

# **Course : CM-072**

## **Presentation 4**

# Combining Models

**Classification/regression performance can be improved by combining multiple models**

1. Committee: train  $L$  different models
  - a. Make predictions by average of the predictions
2. Boosting: Variant of Committee
  - a. Train multiple models in sequence
  - b. Error function used to train a model depends on performance of previous models
3. .Decision Tree
  - a. Select one of  $L$  models to make the prediction
    - i. Choice of model is a function of input variables

# When to consider Decision Tree

- Instances describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

## Examples

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

# Learning simplest decision tree is NP-hard

- Learning the simplest (smallest) decision tree is an NP---complete problem [Hyafil & Rivest ]
- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on next best attribute (feature)
  - Recurse

# Decision Tree

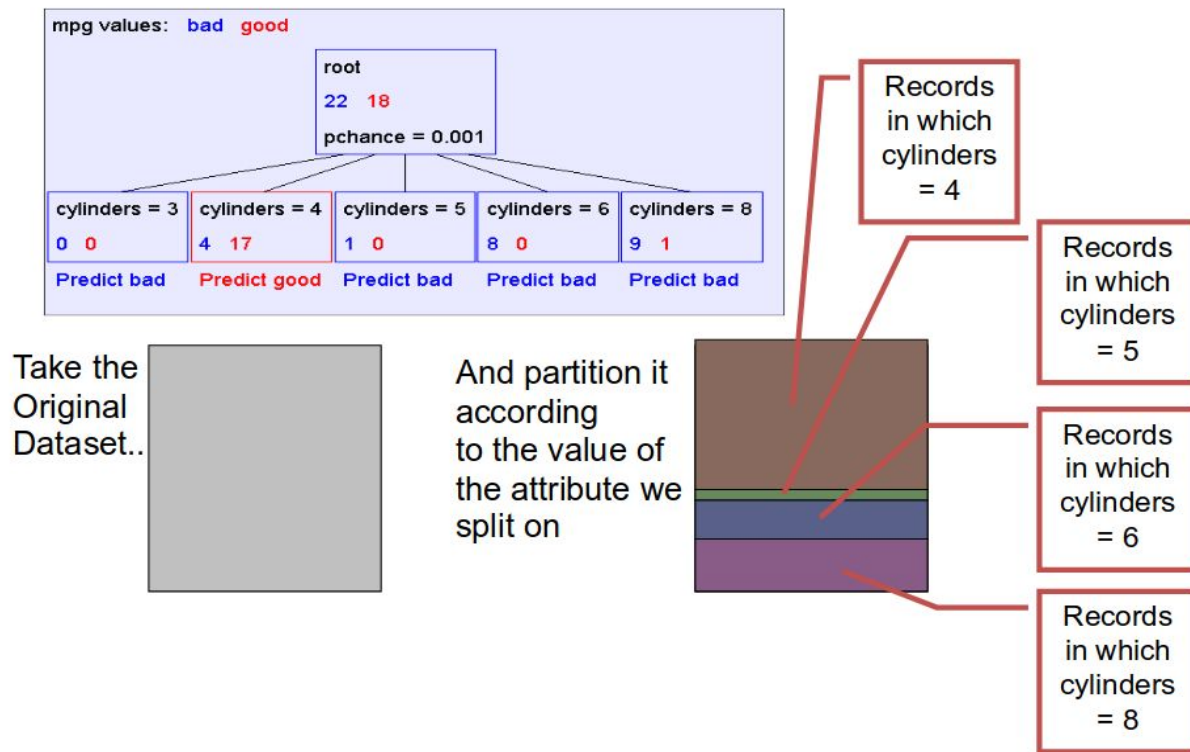
- Choice of model is function of input variables
  - Different models become responsible for making predictions in different regions of input space.
  - A sequence of binary selections corresponding to traversal of a tree structure

# Tree-based Models

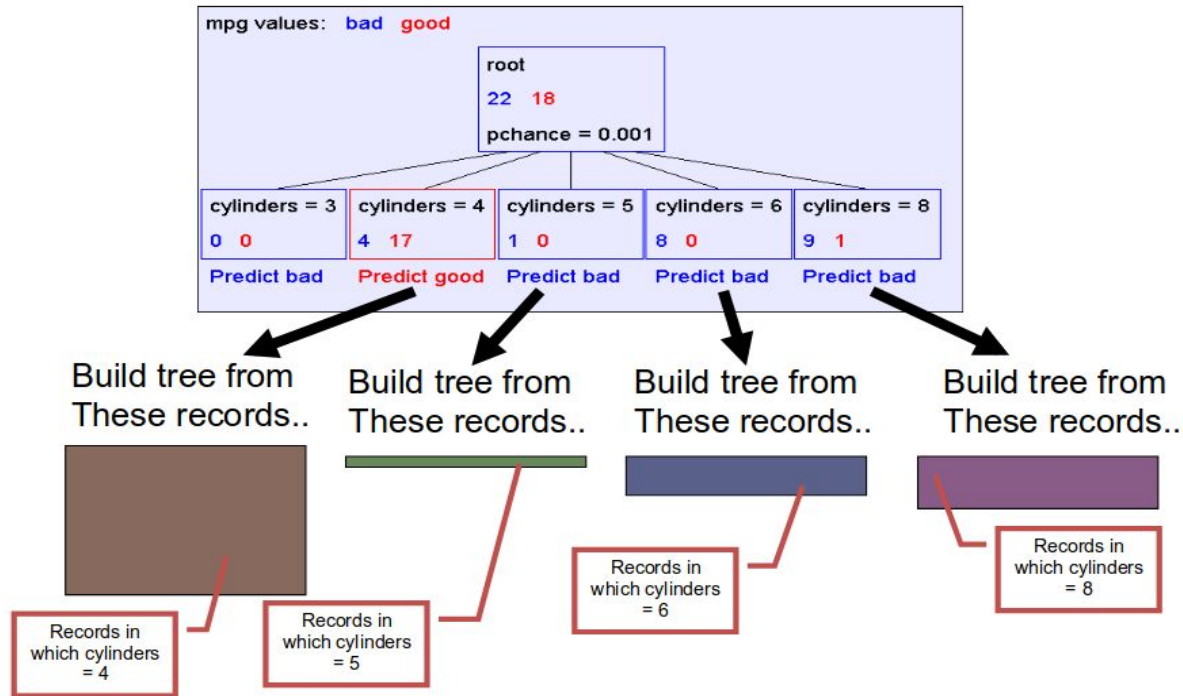
- Partition the inputs space into cuboid regions
  - Edges align the axes
  - Assign a simple models, e.g., a constant to each region
- Can be viewed as a models combination method in which only one model is responsible for making predictions at any given point in space
- Process of selecting a model for input **x corresponds to a traversal of a binary tree.**

# Greedy learn trees using recursion

- Key idea:

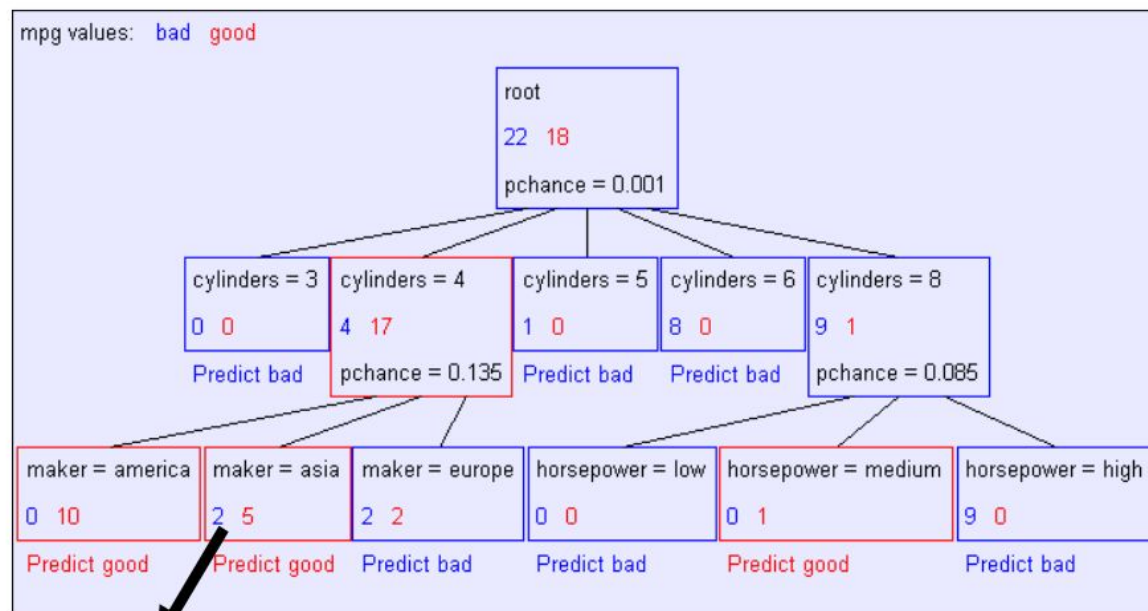


# Recursive step





# Second level of tree

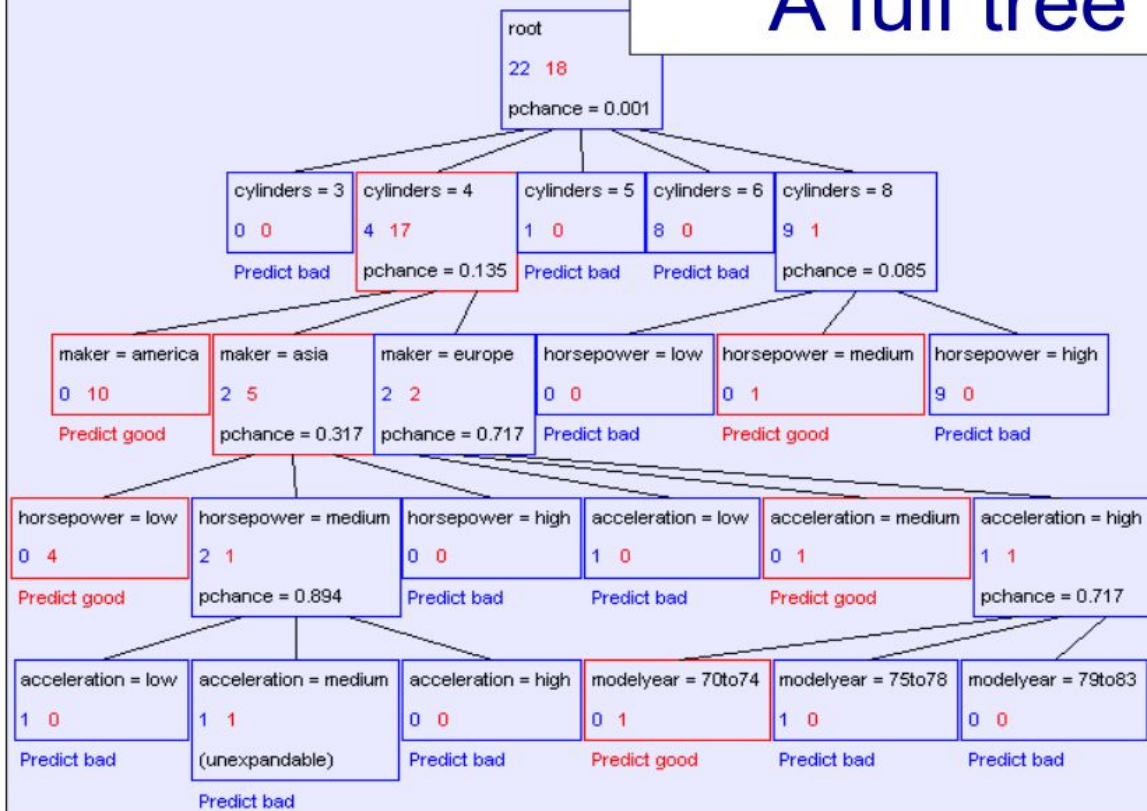


Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

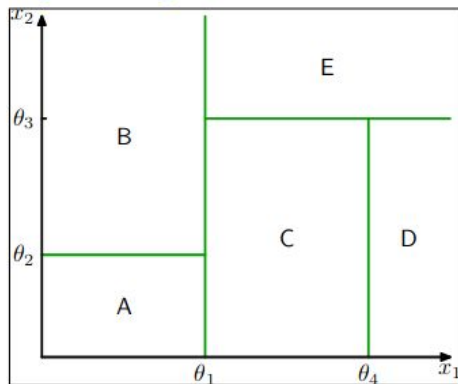
# A full tree

mpg values: bad good

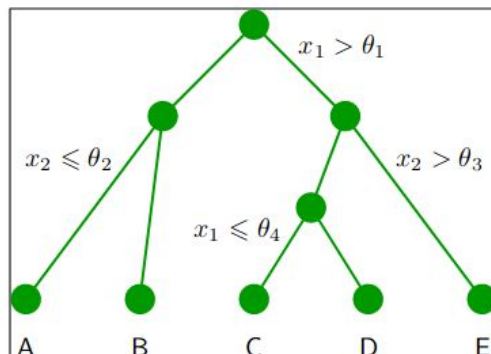


# Recursive Binary Tree Partitioning

2-D input space  $\mathbf{x} = [x_1, x_2]$   
Partitioned into five regions  
using axis aligned boundaries



Binary Tree corresponding to  
Partitioning of input space



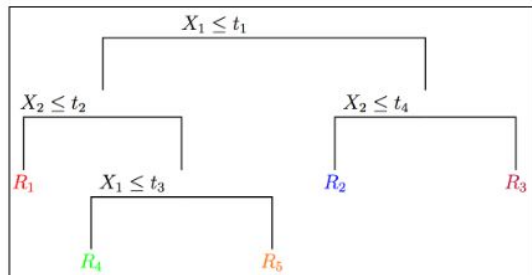
First step divides the whole of input space into two regions according to whether  $x_1 \leq \theta_1$  or  $x_1 > \theta_1$  where  $\theta_1$  is a parameter of the model. This creates two subregions, each of which can be subdivided independently e.g., region  $x_1 \leq \theta_1$  is further subdivided according to whether  $x_2 \leq \theta_2$  or  $x_2 > \theta_2$  giving rise to the regions denoted A and B. For a new input  $\mathbf{x}$  we determine which region it falls into by starting at top of tree.

# Use of Tree Models

- Can be used for both regression and classification
- So they are called Classification and Regression Trees (CART)
- Within each region there is a separate model to predict a target variable
  - In regression, we might simply predict a constant over each region
  - In classification, we might assign each region to a specific class
- Key property: they are interpretable by humans
  - To predict a disease, is temperature greater than a threshold?, is BP less than a threshold? Each leaf is a diagnosis.

# Regression Tree with 2 inputs

Two input attributes  $(x_1, x_2)$ , a real output



Result of axis-parallel splits:

2-d space partitioned into 5 regions

Each region is associated with a mean response

Result: piecewise constant function

Model can be written as:

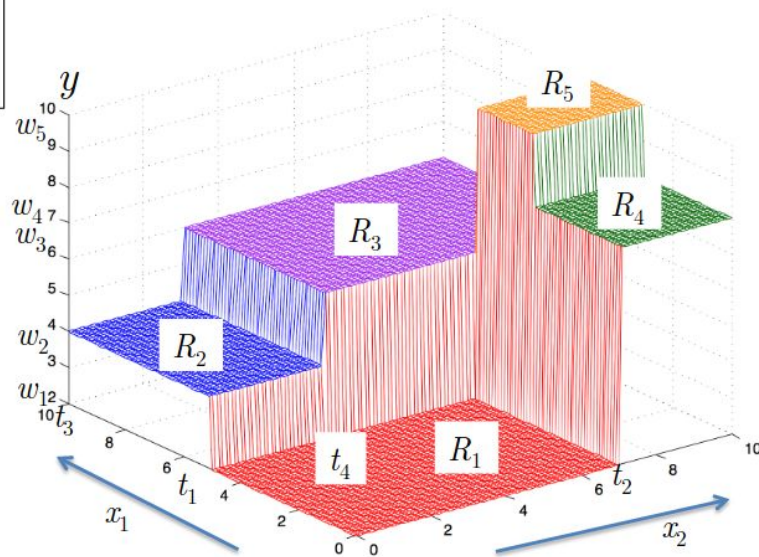
$$f(\mathbf{x}) = E[y | \mathbf{x}] = \sum_{m=1}^M w_m I(\mathbf{x} \in R_m) = \sum_{m=1}^M w_m \phi(\mathbf{x}; \mathbf{v}_m)$$

First node asks if  $x_1 \leq t_1$

If yes, ask if  $x_2 \leq t_2$


If yes, we are at quadrant  $R_1$

Associate a value of  $y$  with each region



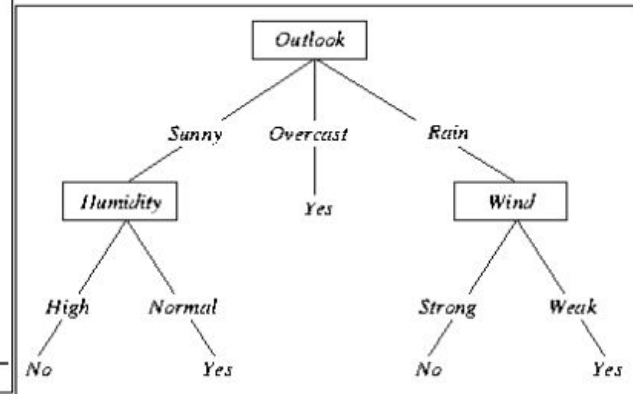
# Classification Tree (Binary Output)

PlayTennis has 4 input attributes



Day	Outlook	Temp	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

Learned function is a tree





# CART as adaptive basis model

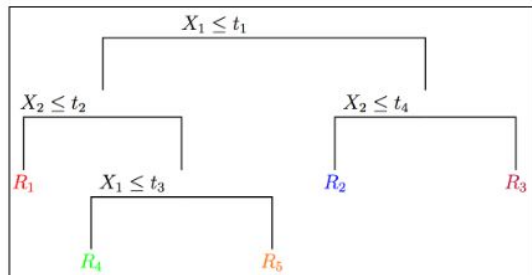
- An adaptive basis function model (ABM) is a model of the form

$$f(\mathbf{x}) = w_0 + \sum_{m=1}^M w_m \phi_m(\mathbf{x})$$

- Where  $\phi_m(\mathbf{x})$  is the  $m$ th basis function which is learned from the data
- Typically the basis functions are parametric
  - We can write  $\phi_m(\mathbf{x}) = \phi(\mathbf{x}; \mathbf{v}_m)$ 
    - Where  $\mathbf{v}_m$  are the parameters of the basis function itself
    - We can use  $\theta = (w_0, \mathbf{w}, \mathbf{v})$  to denote the entire parameter set

# Regression Tree with 2 inputs

Two input attributes  $(x_1, x_2)$ , a real output



Result of axis-parallel splits:  
2-d space partitioned into 5 regions

Each region is associated with a mean response  
Result: piecewise constant function

Model can be written as:

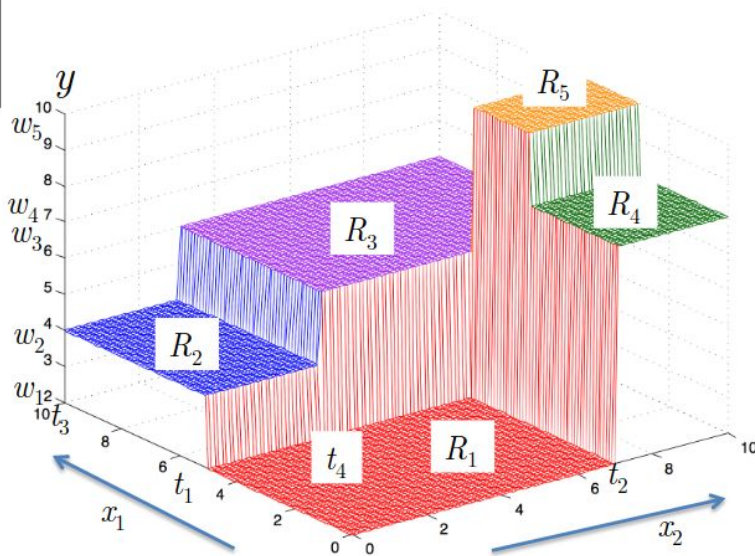
$$f(\mathbf{x}) = E[y | \mathbf{x}] = \sum_{m=1}^M w_m I(\mathbf{x} \in R_m) = \sum_{m=1}^M w_m \phi(\mathbf{x}; \mathbf{v}_m)$$

First node asks if  $x_1 \leq t_1$

If yes, ask if  $x_2 \leq t_2$

If yes, we are at quadrant  $R_1$

Associate a value of  $y$  with each region





# Tree as adaptive basis model


- The axis parallel splits partition the 2-D space into 5 regions
  - We associate a mean response with each region resulting in a piecewise constant surface
- We can write the model in the form

$$f(\mathbf{x}) = E[y | \mathbf{x}] = \sum_{m=1}^M w_m I(\mathbf{x} \in R_m) = \sum_{m=1}^M w_m \phi(\mathbf{x}; \mathbf{v}_m)$$

- Where  $R_m$  is the  $m^{\text{th}}$  region,  $w_m$  is the mean response of this region and  $\mathbf{v}_m$  encodes the choice of variable to split on and the threshold value on the path from the root to the  $m^{\text{th}}$  leaf

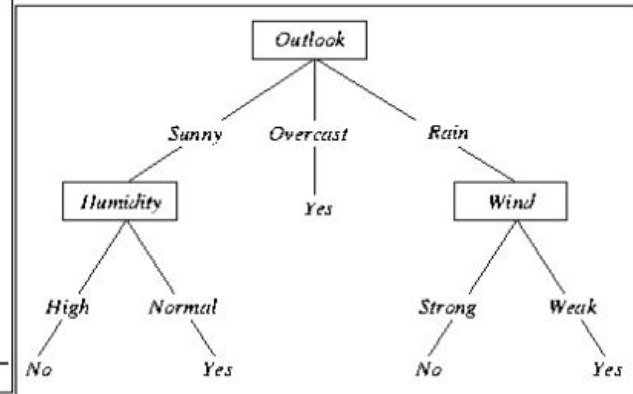
# Classification Tree (Binary Output)

PlayTennis has 4 input attributes



Day	Outlook	Temp	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

Learned function is a tree



# Bagging with Trees

- One way to reduce the variance of an estimate is to average together many estimates
  - E.g., we can train  $M$  different trees on different random subsets of data, with replacement, and then compute the ensemble

$$f(\mathbf{x}) = \sum_{m=1}^M \frac{1}{M} f_m(\mathbf{x})$$

- where  $f_m$  is the  $m^{\text{th}}$  tree
- This technique is called bagging
  - Stands for bootstrap aggregation

# Tree-based Models

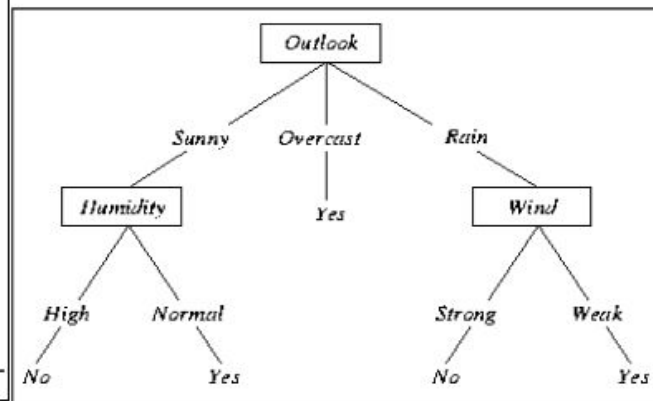
- Unfortunately, simply re-running the same learning algorithm on different subsets of data can result in highly correlated predictors
  - Which limits the amount of variance reduction possible.
- Random forests tries to decorrelate the base learners
  - By learning trees based on a randomly chosen subset of input variables, as well as a randomly chosen subset of data cases.

# Classification Tree

Data Set  $x = [x_1, x_2, x_3, x_4]$

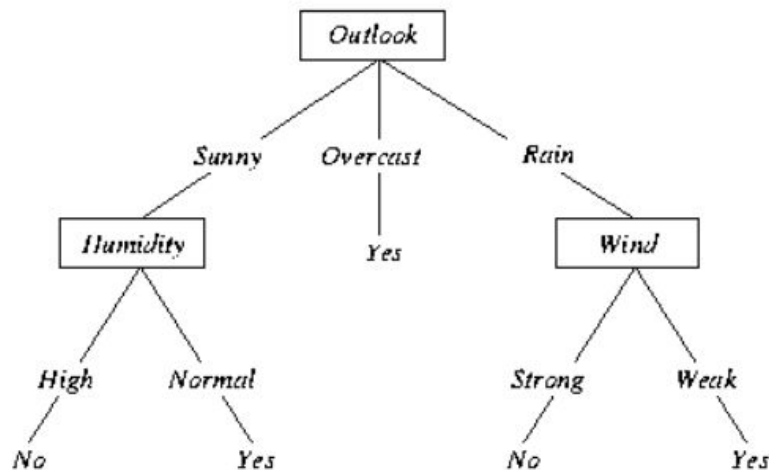
					$y$
Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	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

Learned function  $y(x)$ : a tree



# Components of Classification Tree

- Classify instances
  - by traversing from root to leaf node,
- Each node specifies a test of an attribute
- Each branch descending from a node corresponds a possible value of this attribute

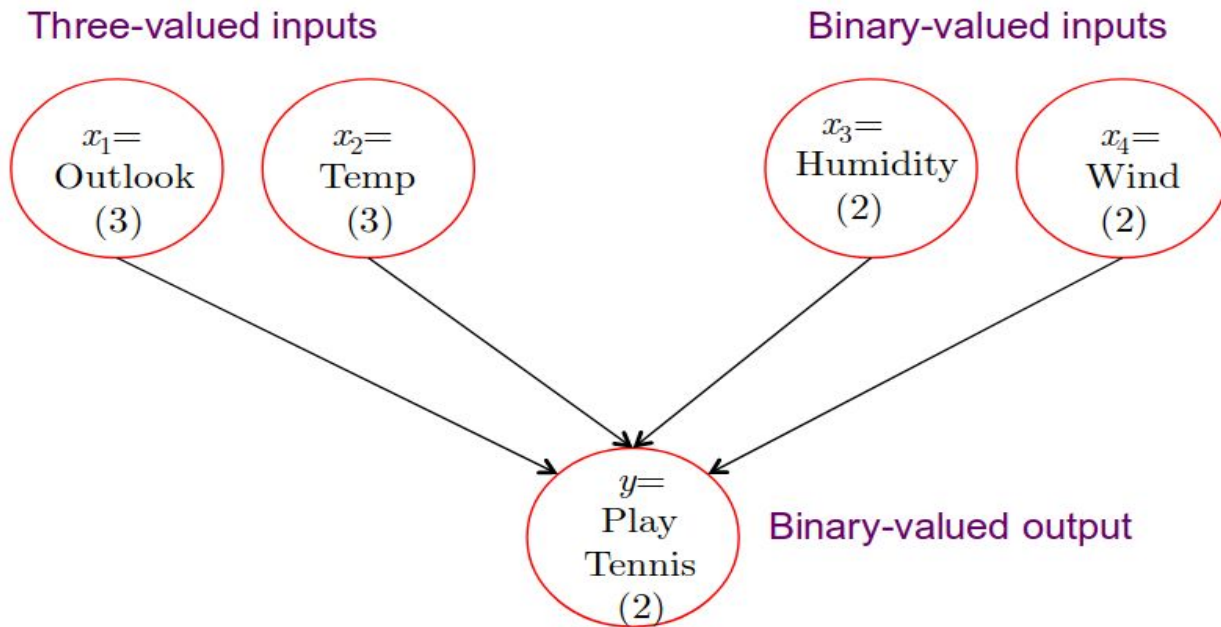




# Classification Tree Learning

- Tree approximates discrete-valued target function  $y(\mathbf{x})$ 
  - Robust to noisy data and capable of learning disjunctive expressions
- A family of decision tree learning algorithms includes ID3, ASSISTANT and C4.5

# Graphical Model for Play Tennis

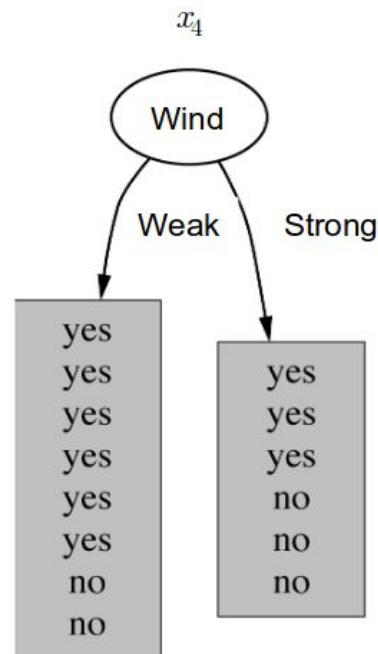


Need  $3 \times 3 \times 2 \times 2 \times 2 = 76$  probabilities (parameters) for joint distribution  $p(\mathbf{x}, y)$   
Can we do better?

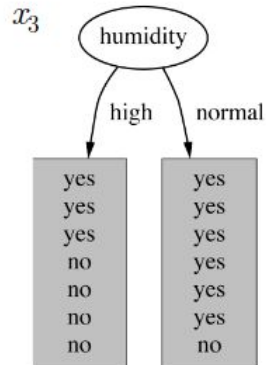
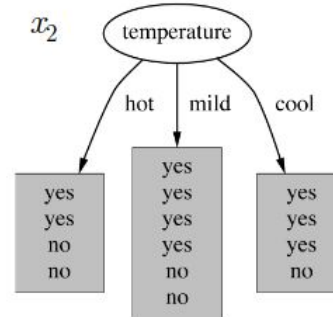
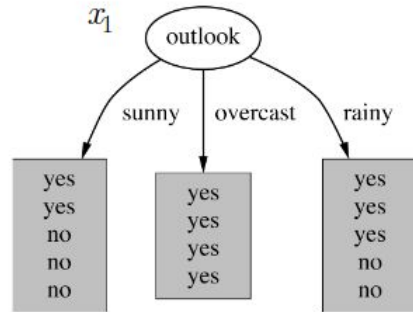


# Tree Stump for a binary-valued input variable

Day	Outlook	Temp	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



# Tree Stump for other three inputs



Day	Outlook	Temp	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

# Good and Poor attributes

- **An binary-valued attributes is good when:**
  - For one value we get all instances as positive
  - For other value we get all instances as negative
- **An attribute is poor when:**
  - It provides no discrimination
  - Attributes is immaterial to the decision
  - For each value we have same number of positive and negative instances.

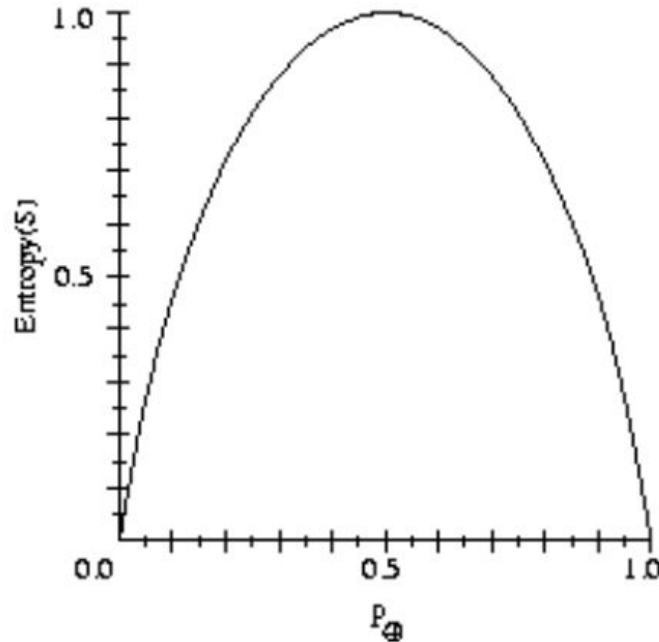
# Entropy: Measure of homogeneity of Examples

- Entropy: Characterizes the (im)purity of an arbitrary collection of examples
- Given a collection  $S$  of positive and negative examples, entropy of  $S$  relative to boolean classification is

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

Where  $p_+$  is proportion of positive examples and  $p_-$  is proportion of negative examples

# Entropy Function Relative to a Boolean Classification



# Entropy of data set

- Dataset  $S$  has 9 positive, 5 negative examples
- Entropy of  $S$  is:

$$\begin{aligned}\text{Entropy}(9+, 5-) = \\ -(9/14)\log_2(9/14) - \\ (5/14)\log_2(5/14) = 0.94\end{aligned}$$

Day	Outlook	Temp	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 is zero if all in  $S$  belong to the same class

# Entropy for multi-valued target function

- If the target attribute can take on  $c$  different values, the entropy of  $S$  relative to this  $c$ -wise classification is

$$\text{Entropy}(S) \equiv \sum_{i=1}^c -p_i \log_2 p_i$$

# Information Gain

- Entropy measures the impurity of a collection
- Information Gain is defined in terms of Entropy
  - Expected reduction in entropy caused by partitioning the examples according to this attributes



# Information Gain Measures the Expected Reduction in Entropy

- Information gain of attribute  $A$  is the reduction in entropy caused by partitioning the set of examples  $S$

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

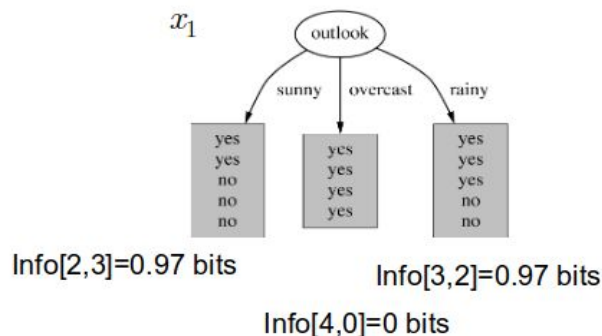
- where  $Values(A)$  is the set of all possible values for attribute  $A$  and  $S_v$  is the subset of  $S$  for which attribute  $A$  has value  $v$

# Information Gain for attributes $x_1$

## *Outlook*

For first value (*Sunny*) there are 2 positive and 3 negative examples

$$\begin{aligned}\text{Info}[2,3] &= \text{entropy}(2/5, 3/5) \\ &= -2/5 \log 2/5 - 3/5 \log 3/5 \\ &= 0.97 \text{ bits}\end{aligned}$$



Average info of subtree(weighted)=

$$\begin{aligned}0.97 \times 5/14 + \\ 0 \times 4/14 +\end{aligned}$$

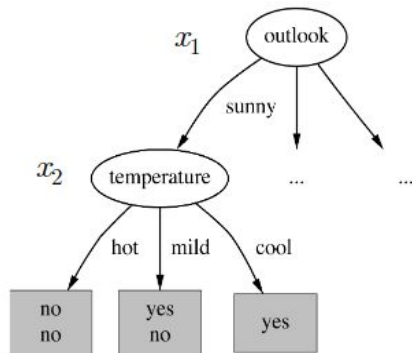
$$0.97 \times 5/14 = 0.693 \text{ bits}$$

Info of all training samples,  $\text{info}[9,5] = 0.94$

$$\text{Gain}(S, \text{Outlook}) = 0.94 - 0.693 = 0.247 \text{ bits}$$

# Expanded Tree Stumps for $x_1$

*Outlook=Sunny*



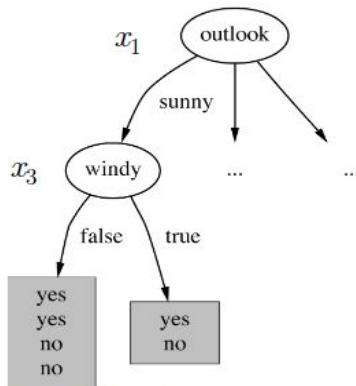
$$\text{Info}(2,3)=0.97$$

$$\text{Info}(0,2)=\text{Info}(1,0)=0$$

$$\text{Info}(1,1)=0.5$$

$$\text{Ave Info}=0+0+(1/5)$$

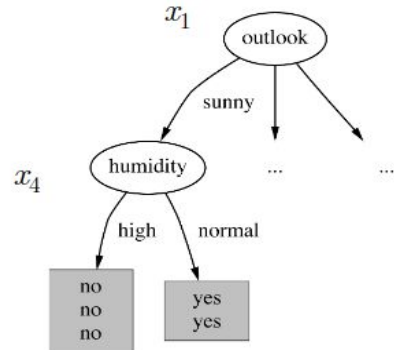
$$\text{Gain}(\text{temp})=0.97-0.2=0.77 \text{ bits}$$



$$\text{Info}(3,3)=1$$

$$\text{Info}(2,2)=\text{Info}(1,1)=1$$

$$\text{Gain}(\text{windy})=1-1=0 \text{ bits}$$



$$\text{Info}(3,2)=0.97$$

$$\text{Info}(0,3)=\text{Info}(2,0)=0$$

$$\text{Gain}(\text{humidity})=0.97 \text{ bits}$$

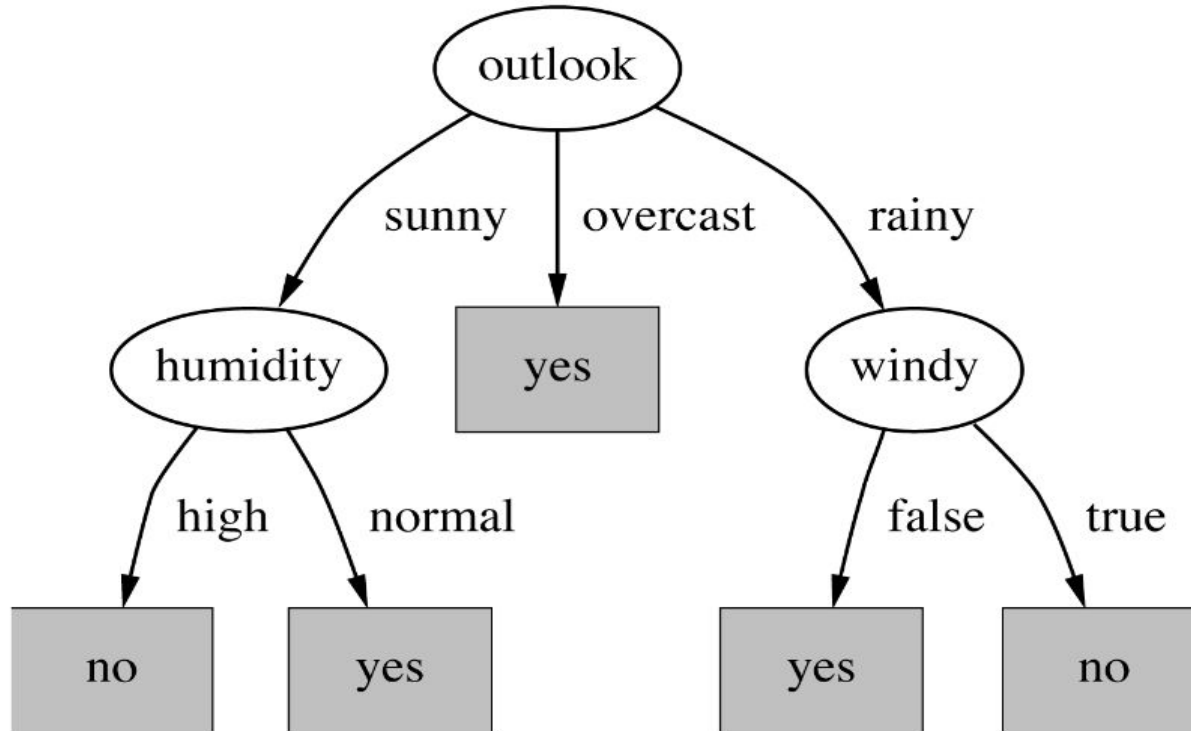
Since  $\text{Gain}(\text{humidity})$  is highest, select humidity as splitting attribute.

No need to split further

# Information Gain for each attribute

- $Gain(Outlook) = 0.94 - 0.693 = 0.247$
- $Gain(Temperature) = 0.94 - 0.911 = 0.029$
- $Gain(Humidity) = 0.94 - 0.788 = 0.152$
- $Gain(Windy) = 0.94 - 0.892 = 0.048$
- $\arg \max \{0.247, 0.029, 0.152, 0.048\} = Outlook$
- **Select *Outlook* as the splitting attribute of tree**

# Decision Tree for the Weather Data



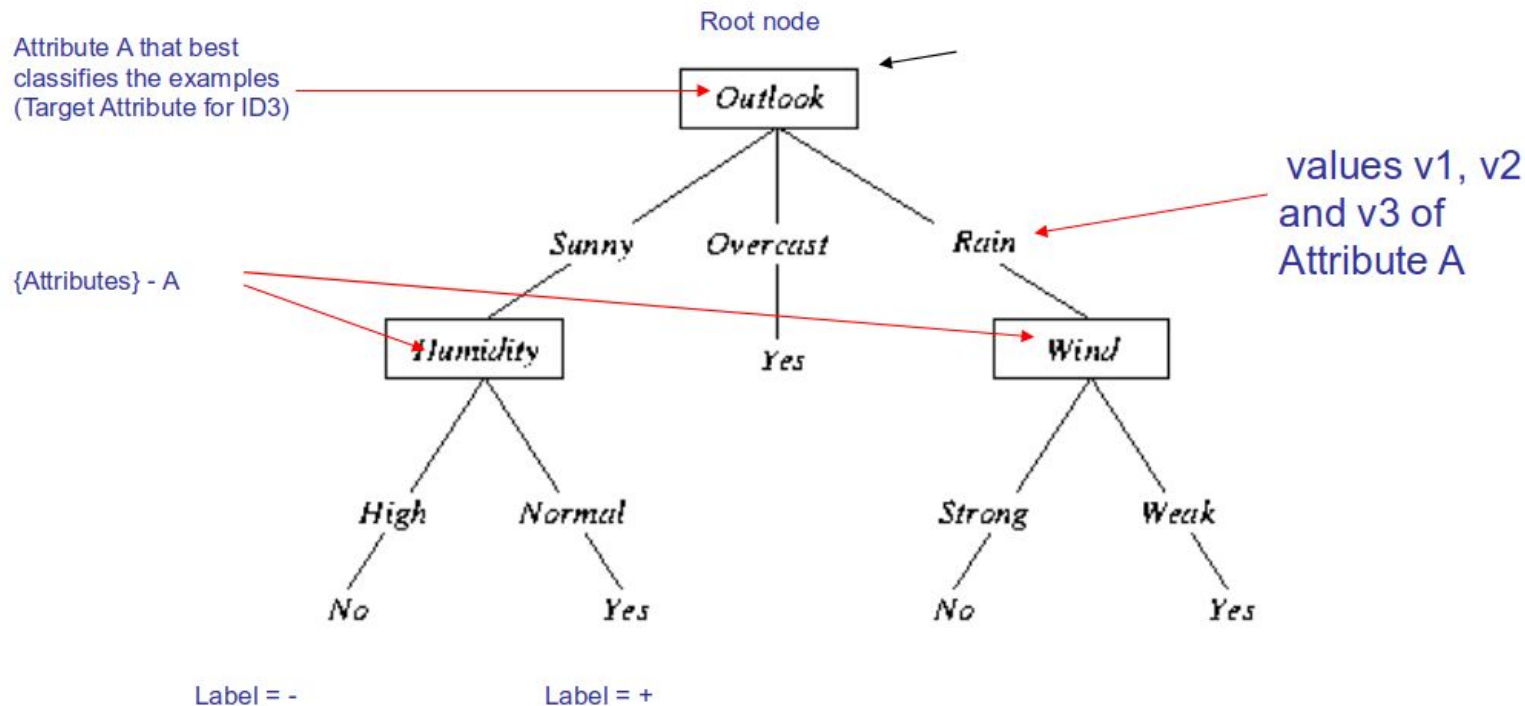
# The Basic Decision Tree Learning Algorithm (ID3)

- Top-down, greedy search (no backtracking) through space of possible decision trees
- Begins with the question
  - “which attribute should be tested at the root of the tree?”
- **Answer**
  - evaluate each attribute to see how it alone classifies training examples
- Best attribute is used as root node
  - descendant of root node is created for each possible value of this attribute

# Which Attribute is Best for the Classifier?

- Select attribute that is most useful for classification.
- ID3 uses information gain as a quantitative measure of an attribute.
- **Information Gain**
  - A statistical property that measures how well a given attribute separates that training examples according to their target classification.

# ID3 Algorithm Notation





# ID3 Algorithm to learn boolean-valued functions

ID3 (Examples, Target\_attribute, Attributes)

Examples are the training examples. Target\_attribute is the attribute (or feature) whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree (actually the root node of the tree) that correctly classifies the given Examples.

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, Return the single-node tree *Root*, with label = the most common value of *Target\_attribute* in *Examples*
- %Note that we will return the name of a feature at this point

# ID3 Algorithm +

·Otherwise Begin

- A  $[?]$  the attribute from *Attributes* that best\* classifies *Examples*
- The decision attribute (**feature**) for *Root*  $[?]$  A
- For each possible value  $v_i$  of A,
  - Add a new tree branch below *Root*, corresponding to test  $A = v_i$
  - Let  $Examples_{v_i}$  the subset of *Examples* that have value  $v_i$  for A
  - If  $Examples_{v_i}$  is empty'
    - Then below this new branch, add a leaf node with label = most common value of *Target\_attribute* in *Examples*
    - Else, below this new branch add the subtree  
 $ID3(Examples_{v_i}, Target\_attribute, Attributes - \{A\})$

·End

·Return *Root*

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

- The best attribute is the one with the highest *information gain*, as defined in Equation:

# Perfect feature


- If feature outlook has two values: sunny and rainy
- If for sunny all 5 values of play tennis are yes
- If for rainy all 9 values of play tennis are no

$$\begin{aligned}\text{Gain}(S, \text{outlook}) &= 0.94 - (5/9) \cdot 0 - (9/9) \cdot 0 \\ &= 0.94\end{aligned}$$

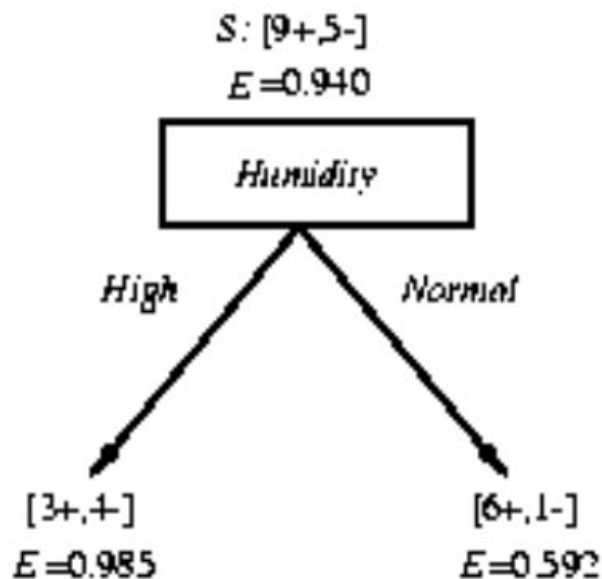
# Training Examples for Target Concept Play Tennis

Day	Outlook	Temp	Humidity	Wind	<i>PlayTennis</i>
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

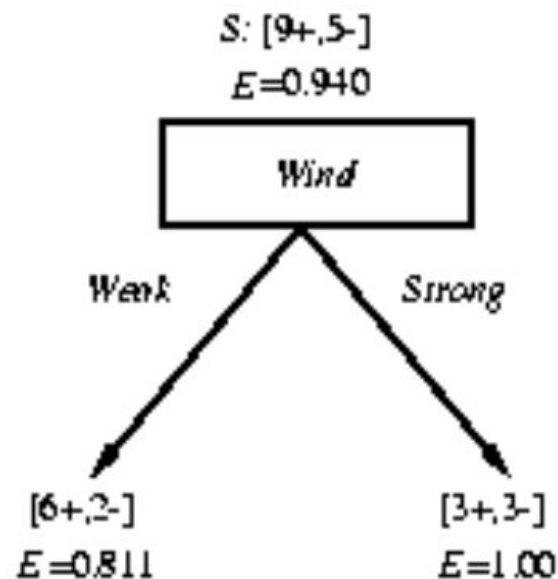
# Stepping through ID3 for the example

- Top Level (with  $S=\{D1,...,D14\}$ )
  - $\text{Gain}(S, \text{Outlook}) = 0.246$
  - $\text{Gain}(S, \text{Humidity}) = 0.151$  
  - $\text{Gain}(S, \text{Wind}) = 0.048$
  - $\text{Gain}(S, \text{Temperature}) = 0.029$
- Example computation of Gain for Wind
  - $\text{Values}(\text{Wind}) = \text{Weak}, \text{Strong}$
  - $S = [9+, 5-]$
  - $S_{\text{weak}} \leftarrow [6+, 2-], S_{\text{strong}} \leftarrow [3+, 3-]$
  - $\text{Gain}(S, \text{Wind}) = 0.940 - (8/14)0.811 - (6/14)1.00$   
 $= 0.048$

# Sample calculations for Humidity and Wind



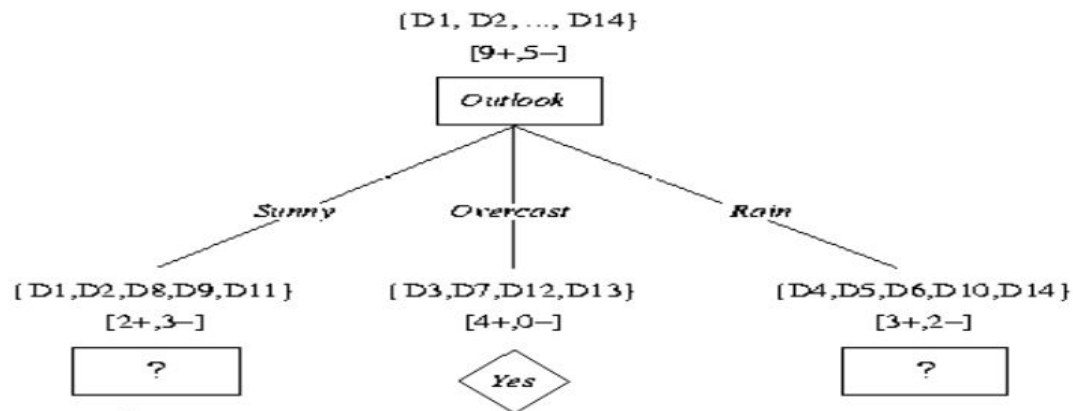
$$\begin{aligned} \text{Gain}(S, \text{Humidity}) &= 0.940 - (7/14) \cdot 0.985 - (7/14) \cdot 0.592 \\ &= 0.151 \end{aligned}$$



$$\begin{aligned} \text{Gain}(S, \text{Wind}) &= 0.940 - (8/14) \cdot 0.811 - (6/14) \cdot 1.0 \\ &= 0.048 \end{aligned}$$



# The Partially Learned Decision Tree



*Which attribute should be tested here?*

$S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}$

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

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

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

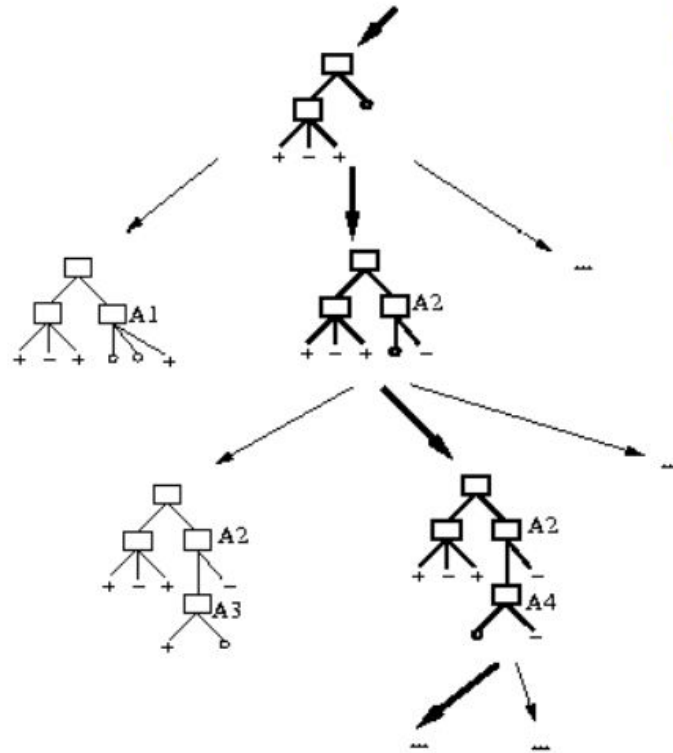
Humidity is a perfect feature since there is no uncertainty left when we know its value

# Hypothesis Space Search in Decision Tree Learning

- ID3 searches a hypothesis space for one that fits training examples
- Hypothesis space searched is set of possible decision trees
- ID3 performs hill-climbing, starting with empty tree, considering progressively more elaborate hypotheses (to find tree to correctly classify training data)
- Hill climbing is guided by evaluation function which is the gain measure.



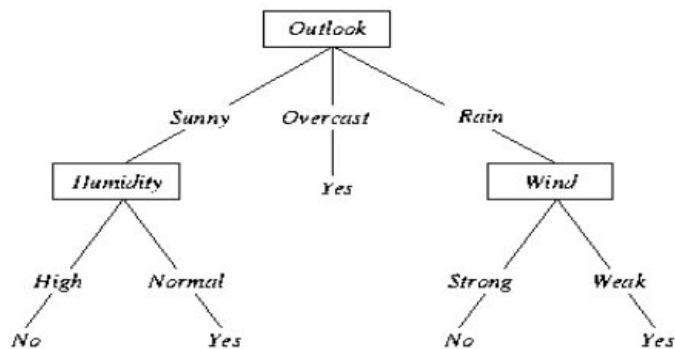
# Hypothesis Space Search by ID3



Information gain heuristic guides search of hypothesis space by ID3

# Decision Trees and Rule-Based Systems

- Learned trees can also be re-represented as sets of *if-then* rules to improve human readability

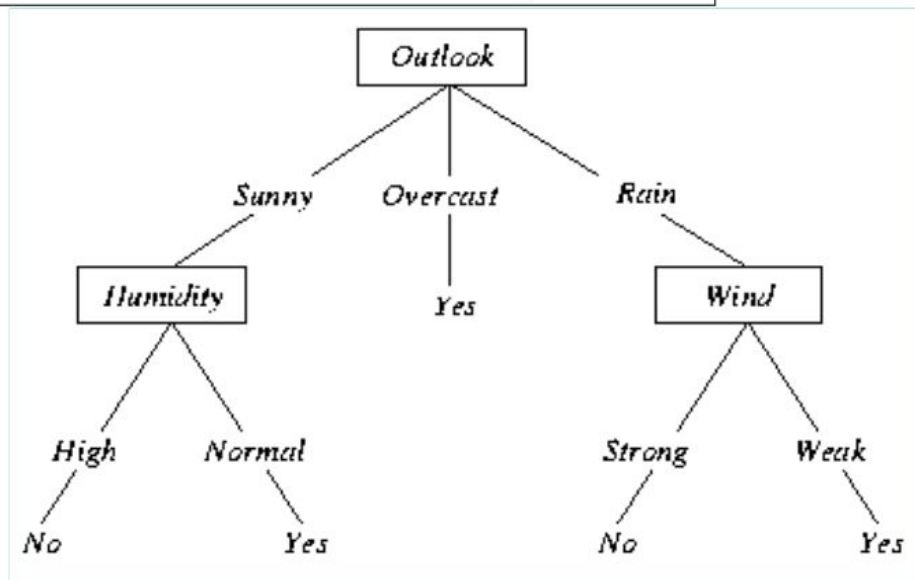


# Decision Trees represent disjunction of conjunctions

$(\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal})$

$\vee (\text{Outlook} = \text{Overcast})$

$\vee (\text{Outlook} = \text{Rain} \wedge \text{Wind} = \text{Weak})$



# Appropriate Problems for Decision Tree Learning

- Instances are represented by attribute-value pairs
  - each attribute takes on a small no of disjoint possible values, eg, hot, mild, cold
  - extensions allow real-valued variables as well, eg temperature
- The target function has discrete output values
  - eg, Boolean classification (yes or no)
  - easily extended to multiple-valued functions
  - can be extended to real-valued outputs as well

# Appropriate Problems for Decision Tree Learning

- Disjunctive descriptions may be required
  - naturally represent disjunctive expressions
- The training data may contain errors
  - robust to errors in classifications and in attribute values
- The training data may contain missing attribute values
  - eg, humidity value is known only for some training examples

# Appropriate Problems for Decision Tree Learning

- Practical problems that fit these characteristics are:
  - learning to classify
    - medical patients by their disease
    - equipment malfunctions by their cause
    - loan applications by likelihood of default of payments

# Capabilities and Limitations of ID3

- Hypothesis space is a complete space of all discrete valued functions
- Cannot determine how many alternative trees are consistent with training data (follows from maintaining a single current hypothesis)
- ID3 in its pure form performs no backtracking (usual risks of hill-climbing- converges to local optimum)
- ID3 uses all training examples at each step to make statistically based decisions regarding how to refine its current hypothesis
  - more robust than Find-S and Candidate Elimination which are incrementally-based