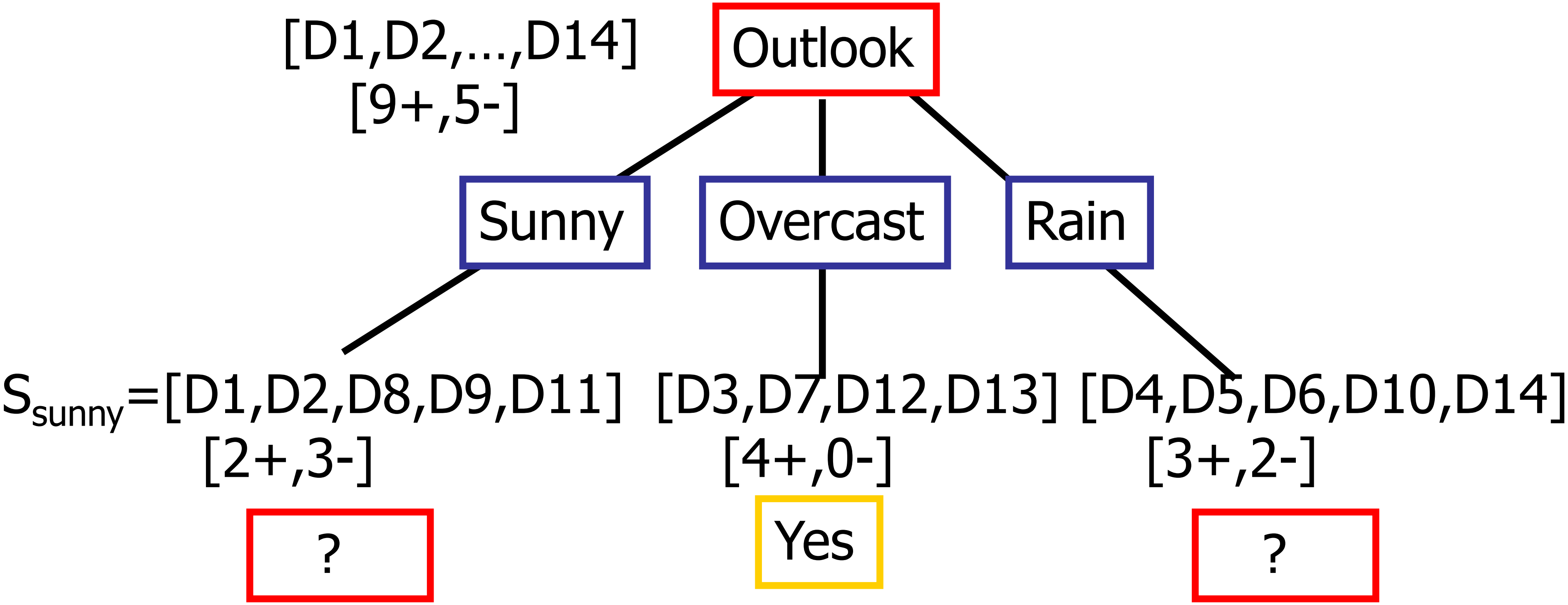
Selecting the Next Attribute

The information gain values for the 4 attributes are:

- Gain(S,Outlook) =0.247
- Gain(S,Humidity) =0.151
- Gain(S,Wind) =0.048
- Gain(S,Temperature) =0.029

where S denotes the collection of training examples

ID3 Algorithm

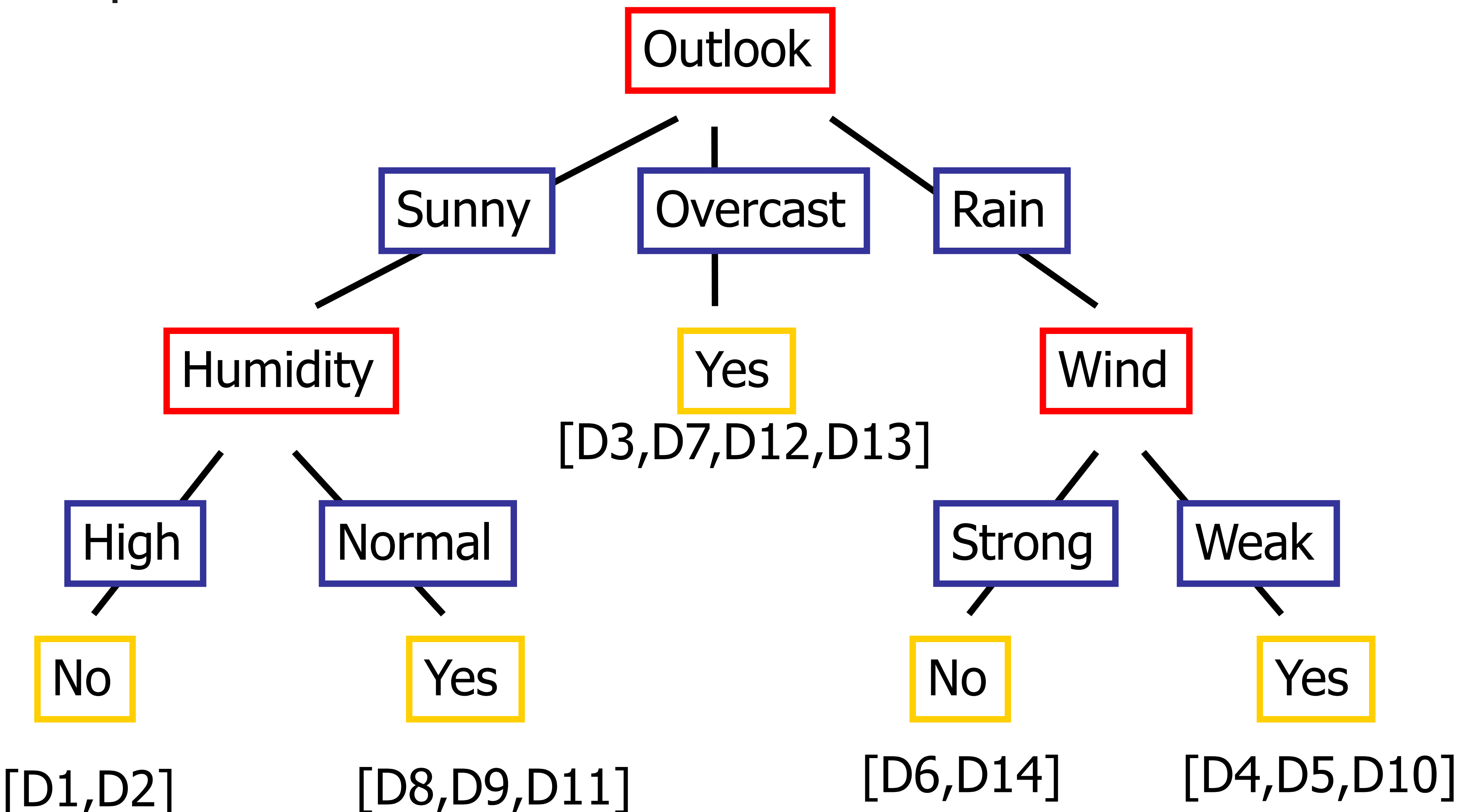


Gain(S_{sunny} , Humidity)=0.970-(3/5)0.0 – 2/5(0.0) = 0.970

Gain(S_{sunny} , Temp.)=0.970-(2/5)0.0 –2/5(1.0)-(1/5)0.0 = 0.570

Gain(S_{sunny} , Wind)=0.970= -(2/5)1.0 – 3/5(0.918) = 0.019

ID3 Algorithm



Occam’s Razor

"*Pluralitas non est ponenda sine neccesitate*" or "plurality should not be posited without necessity.“William of Ockham (ca. 1285-1349)

Why prefer short hypotheses?

Argument in favor:

- Fewer short hypotheses than long hypotheses
- A short hypothesis that fits the data is unlikely to be a coincidence
- A long hypothesis that fits the data might be a coincidence

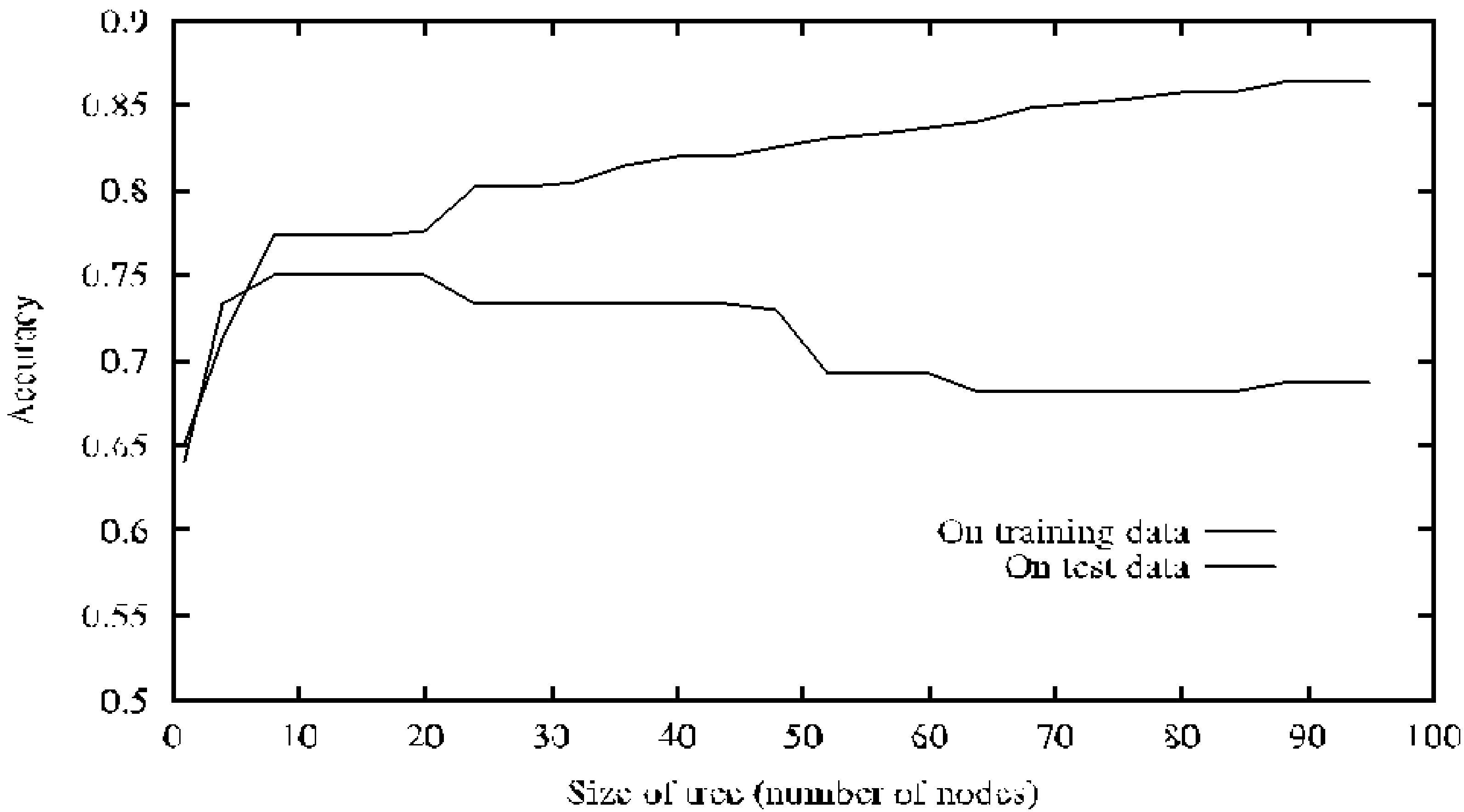
Argument opposed:

- There are many ways to define small sets of hypotheses
- E.g. All trees with a prime number of nodes that use attributes beginning with ”Z”
- What is so special about small sets based on *size* of hypothesis

Overfitting

- One of the biggest problems with decision trees is **Overfitting**

Overfitting in Decision Tree Learning



Avoid Overfitting

How can we avoid overfitting?

- Stop growing when data split not statistically significant
- Grow full tree then post-prune
- Minimum description length (MDL):

  Minimize:

  size(tree) + size(misclassifications(tree))

Reduced-Error Pruning

Split data into *training* and validation set

Do until further pruning is harmful:

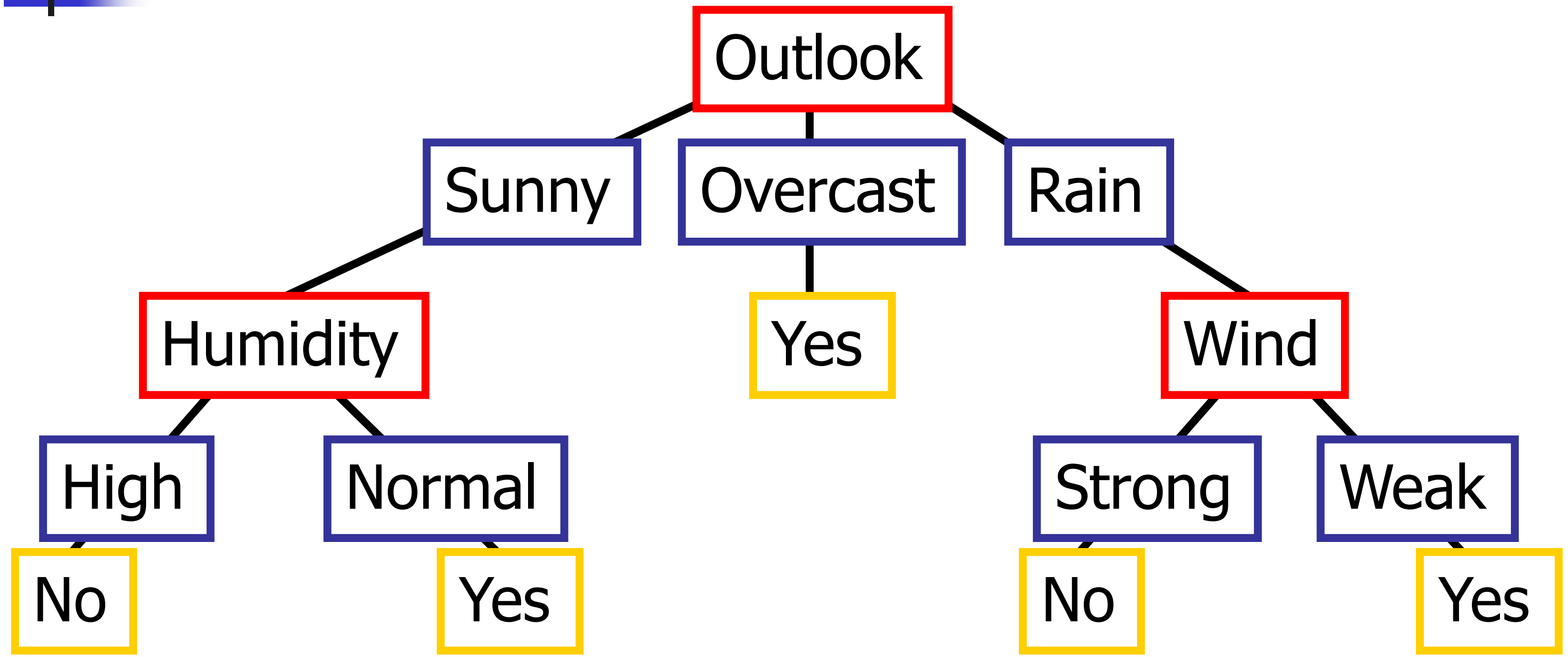
- Evaluate impact on *validation* set of pruning each possible node (plus those below it)
- Greedily remove the one that most improves the *validation* set accuracy

Produces smallest version of most accurate subtree

Rule-Post Pruning

- Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing over-fitting to occur.
- Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.
- Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.
- Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

Converting a Tree to Rules



R_1: If (Outlook=Sunny) $\wedge$ (Humidity=High) Then PlayTennis=No

R_2: If (Outlook=Sunny) $\wedge$ (Humidity=Normal) Then PlayTennis=Yes

R_3: If (Outlook=Overcast) Then PlayTennis=Yes

R_4: If (Outlook=Rain) $\wedge$ (Wind=Strong) Then PlayTennis=No

R_5: If (Outlook=Rain) $\wedge$ (Wind=Weak) Then PlayTennis=Yes

Continuous Valued Attributes

Create a discrete attribute to test continuous

- Temperature = 24.5^0C
- (Temperature > 20.0^0C) = {true, false}

Where to set the threshold?

Temperature	15^0C	18^0C	19^0C	22^0C	24^0C	27^0C
PlayTennis	No	No	Yes	Yes	Yes	No

(see paper by [Fayyad, Irani 1993])

Attributes with many Values

- **Problem:** if an attribute has many values, maximizing *InformationGain* will select it.
- E.g.: Imagine using Date=27.3.2002 as attribute perfectly splits the data into subsets of size 1

A Solution:

Use *GainRatio* instead of information gain as criteria:

$GainRatio(S,A) = Gain(S,A) / SplitInformation(S,A)$

$SplitInformation(S,A) = -\Sigma_{i=1..c} |S_i|/|S| \log_2 |S_i|/|S|$

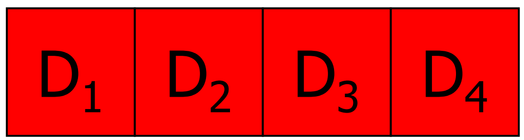
Where S_i is the subset for which <u>attribute</u> A has the value v_i

Cross-Validation

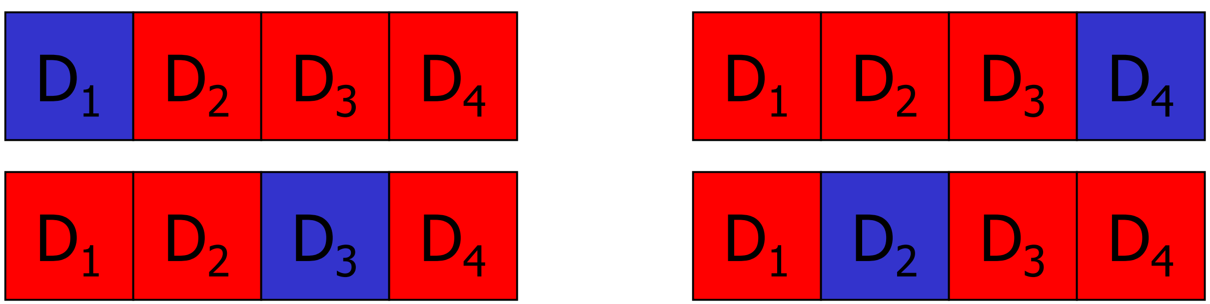
- Estimate the accuracy of a hypothesis induced by a supervised learning algorithm
- Predict the accuracy of a hypothesis over future unseen instances
- Select the optimal hypothesis from a given set of alternative hypotheses
  - Pruning decision trees
  - Model selection
  - Feature selection
- Combining multiple classifiers (boosting)

Cross-Validation

- k-fold cross-validation splits the data set D into k mutually exclusive subsets $D_1, D_2, \ldots, D_k$



- Train and test the learning algorithm k times, each time it is trained on $D \backslash D_i$ and tested on D_i



$$acc_{cv} = 1/n \sum_{(vi,yi) \in D} \delta(I(D \backslash D_i, v_i), y_i)$$

Cross-Validation

- Uses all the data for training and testing
- Complete k-fold cross-validation splits the dataset of size m in all (m over m/k) possible ways (choosing m/k instances out of m)
- Leave n-out cross-validation sets n instances aside for testing and uses the remaining ones for training (leave one-out is equivalent to n-fold cross-validation)