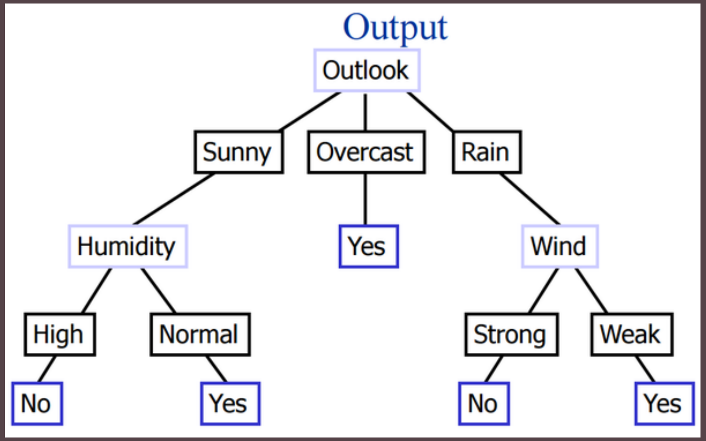
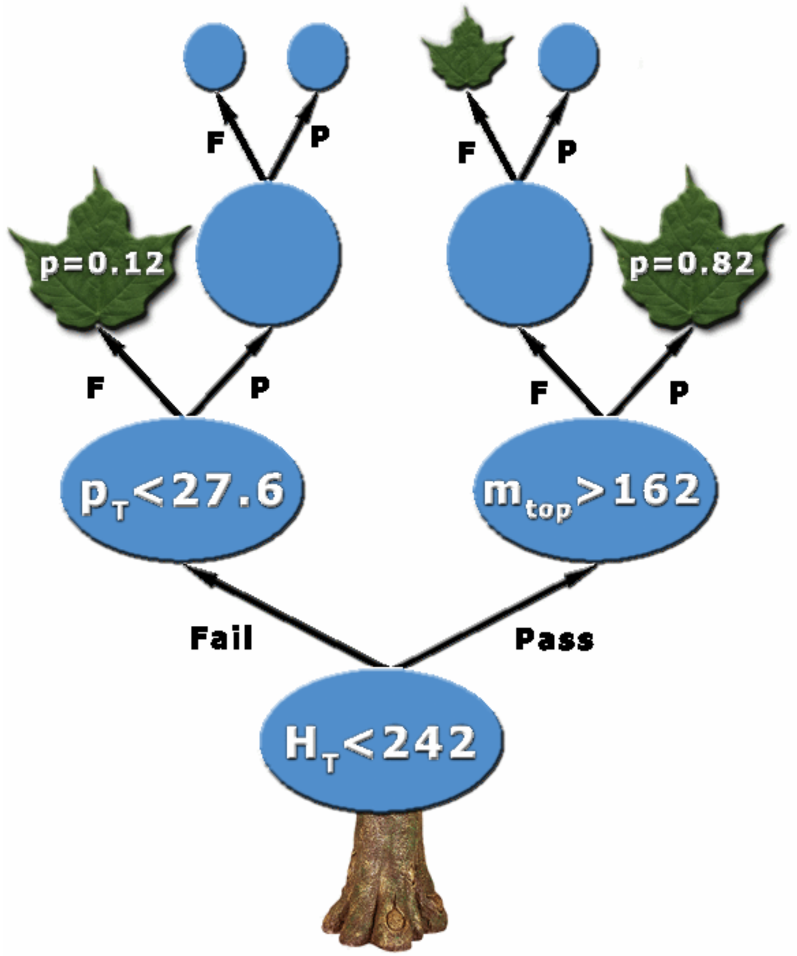




Decision Tree Learning

Module 2

Reference : Machine Learning by Tom M Mitchell (Chapter 3)

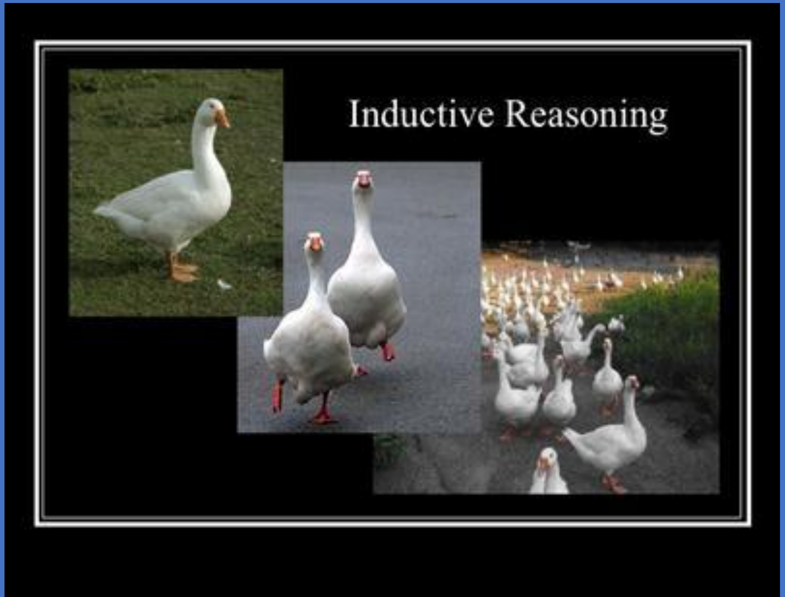
ಡಾ|| ತ್ಯಾಗರಾಜು ಜಿ.ಎಸ್
Head Dept of CSE
SDMIT Ujire

Context Innovations Lab

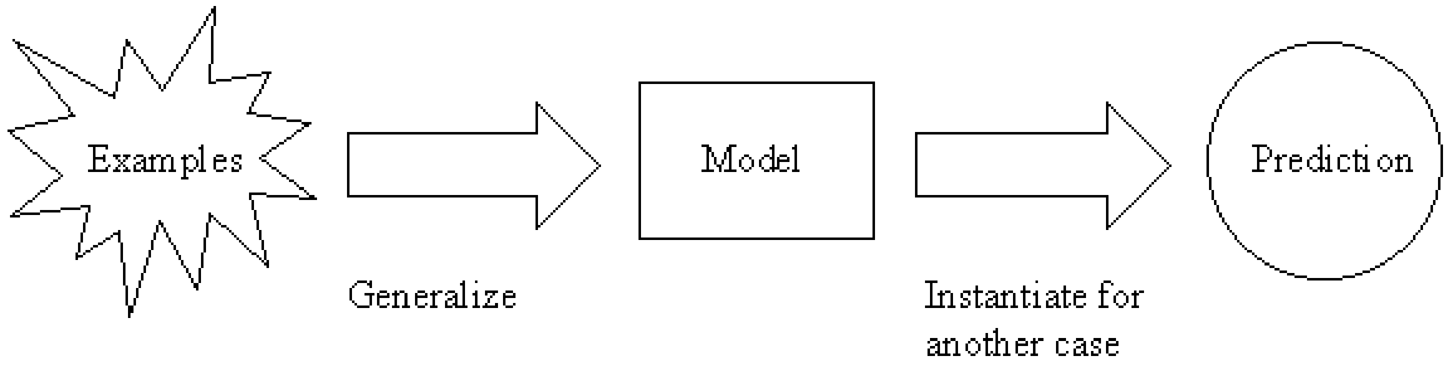
Topics

1. Decision Tree Learning
2. Decision Trees
3. Decision Tree Representation
4. Appropriate Problems for Decision Tree Learning
5. Basic Decision Tree Learning Algorithm (ID3)
6. Hypothesis Space Search in decision Tree Learning
7. Inductive Bias in Decision Tree Learning
8. Issues in Decision Tree Learning

1. Decision Tree Learning



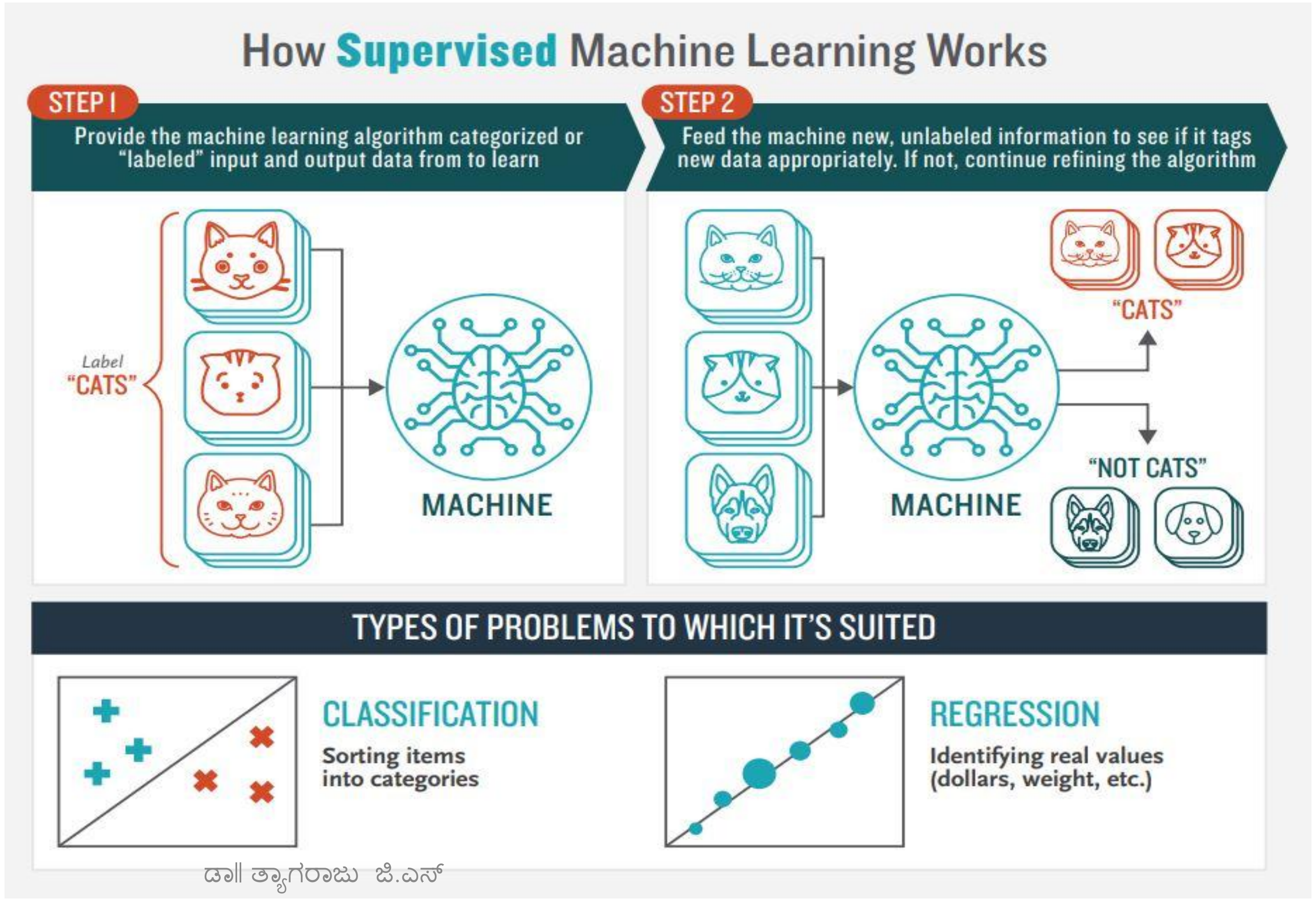
- **Decision tree learning is one of the most widely used and practical methods for inductive inference. It is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree.**



Inductive inference is the process of reaching a general conclusion from specific examples.

1. Decision Tree Learning

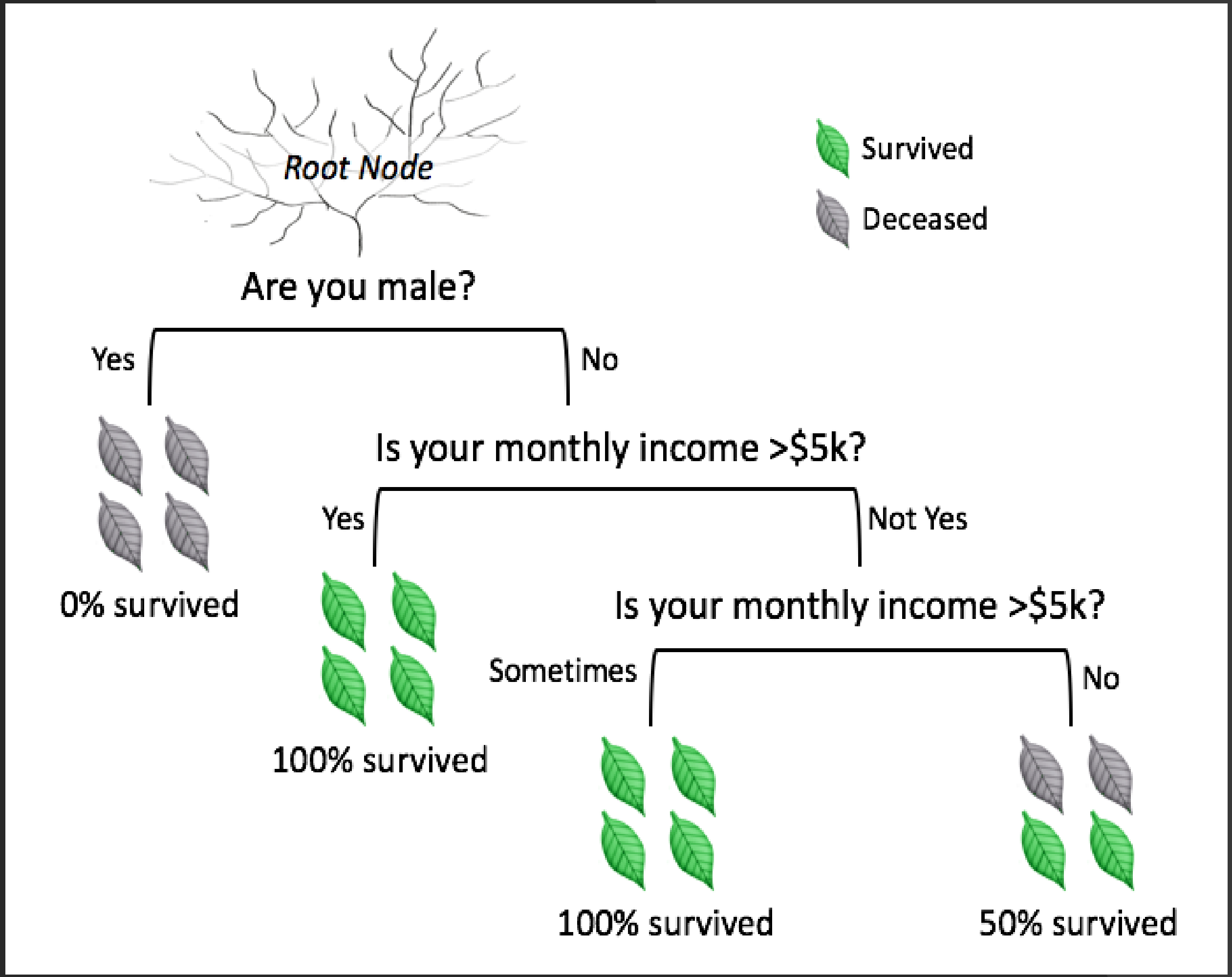
- Is a type of supervised machine learning that is mostly used in classification .



How Supervised Machine Learning Works
STEP 1
Provide the machine learning algorithm categorized or "labeled" input and output data from to learn
STEP 2
Feed the machine new, unlabeled information to see if it tags new data appropriately. If not, continue refining the algorithm
Label "CATS"
MACHINE
MACHINE
"CATS"
"NOT CATS"
TYPES OF PROBLEMS TO WHICH IT'S SUITED
CLASSIFICATION
Sorting items into categories
REGRESSION
Identifying real values (dollars, weight, etc.)
ಡಾ॥ ತ್ಯಾಗರಾಜು ಜಿ.ಎಸ್

2. Decision Trees

- A decision tree is a graphical representation of all possible solutions to decisions based on certain conditions.
- A tree structured classifier with two types of nodes :
  - **Decision Nodes** and
  - **Leaf Nodes**.



Inductive inference with decision trees

- *Decision Trees* is one of the most widely used and practical methods of *inductive inference*
- Features
  - Method for approximating *discrete-valued* functions (including boolean)
  - Learned functions are represented as *decision trees* (or *if-then-else* rules)
  - Expressive hypotheses space, including disjunction
  - Robust to noisy data

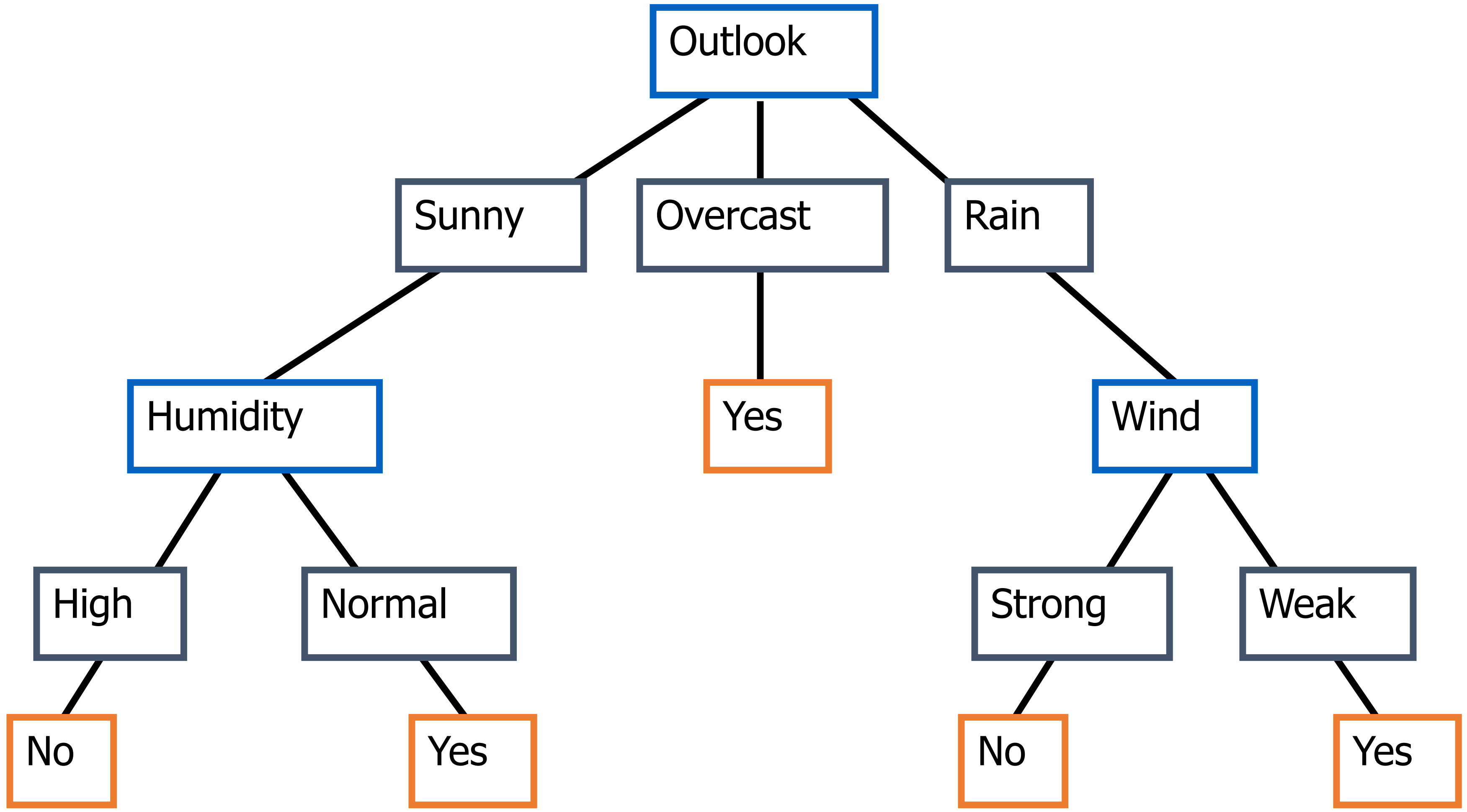
3. Decision Tree Representation

- **Decision trees** classify instances by sorting them down the tree *from the root to some leaf node*, which provides the classification of the instance.
- In the **decision tree representation**:
  - An inner node represents an attribute
  - Edge / branch represents an attribute value
  - Leaf represents a class
  - The paths from root to leaf represent **classification rules**.
- **Decision trees represent a disjunction of conjunctio**ns.
  - Each path from root to a leaf is a conjunction of attribute tests.
  - The tree itself is a disjunction of these conjunctions.

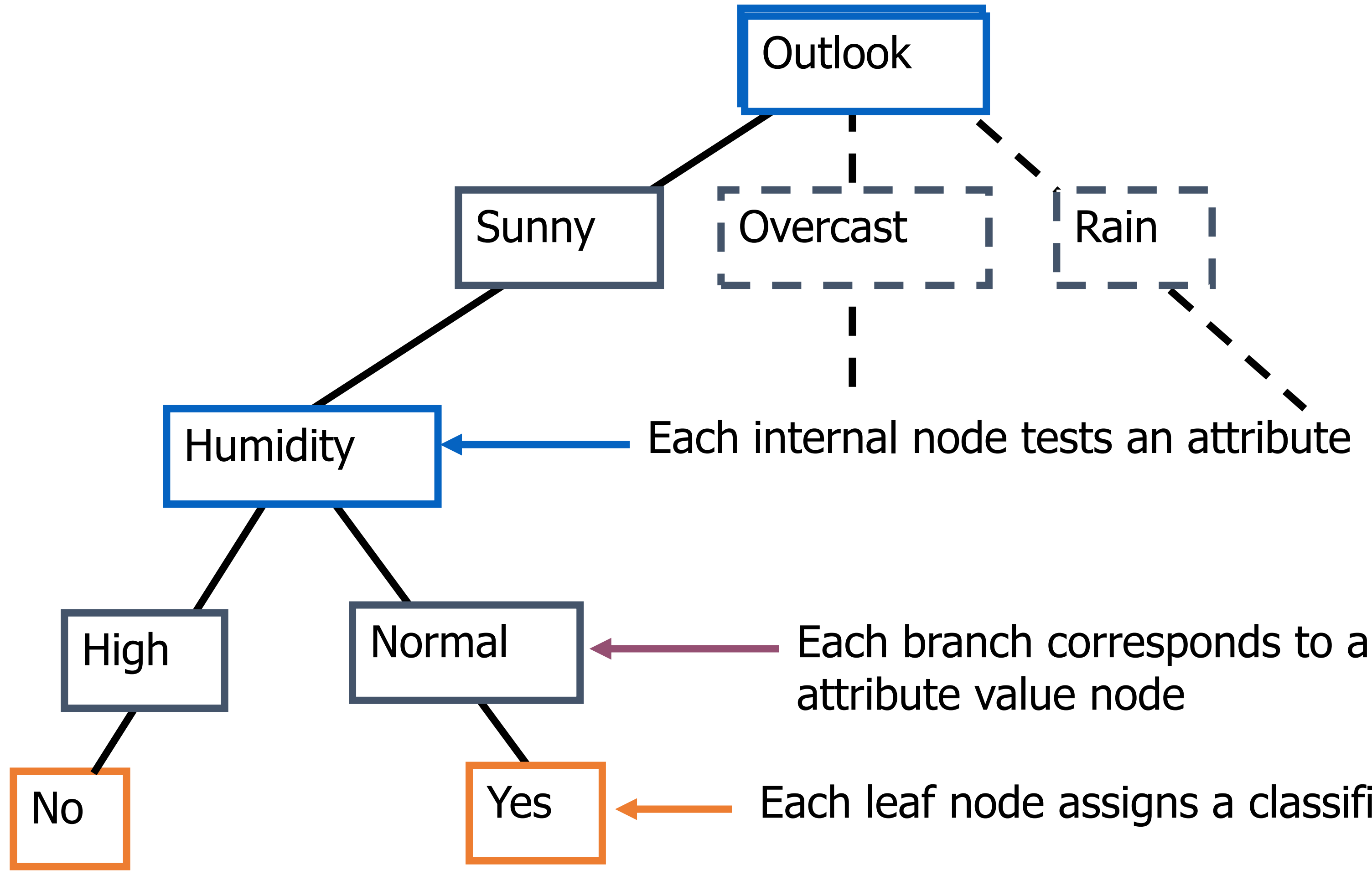
Training Examples (Data Set)

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

Decision Tree for PlayTennis



Decision Tree for PlayTennis



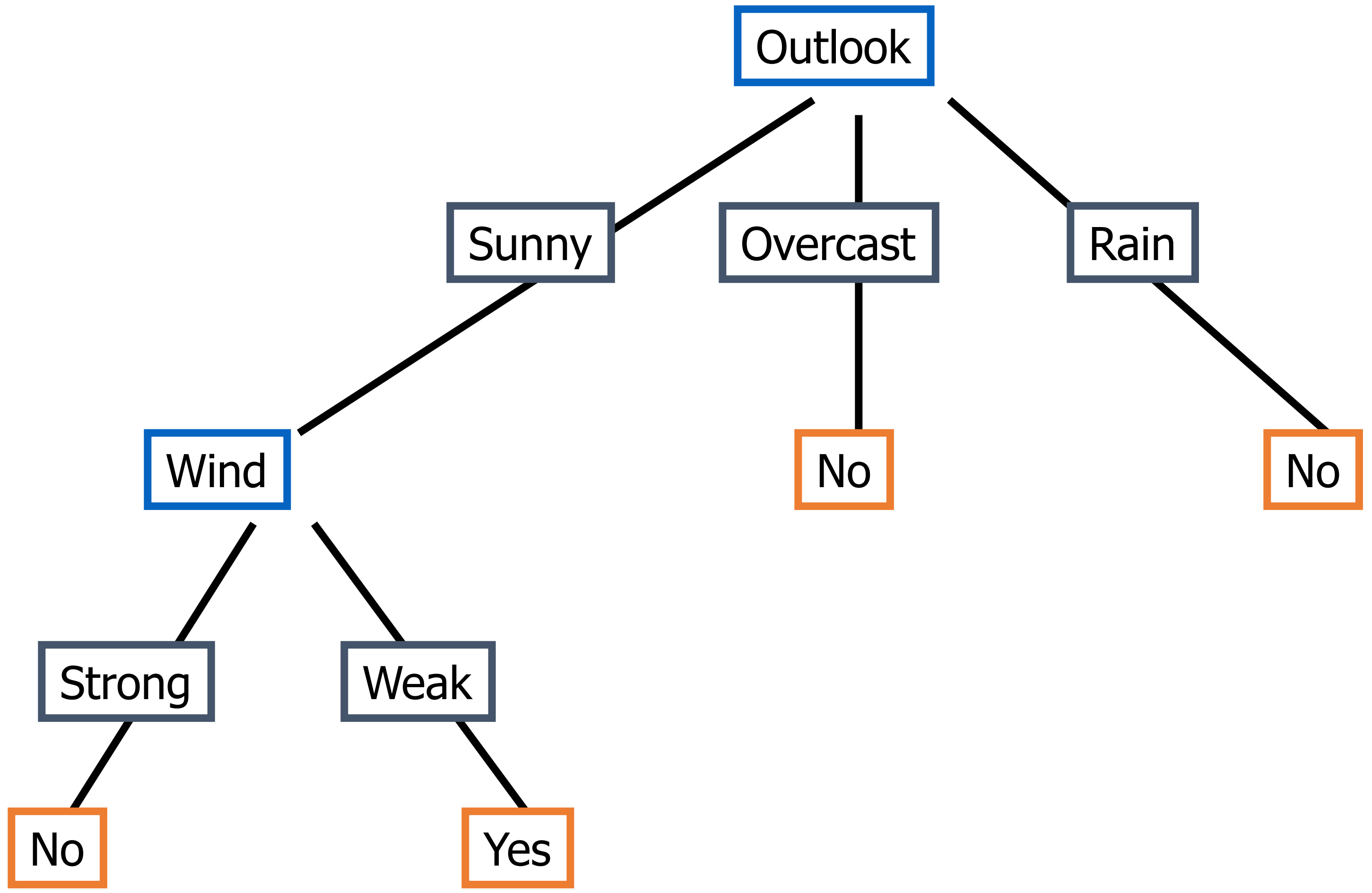
Each internal node tests an attribute

Each branch corresponds to an attribute value node

Each leaf node assigns a classification

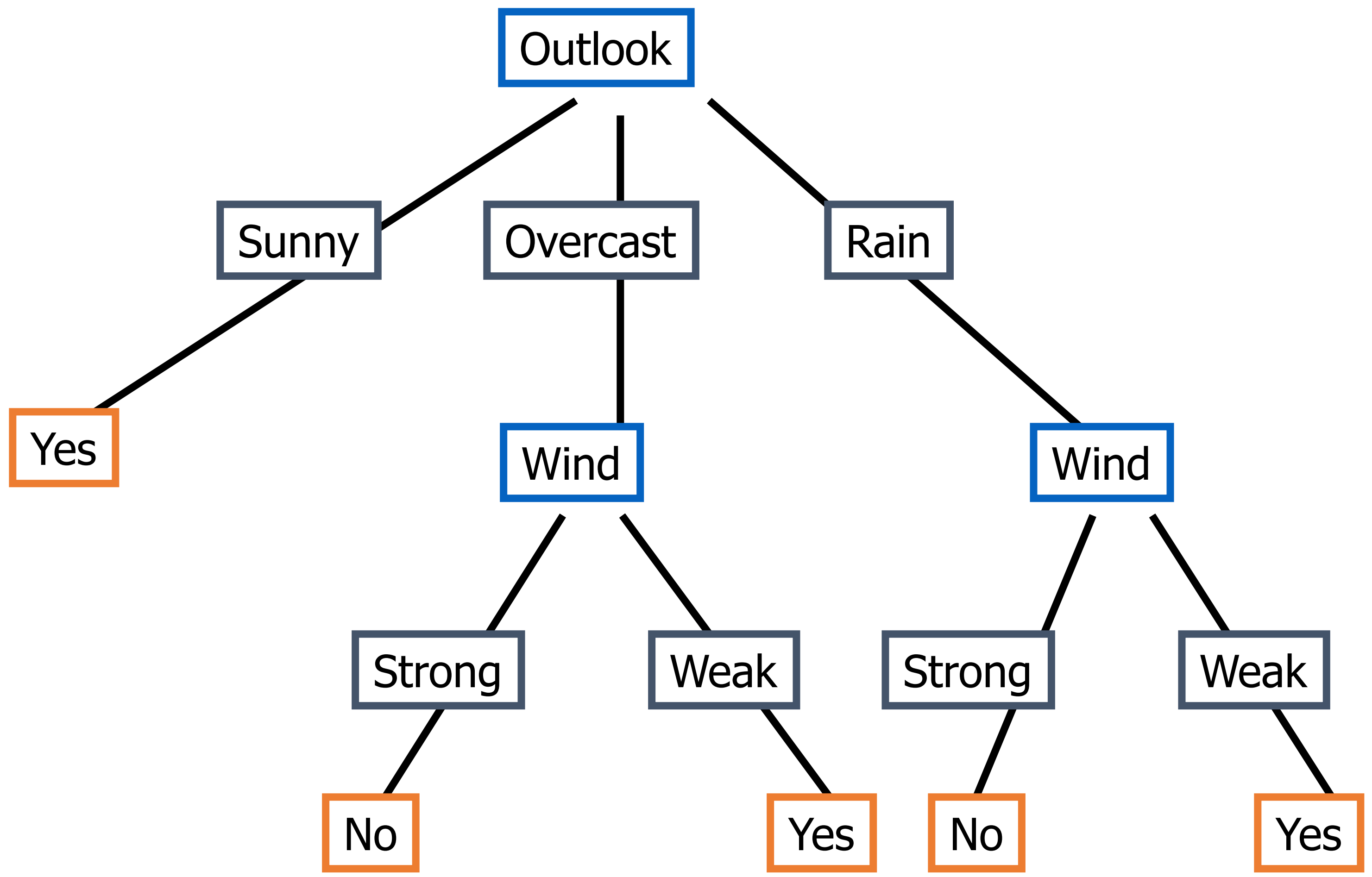
Decision Tree for Conjunction

Outlook=Sunny $\wedge$ Wind=Weak



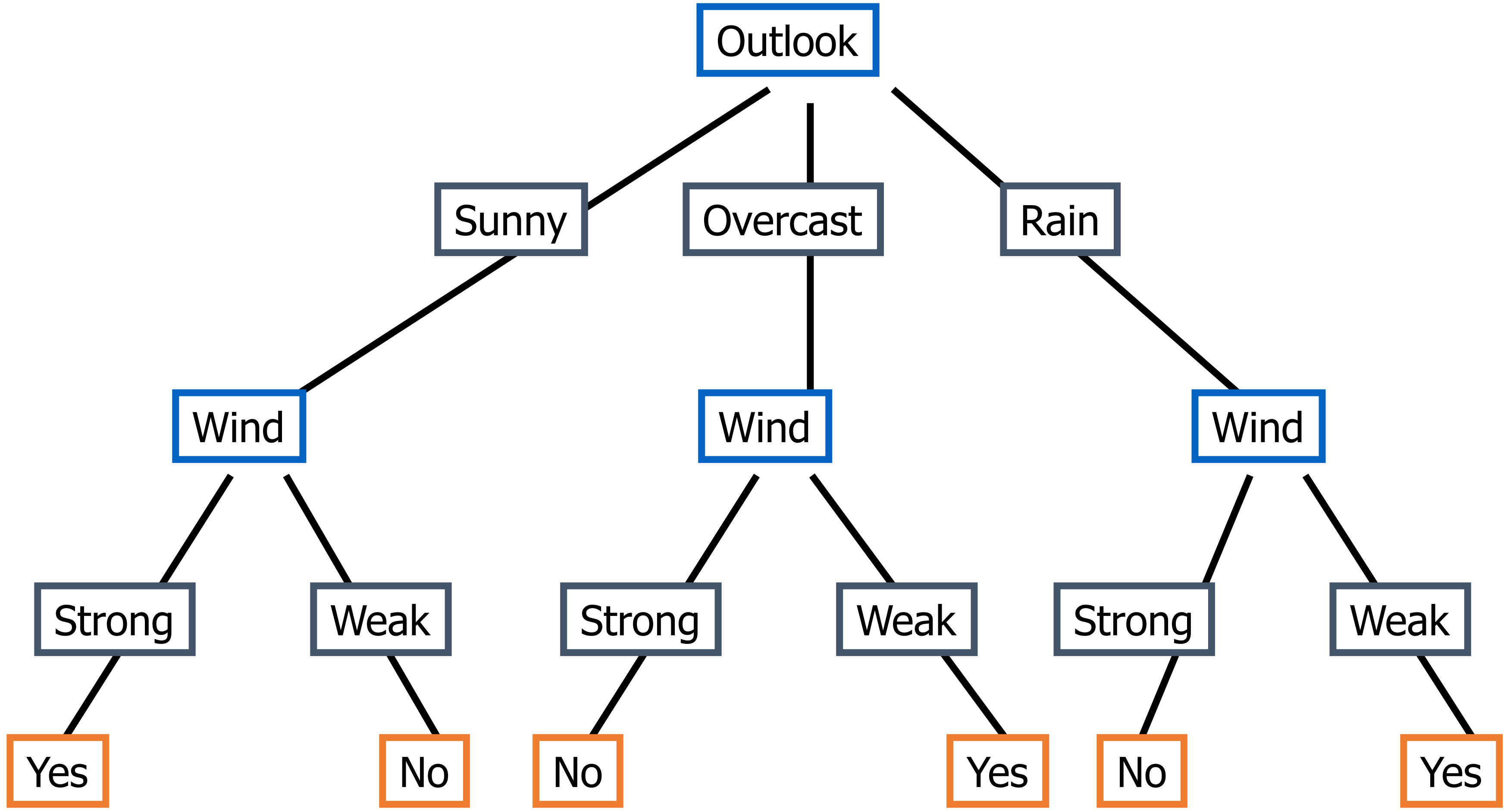
Decision Tree for Disjunction

Outlook=Sunny ∨ Wind=Weak



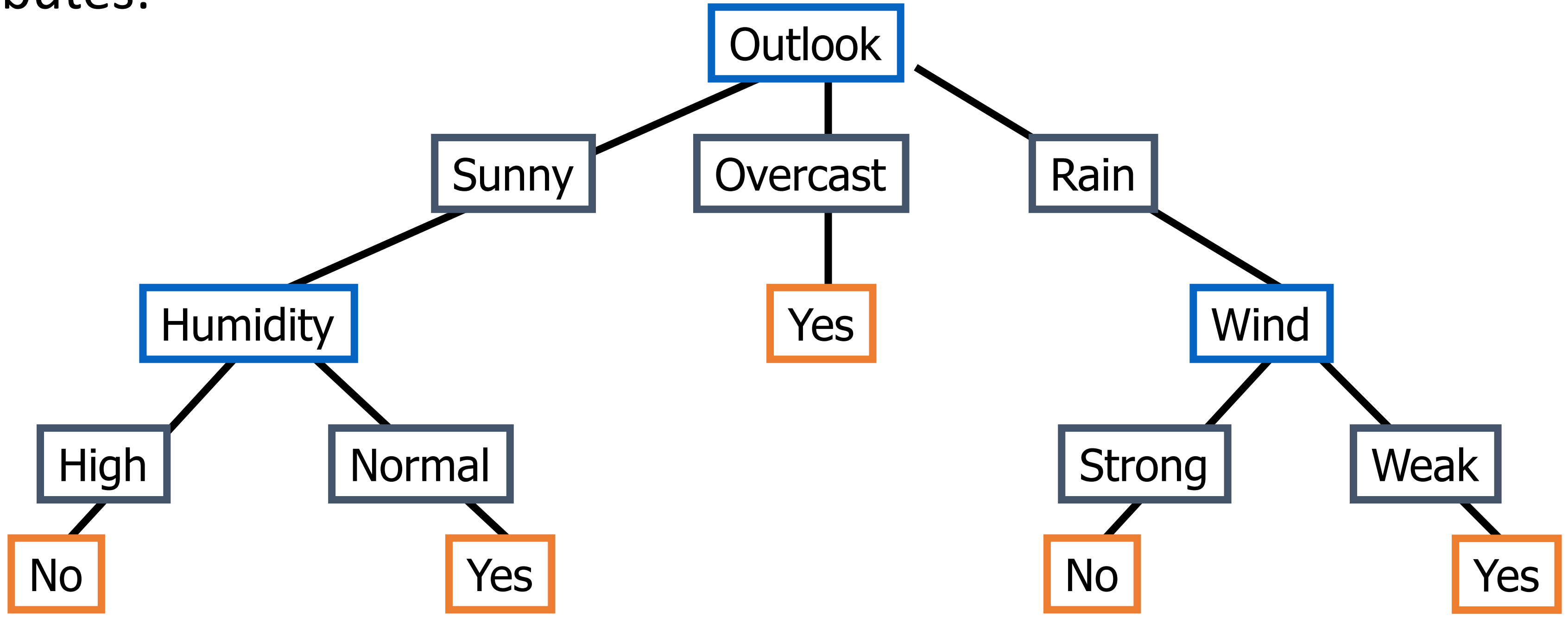
Decision Tree for XOR

Outlook = Sunny XOR Wind = Weak



Decision Tree Expressivity

- Decision trees represent a disjunction of conjunctions on constraints on the value of attributes:

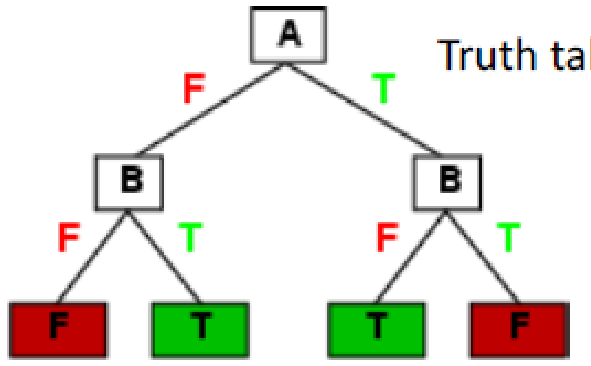


(Outlook=Sunny $\wedge$ Humidity=Normal)
$\vee$ **(Outlook=Overcast)**
$\vee$ **(Outlook=Rain $\wedge$ Wind=Weak)**

Expressiveness

- Decision trees can represent any boolean function of the input attributes

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



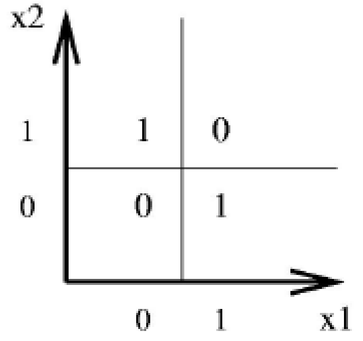
Truth table row → path to leaf

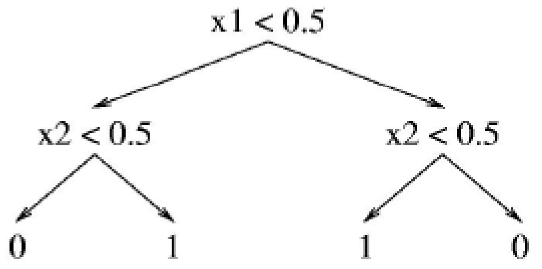
- In the worst case, the tree will require exponentially many nodes

Expressiveness

Decision trees have a variable-sized hypothesis space

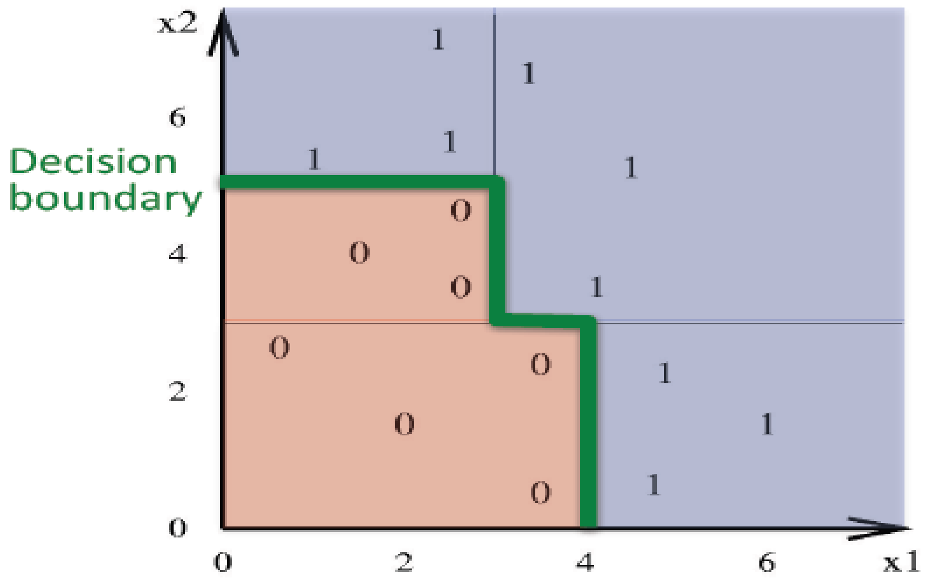
- As the #nodes (or depth) increases, the hypothesis space grows
  - Depth 1 ("decision stump"): can represent any boolean function of one feature
  - Depth 2: any boolean fn of two features; some involving three features (e.g., $(x_1 \land x_2) \lor (\neg x_1 \land \neg x_3)$)
  - etc.

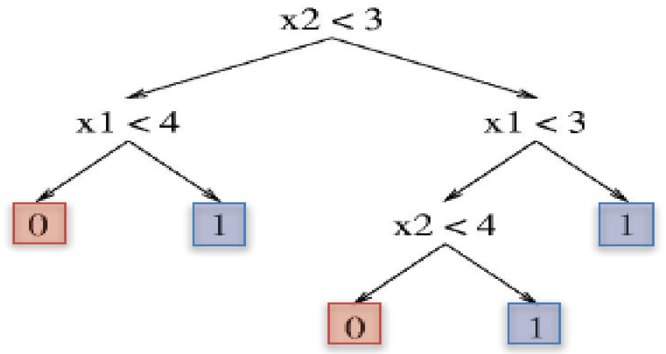




Decision Tree – Decision Boundary

- Decision trees divide the feature space into axis-parallel (hyper-)rectangles
- Each rectangular region is labeled with one label
  - or a probability distribution over labels





4. Appropriate Problems for Decision Tree Learning

- **Decision tree learning is generally best suited to problems with the following characteristics**:
  - Instances describable by attribute-value pairs. (Hot, Mild, Cold)
  - Target function has discrete output values (yes or no).
  - Disjunctive hypothesis/description may be required.
  - The training data may contain errors/noisy data.
  - The training data may contain missing attribute values.
- **Some examples of problems that fit to these characteristics are:**
  - Medical or equipment diagnosis
  - Credit risk analysis
  - Modelling calendar scheduling preferences

5.Basic Decision Tree Learning Algorithm (ID3)

- The basic decision tree learning algorithm, ID3, employs a top-down, greedy search through the space of possible decision trees, beginning with the question ”**which attribute should be tested at the root of the tree?**”.

ID3 - Algorithm

```
ID3(Examples, TargetAttribute, 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, Return the single-node tree Root, with label = most common
    value of TargetAttribute in Examples
•   Otherwise Begin
     –  A ← the attribute from Attributes that best classifies Examples
     –  The decision 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 value vi for A
          •  If $Examples_{vi}$ is empty
               –  Then below this new branch add a leaf node with label = most common
                  value of TargetAttribute in Examples
               –  Else below this new branch add the subtree
                     ID3($Examples_{vi}$, TargetAttribute, Attributes – {A})
•   End
•   Return Root
```

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

Choosing the Best Attribute

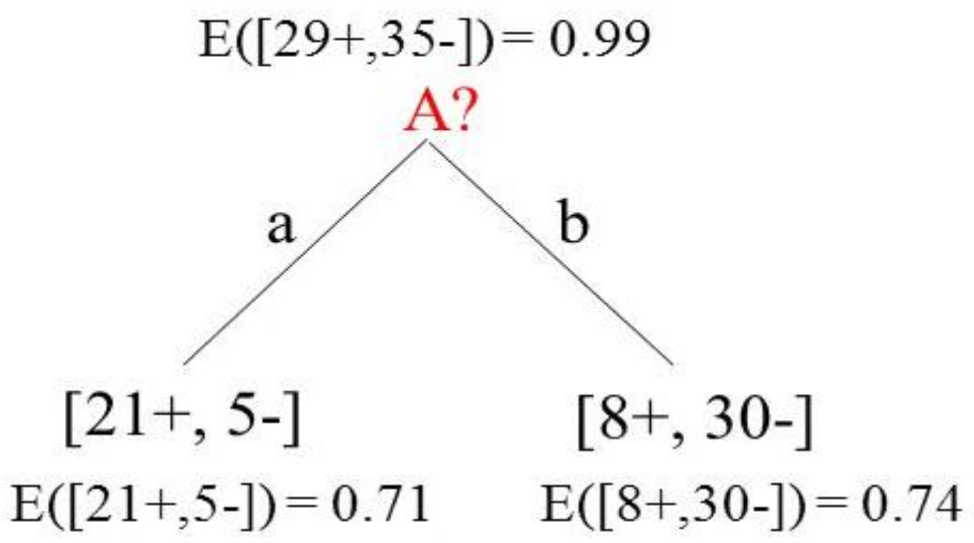
Key problem: choosing which attribute to split a given set of examples

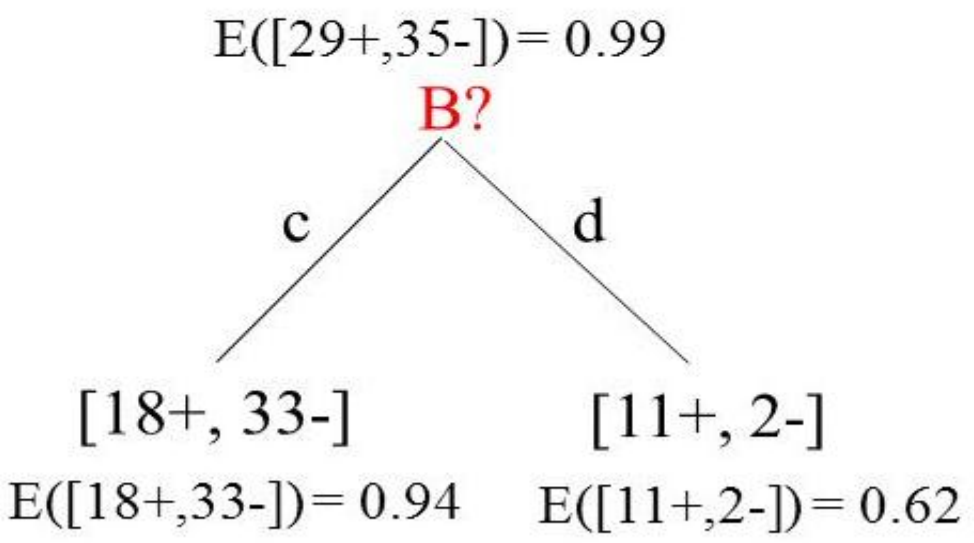
- Some possibilities are:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose the attribute with the smallest number of possible values
  - **Most-Values:** Choose the attribute with the largest number of possible values
  - **Max-Gain:** Choose the attribute that has the largest expected *information gain*
    - i.e., attribute that results in smallest expected size of subtrees rooted at its children
- The ID3 algorithm uses the Max-Gain method of selecting the best attribute

Which attribute is the Best Classifier ?

- The central choice in the ID3 algorithm is selecting which attribute to test at each node in the tree. The attribute must be selected that is most useful for classifying examples .
- The statistical property called information gain is the good quantitative measure of the worth of an attribute .
- Information Gain measures how well a given attribute separates the training examples according to their target classification.
- ID3 uses this information gain measure to select among the candidate attributes at each step while growing the tree.
- Information gain uses the notion of *entropy*, commonly used in information theory
- Information gain = expected reduction of entropy

Which attribute is the best classifier?



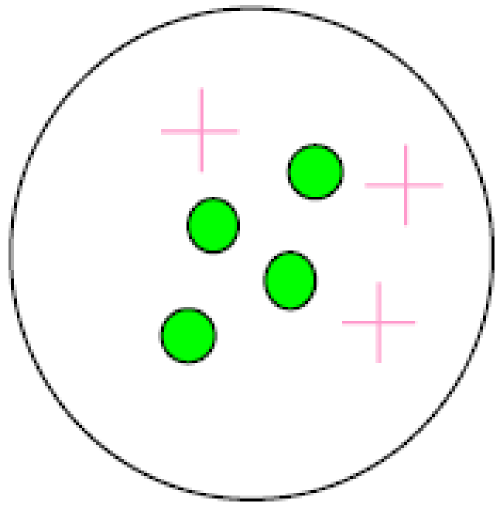


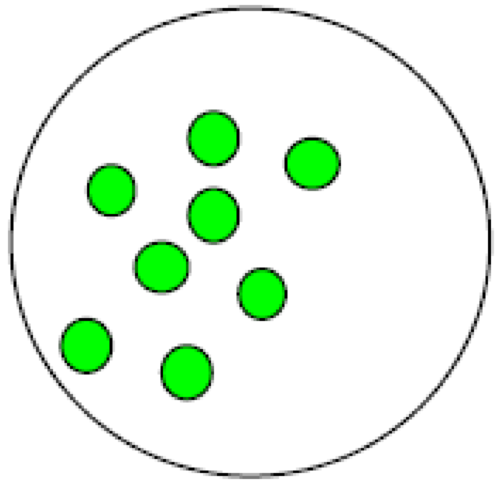
Gain(S,A) = Entropy(S)
-26/64*Entropy([21+,5-])
-38/64*Entropy([8+,30-])
= **0.27**

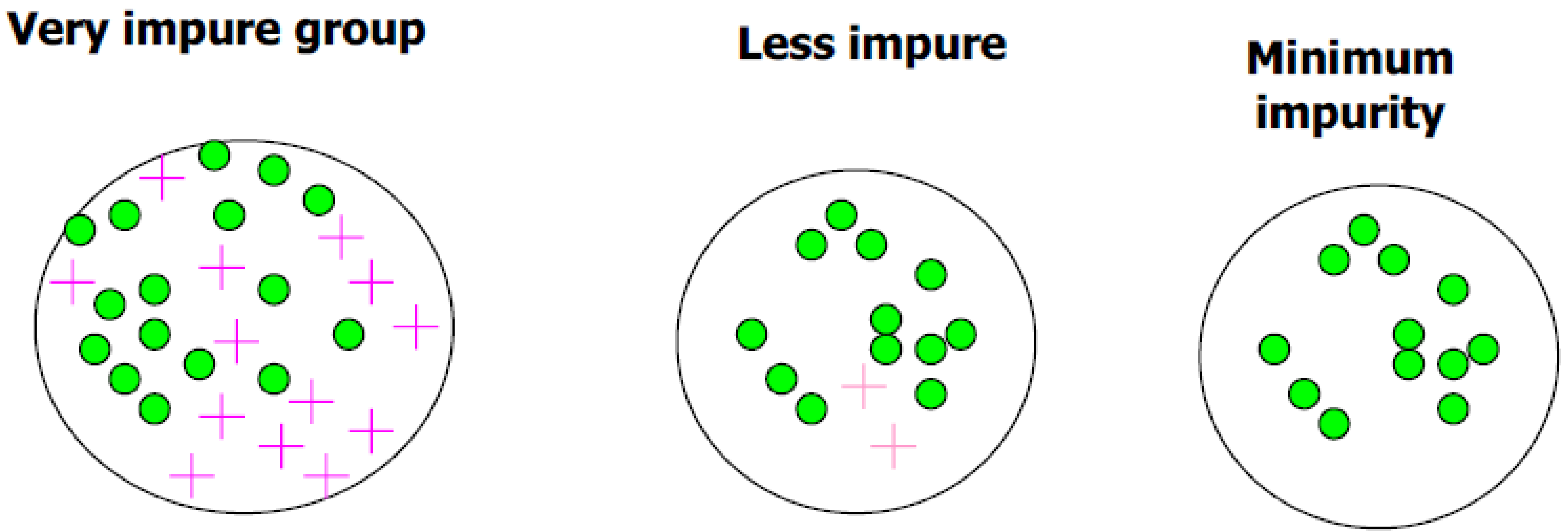
Gain(S,B) = Entropy(S)
-51/64*Entropy([18+,33-])
-13/64*Entropy([11+,2-])
= **0.12**

Impurity/Entropy (informal)

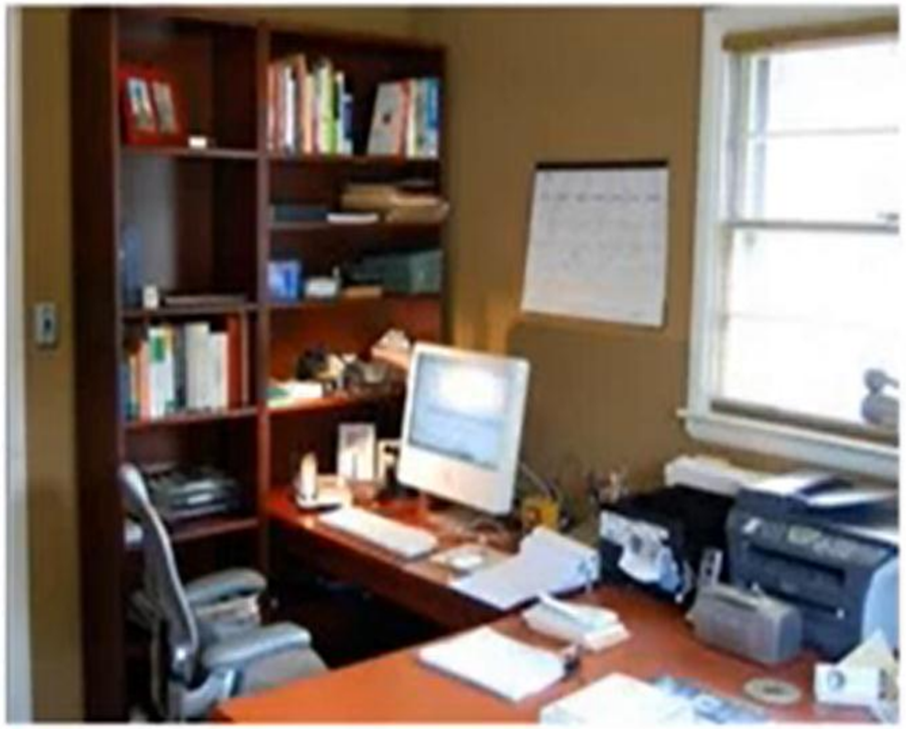
– Measures the level of **impurity** in a group of examples





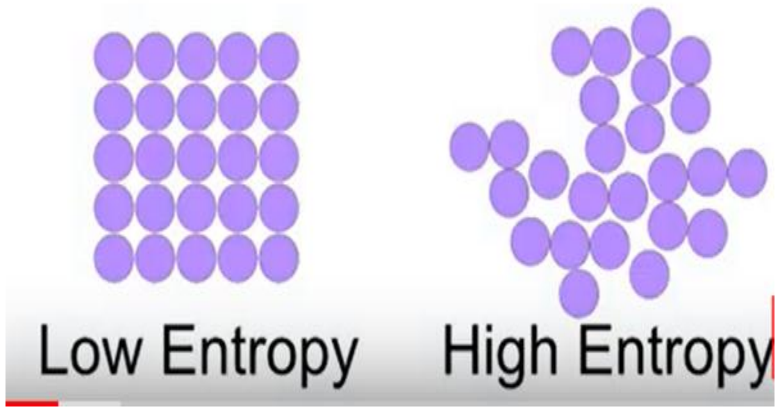

Very impure group
Less impure
Minimum impurity

High Entropy (messy)



Low Entropy (Clean)

What is Entropy ?



- Define and measures Randomness in the Data
- Entropy is just a metric which measures the impurity
- Minimum number of bits of information needed to encode the classification of an arbitrary member of Examples Space S.
- If the target attribute can take on **c** possible values the entropy can be as large as $\boldsymbol{log_2\ c}$

1. Entropy measures Homogeneity of Examples

- In order to define information gain precisely, we begin by defining a measure called entropy, that characterizes the (im)purity of an arbitrary collection of examples. That is, it measures the homogeneity of examples.

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

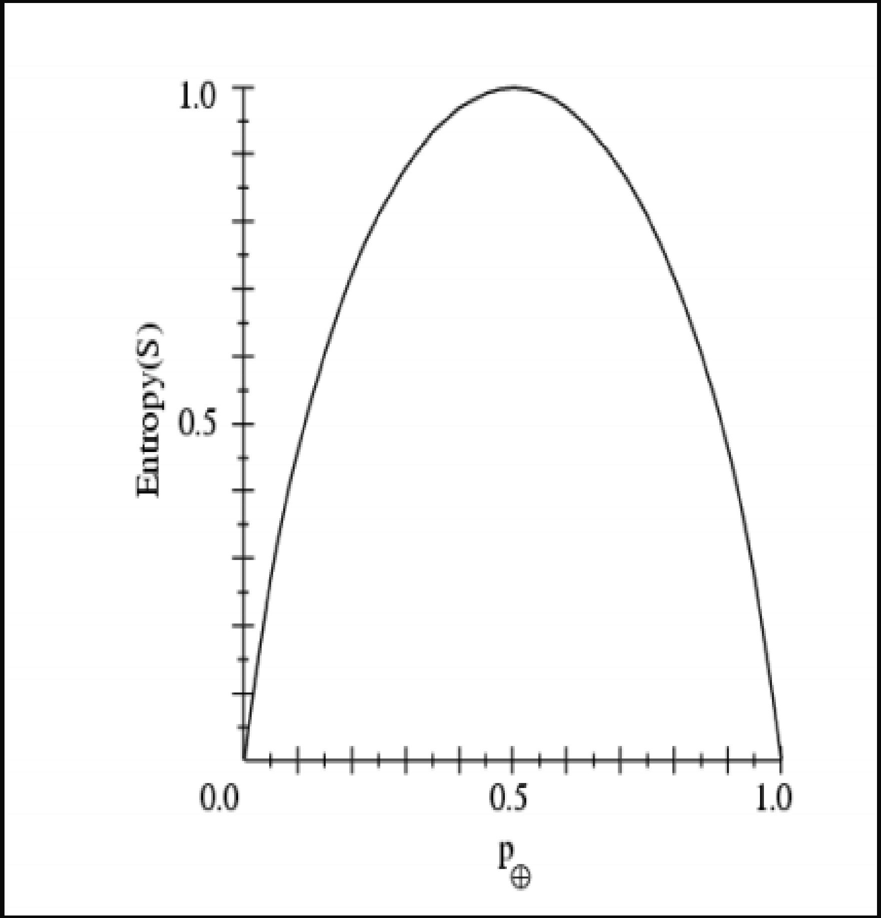
Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)

= Entropy (0.36, 0.64)

= - $(0.36 \log_2 0.36)$ - $(0.64 \log_2 0.64)$

= 0.94



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

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

Examples

- E(s) = - $\boldsymbol{p_+ log_2 p_+}$ - $\boldsymbol{p_- \; log_2 \; p_-}$

- E(9+,5-) = -(9/4)$\boldsymbol{log_2}$(9/14) - (5/14) $\boldsymbol{log_2}$(5/14)

  = 0.940

Entropy in 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

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

$$Entropy\ ([14+, 0–]) = -14/14 \log_2 (14/14) - 0 \log_2 (0) = 0$$

$$Entropy\ ([9+, 5–]) = -9/14 \log_2 (9/14) - 5/14 \log_2 (5/14) = 0{,}94$$

$$Entropy\ ([7+, 7–]) = -7/14 \log_2 (7/14) - 7/14 \log_2 (7/14) =$$

$$= 1/2 + 1/2 = 1 \qquad [\log_2 1/2 = -1]$$

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

Entropy and information theory

- Entropy specifies the number the average length (in bits) of the message needed to transmit the outcome of a random variable. This depends on the probability distribution.
- Optimal length code assigns $\lceil -log_2\, p \rceil$ bits to messages with probability p. Most probable messages get shorter codes.
- Example: 8-sided [unbalanced] die

1	2	3	4	5	6	7	8
4/16	4/16	2/16	2/16	1/16	1/16	1/16	1/16
2 bits	*2 bits*	*3 bits*	*3 bits*	*4bits*	*4bits*	*4bits*	*4bits*

$E = (1/4\ log_2\ 4) \times 2 + (1/8\ log_2\ 8) \times 2 + (1/16\ log_2\ 16) \times 4 = 1+3/4+1 = 2{,}75$

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]

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

  for classification in *c* classes

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

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

2. Information Gain Measures the expected reduction in Entropy

- **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.

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

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

Example:

- Values (Wind) = Weak, Strong
- S = [9+,5-]
- $S_{weak} = [9+, 5 -]$
- $S_{strong} = [3+, 3 -]$

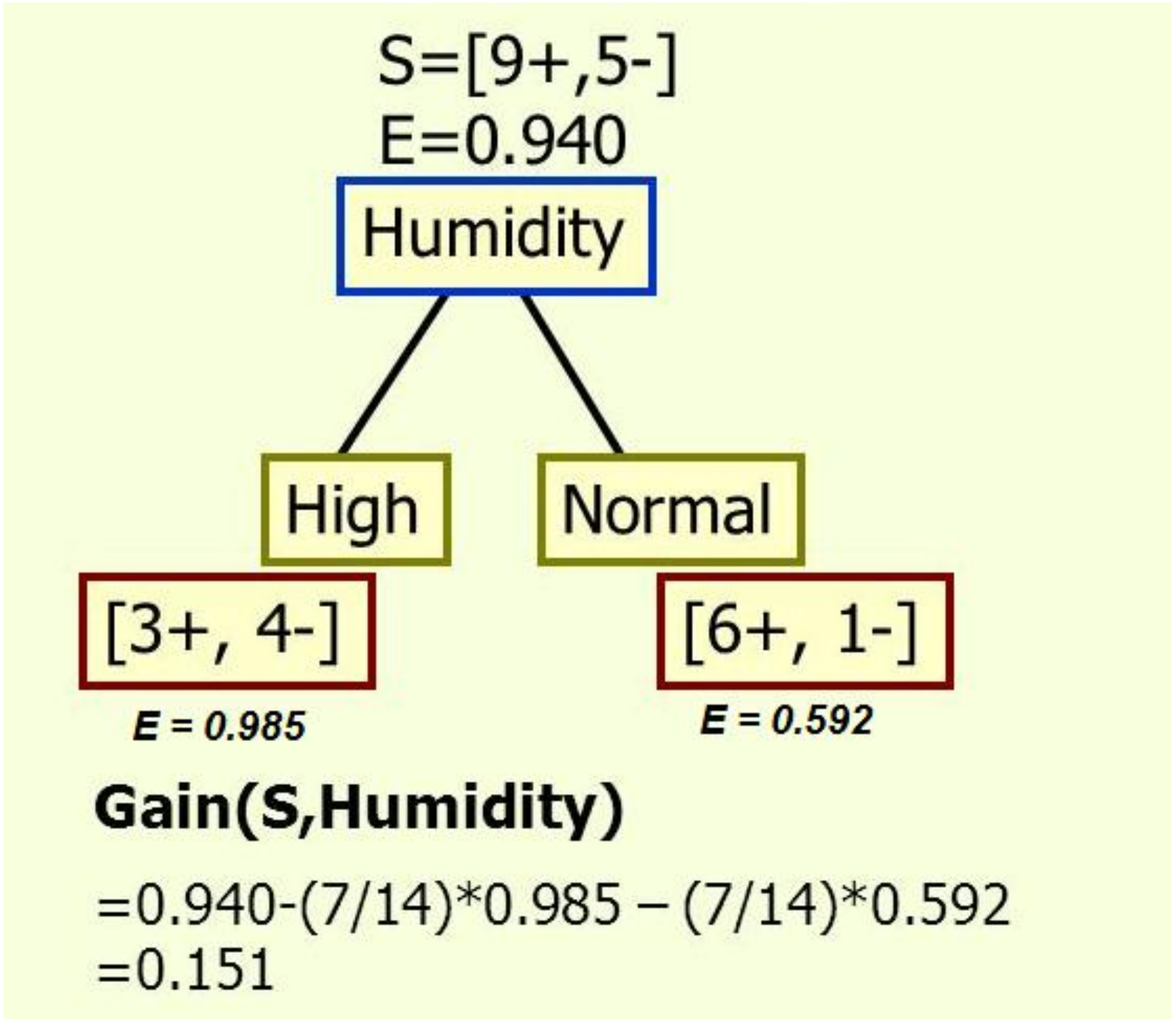
$$\text{Gain}(S, \text{wind}) = \text{Entropy}(S) - \sum_{v \in \{weak, strong\}} (|S_v| / |S|)\ \text{Entropy}(S_v)$$

= Entropy (S) – (8/14)Entropy(S_{weak}) – (6/4)Entropy(S_{strong})

= 0.940 – (8/14)0.811 – (6/14)1.00

= 0.048

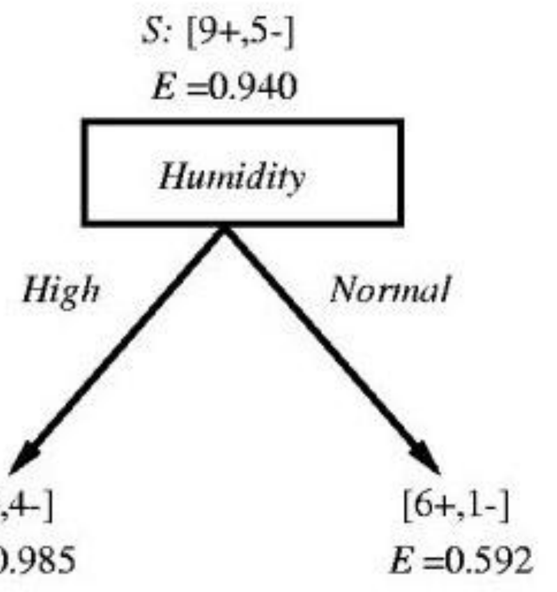
For Humidity



Gain(S,Humidity)

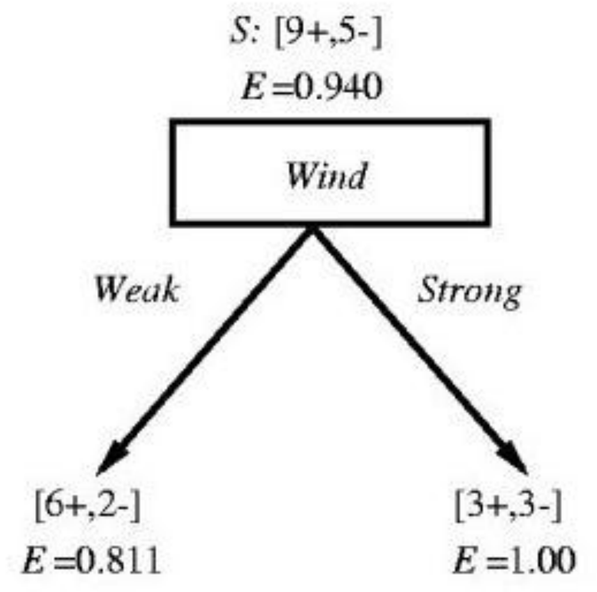
=0.940-(7/14)*0.985 – (7/14)*0.592
=0.151

For Humidity and wind



Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151



Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

Entropy-Based Automatic Decision Tree Construction

Training Set X

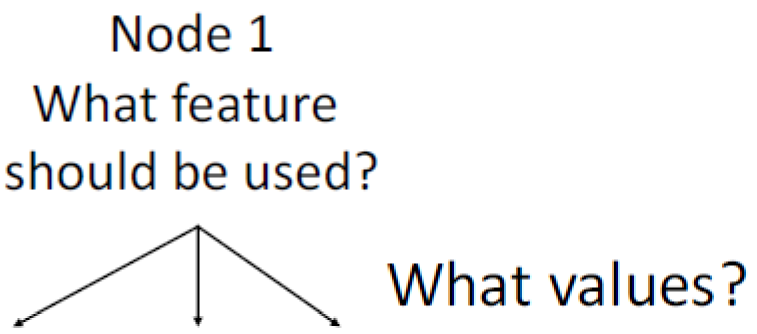
x1=(f11,f12,...f1m)

x2=(f21,f22,   f2m)

.

.

xn=(fn1,f22,   f2m)

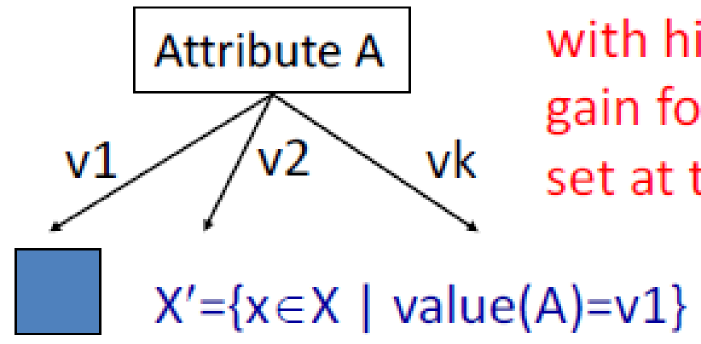


Quinlan suggested information gain in his ID3 system and later the gain ratio, both based on entropy.

Using Information Gain to Construct a Decision Tree

Full Training Set X

Choose the attribute A with highest information gain for the full training set at the root of the tree.



Construct child nodes for each value of A. Each has an associated subset of vectors in which A has a particular value.

repeat recursively till when?

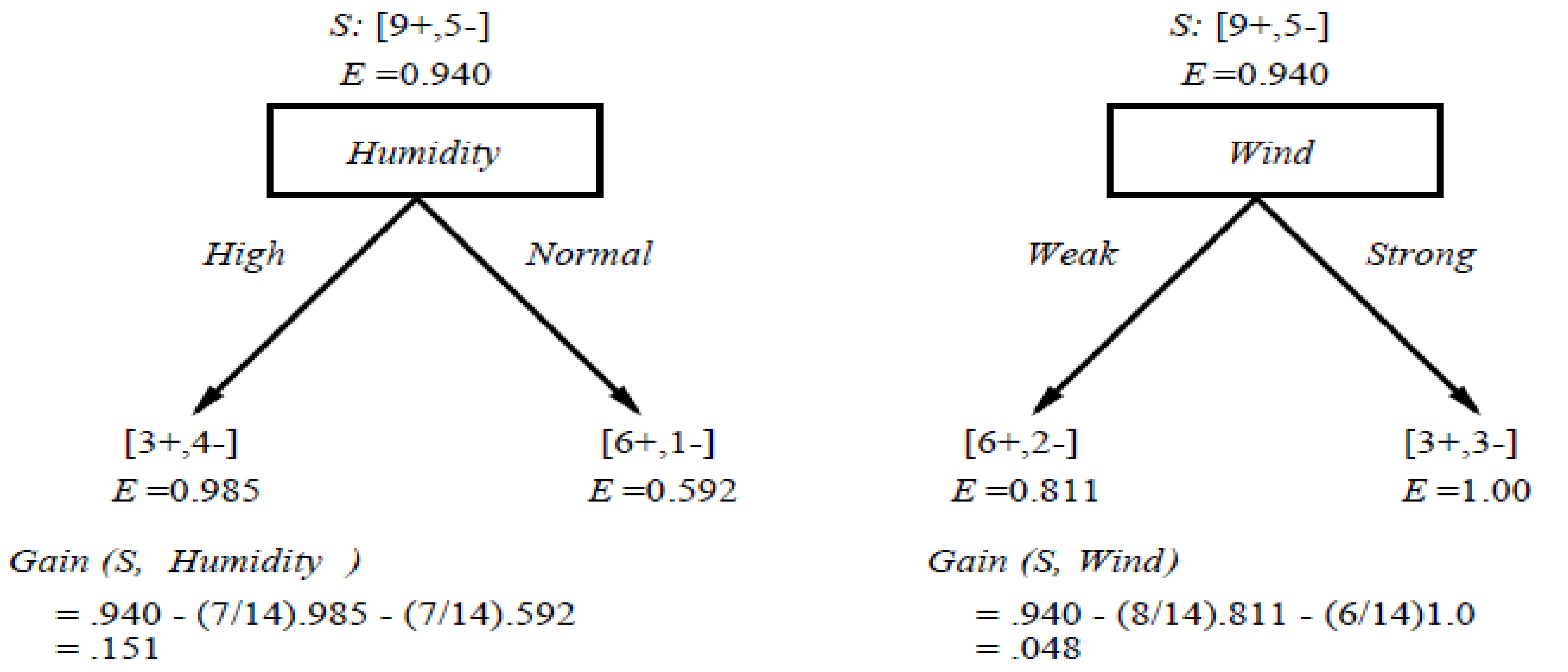
An Illustrative Example

To illustrate the operation of ID3, let's consider the learning task represented by the below examples.

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**

Compute the Gain and identify which attribute is the best as illustrated below

Which attribute is the best classifier?

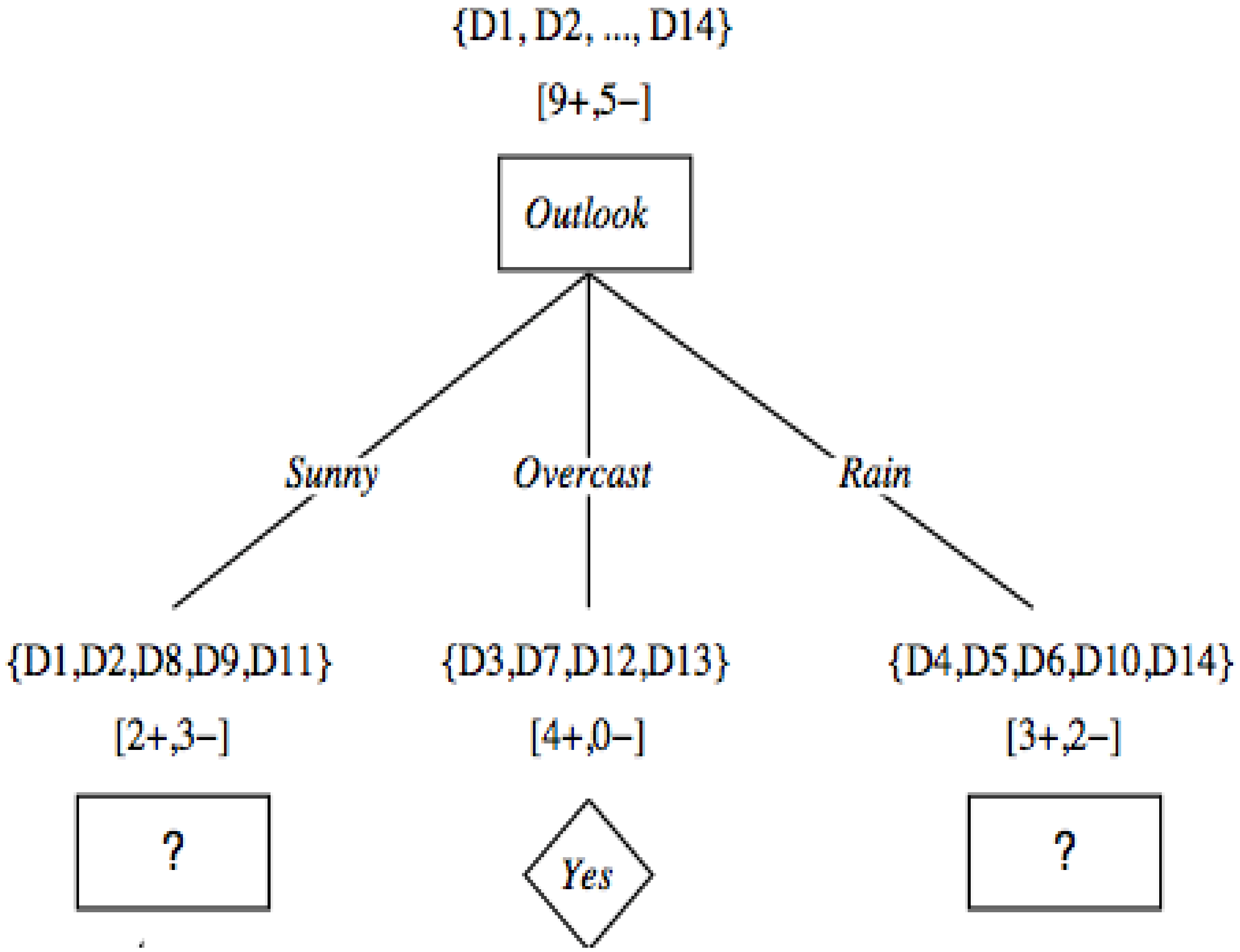


Context Innovations Lab

Which attribute to test at the root?

- 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*

After first step



Second step

- Working on *Outlook=Sunny* node:

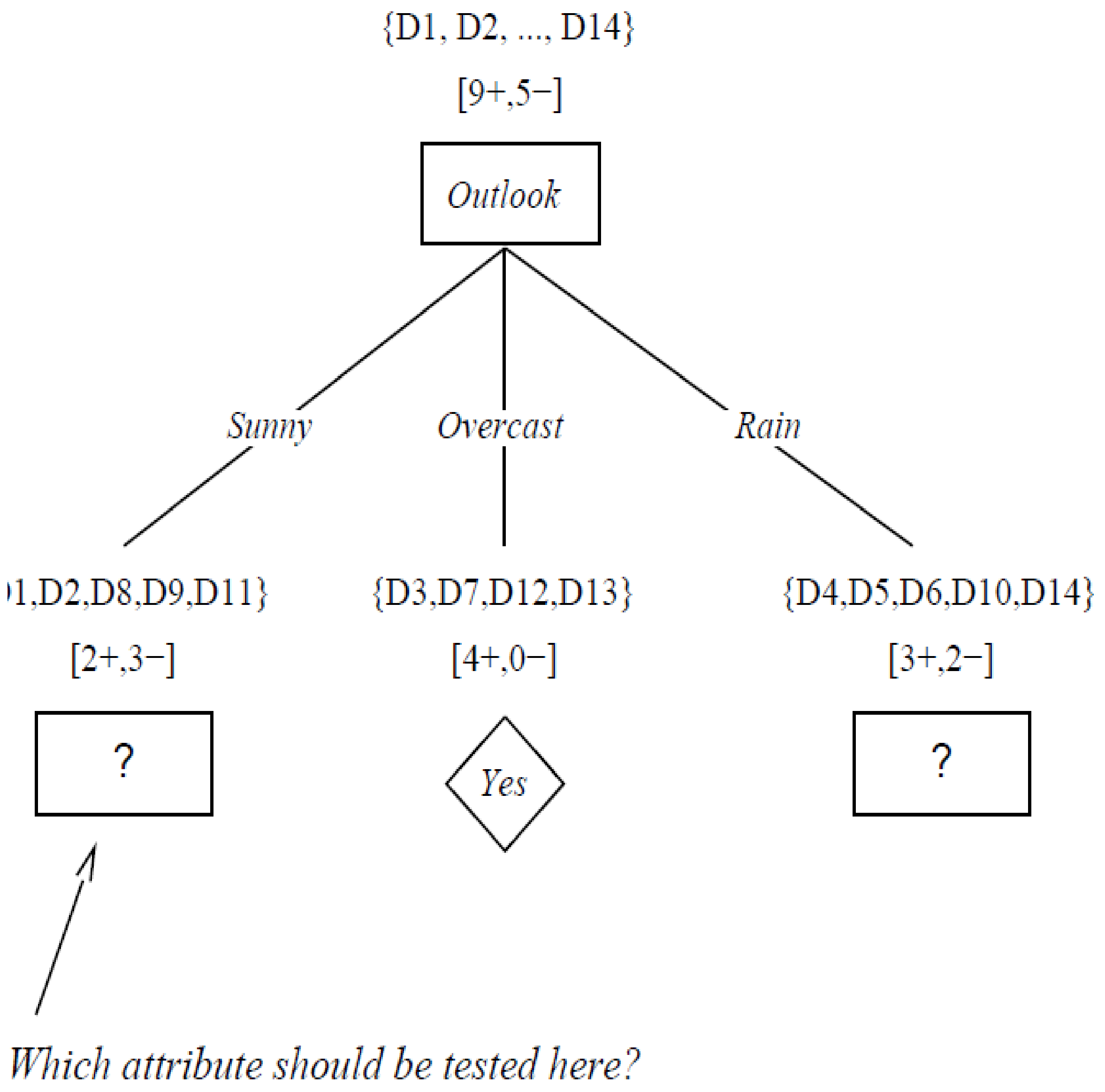
  $Gain(S_{Sunny}, Humidity) = 0.970 - 3/5 \times 0.0 - 2/5 \times 0.0 = \mathbf{0.970}$

  $Gain(S_{Sunny}, Wind) = 0.970 - 2/5 \times 1.0 - 3.5 \times 0.918 = \mathbf{0\ .019}$

  $Gain(S_{Sunny}, Temp.) = 0.970 - 2/5 \times 0.0 - 2/5 \times 1.0 - 1/5 \times 0.0 = \mathbf{0.570}$

- *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*

Second and third steps



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

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

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

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

ID3: algorithm

```
ID3(X, T, Attrs)        X: training examples:
                        T: target attribute (e.g. PlayTennis),
                        Attrs: other attributes, initially all attributes
Create Root node
If all X's are +, return Root with class +
If all X's are –, return Root with class –
If Attrs is empty return Root with class most common value of T in X
else
    A ← best attribute; decision attribute for Root ← A
    For each possible value $v_i$ of A:
        - add a new branch below Root, for test A = $v_i$
        - $X_i$ ← subset of X with A = $v_i$
        - If $X_i$ is empty then add a new leaf with class the most common value of T in X
          else add the subtree generated by ID3($X_i$, T, Attrs – {A})
    return Root
```

Advantages and Disadvantages

Advantages :

- **Computationally Inexpensive**
- **Handles both numerical and categorical attributes**
- **Outputs are easy to interpret**
- **Works well with both linear and nonlinear data**
- **Sensitive to small variations in the training data**
- **Robust with redundant and correlated data**

Disadvantages :

- **Overfitting**
- **Too many Layers**
- **Lack of training Data**
- **Biased Data in training set**
- **Multicollinearity among variables**

6.Hypothesis Space Search in Decision Tree Learning

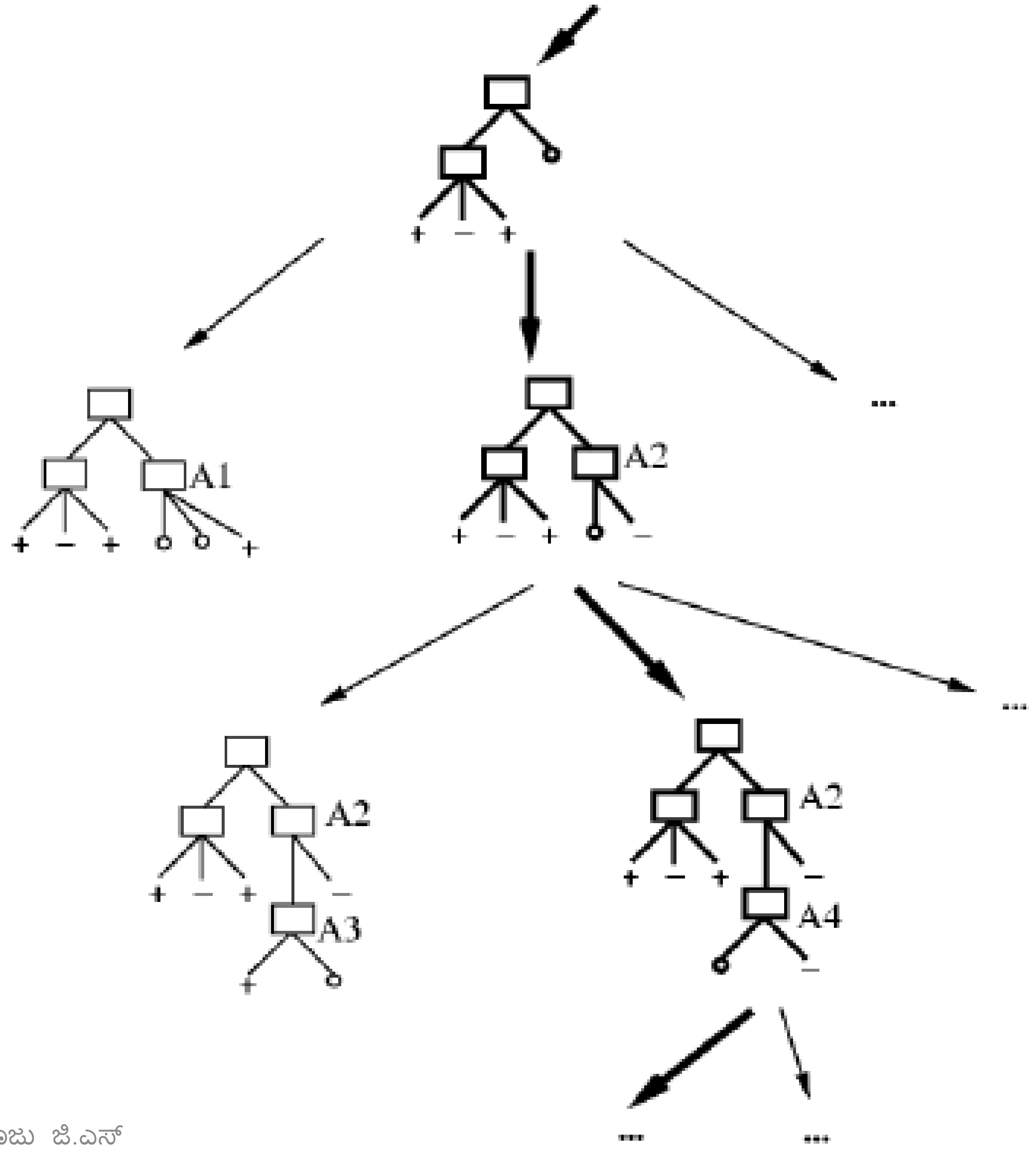
- **Hypothesis Space**: Set of possible decision trees (i.e., complete space of finite discrete-valued functions).
- **Search Method**: Simple-to-Complex *Hill-Climbing* Search (only a single current hypothesis is maintained (≠ from candidate-elimination method)). **No Backtracking!!!**
- **Evaluation Function**: Information Gain Measure
- **Batch Learning:** ID3 uses all training examples at each step to make statistically-based decisions (≠ from candidate-elimination method which makes decisions incrementally). ==> the search is less sensitive to errors in individual training examples.

Some Insight into ID3 capabilities and limitations

- **The hypotheses space is complete (represents all discrete-valued functions)**
- **The search maintains a single current hypothesis**
- **The search space is made by partial decision trees**
- **No backtracking; no guarantee of optimality**
- **It uses all the available examples (not incremental)**
  - **The algorithm is *hill-climbing***
  - **The evaluation function is *information gain***
  - **May terminate earlier, accepting noisy classes**

Which Tree Should We Output?

- **ID3 performs heuristic search through space of decision trees**
- **It stops at smallest acceptable tree. Why?**



7. Inductive bias in decision tree learning

- What is the inductive bias of DT learning? : Generalizing from observed training examples to classify unseen instances :
- *ID3 search strategy*
  1. *Approximate inductive bias of ID3 : Shorter trees are preferred over longer trees*

     Not enough. This is the bias exhibited by a simple breadth first algorithm generating all DT's e selecting the shorter one
  2. *A Closer approximation to the inductive bias of ID3 : Shorter trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred over those do not.*
- *Note*: DT's are not limited in representing all possible function.

Two kinds of biases

Sl.No	INDUCTIVE BIAS EXHIBITED BY ID3	INDUCTIVE BIAS EXHIBITED BY CANDIDATE ELIMINATION ALGORITHM
1	ID3 searches a complete hypothesis space	It searches an incomplete hypothesis space
2	It searches incompletely through this space from simple to complete hypothesis until its termination condition is met.	It searches this space completely finding every hypothesis consistent with the training data
3	Its inductive bias is solely a consequences of the ordering of hypothesis by its search strategy.	Its inductive bias is solely a consequence of expressive power of its hypothesis representation.
4	Its hypothesis space introduces no additional bias	Its search strategy introduces no additional bias
5	Follows from its search strategy	Follows from the definition of its search space.
6	It is a preference for certain hypothesis over others (Preference bias)/ search bias	Restriction bias (or alternatively a language bias)

Why Prefer Short Hypotheses

- **Occam's razor: Prefer the simplest hypothesis that fits the data** [William of Occam (Philosopher), circa 1320]
- Scientists seem to do that: E.g., Physicist seem to prefer simple explanations for the motion of planets, over more complex ones
- **Argument:** Since there are fewer short hypotheses than long ones, it is less likely that one will find a short hypothesis that coincidentally fits the training data.
- **Problem with this argument:** it can be made about many other constraints. Why is the "short description" constraint more relevant than others?
- **Nevertheless:** Occam's razor was shown experimentally to be a successful strategy!

8.Issues in Decision Tree Learning:

Practical issues in learning DT include determining how deeply to grow the DT , handling continuous attributes , choosing and appropriate attribute selection measure , handling training data with missing attribute values , handling attributes with different costs and improving computational efficiency.

1. Avoiding overfitting the data
2. Incorporating Continuous-Valued Attributes
3. Alternative Measures for Selecting Attributes
4. Handling Training Examples with Missing Attribute Values
5. Handling Attributes with Differing Costs.

8.1 Avoiding Overfitting the Data :

- 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

$$error_D(h) < error_D(h') \quad \text{and}$$

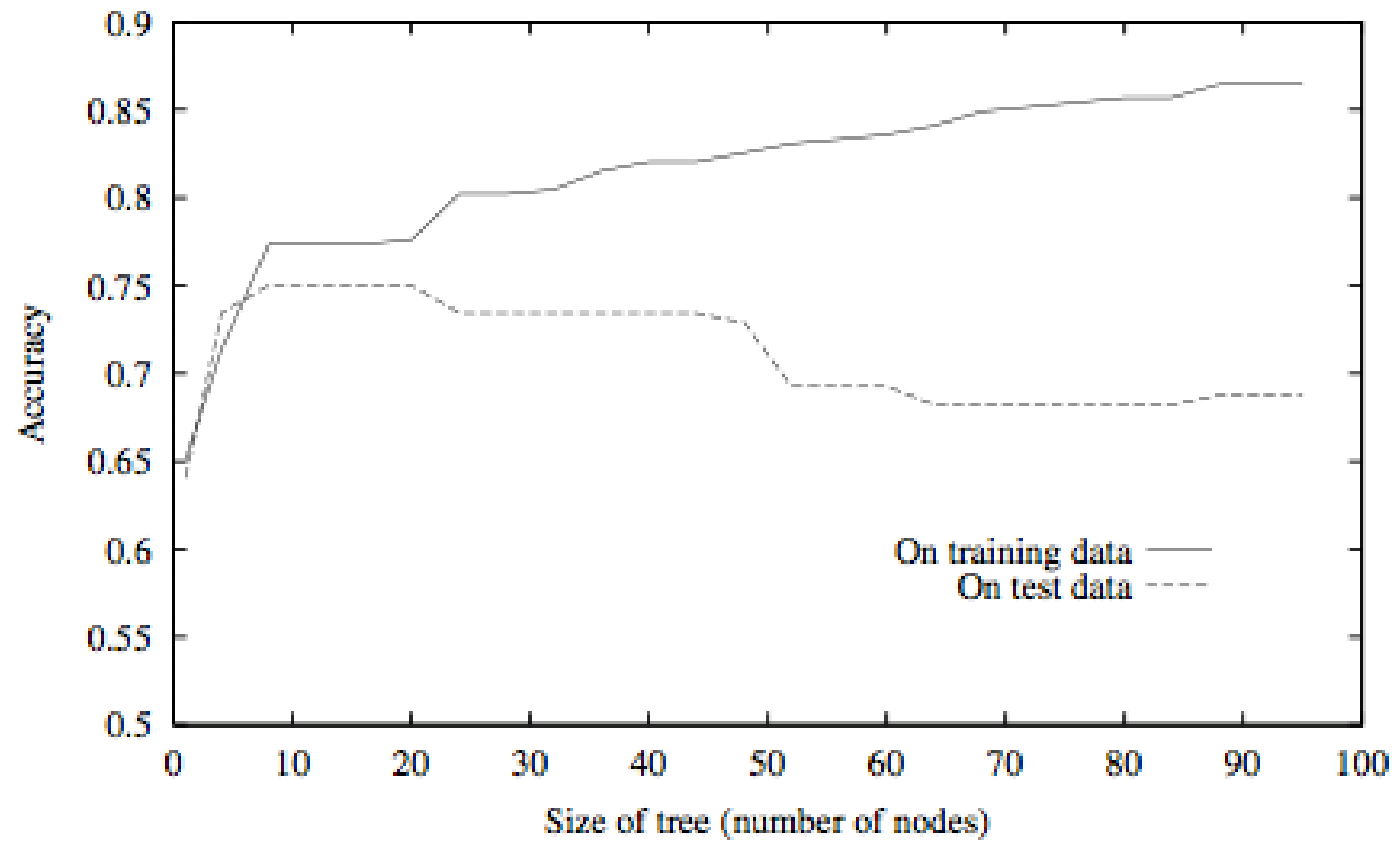
$$error_X(h') < error_X(h)$$

i.e. h' behaves better over unseen data

8.1 Avoiding Overfitting the Data :

- **Definition:** Given a hypothesis space *H*, a hypothesis $h \in H$ is said to ***overfit*** the training data if there exists some alternative hypothesis $h' \in H$, such that *h* has smaller error than *h′* over the training examples, but *h′* has a smaller error than *h* over the entire distribution of instances. (See curves in [Mitchell, p.67])
- Figure below illustrates the impact of overfitting in a typical application of decision tree learning in a typical application of decision tree learning for medical patients have a form of diabetes

Overfitting in decision tree learning



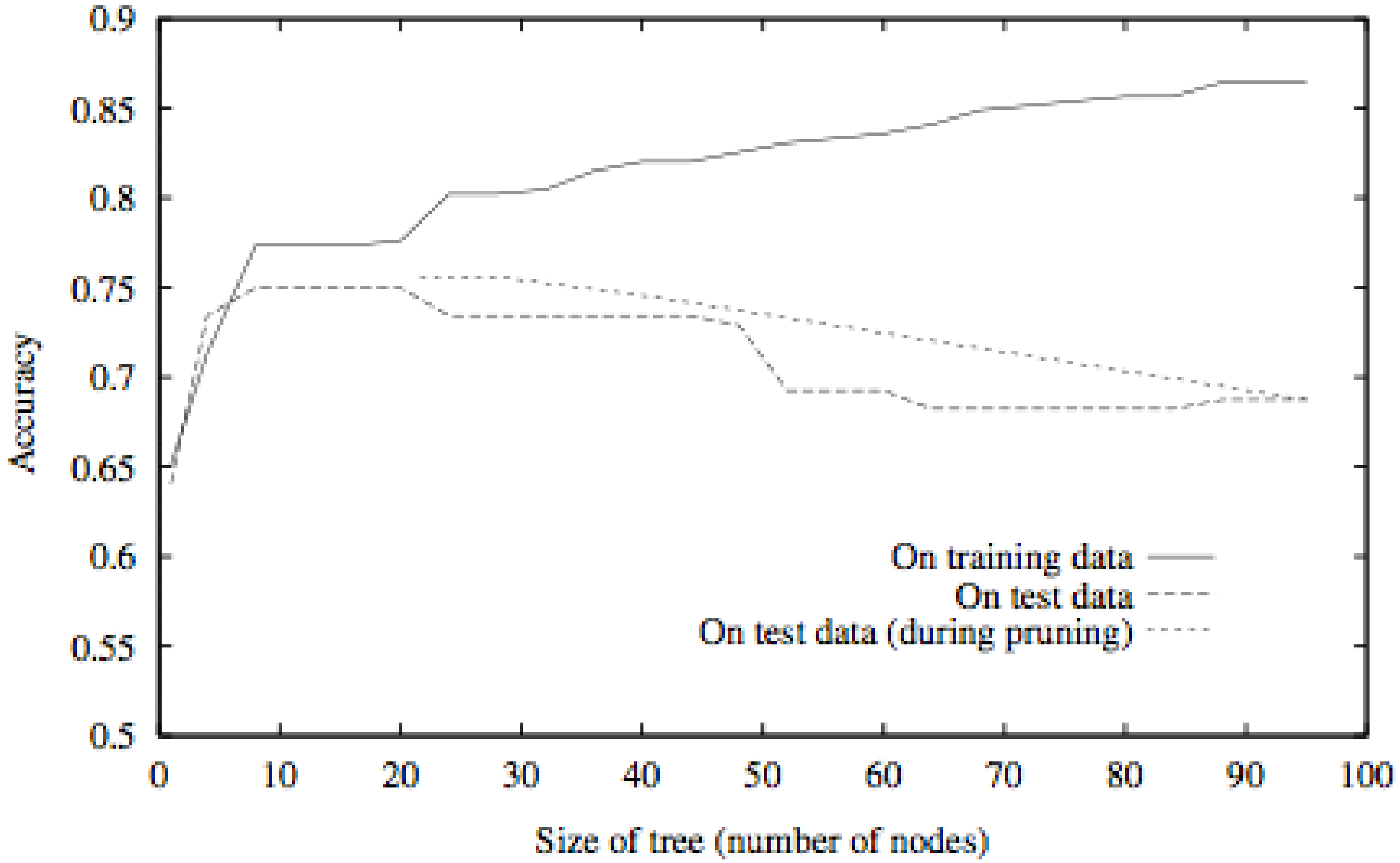
Avoid overfitting in Decision Trees

- Two strategies:
  1. Stop growing the tree earlier, before perfect classification
  2. 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 (Quinlan 1987)

- 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

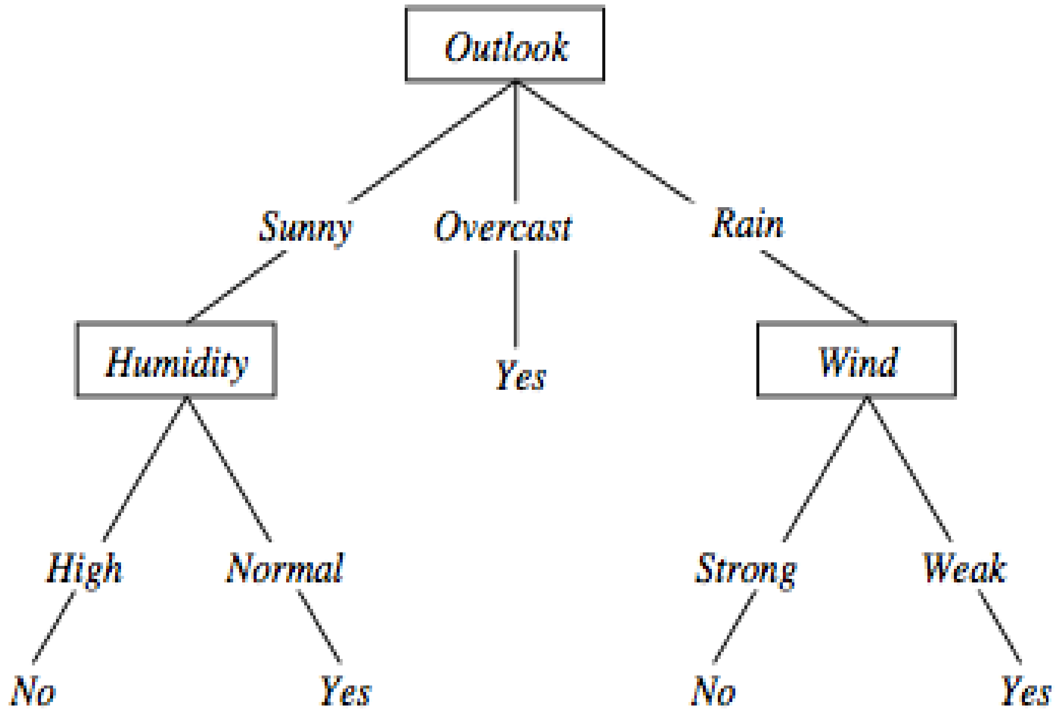
Effect of reduced error pruning



Rule post-pruning

1. Create the decision tree from the training set
2. 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
3. Prune (generalize) each rule by removing those preconditions whose removal improves accuracy …
    - … over validation set
    - … over training with a pessimistic, statistically inspired, measure
4. Sort the rules in estimated order of accuracy, and consider them in sequence when classifying new instances

Converting to rules



Example : The leftmost path of the tree in Figure is translated into the rule

IF (*Outlook=Sunny*)$\wedge$(*Humidity=High*)
THEN *PlayTennis=No*

Why converting to rules?

- It allows distinguishing among the different contexts in which a decision node is used.
- Converting to rules removes the distinction between attribute tests that occur near the root of the tree and those that occurs near the leaves.
- Converting to rules improves readability for humans. Rules are often easier for people to understand.

2. Incorporating 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

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

- 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 $\mathit{Temperature}_{>54}$.The new attribute competes with the other ones

Problems with *information gain*

- **Natural bias of information gain**: it favors 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 …

3. An alternative measure for selecting attribute: *Gain ratio*

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

- 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.

$$\boldsymbol{GainRatio(S, A)} \equiv \frac{\boldsymbol{Gain(S, A)}}{\boldsymbol{SplitInformation(S, A)}}$$

- *GainRatio* penalizes attributes that split examples in many small classes such as *Date.* Let $|S|=n$, *Date* splits examples in n classes
  - $\boldsymbol{SplitInformation(S, Date) = -[(1/n \log_2 1/n)+\ldots+(1/n \log_2 1/n)] = -\log_2 1/n = \log_2 n}$
- Compare with A, which splits data in two even classes:
  - $\boldsymbol{SplitInformation(S, A) = -[(1/2 \log_2 1/2)+(1/2 \log_2 1/2)] = -[-1/2 - 1/2] = 1}$

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:
  1. compute $Gain$ for each attribute
  2. apply $GainRatio$ only to attributes with $Gain$ above average
- 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

4. Handling Training Examples with Missing complete 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
  1. Assign the value that is most common among the training examples at the node
  2. 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)

5. Handling attributes with different costs

- 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:

1. Tan and Schlimmer (1990)

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

2. Nunez (1988)

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

How well does it work?

Many case studies have shown that decision trees are at least as accurate as human experts.

– A study for diagnosing breast cancer had humans correctly classifying the examples 65% of the time; the decision tree classified 72% correct

– British Petroleum designed a decision tree for gas-oil separation for offshore oil platforms that replaced an earlier rule-based expert system

– Cessna designed an airplane flight controller using 90,000 examples and 20 attributes per example

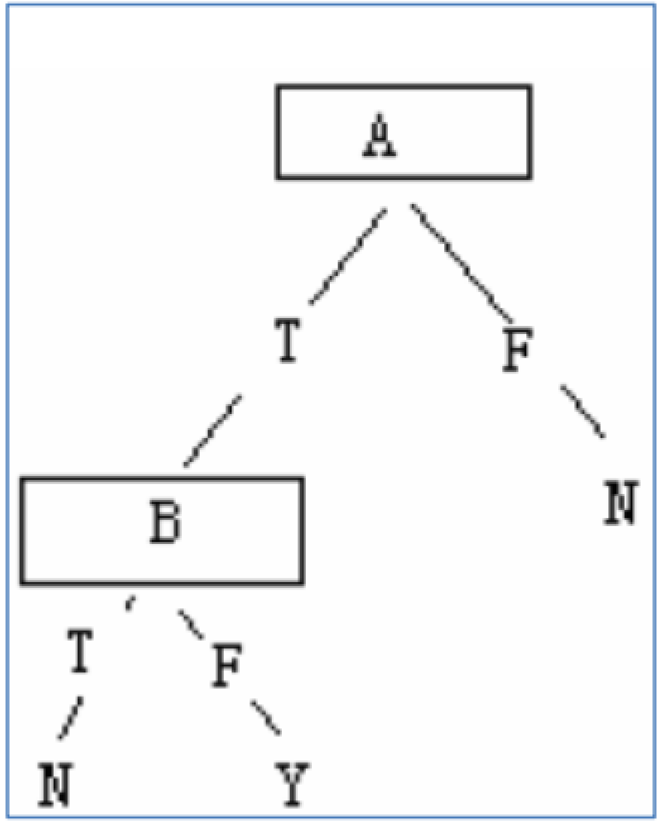
Extensions of ID3

- Using gain ratios
- Real-valued data
- Noisy data and overfitting
- Generation of rules
- Setting parameters
- Cross-validation for experimental validation of performance
- C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, and so on

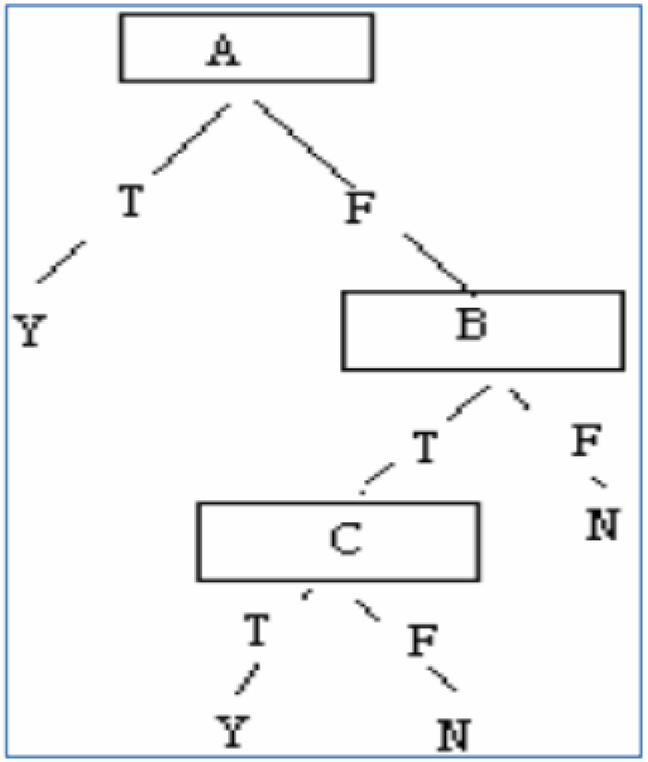
Example : Give decision trees to represent the following Boolean functions:

1) $A \wedge \neg B$
2) $A \vee [B \wedge C]$
3) A XOR B
4) $[A \wedge B] \vee [C \wedge D]$

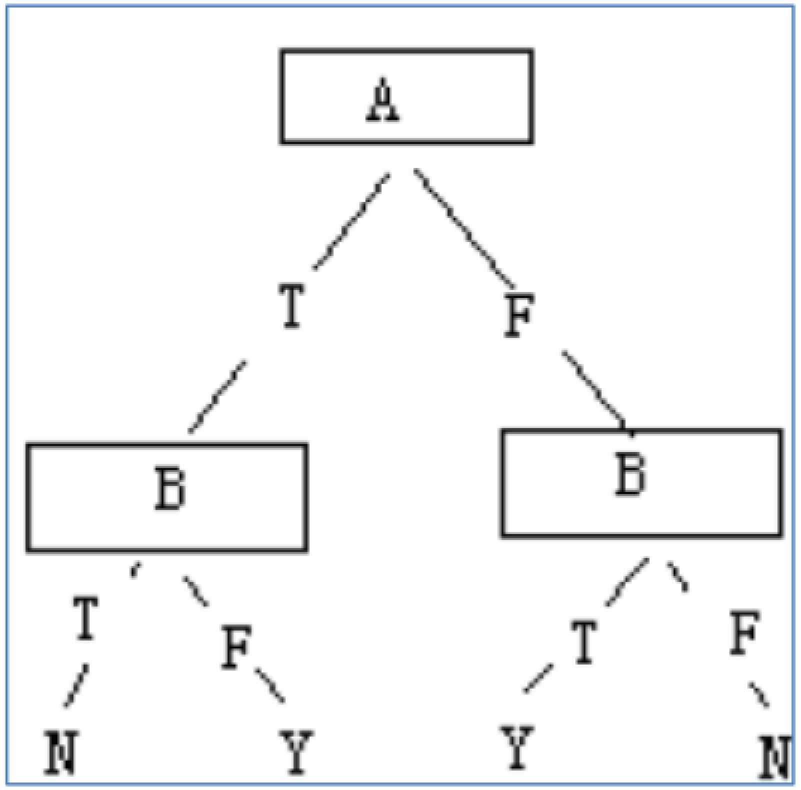
1) $A \wedge \neg B$



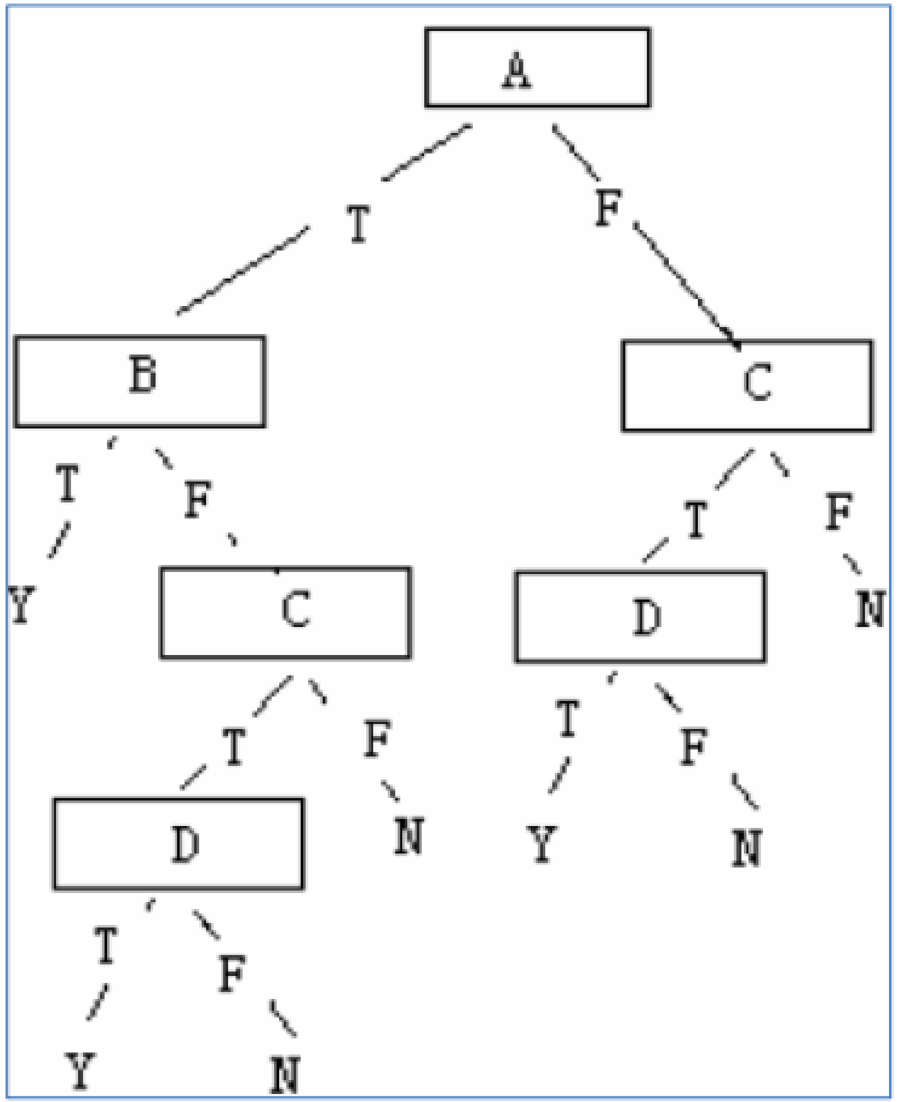
2) $A \vee [B \wedge C]$



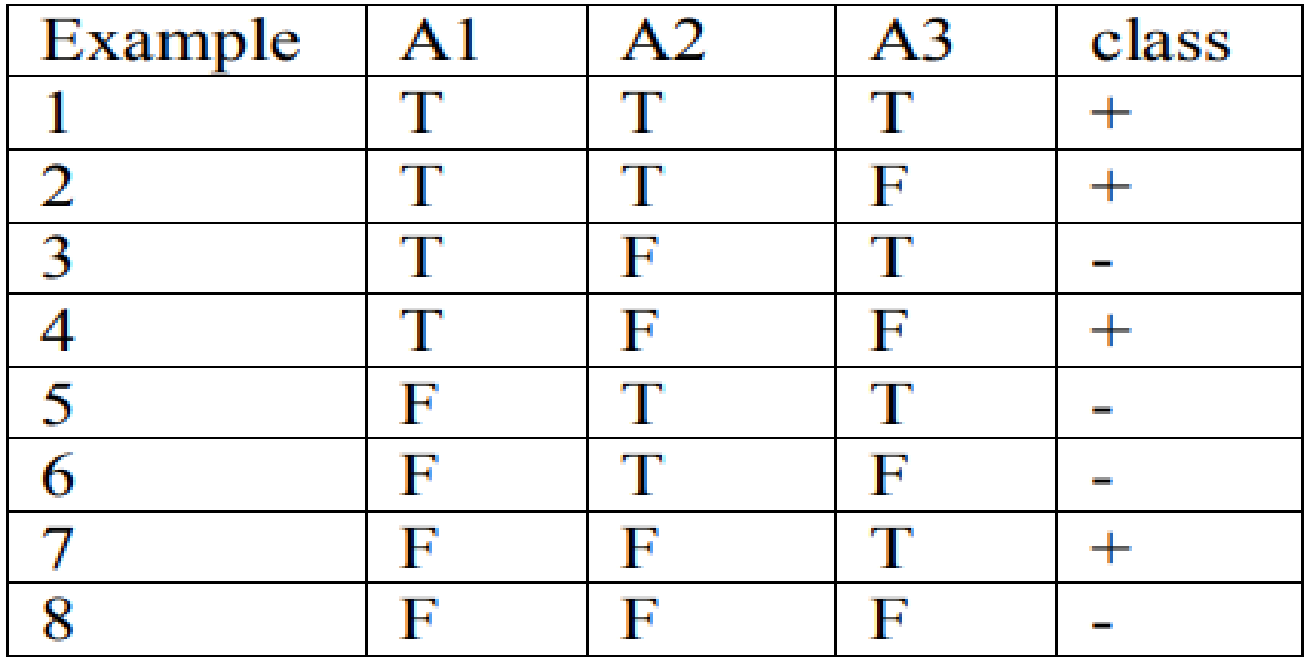
3) A XOR B = $(A \wedge \neg B) \vee (\neg A \wedge B)$



4) $[A \wedge B] \vee [C \wedge D]$



2. Consider the following set of training examples:



Example	A1	A2	A3	class
1	T	T	T	+
2	T	T	F	+
3	T	F	T	-
4	T	F	F	+
5	F	T	T	-
6	F	T	F	-
7	F	F	T	+
8	F	F	F	-

A) What is the entropy of this collection of training examples with respect to the target function classification?

- The entropy of this collection of training examples with respect to the target function classification is E(S) = 1
- because it contains equal numbers of positive and negative examples.

B) What is the information gain of feature A2 relative to these training examples?

- The information gain of feature A2 relative to these training examples $G(S,A2) = E(S) - \sum|S_{A2}| / |S| * E(S) = 1 – (4/8 * 1 + 4/8 * 1) = 0$

C) What is the best feature relative to these training examples, using Gain Ratio?

- The best feature relative to these training examples is the feature with the maximum information gain, and in this example any feature can selected because the gain of all features are the same.

End of Module2