Decision Tree Learning

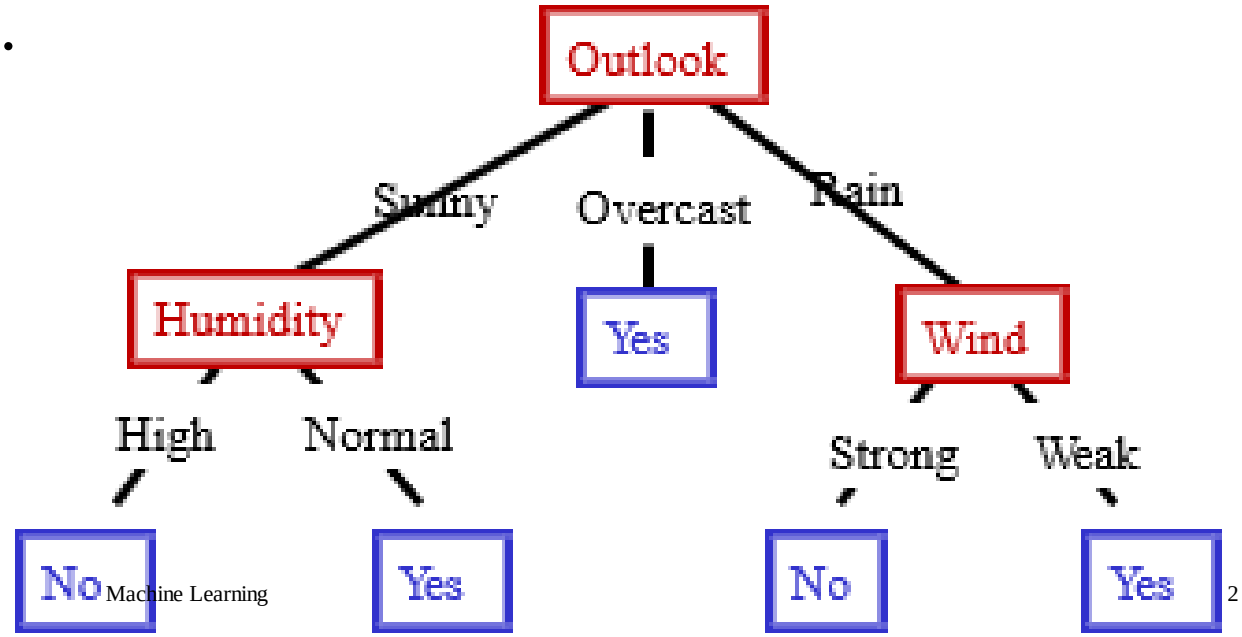
- **Decision tree learning** is a method for approximating discrete-valued target functions.
- The learned function is represented by **a decision tree**.
  - A learned decision tree can also be re-represented as a set of if-then rules.
- Decision tree learning is one of the most widely used and practical methods for inductive inference.
- It is robust to noisy data and capable of learning disjunctive expressions.
- Decision tree learning method searches a completely expressive hypothesis .
  - Avoids the difficulties of restricted hypothesis spaces.
  - Its inductive bias is a preference for small trees over large trees.
- The decision tree algorithms such as ID3, C4.5 are very popular inductive inference algorithms, and they are sucessfully applied to many leaning tasks.

Decision Tree

- Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance.
- Each node in the tree specifies a test of some attribute of the instance.
- Each branch descending from a node corresponds to one of the possible values for the attribute.
- Each leaf node assigns a classification.
- The instance

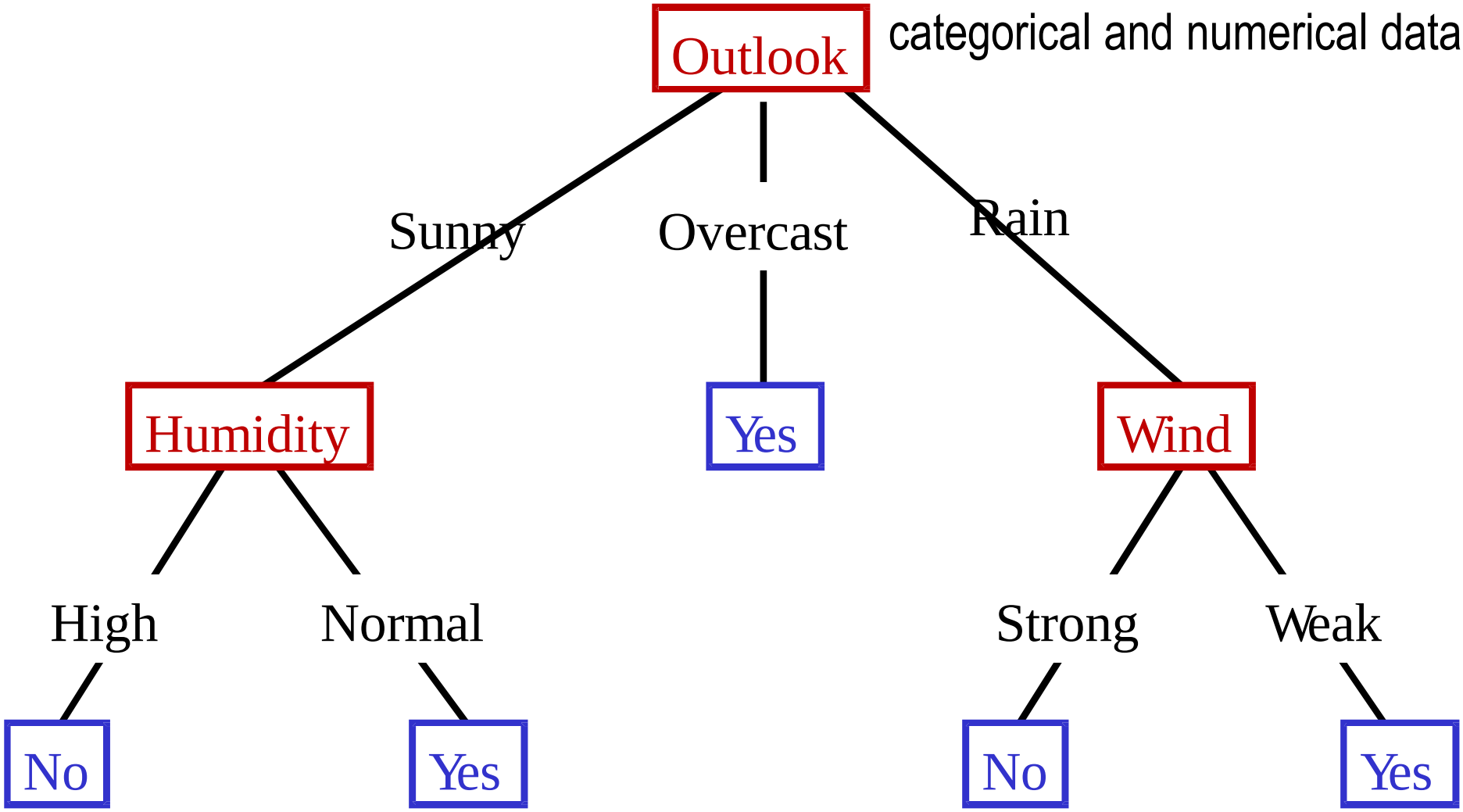
(Outlook=Sunny, Temperature=Hot, Humidity=High, Wind=Strong) is classified as a negative instance.

Decision tree builds classification or regression models in the form of a tree structure.



Decision Tree for PlayTennis

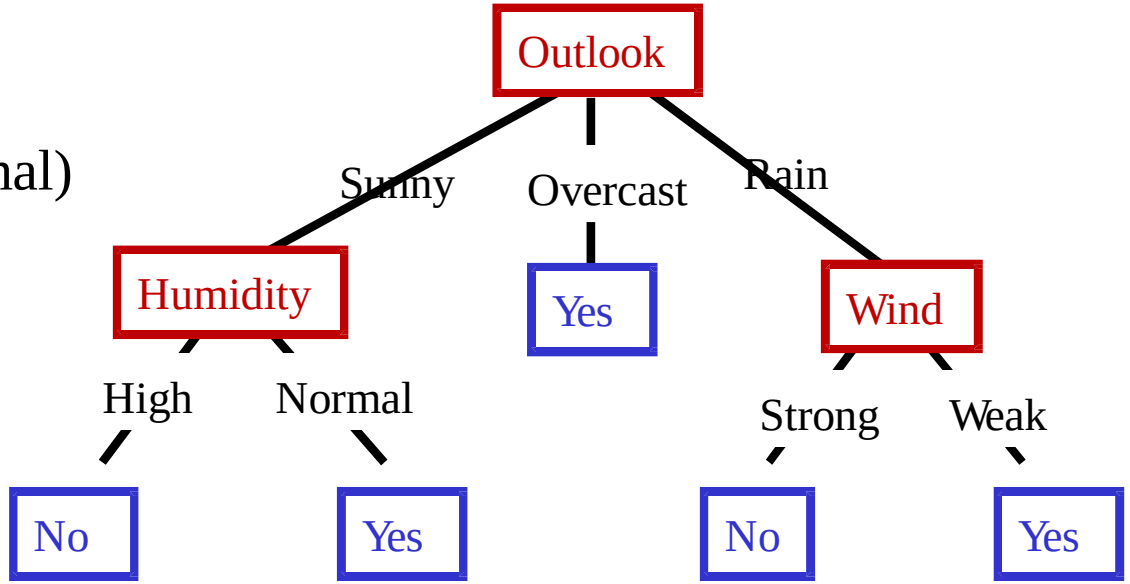
- The topmost decision node in a tree corresponds to the best predictor called root node.
- Decision trees can handle both categorical and numerical data.



Decision Tree

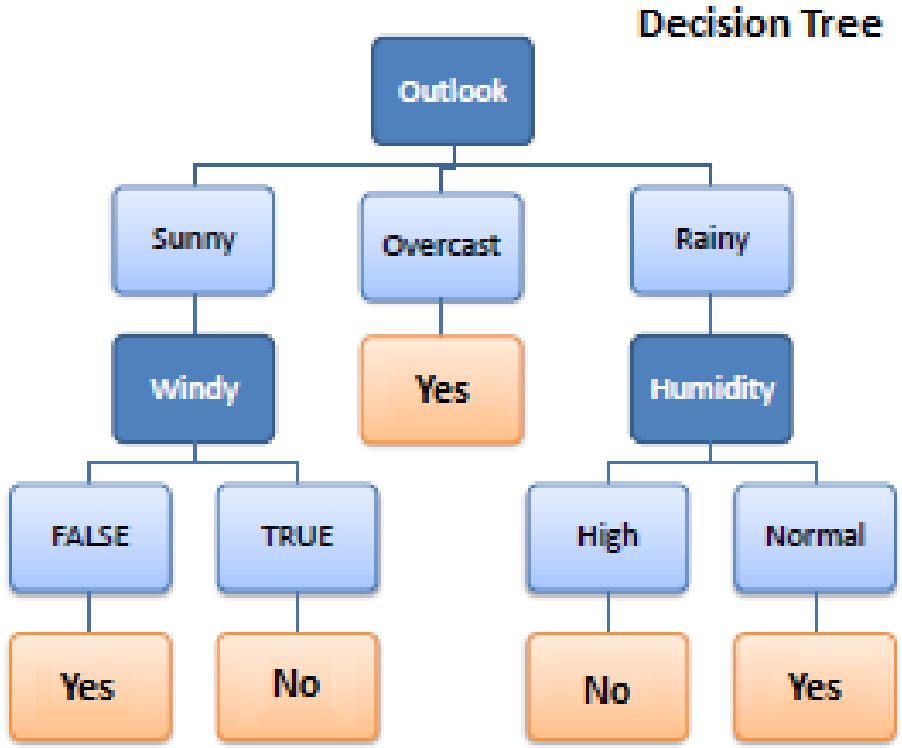
- Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.
- Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and
- The tree itself is a disjunction of these conjunctions.

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

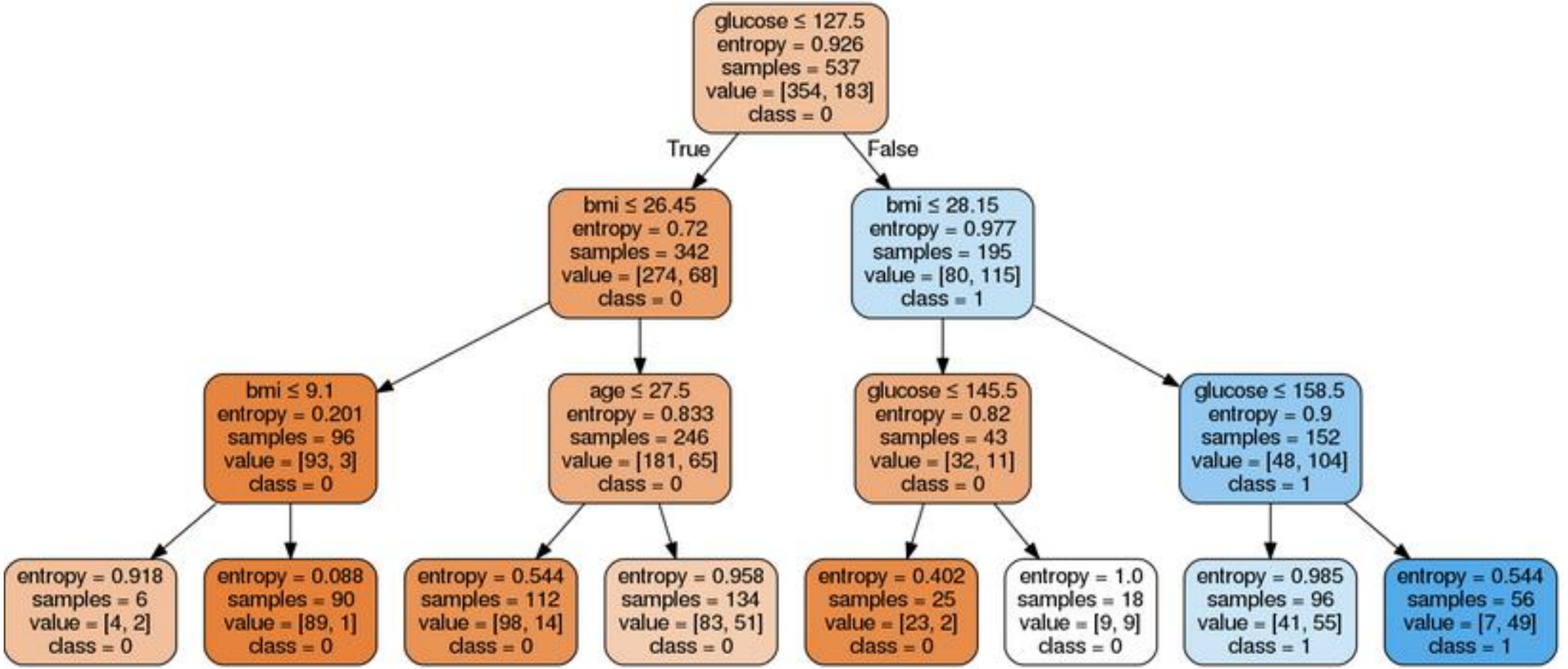


Predictors | Target

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No



Example: Decision Tree



When to Consider Decision Trees

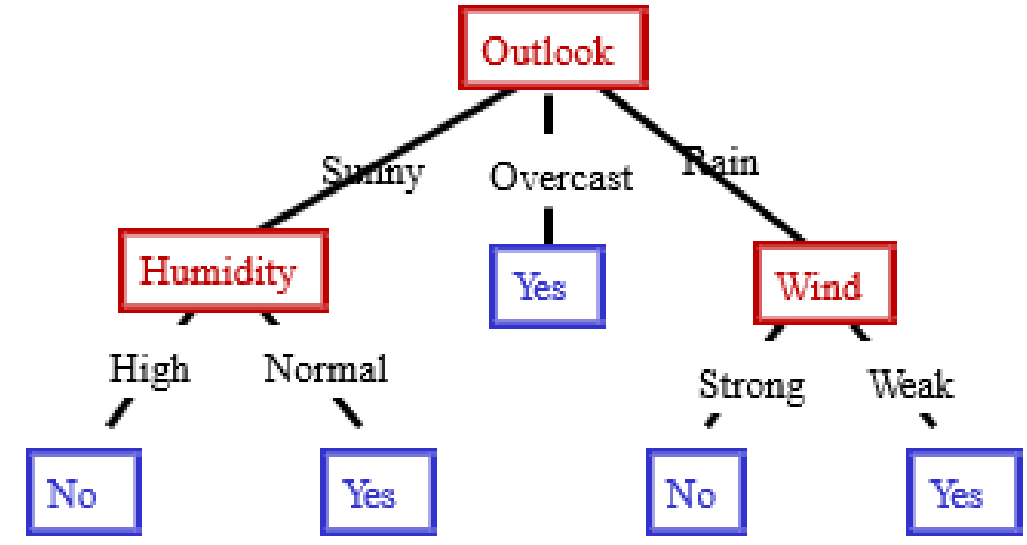
- Instances are represented by attribute-value pairs.
  - Fixed set of attributes, and the attributes take a small number of disjoint possible values.
- The target function has discrete output values.
  - Decision tree learning is appropriate for a boolean classification, but it easily extends to learning functions with more than two possible output values.
- Disjunctive descriptions may be required.
  - decision trees naturally represent disjunctive expressions.
- The training data may contain errors.
  - Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples.
- The training data may contain missing attribute values.
  - Decision tree methods can be used even when some training examples have unknown values.
- Decision tree learning has been applied to problems such as learning to classify
  - medical patients by their disease,
  - equipment malfunctions by their cause, and
  - loan applicants by their likelihood of defaulting on payments.

Top-Down Induction of Decision Trees -- ID3

The core algorithm for building decision trees called **ID3** by J. R. Quinlan.
ID3 uses Entropy and Information Gain to construct a decision tree.

1. A ← the **"best"** decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A create new descendant *node*
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”?

- We would like to select the attribute that is most useful for classifying examples.
- ***Informution 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.
- In order to define information gain precisely, we use a measure commonly used in information theory, called ***entropy***
- ***Entropy*** characterizes the (im)purity of an arbitrary collection of examples.

Entropy

- Given a collection S, containing positive and negative examples of some target concept, the ***entropy of S*** relative to this boolean classification is:

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

- **S** is a sample of training examples
- $\mathbf{p_+}$ is the proportion of positive examples
- $\mathbf{p_-}$ is the proportion of negative examples

Entropy

Entropy([9+,5-] = $-(9/14)\log_2(9/14) - (5/14)\log_2(5/14)$ = 0.940

Entropy([12+,4-] = $-(12/16)\log_2(12/16) - (4/16)\log_2(4/16)$ = 0.811

Entropy([12+,5-] = $-(12/17)\log_2(12/17) - (5/17)\log_2(5/17)$ = 0.874

Entropy([8+,8-] = $-(8/16)\log_2(8/16) - (8/16)\log_2(8/16)$ = 1.0

Entropy([8+,0-] = $-(8/8)\log_2(8/8) - (0/8)\log_2(0/8)$ = 0.0

Entropy([0+,8-] = $-(0/8)\log_2(0/8) - (8/8)\log_2(8/8)$ = 0.0

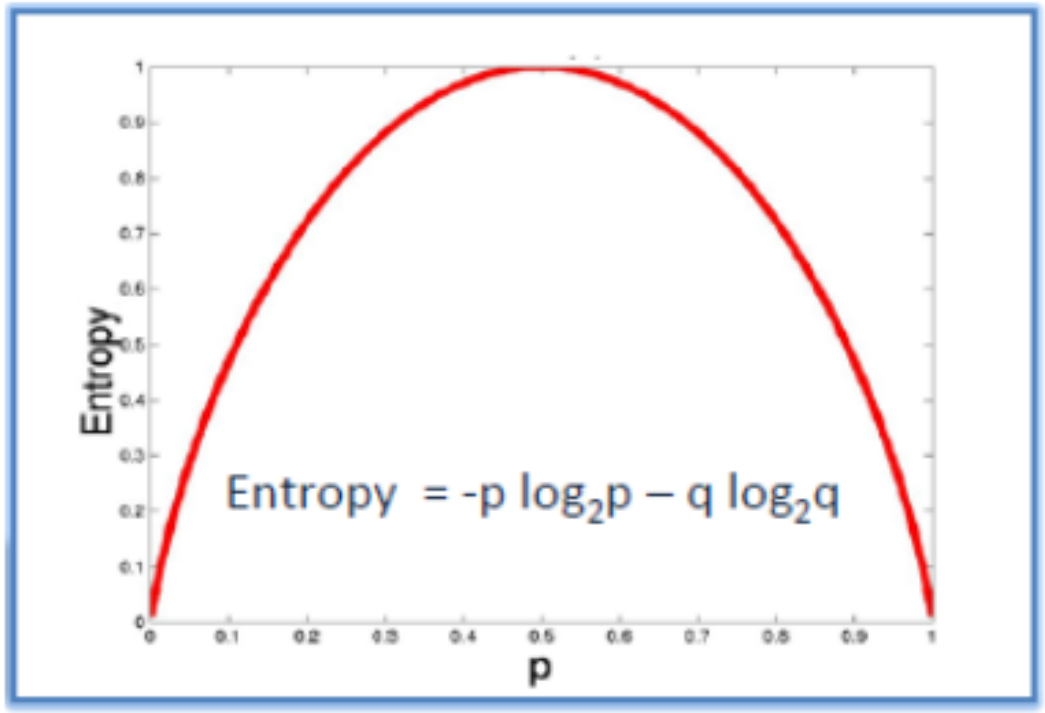
- It is assumed that $\log_2(0)$ is 0

Entropy

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous).

ID3 algorithm uses entropy to calculate the homogeneity of a sample.

If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.



Entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

Entropy – Non-Boolean Target Classification

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

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

- p_i is the proportion of S belonging to class i.
- The logarithm is still base 2 because entropy is a measure of the expected encoding length measured in bits.
- If the target attribute can take on c possible values, the entropy can be as large as $\log_2 c$.

Information Gain

- ***Entropy*** is a measure of the impurity in a collection of training examples
- ***Information Gain*** is a measure of the effectiveness of an attribute in classifying the training data.
- ***Information Gain*** measures the expected reduction in entropy by partitioning the examples according to an attribute.

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

- S – a collection of examples
- A – an attribute
- Values(A) – possible values of attribute A
- S_v – the subset of S for which attribute A has value v

Information Gain

$$Values(Wind) = Weak, Strong$$

$$S = [9+, 5-]$$

$$S_{Weak} \leftarrow [6+, 2-]$$

$$S_{Strong} \leftarrow [3+, 3-]$$

$$Gain(S, Wind) = Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= Entropy(S) - (8/14) Entropy(S_{Weak}) - (6/14) Entropy(S_{Strong})$$

$$= 0.940 - (8/14) 0.811 - (6/14) 1.00$$

$$= 0.048$$

Decision Tree Algorithm-ID3

- To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:
- a) Entropy using the frequency table of one attribute:

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

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

Decision Tree Algorithm-ID3

- Entropy using the frequency table of two attributes:

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

Decision Tree Algorithm-ID3

Information Gain

- The information gain is based on the decrease in entropy after a dataset is split on an attribute.
- Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

- Step 1: Calculate entropy of the target.

$$\begin{aligned}\textbf{Entropy(PlayGolf)} &= \text{Entropy}\ (5,9)\\ &= \text{Entropy}\ (0.36, 0.64)\\ &= -\ (0.36 \log_2 0.36) - (0.64 \log_2 0.64)\\ &= 0.94\end{aligned}$$

Decision Tree Algorithm-ID3

- Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

Decision Tree Algorithm- ID3

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)*__E__(3,2) + **P**(Overcast)*__E__(4,0) + **P**(Rainy)*__E__(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

$$Gain(T,X) = Entropy(T) - Entropy(T,X)$$

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

Decision Tree Algorithm-ID3

- *Step 3*: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

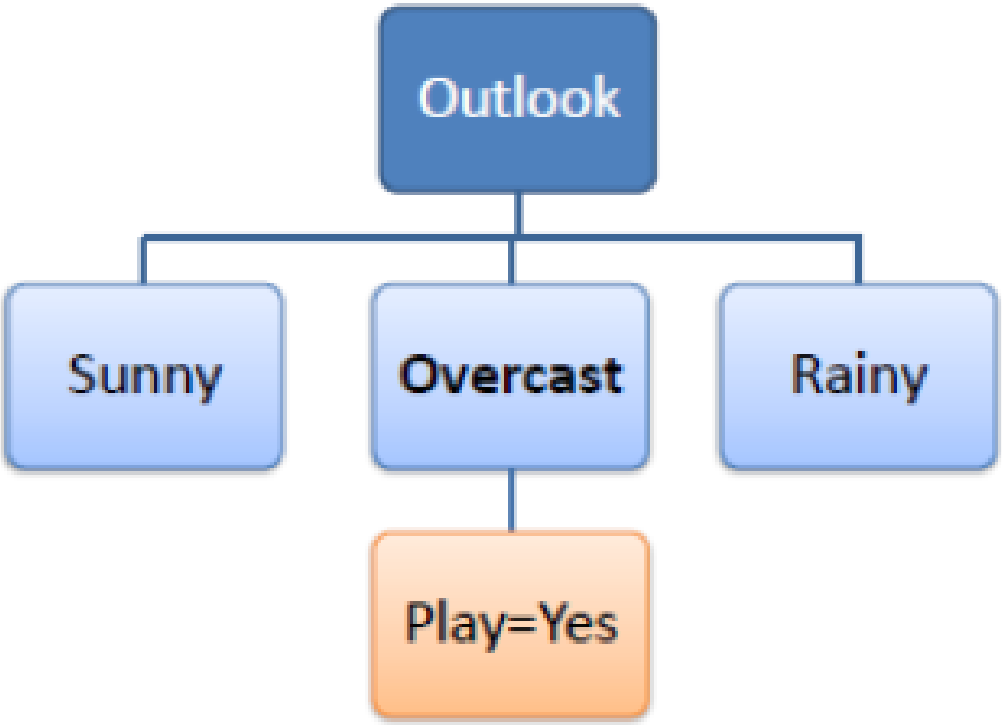
★		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Decision Tree Algorithm-ID3

- *Step 4a*: A branch with entropy of 0 is a leaf node.

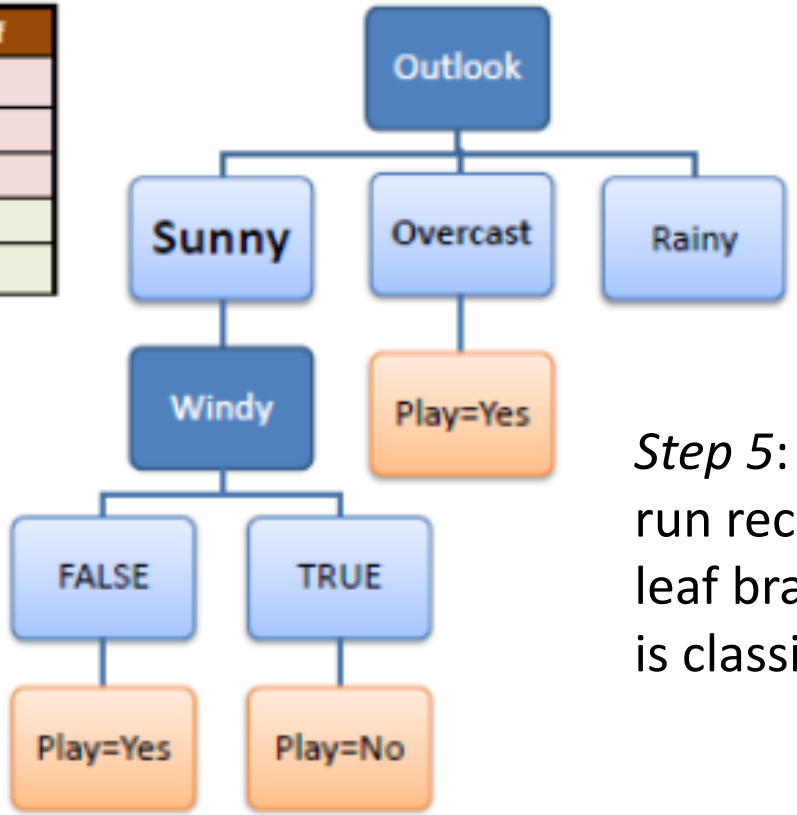
Temp.	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



Decision Tree Algorithm-ID3

- *Step 4b*: A branch with entropy more than 0 needs further splitting.

Temp.	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

Decision Tree Algorithm-ID3

- Decision Tree to Decision Rules
- A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

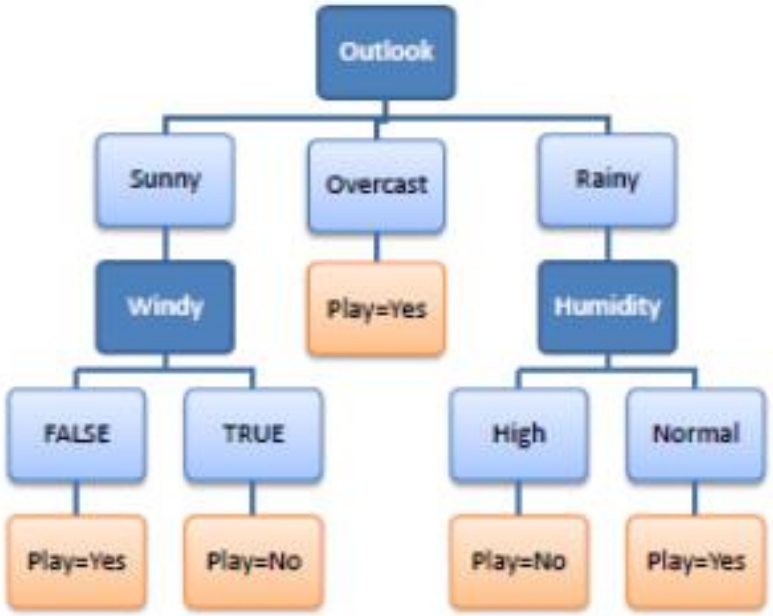
R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

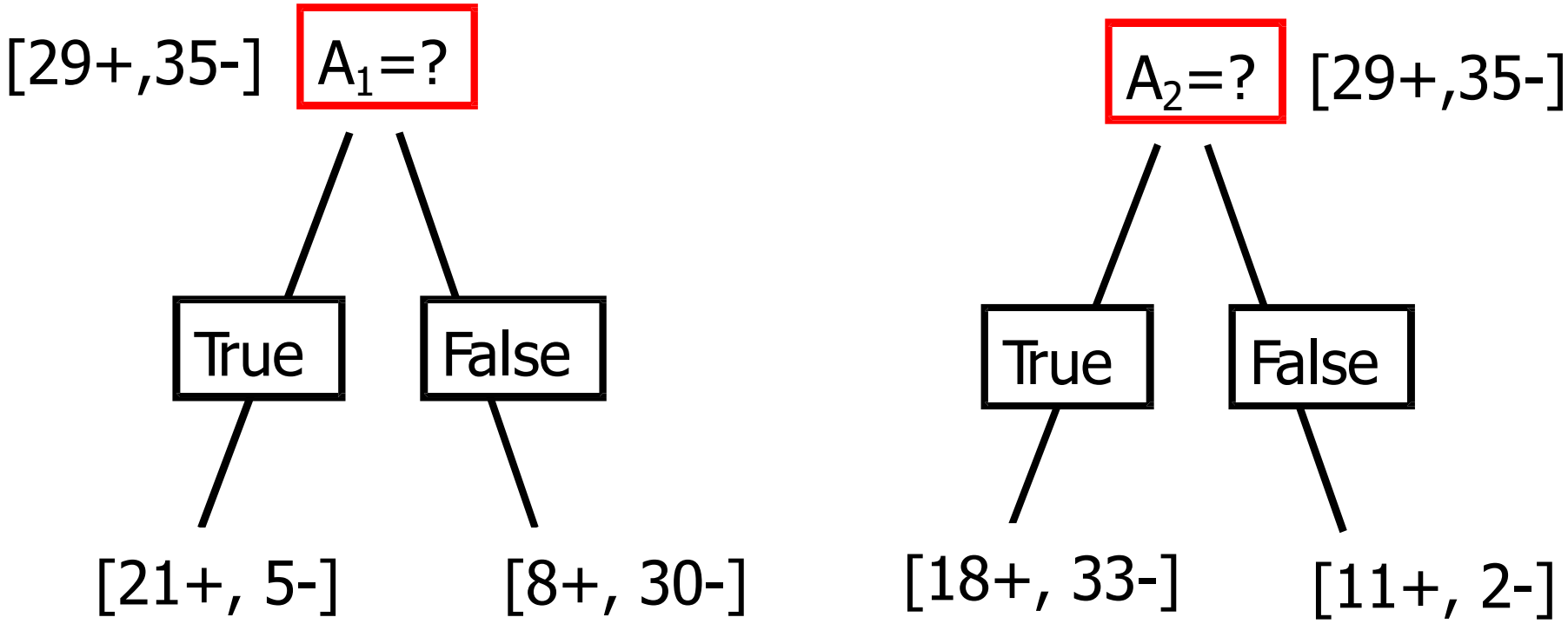
R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes

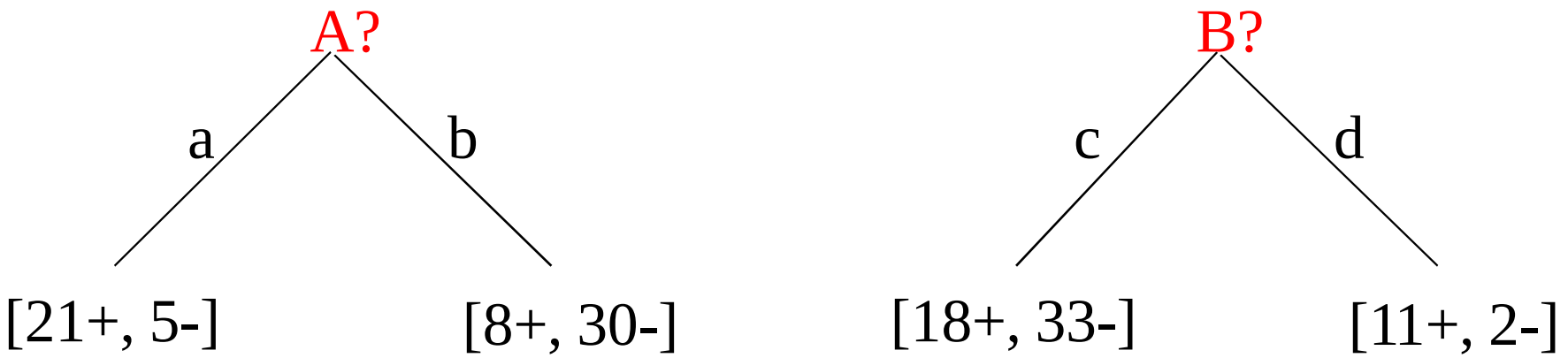


Which Attribute is ”best”?

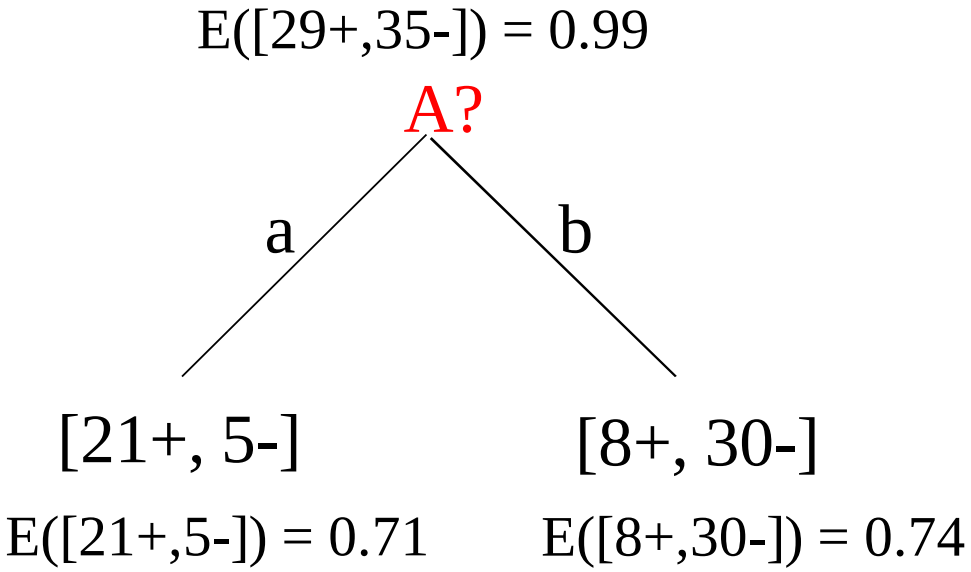


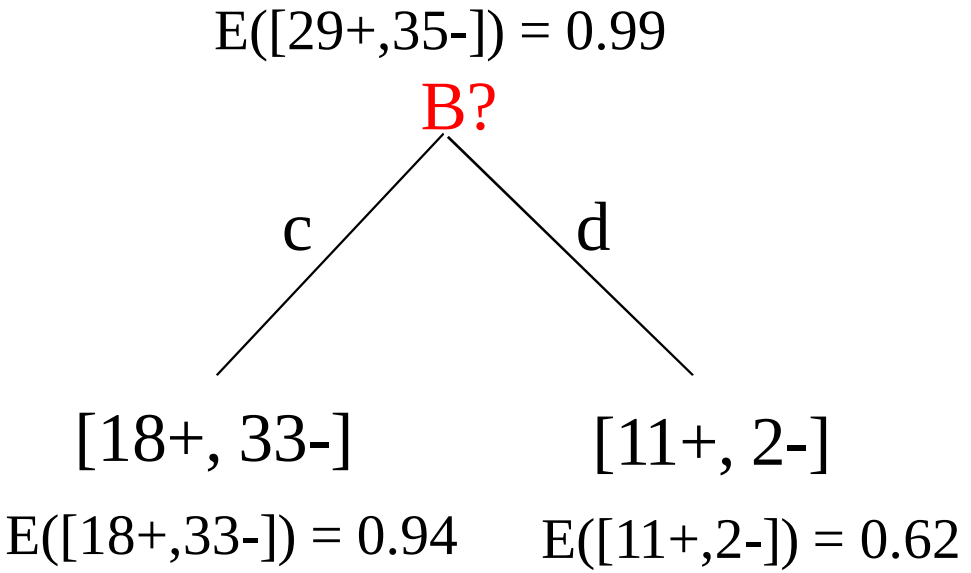
Which attribute is the best classifier?

- S: [29+,35-] Attributes: A and B
- possible values for A: a,b possible values for B: c,d
- Entropy([29+,35-]) = $-29/64 \log_2 29/64 - 35/64 \log_2 35/64 = 0.99$



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**

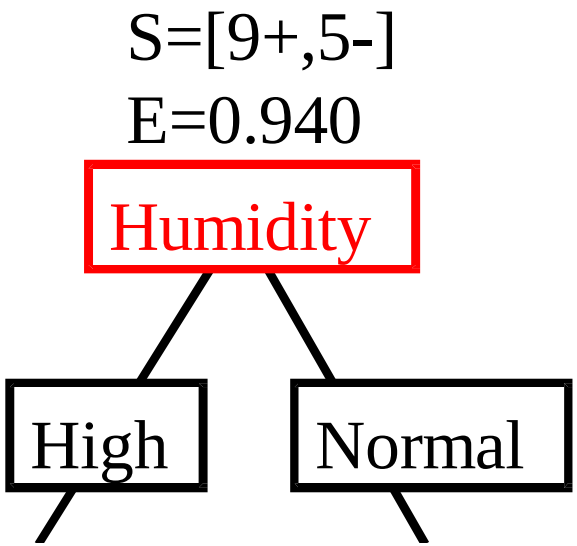
A provides greater information gain than B.
A is a better classifier than B.

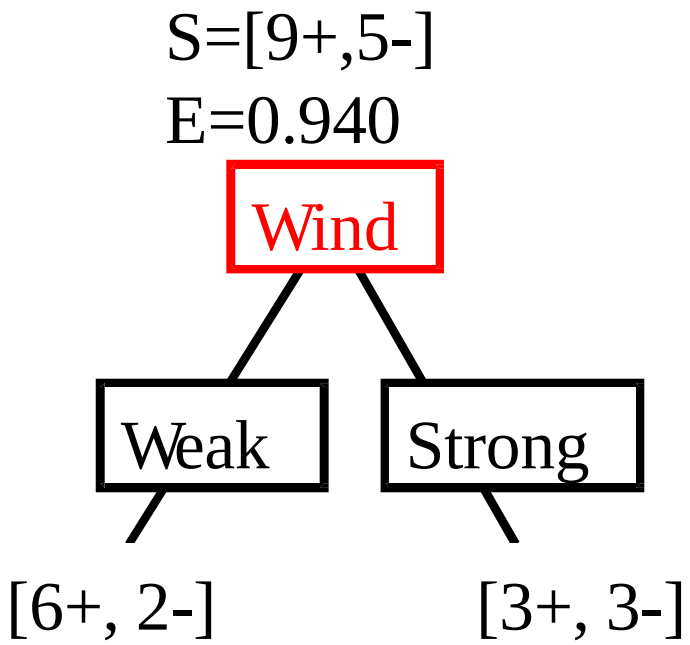
ID3 - Training Examples – [9+,5-]

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

ID3 – Selecting Next Attribute

Entropy([9+,5-] = $-(9/14)\log_2(9/14) - (5/14)\log_2(5/14)$ = 0.940

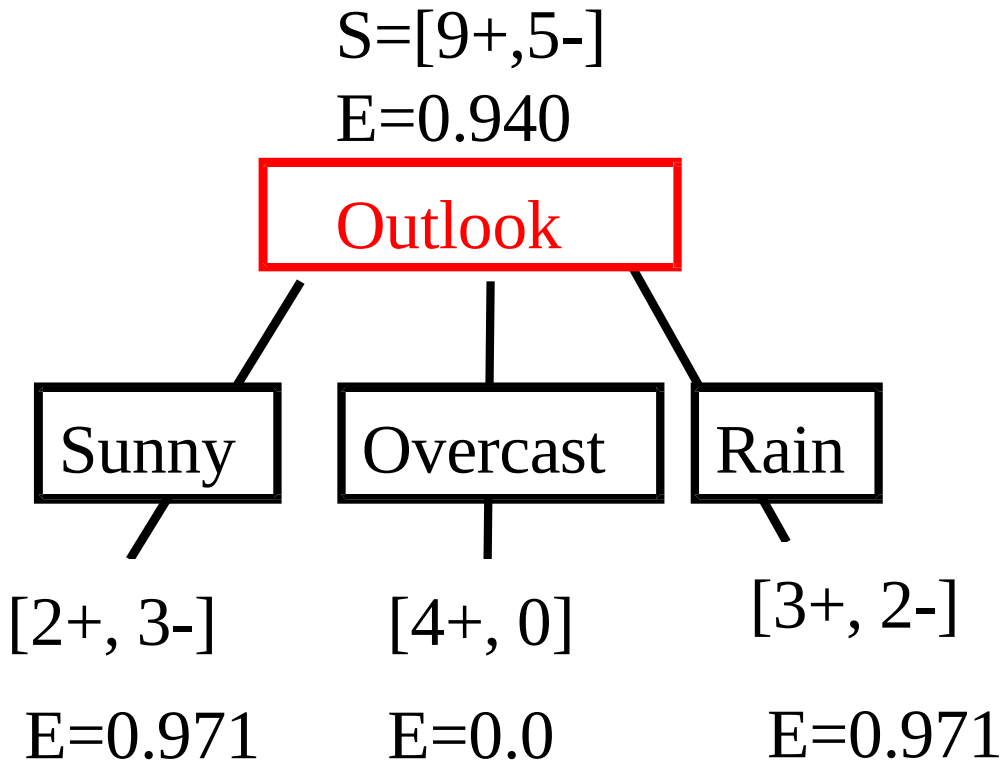




Gain(S,Humidity) =
0.940-(7/14)*0.985-(7/14)*0.592
= **0.151**

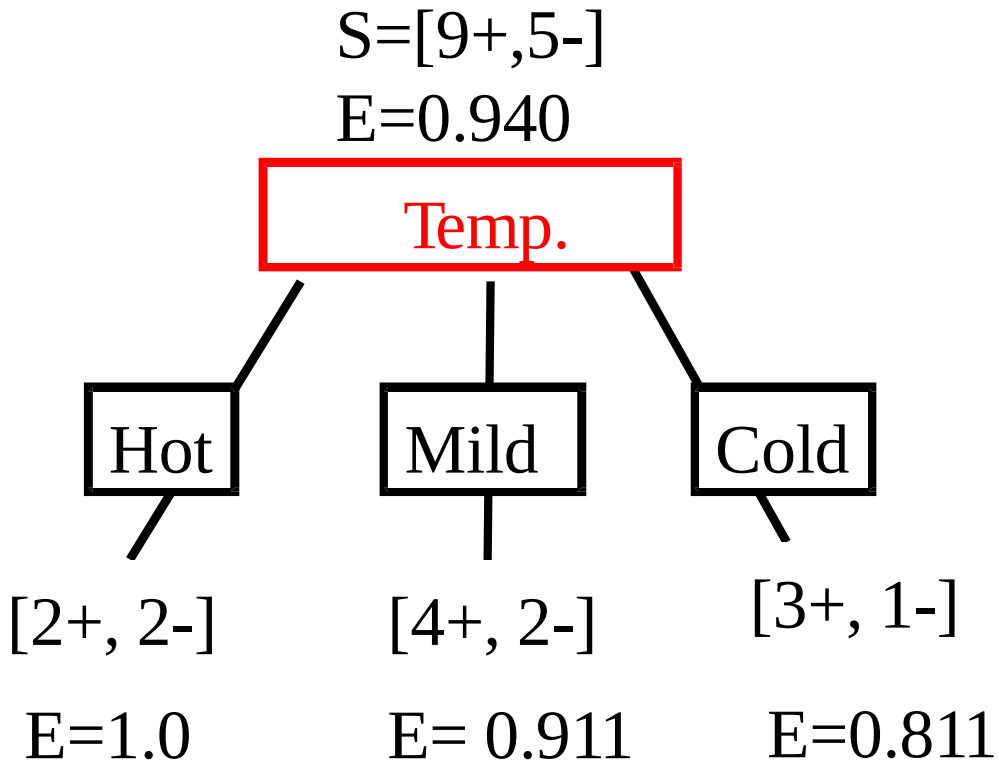
Gain(S,Wind) =
0.940-(8/14)*0.811-(6/14)*1.0
= **0.048**

ID3 – Selecting Next Attribute



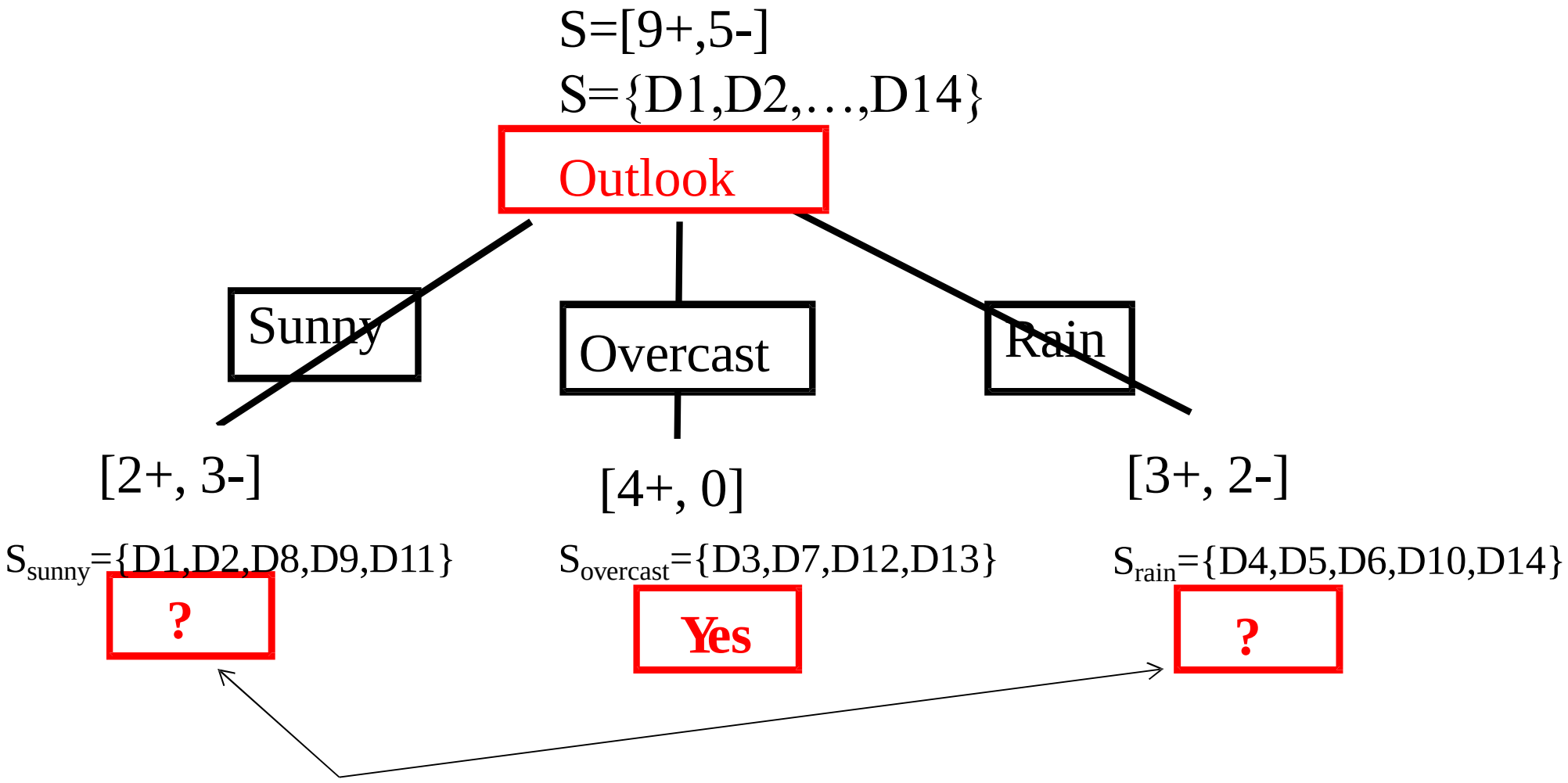
Gain(S,Outlook) =
0.940-(5/14)*0.971 -(4/14)*0.0 -(5/14)*0.0971
= **0.247**

ID3 – Selecting Next Attribute



Gain(S,Outlook) =
0.940-(4/14)*1.0 - (6/14)*0.911 - (4/14)*0.811
= **0.029**

Best Attribute - Outlook



Which attribute should be tested here?

ID3 - S_{sunny}

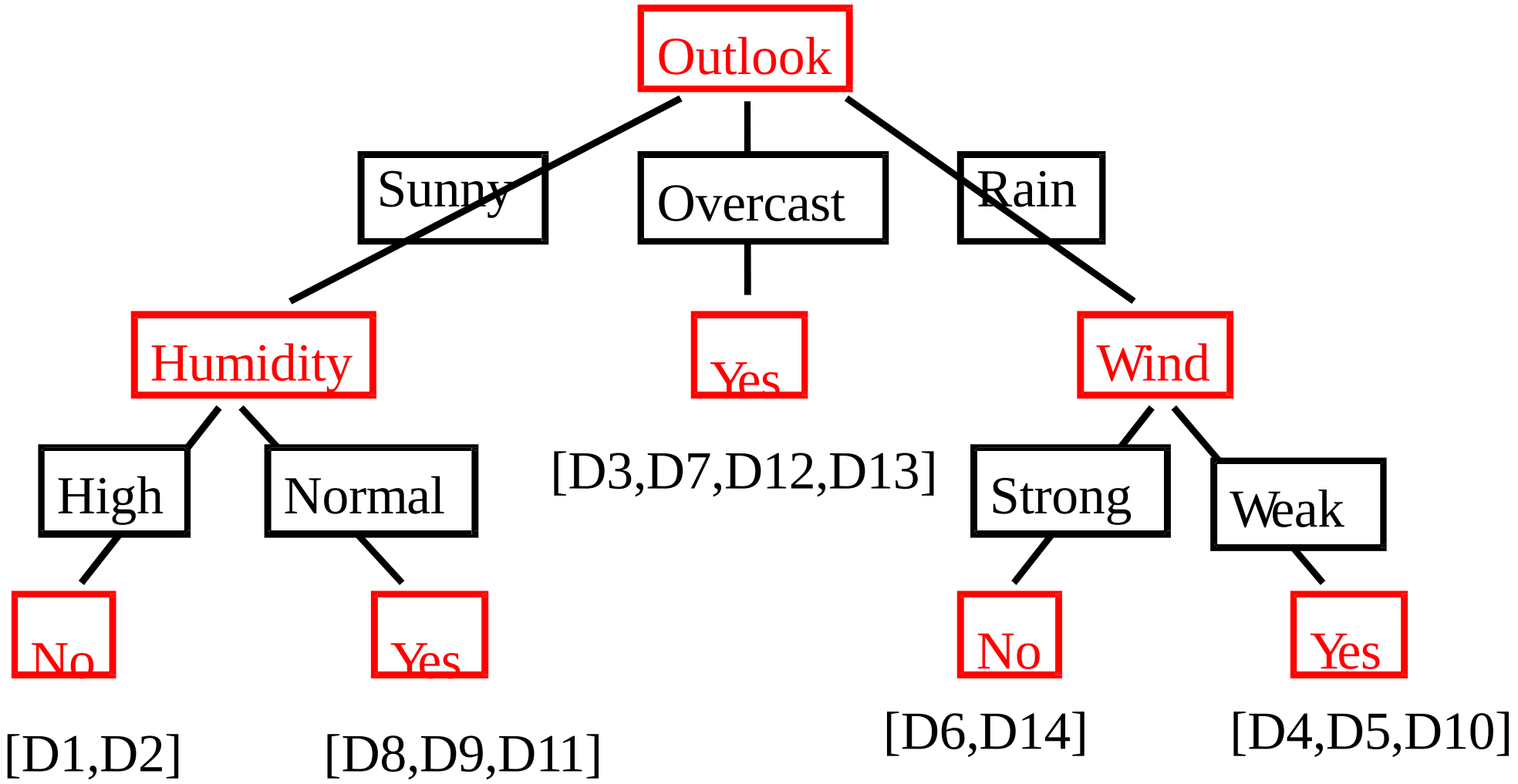
Gain(S_{sunny} , Humidity) = 0.970-(3/5)0.0 – 2/5(0.0) = **0.970**

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

So, Hummudity will be selected

ID3 - Result


Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
[D3,D7,D12,D13]
Strong
Weak
No
Yes
No
Yes
[D1,D2]
[D8,D9,D11]
[D6,D14]
[D4,D5,D10]

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*

Gini Index

Gini Index (also known as Gini impurity) **calculates the amount of probability of a specific feature that is classified incorrectly when selected randomly.**

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} (P_i)^2$$

where Pi denotes the probability of an element being classified for a distinct class.

- ***Gini index varies between values 0 and 1***,
  - *0 expresses that all the elements belong to a specified class or only one class (called* pure)
  - *1 indicates the random distribution of elements across various classes.*
  - *The value of 0.5 of the Gini Index shows an equal distribution of elements over some classes*

Classification and Regression Tree (CART) algorithm deploys the method of the Gini Index to originate binary splits.

Decision tree algorithms exploit Information Gain to divide a node and Gini Index or Entropy is the passageway to weigh the Information Gain.

Gini Index vs Information Gain

- The Gini Index facilitates the bigger distributions so easy to implement whereas the Information Gain favors lesser distributions having small count with multiple specific values.

- The method of the Gini Index is used by CART algorithms, in contrast to it, Information Gain is used in ID3, C4.5 algorithms.

- Gini index operates on the categorical target variables in terms of “success” or “failure” and performs only binary split, in opposite to that Information Gain computes the difference between entropy before and after the split and indicates the impurity in classes of elements.

ID3 Heuristic

- To determine the best attribute, we look at the ID3 heuristic
- ID3 splits attributes based on their *entropy*.
- Entropy is the measure of disinformation…

Entropy

- Entropy is minimized when all values of the target attribute are the same.
  - If we know that commute time will always be *short*, then entropy = 0
- Entropy is maximized when there is an equal chance of all values for the target attribute (i.e. the result is random)
  - If commute time = short in 3 instances, medium in 3 instances and long in 3 instances, entropy is maximized

Entropy

- Calculation of entropy
  - Entropy(S) = $\sum_{(i=1 \text{ to } l)} -|S_i|/|S| * \log_2(|S_i|/|S|)$
    - S = set of examples
    - S_i = subset of S with value v_i under the target attribute
    - l = size of the range of the target attribute

ID3

- ID3 splits on attributes with the lowest entropy
- We calculate the entropy for all values of an attribute as the weighted sum of subset entropies as follows:
  - $\sum_{(i = 1 \text{ to } k)} |S_i|/|S|$ Entropy(S_i), where k is the range of the attribute we are testing
- We can also measure information gain (which is inversely proportional to entropy) as follows:
  - Entropy(S) - $\sum_{(i = 1 \text{ to } k)} |S_i|/|S|$ Entropy(S_i)

ID3

- Given our commute time sample set, we can calculate the entropy of each attribute at the root node

Attribute	Expected Entropy	Information Gain
Hour	0.6511	0.768449
Weather	1.28884	0.130719
Accident	0.92307	0.496479
Stall	1.17071	0.248842

Pruning Trees

- There is another technique for reducing the number of attributes used in a tree - *pruning*
- Two types of pruning:
  - Pre-pruning (forward pruning)
  - Post-pruning (backward pruning)

Prepruning

- In prepruning, we decide during the building process when to stop adding attributes (possibly based on their information gain)

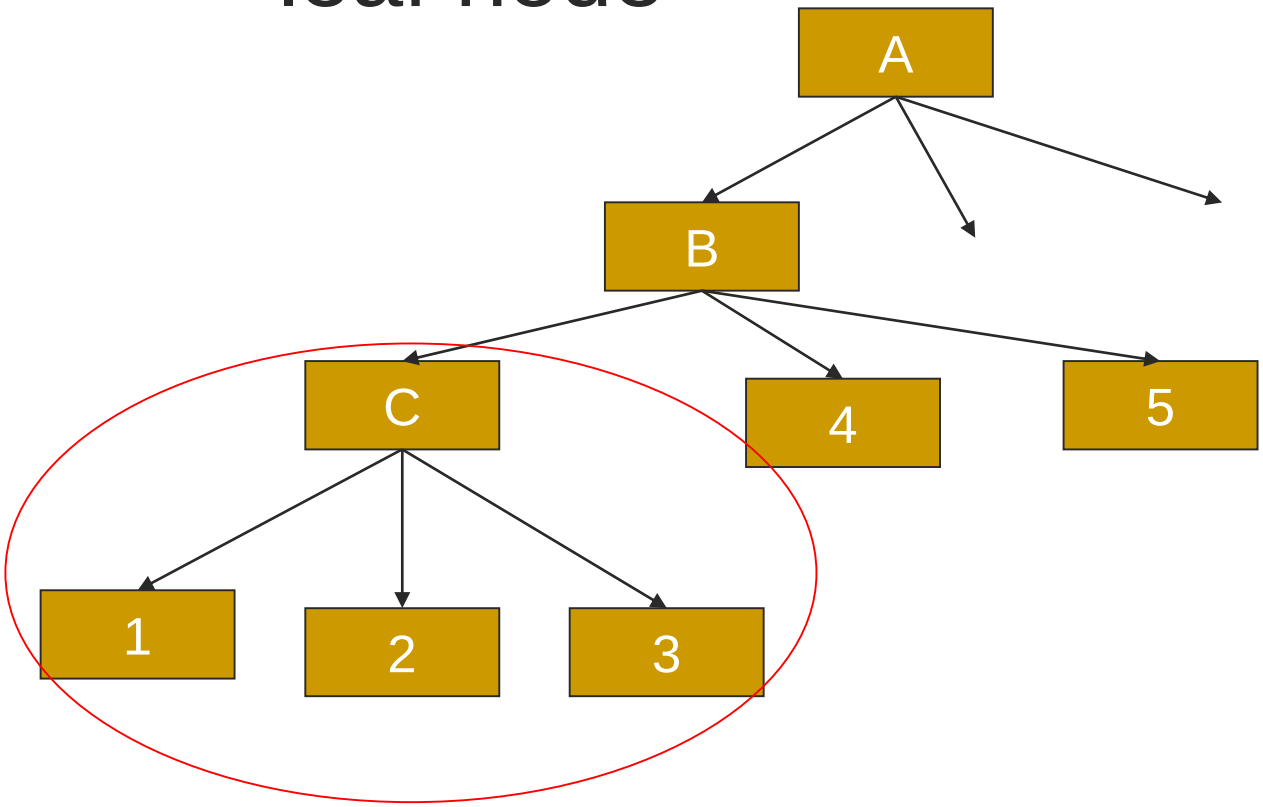
- However, this may be problematic – Why?
  - Sometimes attributes individually do not contribute much to a decision, but combined, they may have a significant impact

Postpruning

- Postpruning waits until the full decision tree has built and then prunes the attributes
- Two techniques:
  - Subtree Replacement
  - Subtree Raising

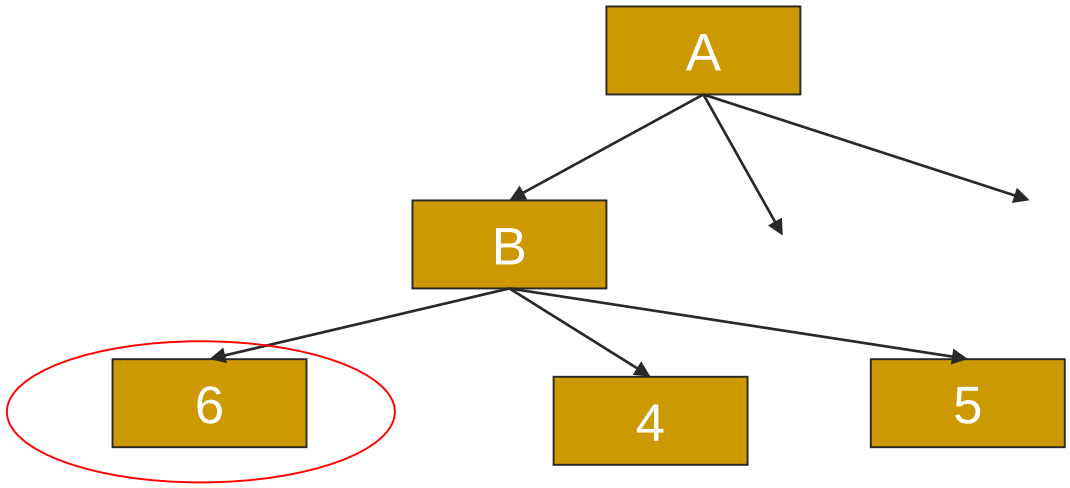
Subtree Replacement

- Entire subtree is replaced by a single leaf node



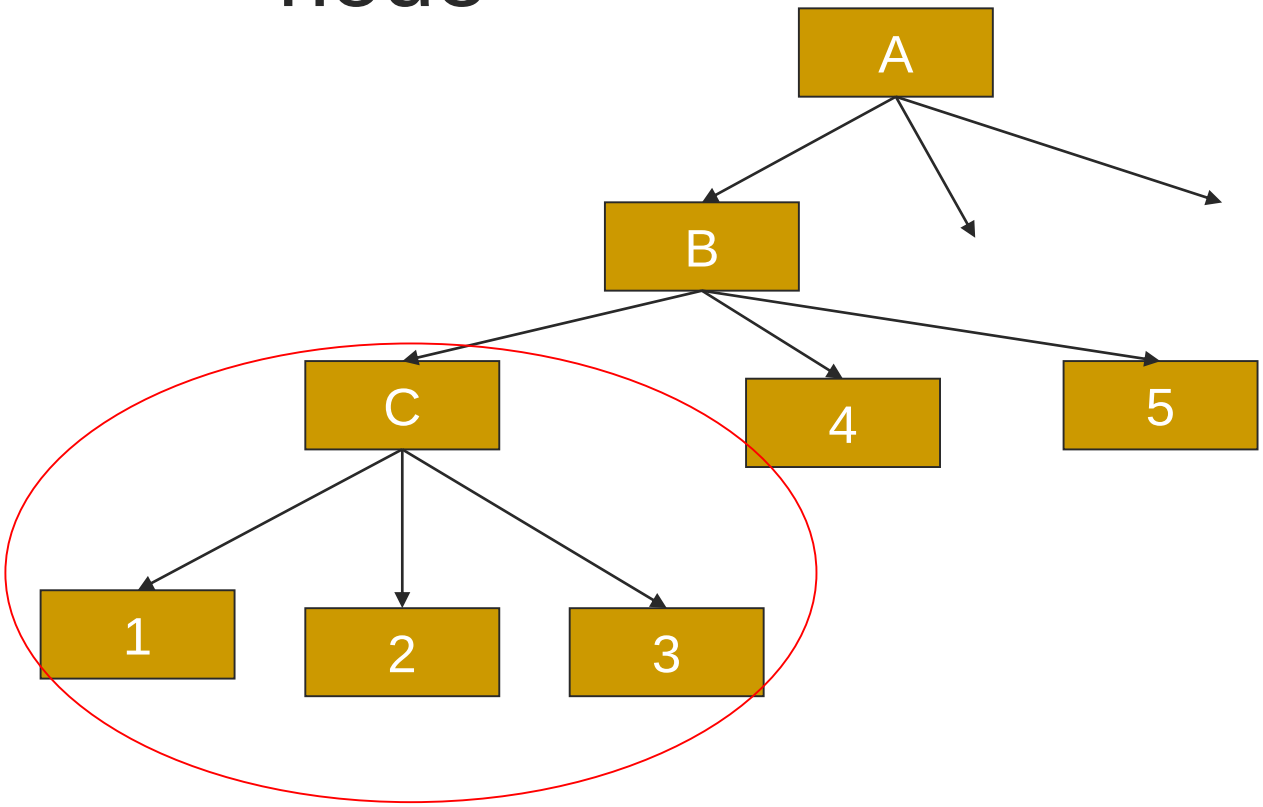
Subtree Replacement

- Node 6 replaced the subtree
- Generalizes tree a little more, but may increase accuracy



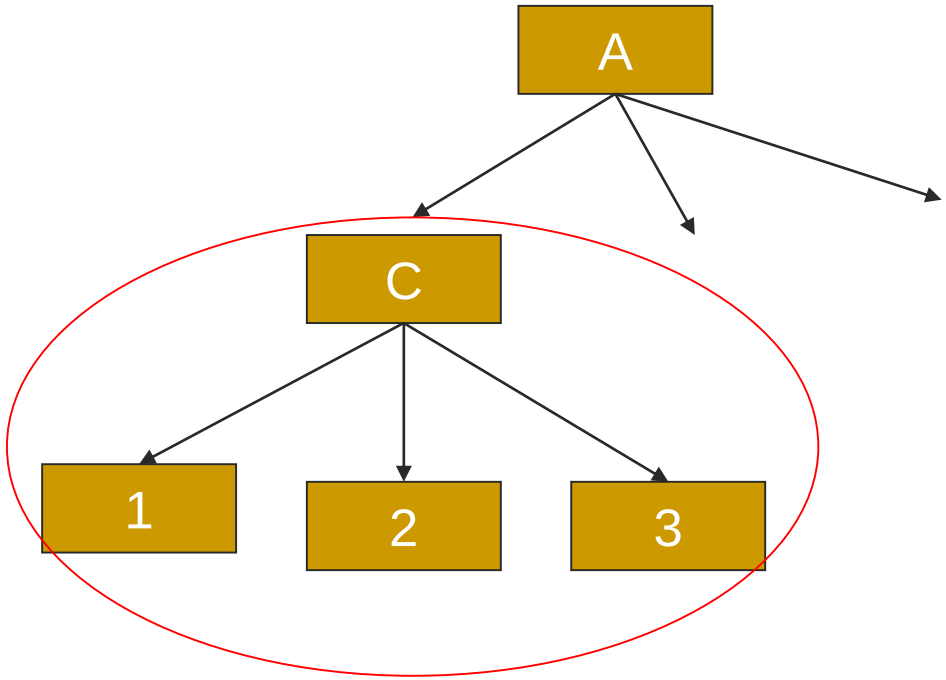
Subtree Raising

- Entire subtree is raised onto another node



Subtree Raising

- Entire subtree is raised onto another node
- This was not discussed in detail as it is not clear whether this is really worthwhile (as it is very time consuming)



Problems with ID3

- ID3 is not optimal
  - Uses *expected* entropy reduction, not actual reduction
- Must use discrete (or discretized) attributes
  - What if we left for work at 9:30 AM?
  - We could break down the attributes into smaller values…

Problems with Decision Trees

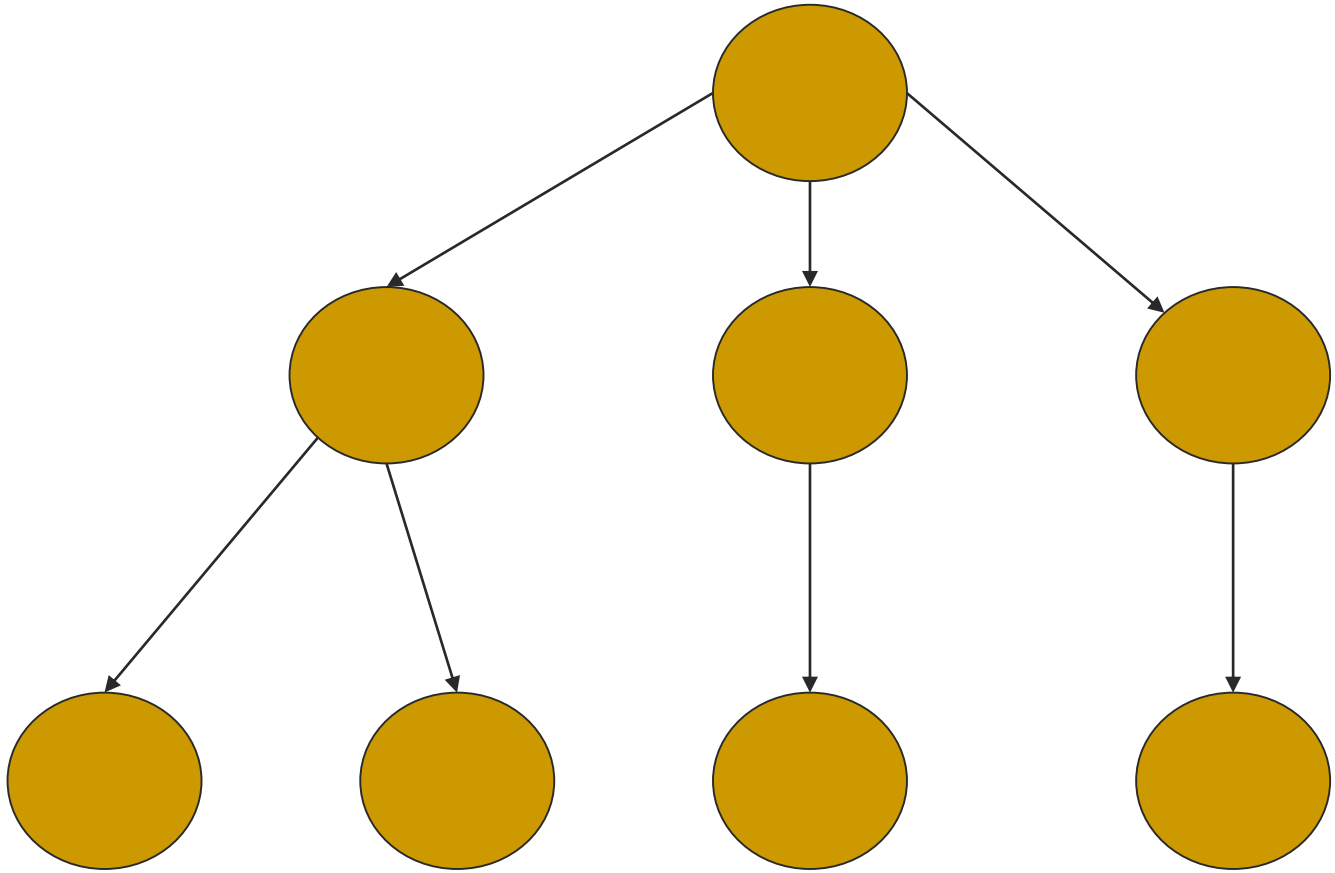
- While decision trees classify quickly, the time for building a tree may be higher than another type of classifier

- Decision trees suffer from a problem of errors propagating throughout a tree
  - A very serious problem as the number of classes increases

Error Propagation

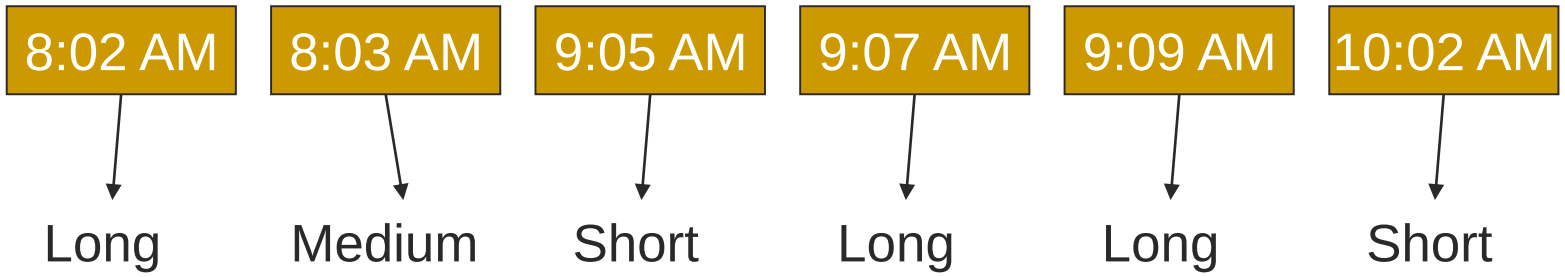
- Since decision trees work by a series of local decisions, what happens when one of these local decisions is wrong?
  - Every decision from that point on may be wrong
  - We may never return to the correct path of the tree

Error Propagation Example



Problems with ID3

- If we broke down leave time to the minute, we might get something like this:



Since entropy is very low for each branch, we have n branches with n leaves. This would not be helpful for predictive modeling.

Problems with ID3

- We can use a technique known as discretization
- We choose *cut points*, such as 9AM for splitting continuous attributes
- These cut points generally lie in a subset of *boundary points*, such that a boundary point is where two adjacent instances in a sorted list have different target value attributes

Problems with ID3

- Consider the attribute commute time

8:00 (L), 8:02 (L), 8:07 (M), 9:00 (S), 9:20 (S), 9:25 (S), 10:00 (S), 10:02 (M)

When we split on these attributes, we increase the entropy so we don't have a decision tree with the same number of cut points as leaves