Decision Trees

Introduction

Entropy

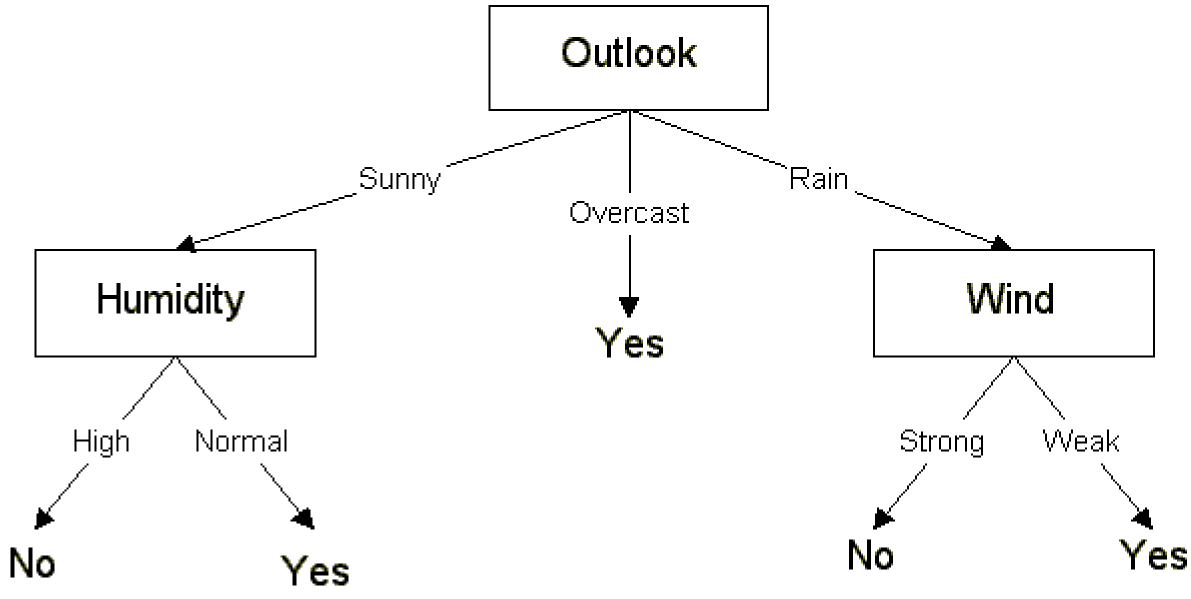
Information gain

Tree models

- Trees are expressive and easy to understand.
- A feature tree is a tree such that each internal node is labeled with a feature and each edge emanating from an internal node is labeled with a literal.
- The set of literals at a node is called a *split*.
- Each leaf of the tree represents a logical expression, which is the conjunction of literals on the path from root to the leaf.
- The extension of that conjunction is called the *instance space segment associated with the leaf*.

Tree models

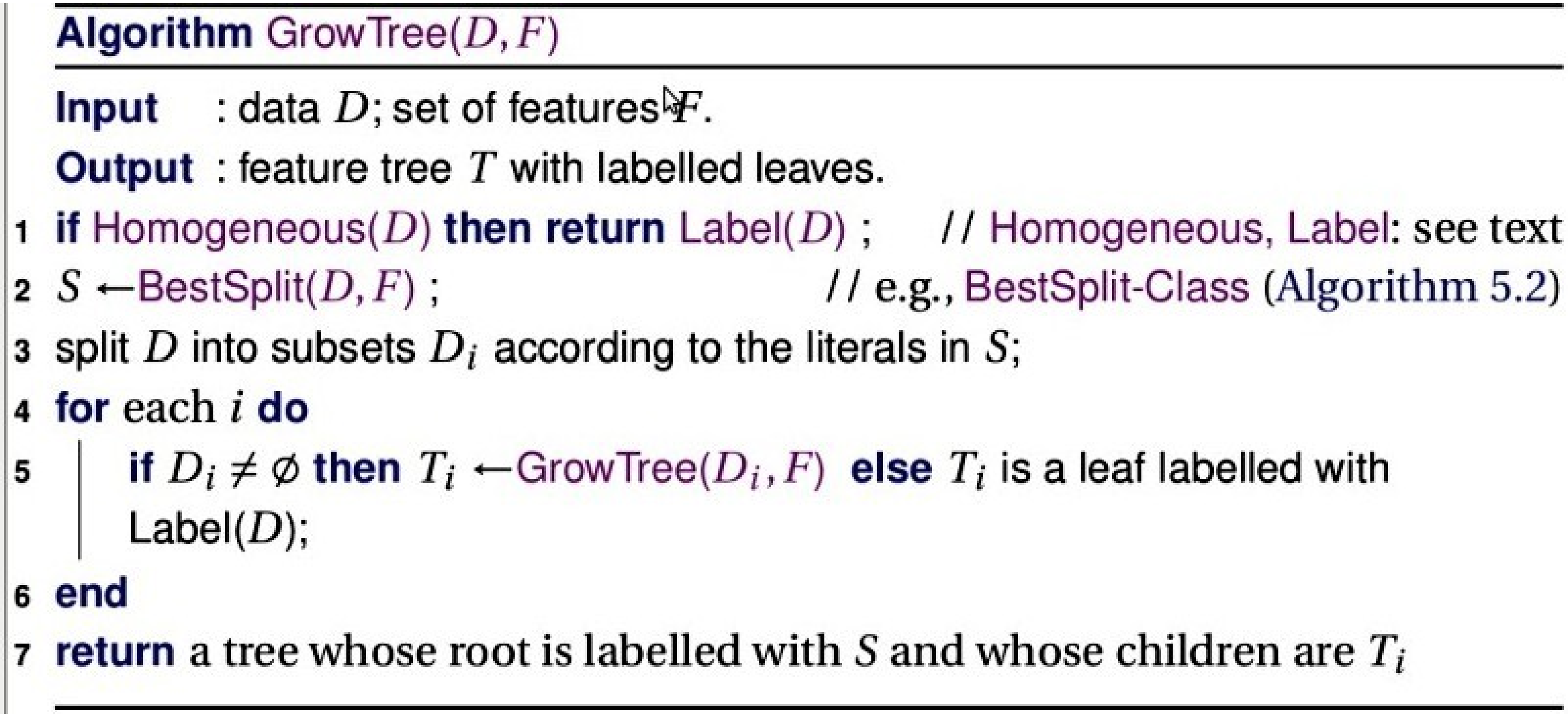
- An internal node test the value of particular features.
- A branch represents an outcome of the test.
- A leaf node represents a class label *h*(x) or class label distribution.



Building decision trees

- Top-down tree construction
  - At start, all training examples are at the root.
  - Partition the examples recursively by choosing one attribute each time.
- Bottom-up tree pruning
  - Remove subtrees or branches, in a bottom-up manner, to improve the estimated accuracy on new cases.

Building decision trees



```
Algorithm GrowTree($D, F$)
Input  : data $D$; set of features $F$.
Output : feature tree $T$ with labelled leaves.
if Homogeneous($D$) then return Label($D$) ;      // Homogeneous, Label: see text
$S$ ←BestSplit($D, F$) ;                     // e.g., BestSplit-Class (Algorithm 5.2)
split $D$ into subsets $D_i$ according to the literals in $S$;
for each $i$ do
    if $D_i \neq \emptyset$ then $T_i$ ←GrowTree($D_i, F$) else $T_i$ is a leaf labelled with
    Label($D$);
end
return a tree whose root is labelled with $S$ and whose children are $T_i$
```

Building decision trees

Algorithm 5.1 gives the generic learning procedure common to most tree learners. It assumes that the following three functions are defined:

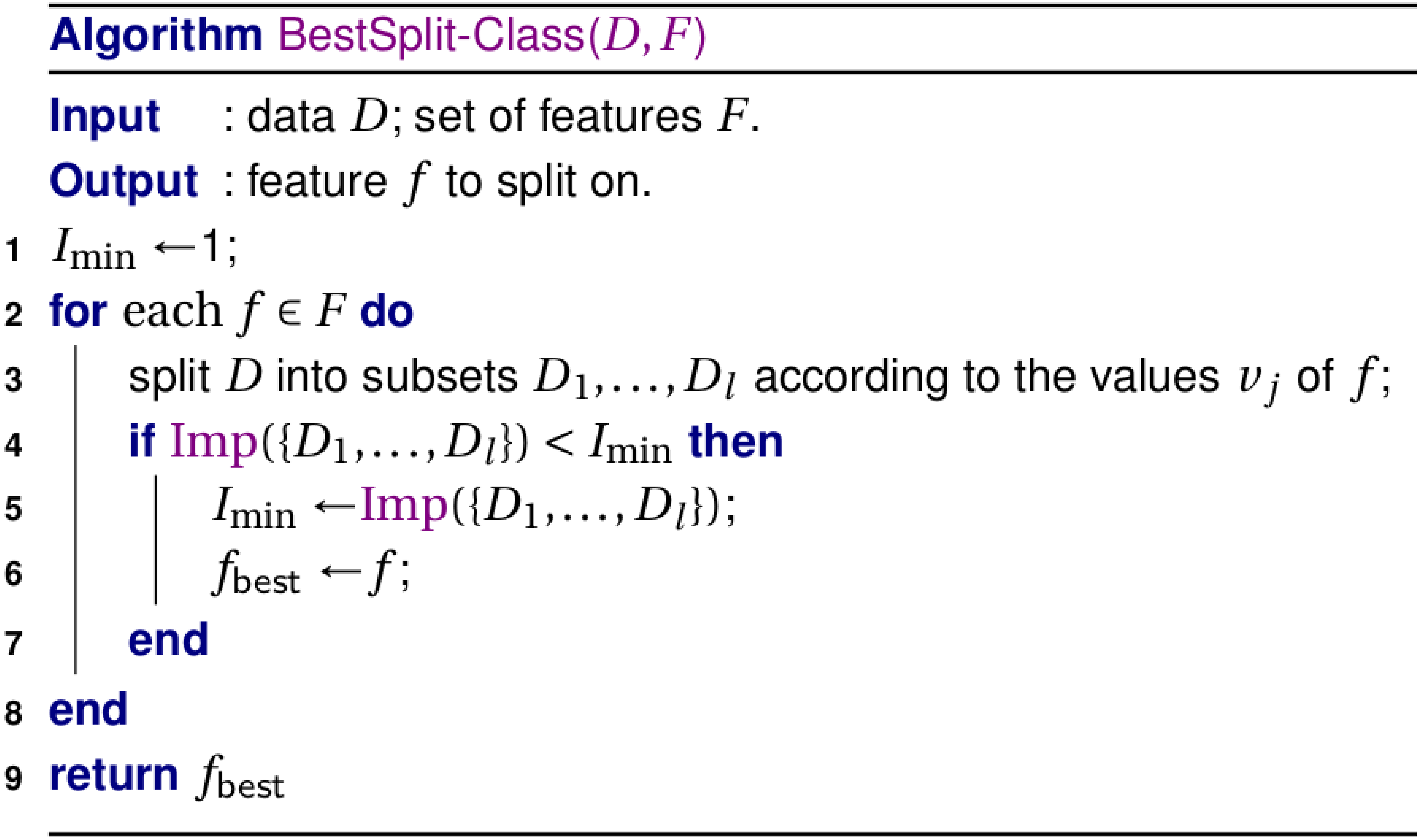
Homogeneous(D) returns true if the instances in D are homogeneous enough to be labelled with a single label, and false otherwise;

Label(D) returns the most appropriate label for a set of instances D;

BestSplit(D, F) returns the best set of literals to be put at the root of the tree.

These functions depend on the task at hand: for instance, for classification tasks a set of instances is homogeneous if they are (mostly) of a single class, and the most appropriate label would be the majority class. For clustering tasks a set of instances is homogenous if they are close together, and the most appropriate label would be some exemplar such as the mean.

Building decision trees



```
Algorithm BestSplit-Class($D, F$)
Input  : data $D$; set of features $F$.
Output : feature $f$ to split on.
$I_{\min} \leftarrow 1$;
for each $f \in F$ do
    split $D$ into subsets $D_1, \ldots, D_l$ according to the values $v_j$ of $f$;
    if $\mathrm{Imp}(\{D_1, \ldots, D_l\}) < I_{\min}$ then
        $I_{\min} \leftarrow \mathrm{Imp}(\{D_1, \ldots, D_l\})$;
        $f_{\mathrm{best}} \leftarrow f$;
    end
end
return $f_{\mathrm{best}}$
```

Building decision trees

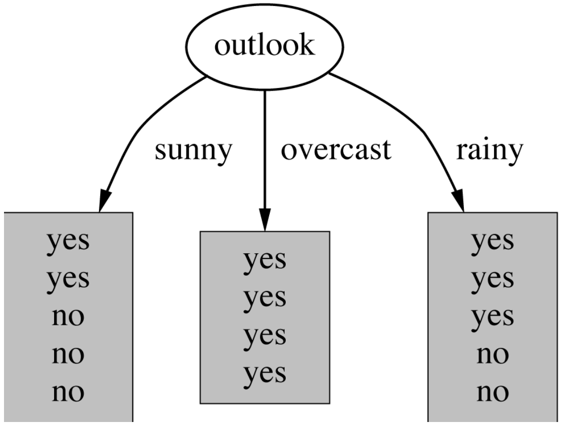
- The construction of a tree involves the following three general elements:
    - The selection of the splits i.e., which node to split and how to split it?
    - When to declare a node terminal and stop splitting?
    - We have to assign each terminal node to a class. How do we assign these class labels?

Building decision trees

- The algorithm starts as a single node N, representing the training tuples in D.
- If the tuples in D are all of the same class, then node N becomes a leaf and is labeled with that class.
- Otherwise, the algorithm calls BestSplit to determine the splitting criterion.
  - This tells which attribute to test at node N by determining the way to partition the tuples in D into individual classes.
  - The splitting attribute is determines so that, the resulting partitions is pure – if all tuples in it belong to same class.

Building decision trees

- The node N is labeled with the splitting criterion, which serves as a test at the node.
  - A branch is grown from N for each of the outcome of splitting criterion.
  - The tuples in D are partitioned accordingly.
- The algorithm uses same process recursively to form a decision tree for the tuples at each resulting partition Dj, of D.



Building decision trees

- The recursive partitioning stops under any one of following terminating condition:
  - All of the tuples in partition D (at node N) belong to the same class, or
  - There are no remaining attributes on which the tuples may be further partitioned, or
  - There are no tuples for a given branch, i.e., a partition Dj is empty.

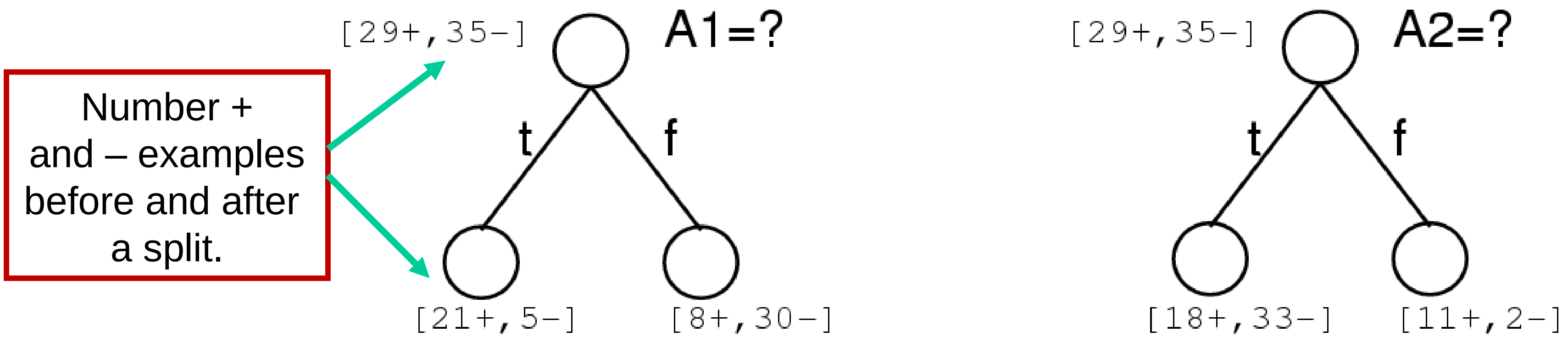
Choosing the Best attribute

- How to choose the best attribute or feature?
- A *goodness of split* is used for this purpose.
- The *goodness of split* in turn is measured by an *impurity function*
- Typical impurity functions:
  - Minority class : $\min(p, 1-p)$
  - Gini index : $2p(1-p)$
  - Entropy : $-p \log_2 p - (1-p) \log_2 (1-p)$

Choosing the Best attribute

A1 and A2 are “attributes” (i.e. features or inputs).

Which attribute is best?

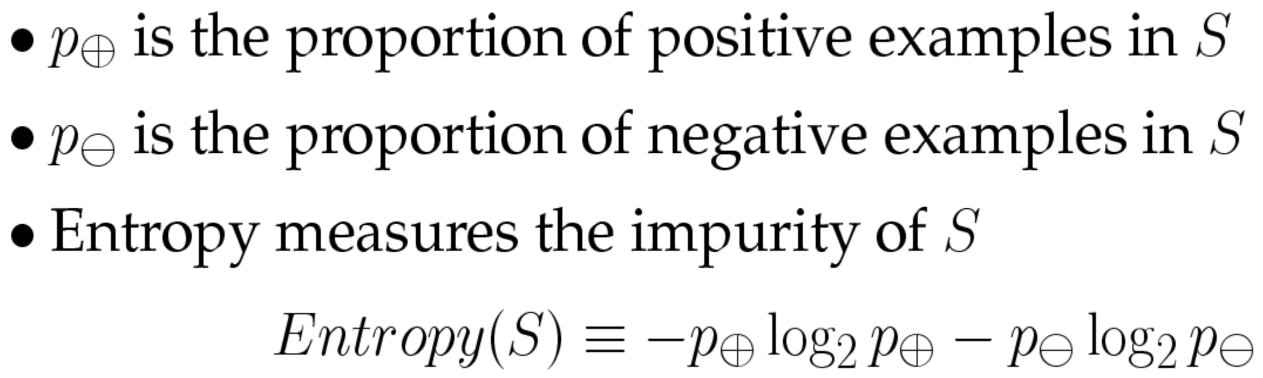


Number + and – examples before and after a split.

- Many different frameworks for choosing BEST have been proposed!
- We will look at Entropy Gain.

Entropy

- A formula to calculate the homogeneity of a sample.
- A completely homogeneous sample has entropy of 0.
- An equally divided sample has entropy of 1.
- p+ = no. of positive samples / total samples in S.
- p- = no. of negative samples / total samples in S.



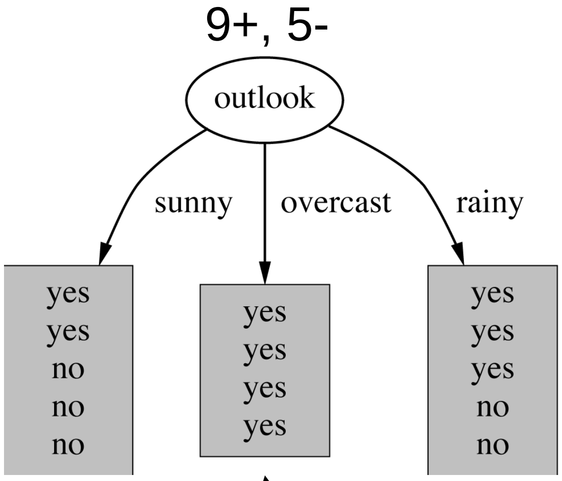
- $p_{\oplus}$ is the proportion of positive examples in S
- $p_{\ominus}$ is the proportion of negative examples in S
- Entropy measures the impurity of S

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

Entropy

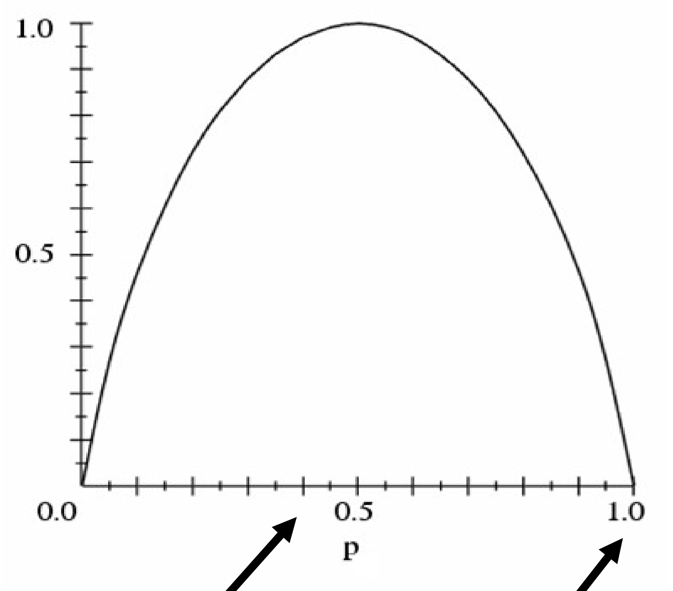
- $Entropy\ (S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$ $[0 \log_2 0 = 0]$
- $Entropy\ ([14+, 0–]) = –14/14 \log_2 (14/14) – 0 \log_2 (0) = 0$
- $Entropy\ ([9+, 5–]) = –9/14 \log_2 (9/14) – 5/14 \log_2 (5/14) = 0.94$
- $Entropy\ ([7+, 7–]) = –7/14 \log_2 (7/14) – 7/14 \log_2 (7/14)$
  $= 1/2 + 1/2 = 1$ $[\log_2 1/2 = –1]$
- Note: $0 \leq p \leq 1$, $0 \leq entropy \leq 1$

Entropy



pure, 100% yes

not pure at all, 40% yes



pure, 100% yes

not pure at all, 40% yes

Information gain (IG)

- Which is the best attribute?
  - The one which will result in the smallest tree.
- Heuristic: choose the attribute that produces the “purest” nodes
- Popular *impurity criterion*: *information gain*
  - Information gain increases with the average purity of the subsets that an attribute produces.
  - Information gain uses entropy H(*p*)
- Strategy: choose attribute that results in greatest information gain.

Information gain (IG)

- The information gain is based on the decrease in entropy after a dataset is split on an attribute.
- Which attribute creates the most homogeneous branches?
- Calculate the entropy of the total dataset D before split.
- 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.

Information gain (IG)

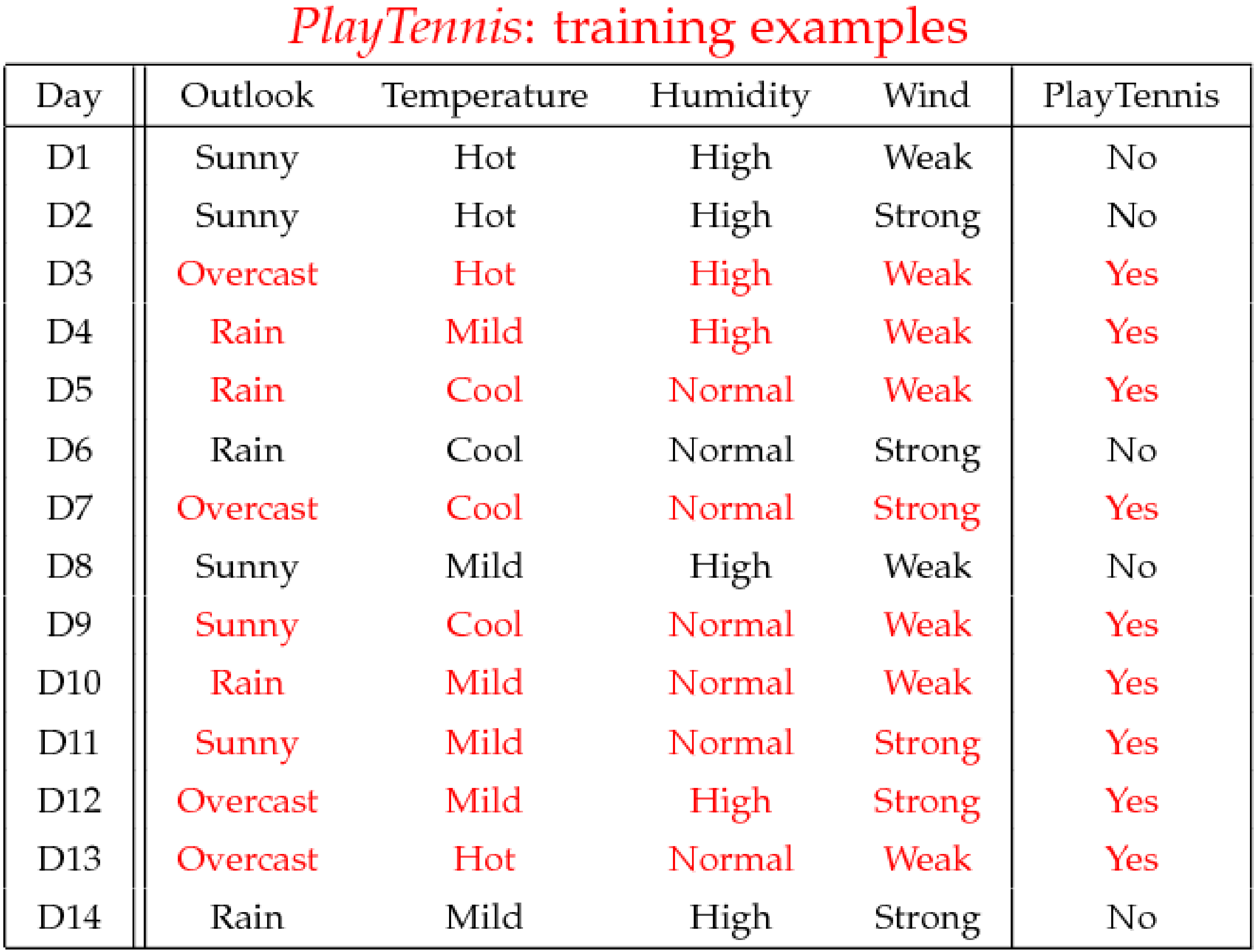
- Information gain = entropy before split – total entropy for the split.
- The attribute that yields the largest IG is chosen for the decision node.

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

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

Decision Trees

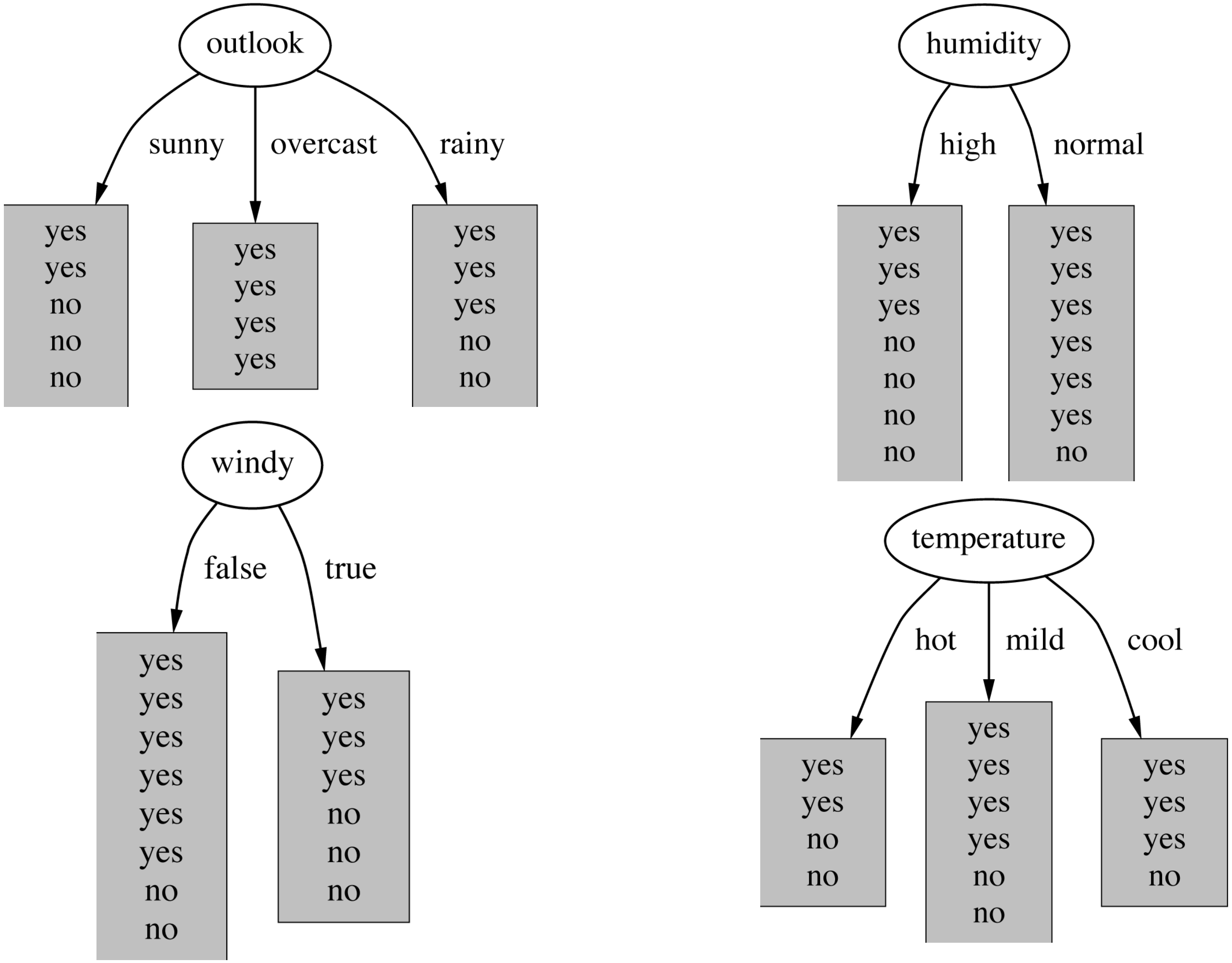
When do I play tennis?



PlayTennis: training 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

Best node?


outlook
sunny
overcast
rainy
yes yes no no no
yes yes yes yes
yes yes yes no no
windy
false
true
yes yes yes yes yes yes no no
yes yes yes no no no
humidity
high
normal
yes yes yes no no no no
yes yes yes yes yes yes no
temperature
hot
mild
cool
yes yes no no
yes yes yes yes no no
yes yes yes no

Attribute: Outlook

- "Outlook" = "Sunny":

$$\text{info}([2,3]) = \text{entropy}(2/5,3/5) = -2/5\log(2/5) - 3/5\log(3/5) = 0.971\,\text{bits}$$

- "Outlook" = "Overcast":

$$\text{info}([4,0]) = \text{entropy}(1,0) = -1\log(1) - 0\log(0) = 0\ \text{bits}$$

- "Outlook" = "Rainy":

$$\text{info}([3,2]) = \text{entropy}(3/5,2/5) = -3/5\log(3/5) - 2/5\log(2/5) = 0.971\,\text{bits}$$

- Expected Information for "Outlook":

$$\text{info}([3,2],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971$$
$$= 0.693\,\text{bits}$$

Information gain

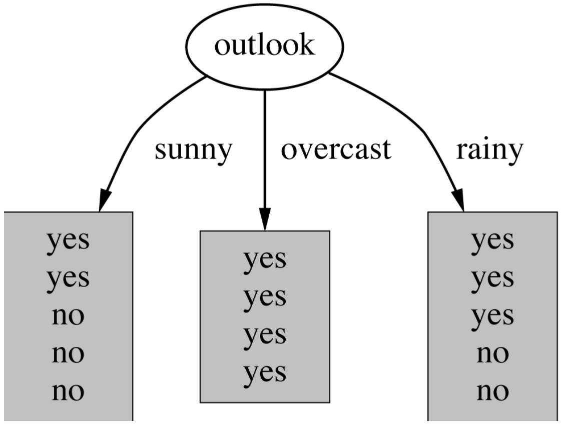
- Entropy(S) = - (9/14) $\log_2$ (9/14) - (5/14) $\log_2$ (5/14) = 0.940
- Information gain:

  (information before split) – (information after split)

$$\text{gain}(\text{"Outlook"}) = \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) = 0.940 - 0.693$$

$$= 0.247 \text{ bits}$$

- Similarly calculate the information gain for 'Wind” attribute!



Attribute: Wind

$Values(Wind) = \{Weak, Strong\}$

$S = [9+, 5-] = 0.94$

$S_{Weak} = [6+, 2-] = \text{entropy}(S_{Weak}) = -\ 6/8 \log(6/8) – 2/8 \log(2/8) = 0.811$

$S_{Strong} = [3+, 3-] = \text{entropy}(S_{Strong}) = -\ 3/3 \log(3/3) – 3/3 \log(3/3) = 1$

- Information gain due to knowing *Wind*:

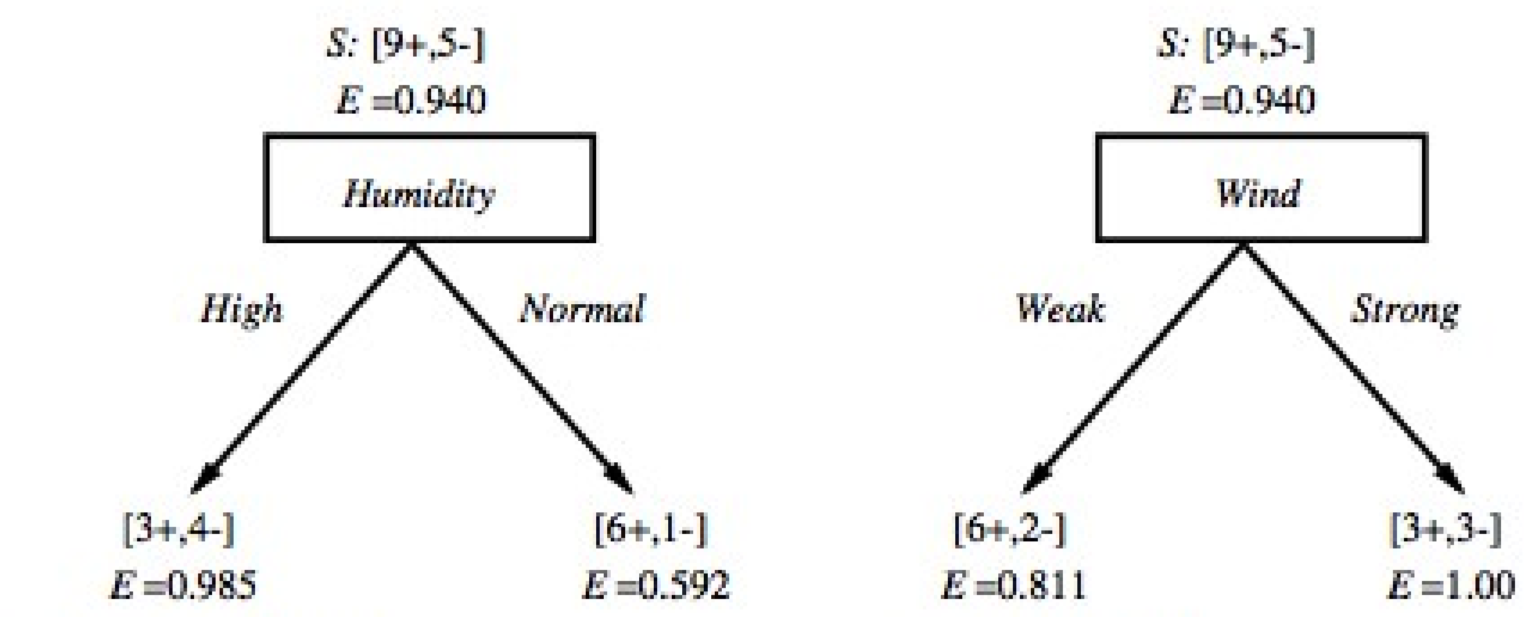
  $Gain(S, Wind) = Entropy(S) - 8/14\ Entropy(S_{Weak}) - 6/14\ Entropy(S_{Strong})$

  $= 0.94 - 8/14 \times 0.811 - 6/14 \times 1.00$

  $= 0.048$

Best Attribute ?

Which attribute is the best classifier?



Gain (S, Humidity)

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

Gain (S, Wind)

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

Best Attribute ?

- Which attribute should be tested at the root?

$$\text{gain}(\text{"Outlook"}) = 0.247 \text{ bits}$$

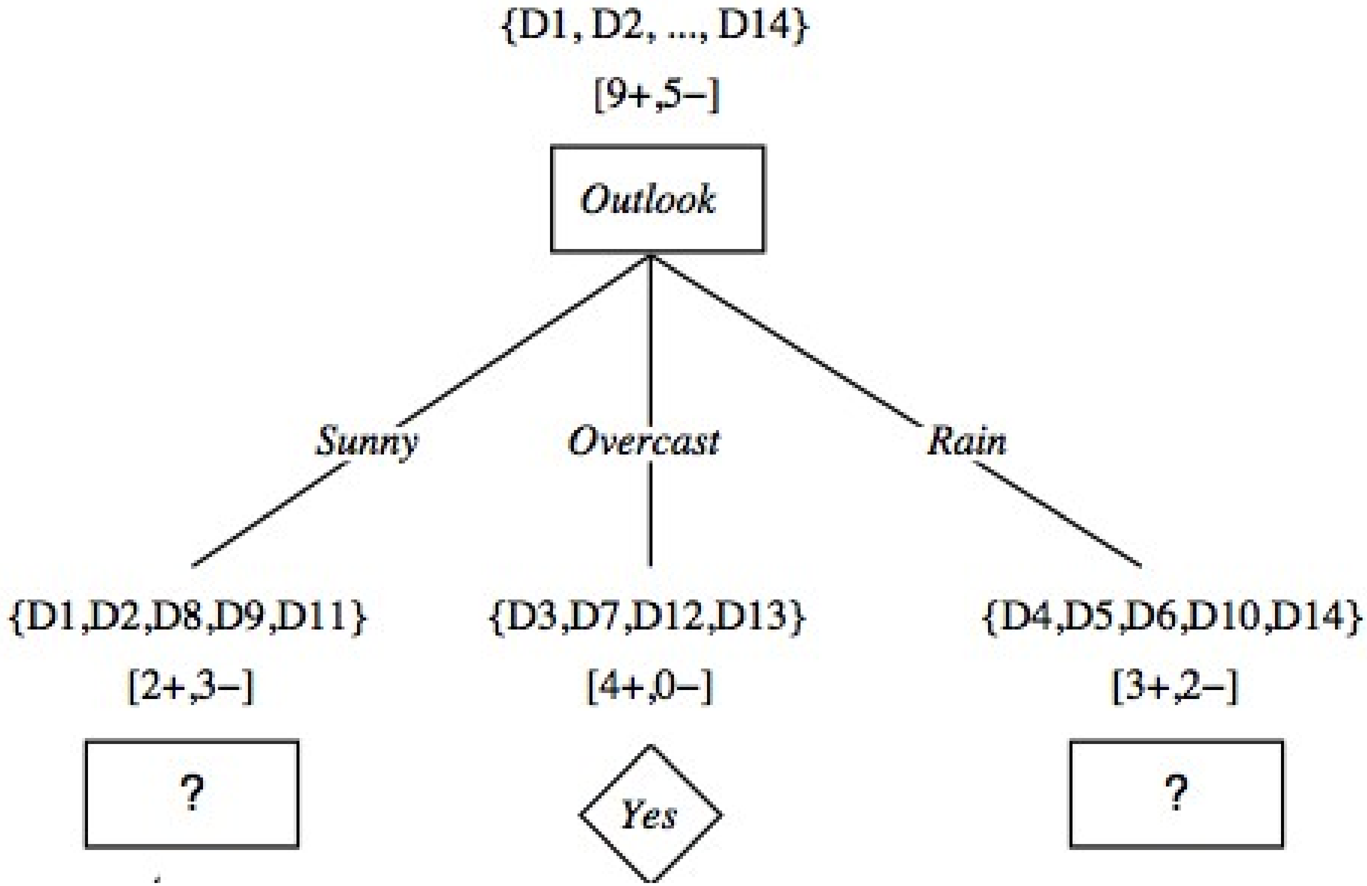
$$\text{gain}(\text{"Temperature"}) = 0.029 \text{ bits}$$

$$\text{gain}(\text{"Humidity"}) = 0.152 \text{ bits}$$

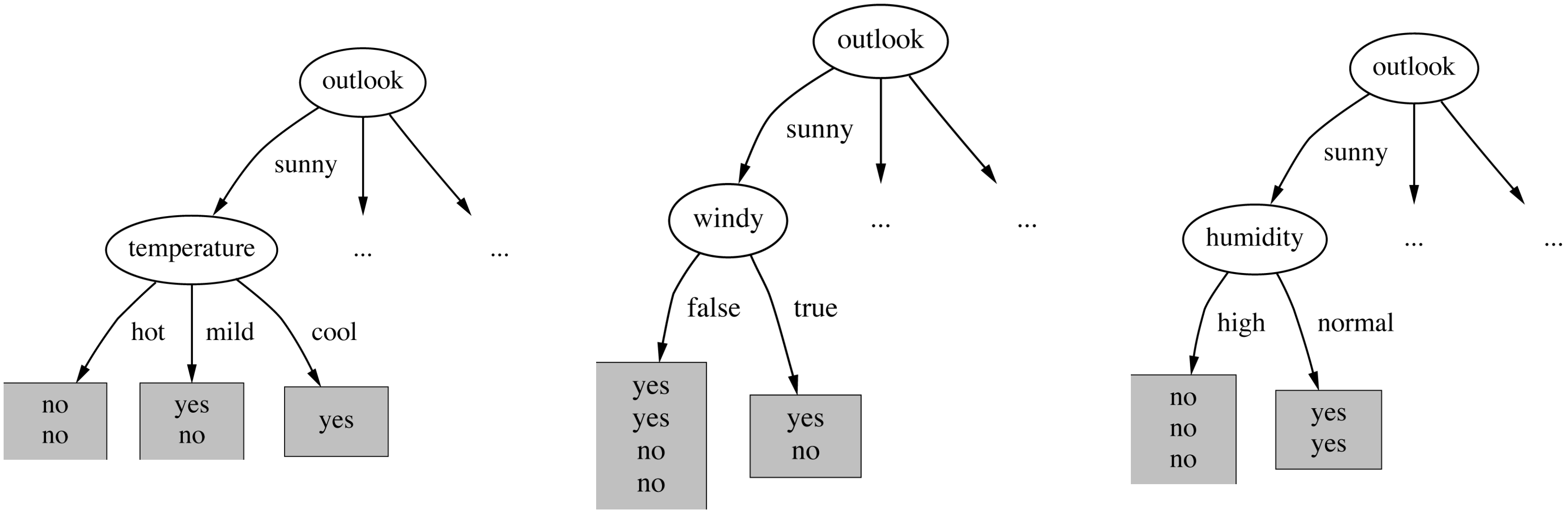
$$\text{gain}(\text{"Windy"}) = 0.048 \text{ bits}$$

- *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



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

$$Gain(S_{Sunny}, Wind) = 0.970 - 2/5 \times 1.0 - 3/5 \times 0.918 = 0\ .019$$

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

Second step

- Working on *Outlook=Sunny* node:

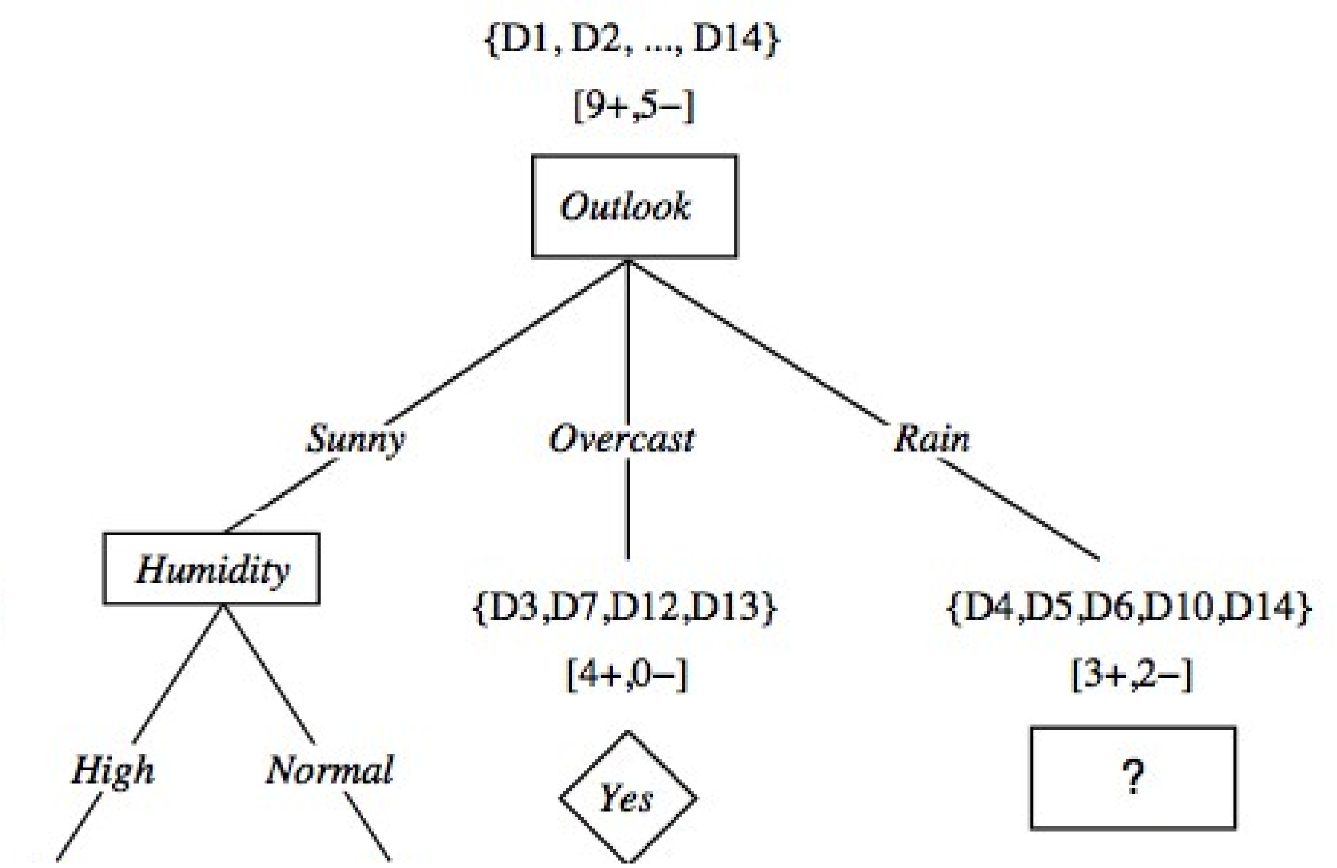
$$\text{gain}(\text{"Temperature"}) = 0.571\ \text{bits}$$

$$\text{gain}(\text{"Windy"}) = 0.020\ \text{bits}$$

$$\text{gain}(\text{"Humidity"}) = 0.971\ \text{bits}$$

- *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*

After second step



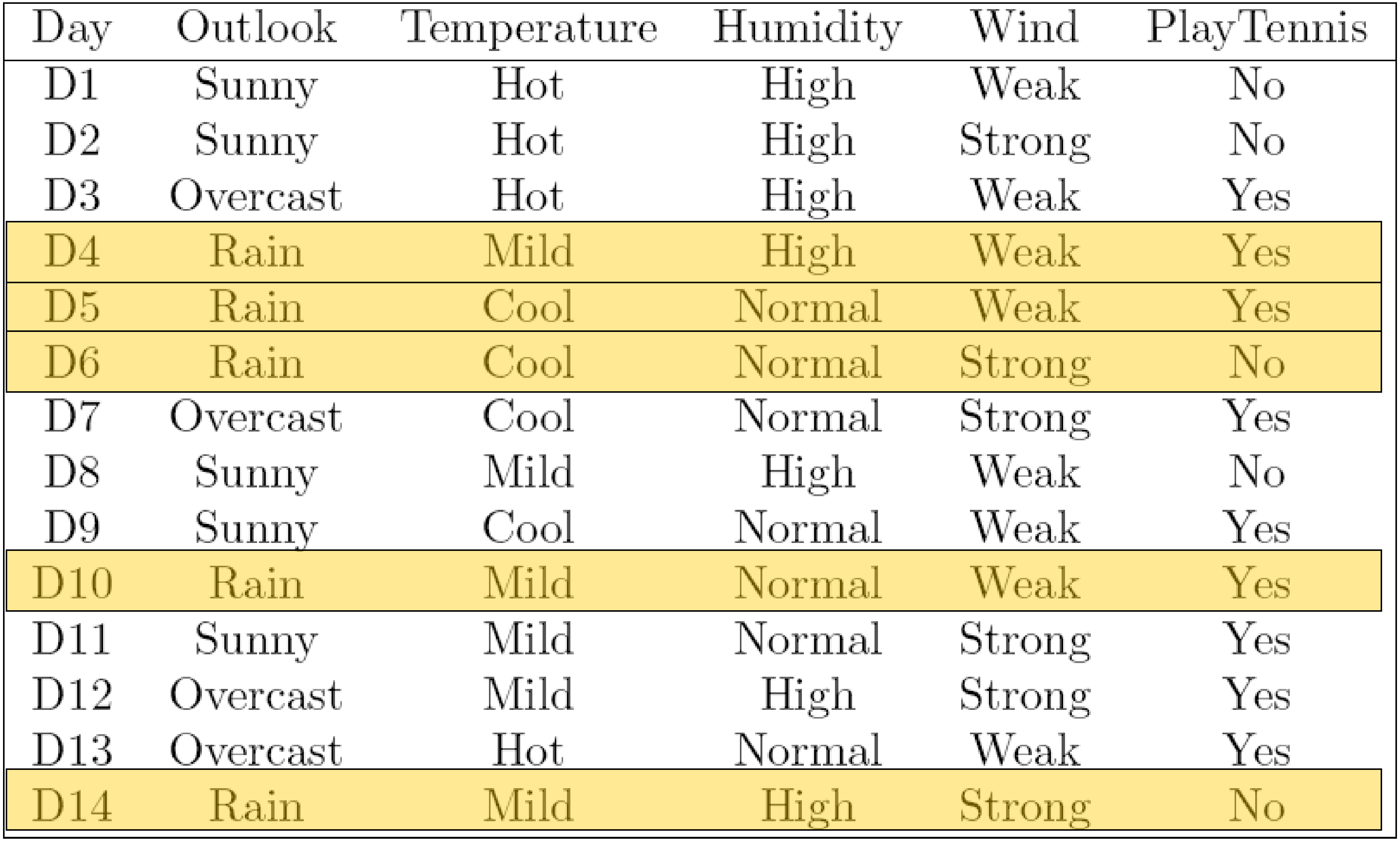
{D1, D2, D8}
No

{D9, D11}
Yes

After second step

- The test on *Humidity* provides two splits, and the resulting tuples of each split falls on the same class.
- Hence leaf nodes are labeled with that class.
- Lets grow the tree by testing the node *rain*:
  - add to the tree a successor for each possible value of *rain.*
  - partition the training samples according to the value of *rain.*

Wind attribute



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

Wind Attribute

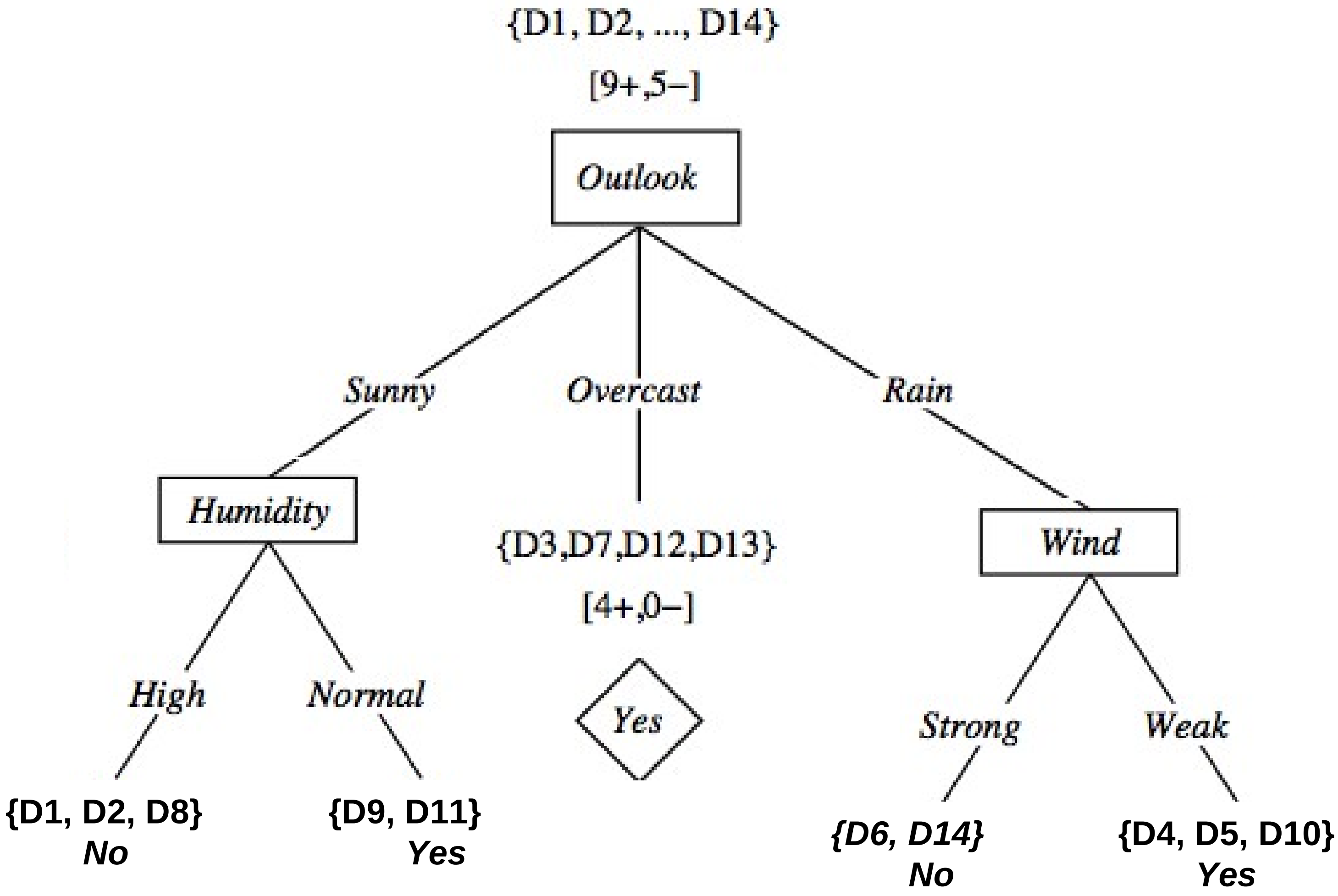
- Let $S = \{D4, D5, D6, D10, D14\}$
- Entropy: $H(S) = -3/5log(3/5) - 2/5log(2/5) = 0.971$
- Information Gain

$$IG(S,Temp) = H(S) - H(S|Temp) = 0.01997$$

$$IG(S,Wind) = H(S) - H(S|Wind)$$

$$= 0.971 - 3/5 \times 0.0 - 2/5 \times 0.0 = 0.971$$

After third step



{D1, D2, ..., D14}

[9+,5−]

Outlook

Sunny *Overcast* *Rain*

Humidity

{D3,D7,D12,D13}

[4+,0−]

Yes

Wind

High *Normal*

Strong *Weak*

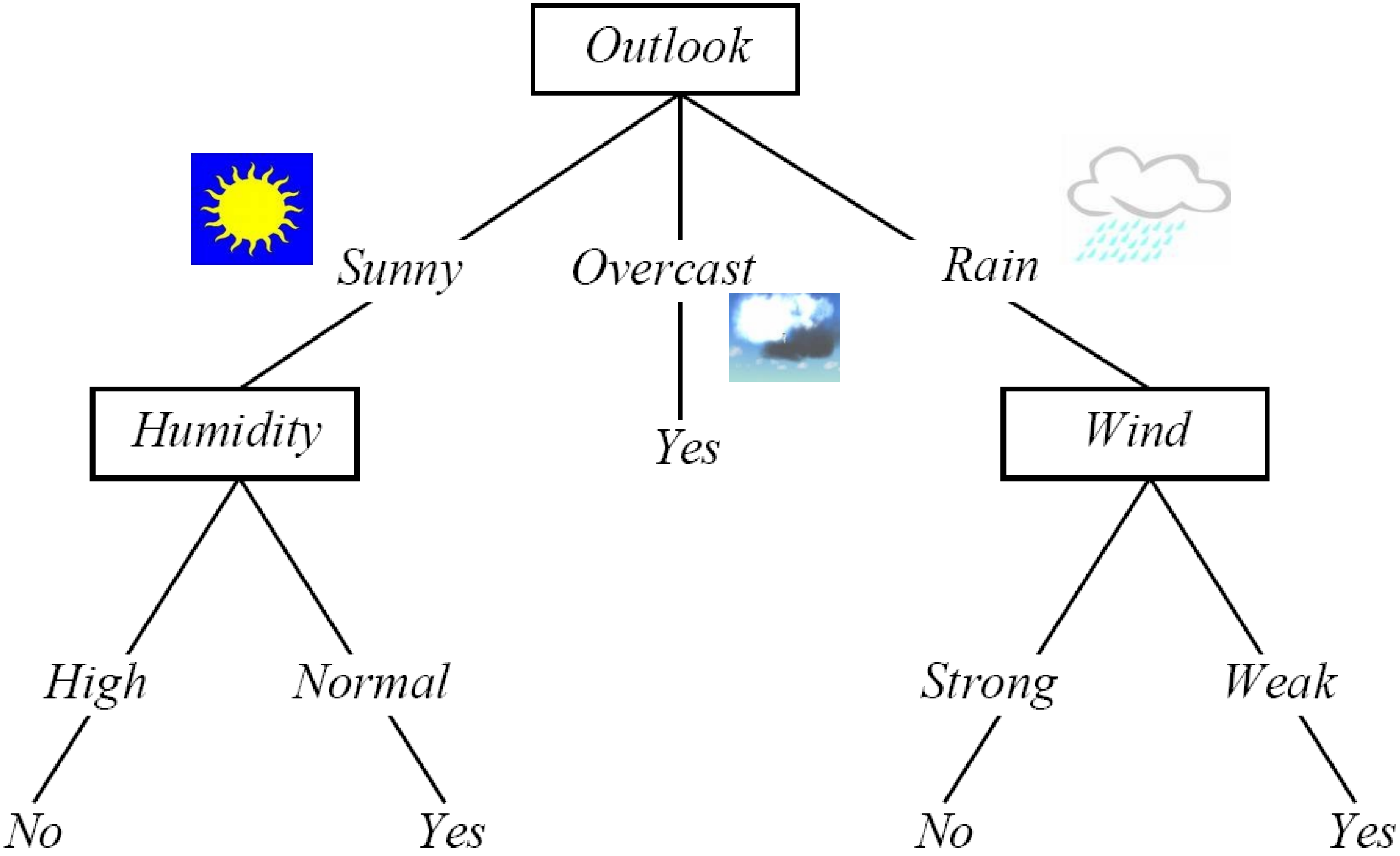
{D1, D2, D8}
No

{D9, D11}
Yes

{D6, D14}
No

{D4, D5, D10}
Yes

Final descision tree



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
```

Acknowledgement

- Machine Learning – Peter Flach
- Notes from Tom Mitchell