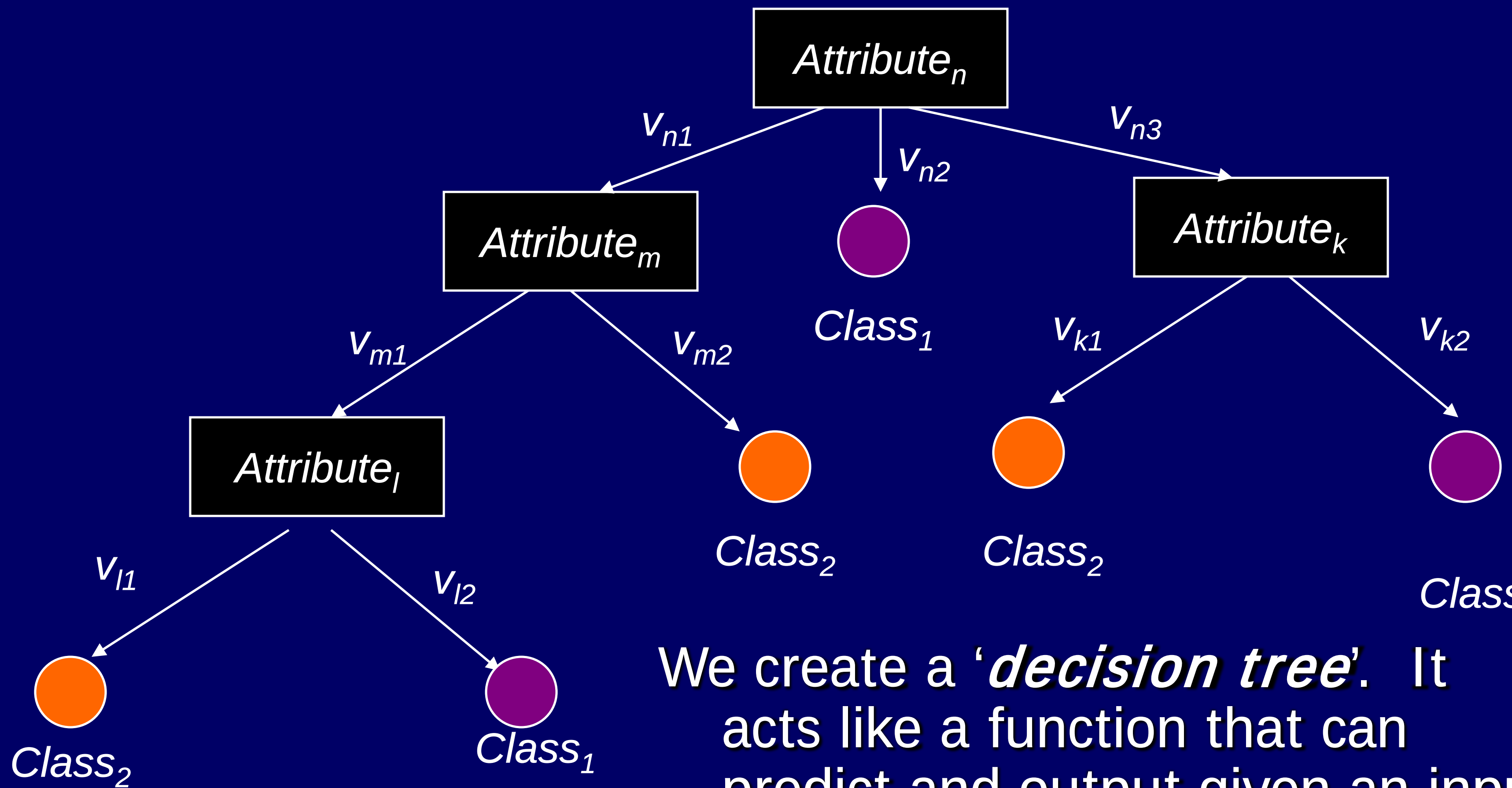
Decision Trees

Defining the Task

- Imagine we've got a set of data containing several types, or ***classes***.
  - E.g. information about customers, and class=whether or not they buy anything.

- Can we predict, i.e ***classify***, whether a previously unseen customer will buy something?

An Example Decision Tree



We create a '***decision tree***'. It acts like a function that can predict and output given an input

Decision Trees

- The idea is to ***ask a series of questions***, starting at the root, that will lead to a leaf node.
- The ***leaf node provides the classification***.

Decision Trees

Consider these data:

A number of examples of weather, for several days, with a classification 'PlayTennis.'

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

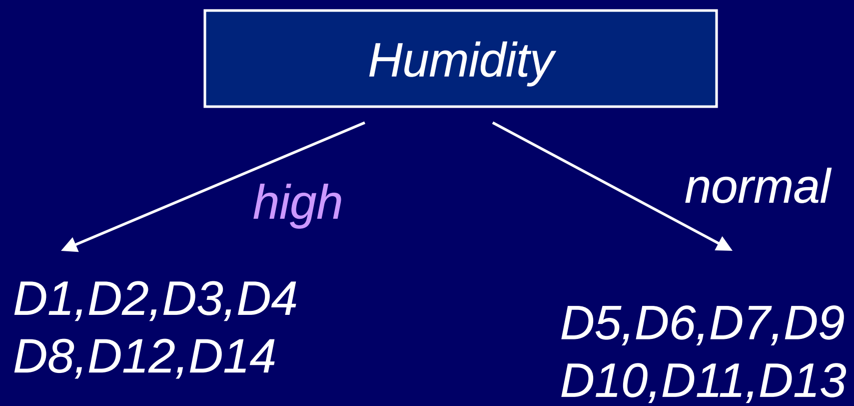
Decision Tree Algorithm

Building a decision tree

1. Select an attribute
2. Create the subsets of the example data for each value of the attribute
3. For each subset
   - *if not all the elements of the subset belongs to same class repeat the steps 1-3 for the subset*

Building Decision Trees

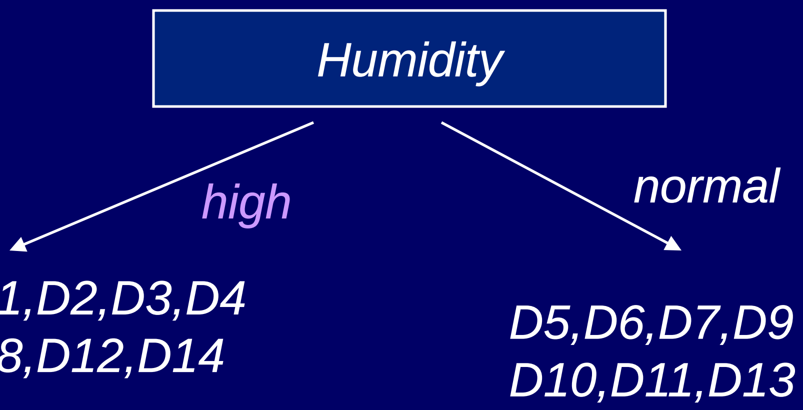
*Let's start building the tree from scratch. We first need to decide which attribute to make a decision. Let's say we selected "**humidity**"*



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

Building Decision Trees

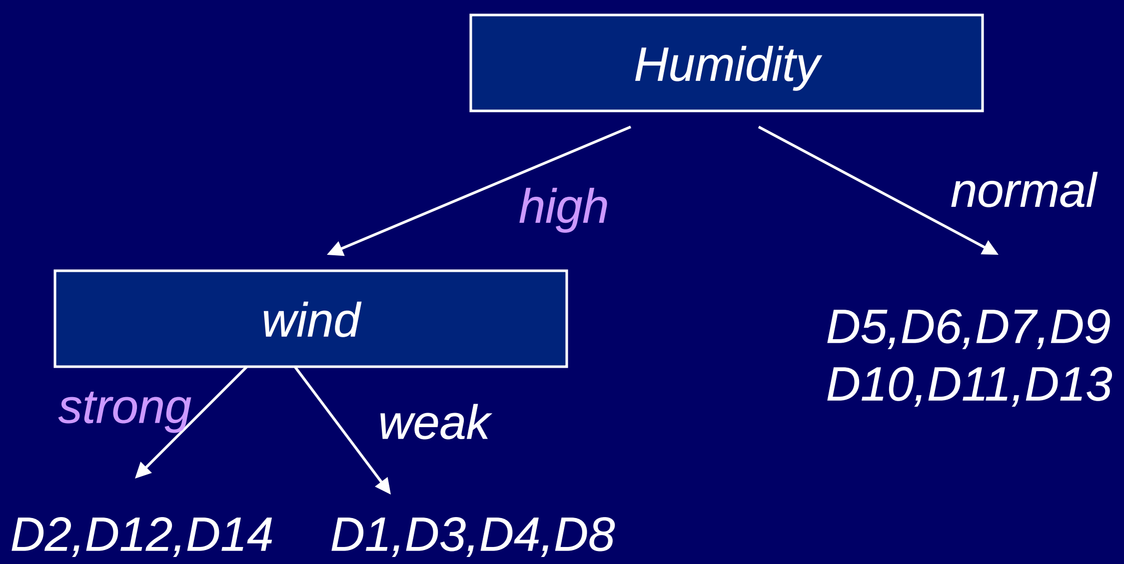
*Now lets classify the first subset D1,D2,D3,D4,D8,D12,D14 using attribute "**wind**"*



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
D8	Sunny	Mild	High	Weak	No
D12	Overcast	Mild	High	Strong	Yes
D14	Rain	Mild	High	Strong	No

Building Decision Trees

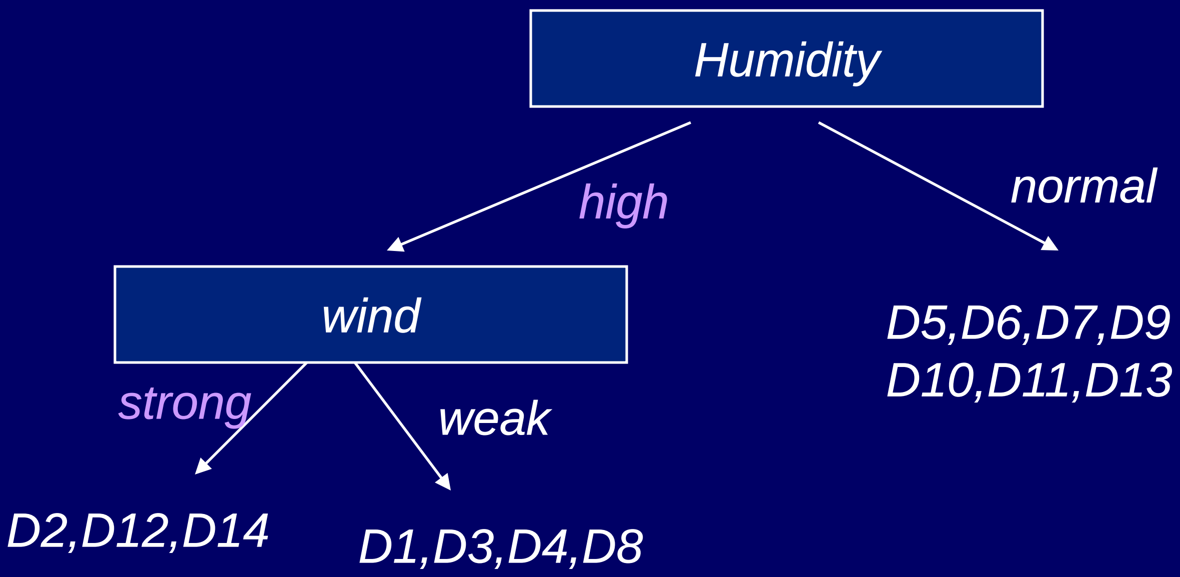
*Subset D1,D2,D3,D4,D8,D12,D14 classified by attribute "**wind**"*



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
D8	Sunny	Mild	High	Weak	No
D12	Overcast	Mild	High	Strong	Yes
D14	Rain	Mild	High	Strong	No

Building Decision Trees

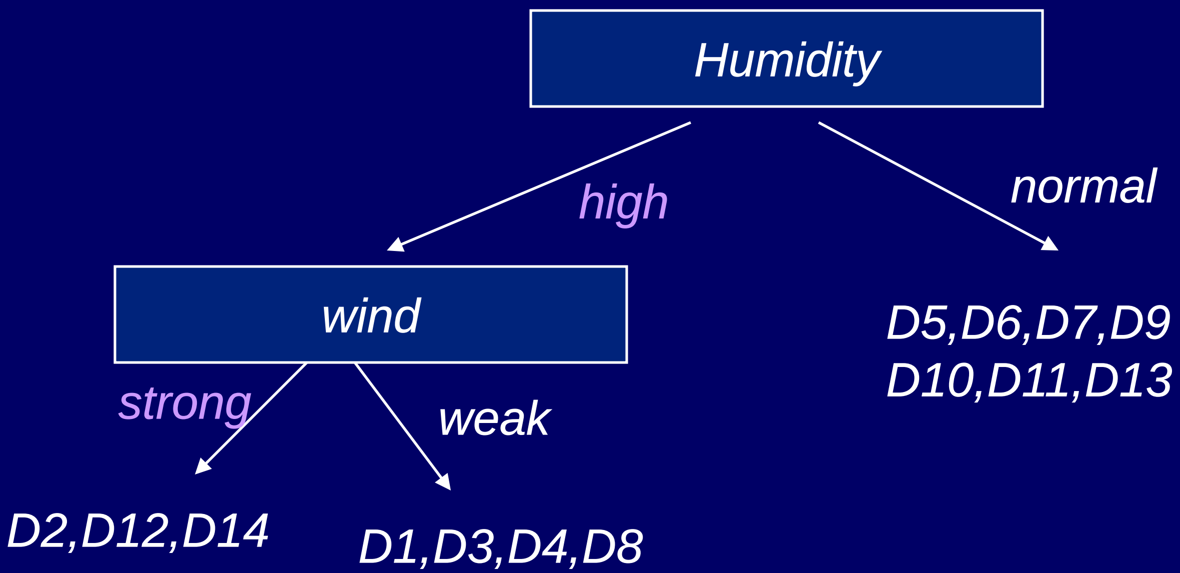
*Now lets classify the subset D2,D12,D14 using attribute "**outlook**"*



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D2	Sunny	Hot	High	Strong	No
D12	Overcast	Mild	High	Strong	Yes
D14	Rain	Mild	High	Strong	No

Building Decision Trees

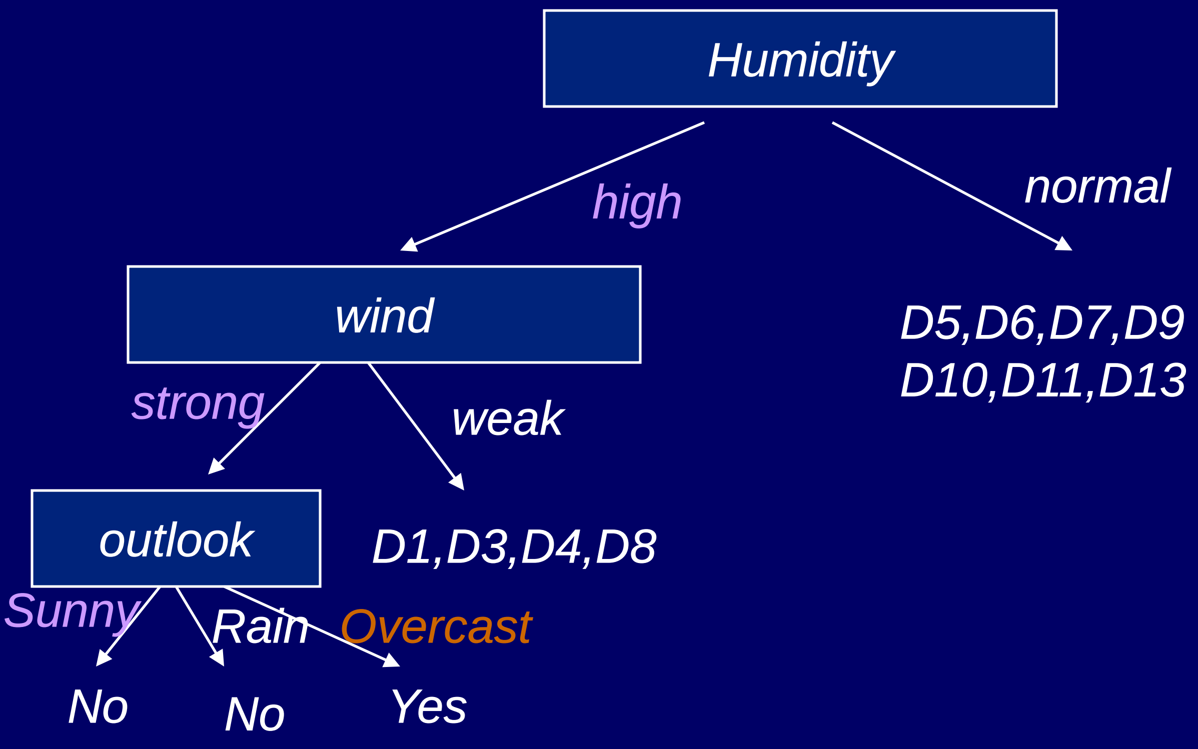
*Subset D2,D12,D14 classified by “**outlook**”*



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D2	Sunny	Hot	High	Strong	No
D12	Overcast	Mild	High	Strong	Yes
D14	Rain	Mild	High	Strong	No

Building Decision Trees

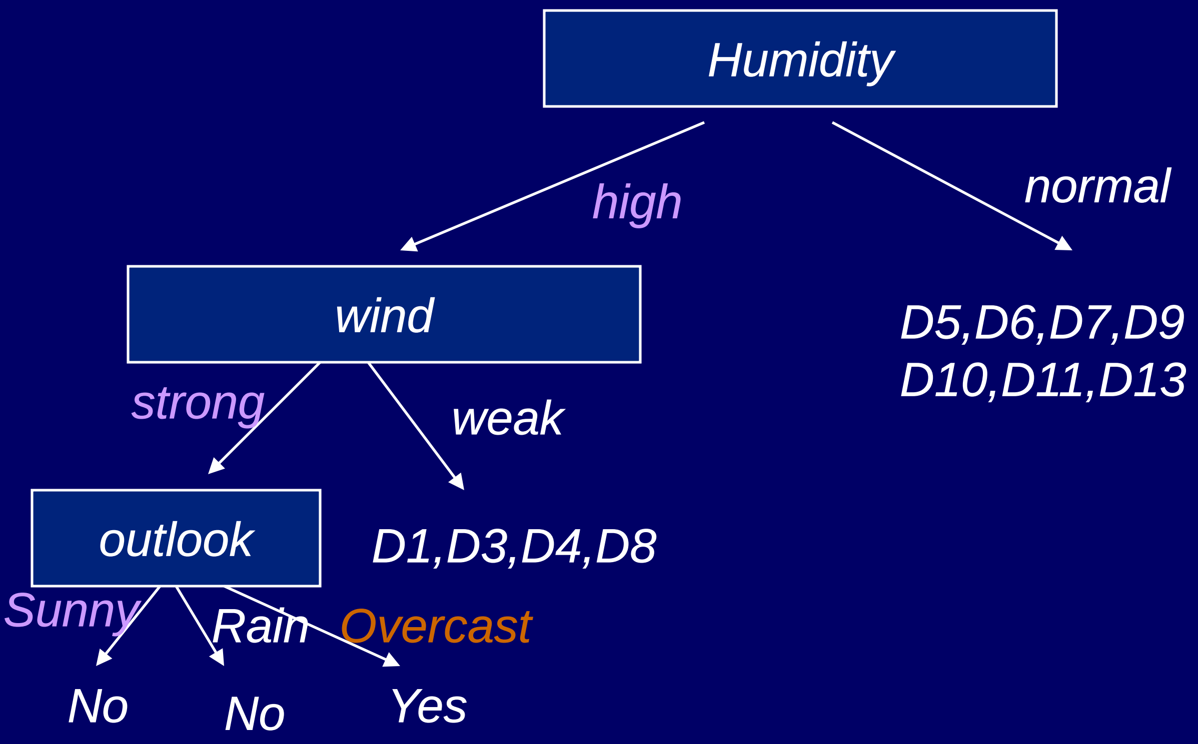
*subset D2,D12,D14 classified using attribute "**outlook**"*

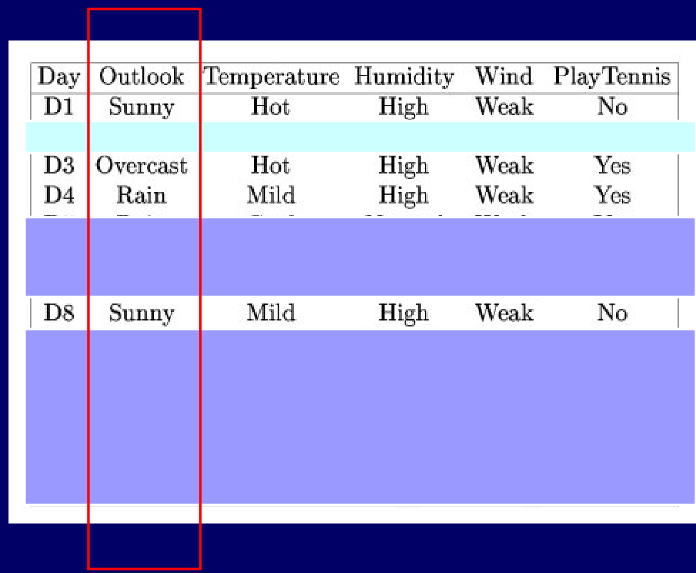


Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D2	Sunny	Hot	High	Strong	No
D12	Overcast	Mild	High	Strong	Yes
D14	Rain	Mild	High	Strong	No

Building Decision Trees

*Now lets classify the subset D1,D3,D4,D8 using attribute "**outlook**"*

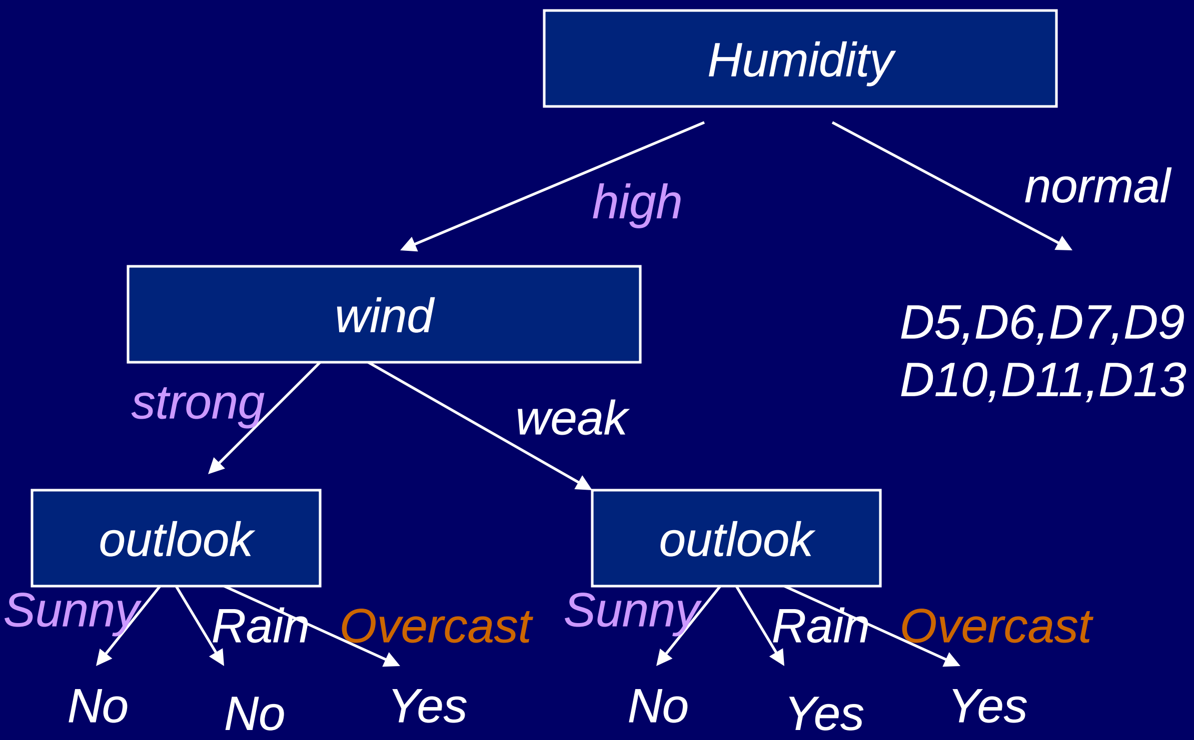




Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D8	Sunny	Mild	High	Weak	No

Building Decision Trees

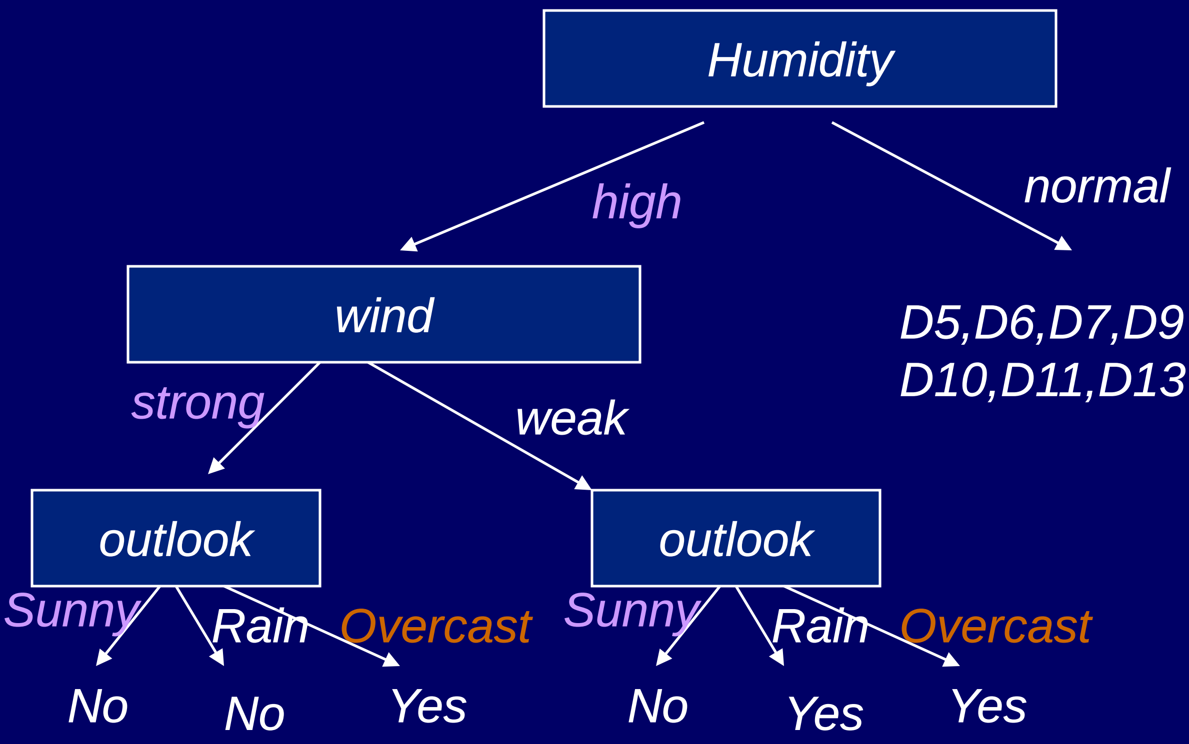
*subset D1,D3,D4,D8 classified by “**outlook**”*



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D8	Sunny	Mild	High	Weak	No

Building Decision Trees

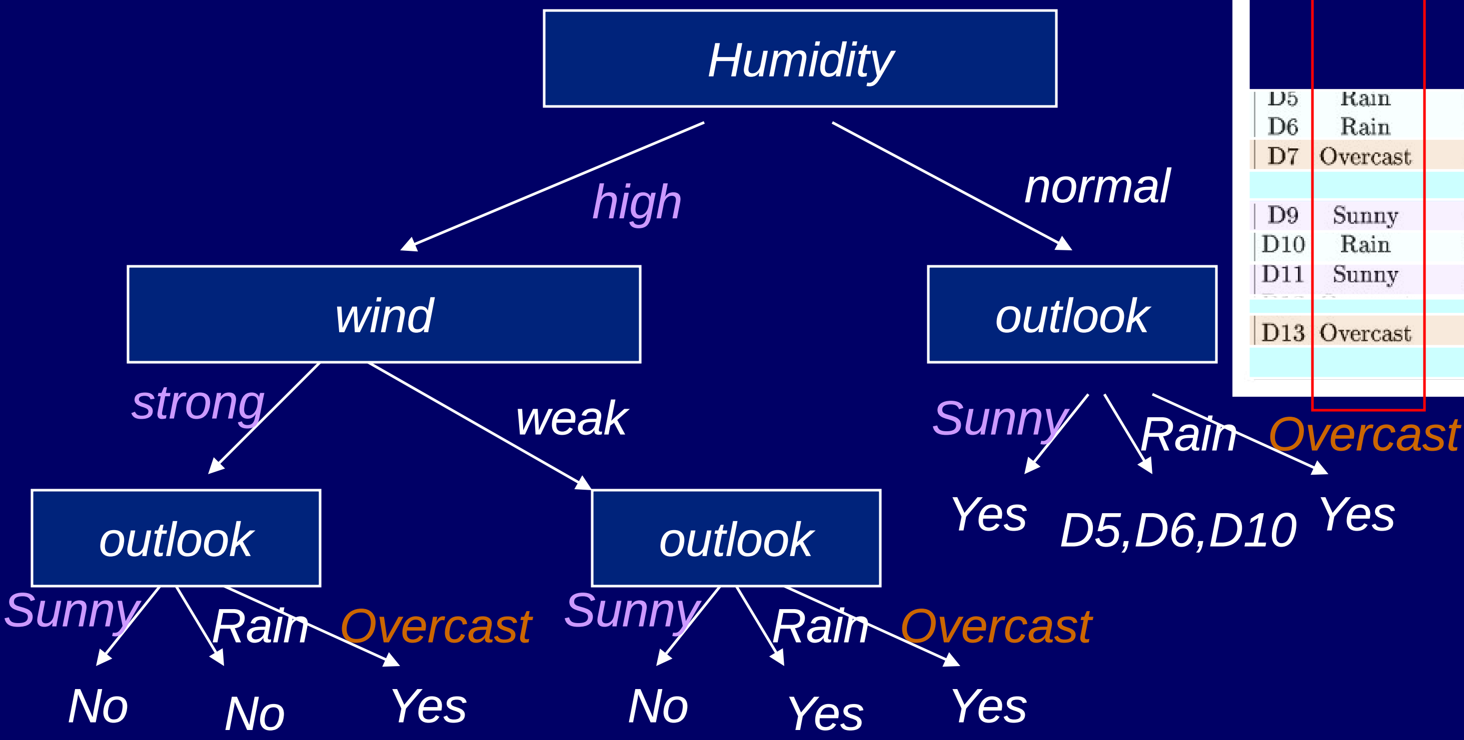
*Now classify the subset D5,D6,D7,D9,D10,D11,D13 using attribute "**outlook**"*



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
[redacted]					
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes

Building Decision Trees

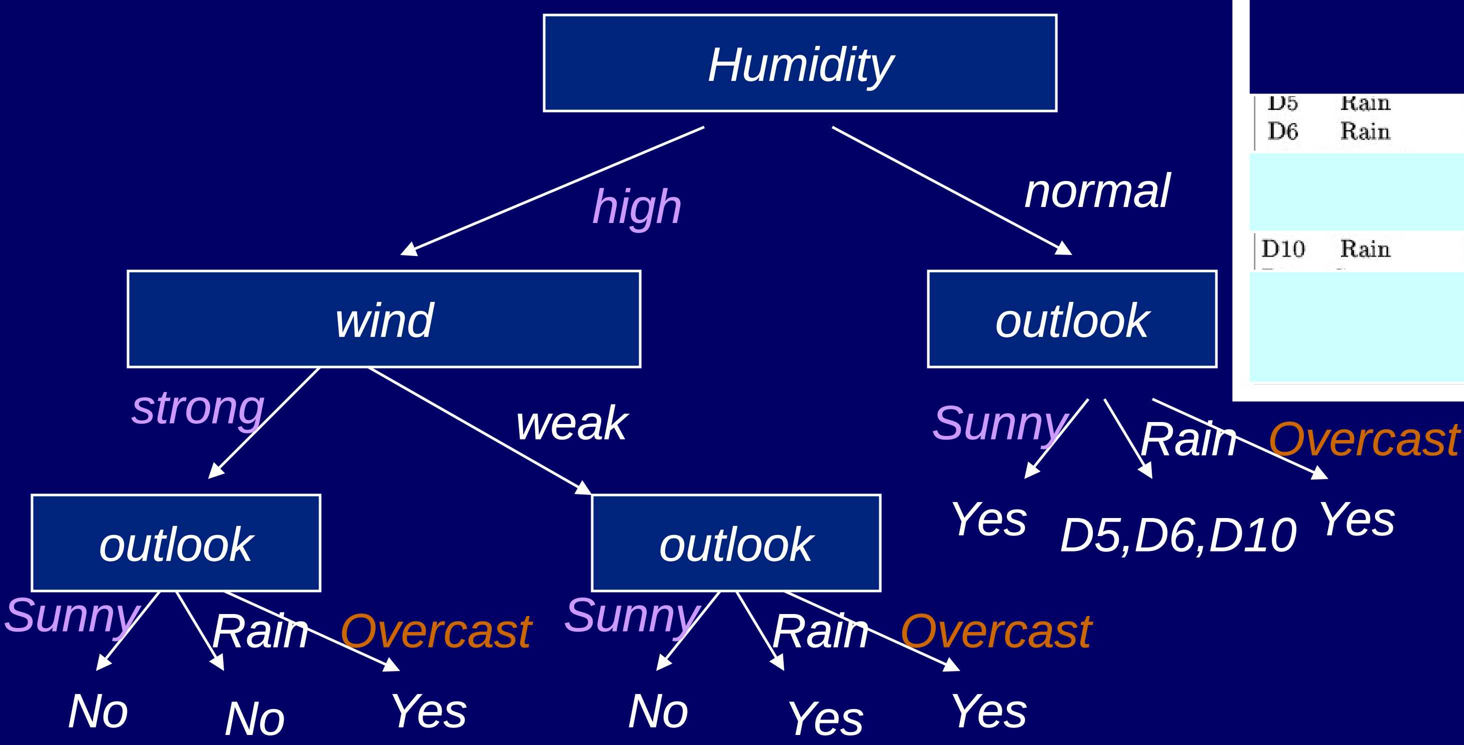
*subset D5,D6,D7,D9,D10,D11,D13 classified by "**outlook**"*



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
[redacted]					
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes

Building Decision Trees

*Finally classify subset D5,D6,D10by “**wind**”*

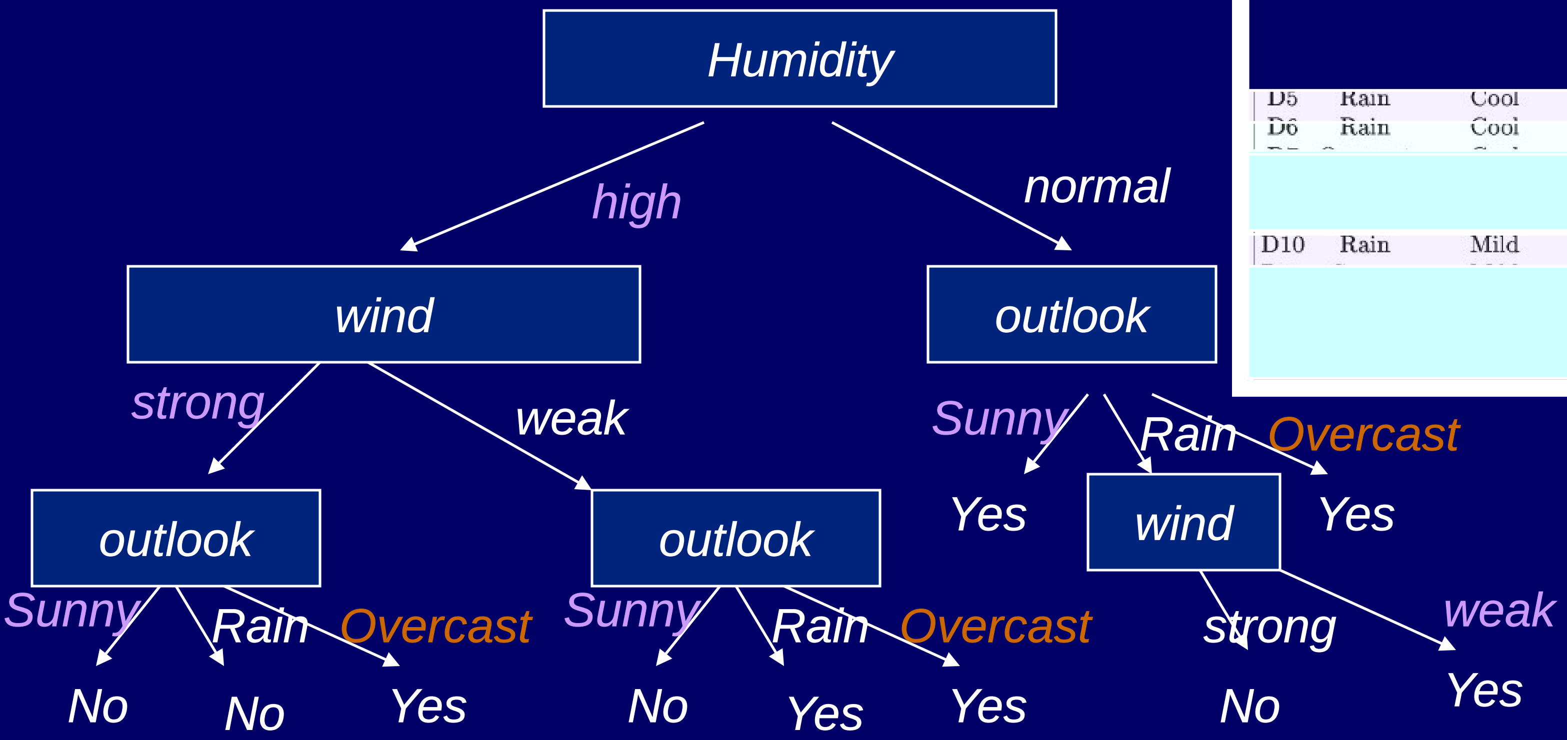


Day	Outlook	Temperature	Humidity	Wind	PlayTennis
[redacted]					
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes

Building Decision Trees

*subset D5,D6,D10 classified by "**wind**"*

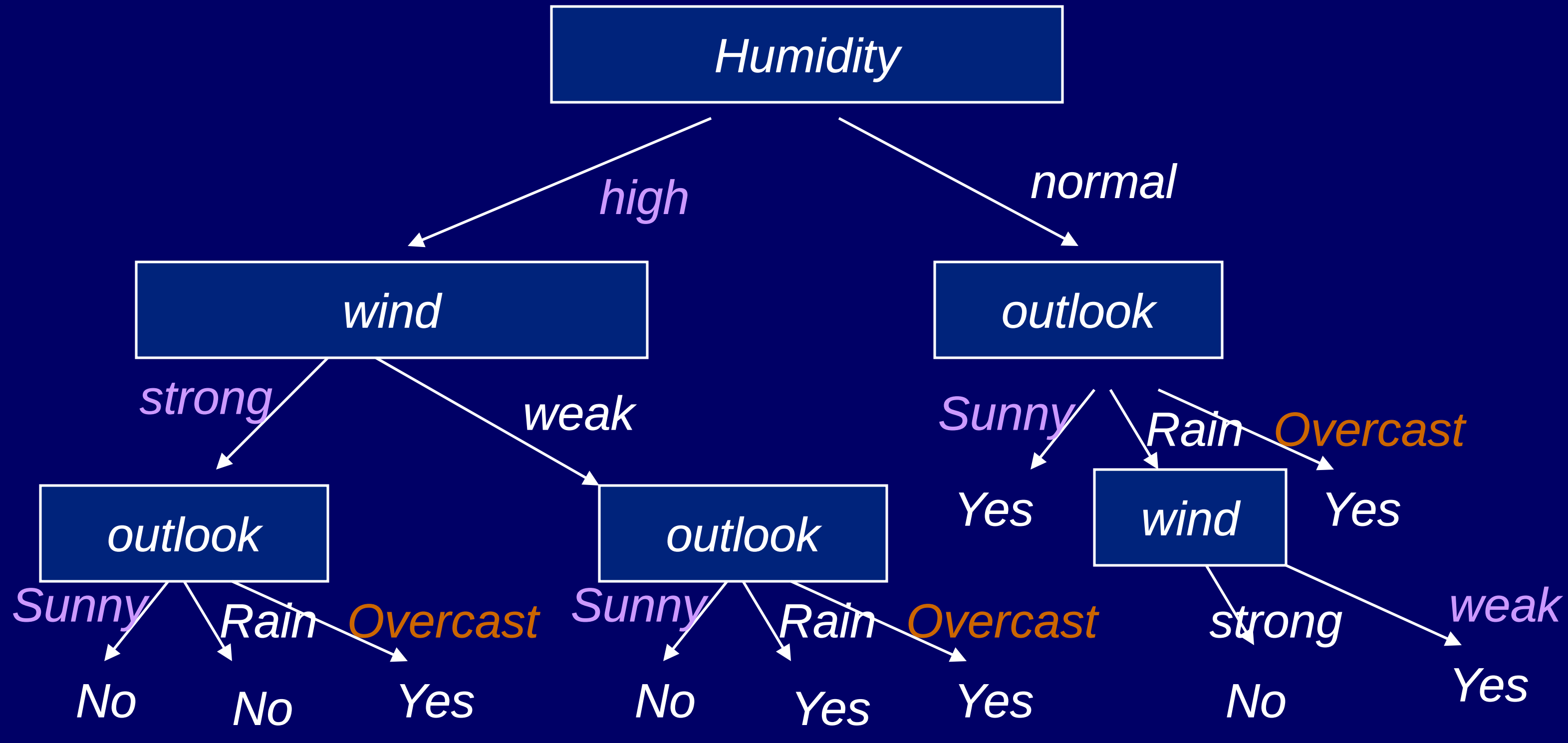
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes



Decision Trees and Logic

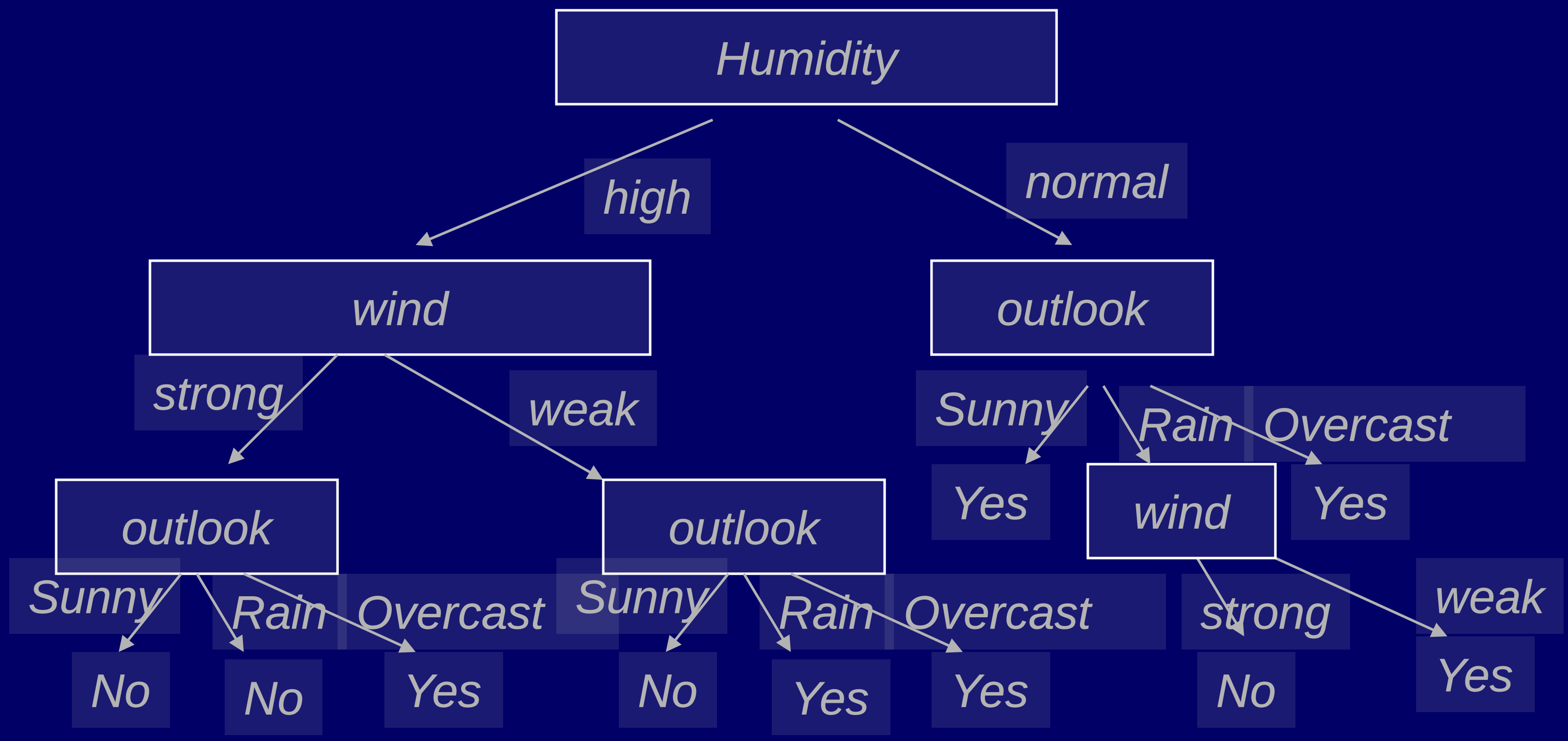
The decision tree can be expressed as an expression or if-then-else sentences:

(humidity=high $\wedge$ wind=strong $\wedge$ outlook=overcast) $\vee$
(humidity=high $\wedge$ wind=weak $\wedge$ outlook=overcast) $\vee$
(humidity=normal $\wedge$ outlook=sunny) $\vee$
(humidity=normal $\wedge$ outlook=overcast) $\vee$
(humidity=normal $\wedge$ outlook=rain $\wedge$ wind=weak) $\Rightarrow$ 'Yes'



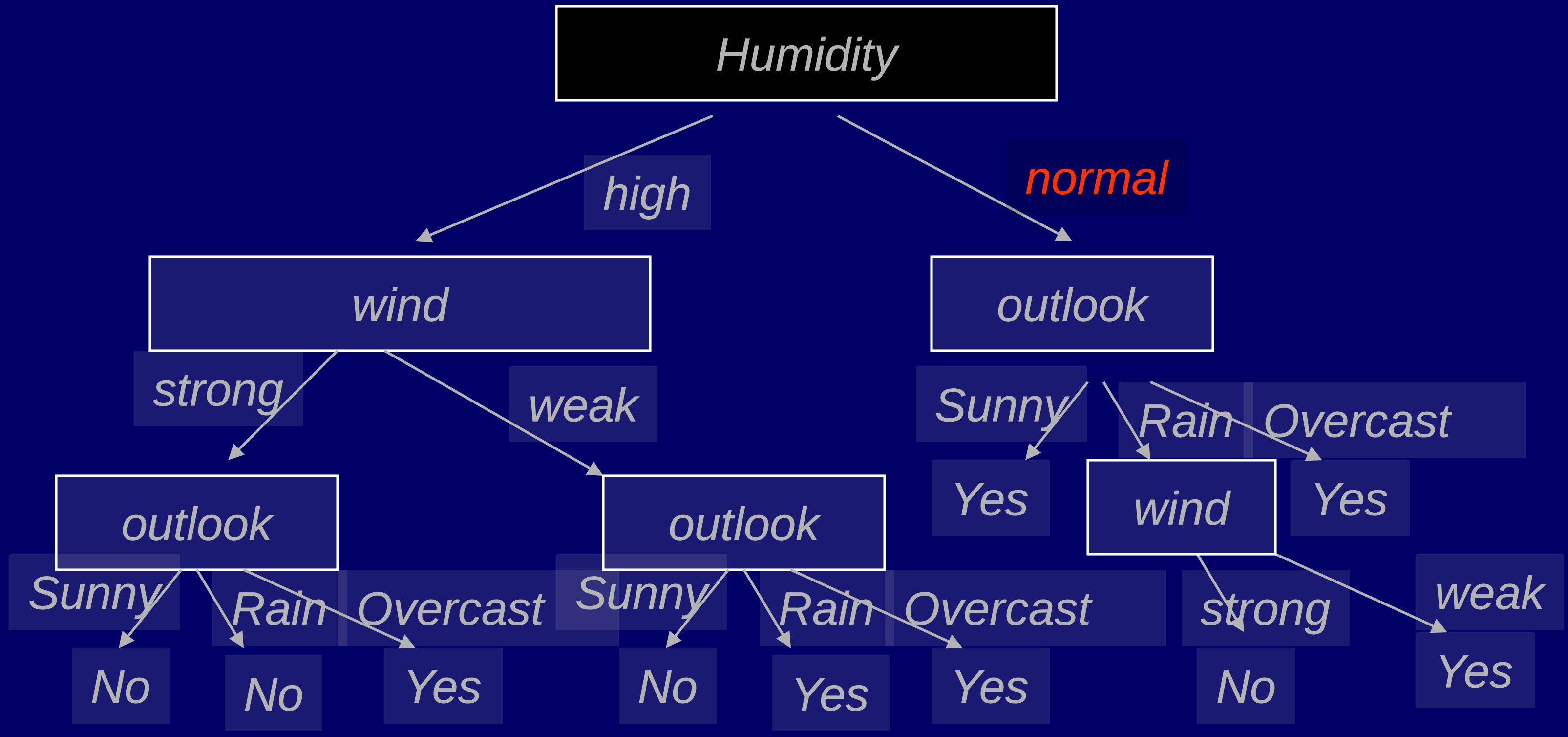
Using Decision Trees

Now let's classify an unseen example: <sunny,hot,normal,weak>=?



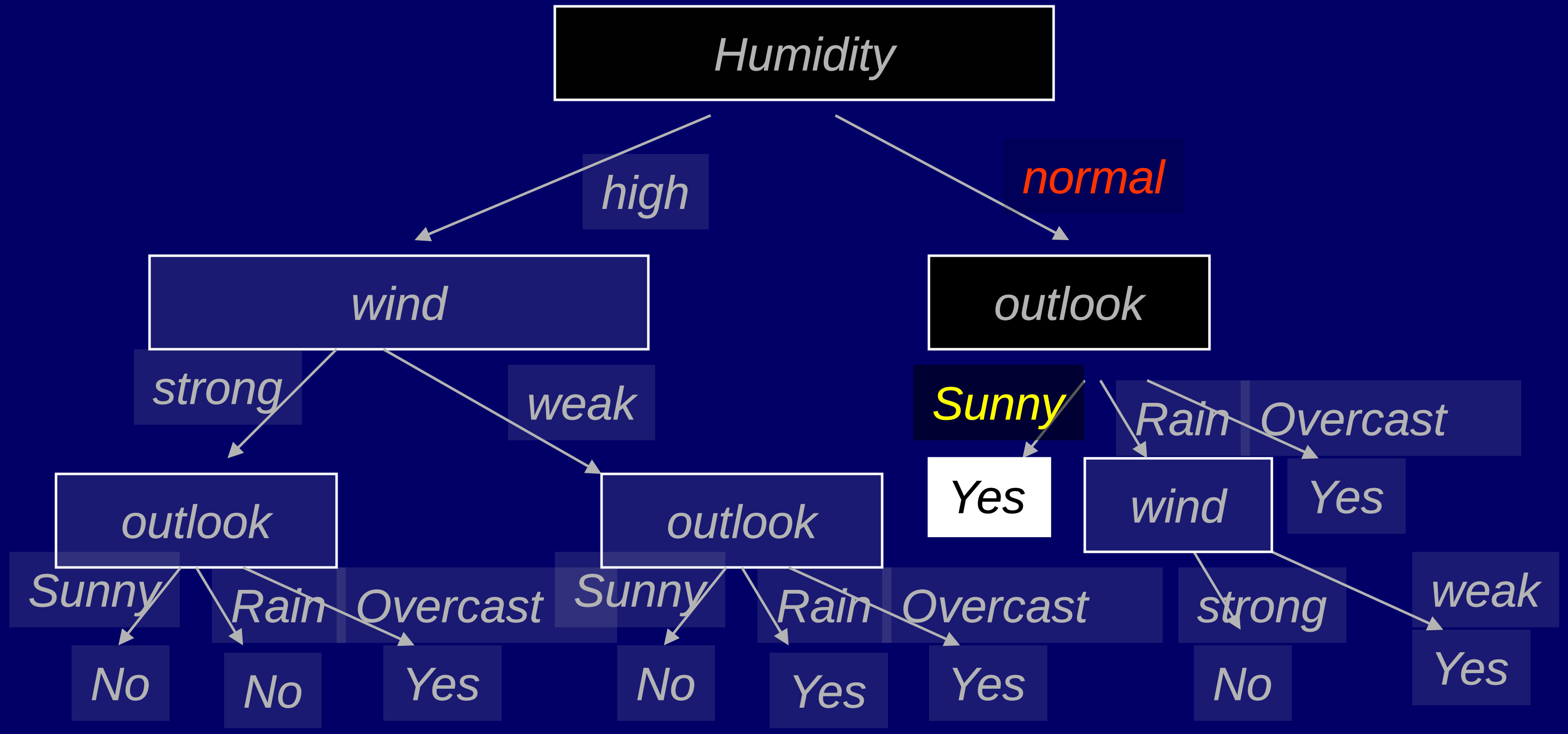
Using Decision Trees

Classifying: <sunny,hot,normal,weak>=?



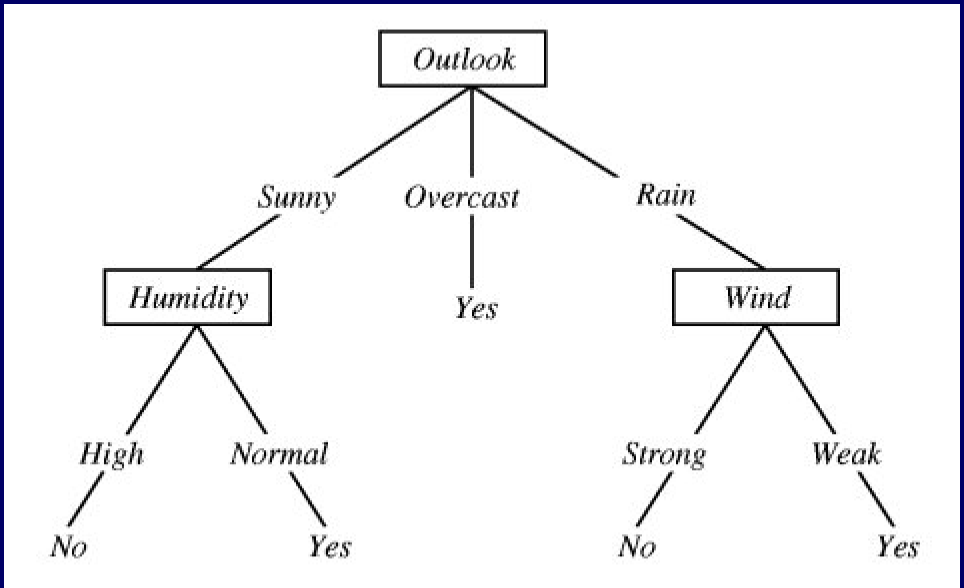
Using Decision Trees

Classification for: <sunny,hot,normal,weak>=Yes



A Big Problem…

Here's another tree from the ***same training data*** that has a different attribute order:



Which attribute should we choose for each branch?

Choosing Attributes

- We need a way of ***choosing the best attribute*** each time we add a node to the tree.
- Most commonly we use a measure called ***entropy***.
- Entropy measure the degree of ***disorder*** in a set of objects.

Entropy

- In our system we have
  - 9 positive examples
  - 5 negative examples
- The ***entropy, E(S),*** of a set of examples is:
  - $E(S) = \sum_{i=1}^{c} -p_i \log p_i$
  - Where c = no of classes and p_i = ratio of the number of examples of this value over the total number of examples.

- P+ = 9/14
- P- = 5/14
- $E = -\ 9/14 \log_2 9/14 - 5/14 \log_2 5/14$
- E = <u>0.940</u>

- In a ***homogenous*** (totally ordered) system, the entropy is 0.

- In a ***totally heterogeneous*** system (totally disordered), all classes have equal numbers of instances; the entropy is 1

Entropy

- We can evaluate ***each attribute*** for their entropy.
  - E.g. evaluate the attribute "*Temperature*"
  - Three values: 'Hot', 'Mild', 'Cool.'
- So we have three subsets, one for each value of 'Temperature'.

$S_{hot}=\{D1,D2,D3,D13\}$

$S_{mild}=\{D4,D8,D10,D11,D12,D14\}$

$S_{cool}=\{D5,D6,D7,D9\}$

We will now find:

$E(S_{hot})$
$E(S_{mild})$
$E(S_{cool})$

Entropy

S_{hot}= {D1,D2,D3,D13}

Examples:
2 positive
2 negative

Totally heterogeneous + disordered therefore:
$p_+ = 0.5$
$p_- = 0.5$

Entropy(S_{hot}),=
$-0.5\log_2 0.5$
*$-0.5\log_2 0.5$ = **<u>1.0</u>***

S_{mild}= {D4,D8,D10, D11,D12,D14}

Examples:
4 positive
2 negative

Proportions of each class in this subset:
$p_+ = 0.666$
$p_- = 0.333$

Entropy(S_{mild}),=
$-0.666\log_2 0.666$
*$-0.333\log_2 0.333$ = **<u>0.918</u>***

S_{cool}={D5,D6,D7,D9}

Examples:
3 positive
1 negative

Proportions of each class in this subset:
$p_+ = 0.75$
$p_- = 0.25$

Entropy(S_{cool}),=
$-0.25\log_2 0.25$
*$-0.75\log_2 0.75$ = **<u>0.811</u>***

Gain

Now we can compare the entropy of the system ***before*** we divided it into subsets using "Temperature", with the entropy of the system ***afterwards***. This will tell us how good "Temperature" is as an attribute.

The entropy of the system after we use attribute "Temperature" is:

$$(|S_{hot}|/|S|)*E(S_{hot}) + (|S_{mild}|/|S|)*E(S_{mild}) + (|S_{cool}|/|S|)*E(S_{cool})$$

$$(4/14)*1.0 \quad + \quad (6/14)*0.918 \quad + \quad (4/14)*0.811 = \underline{\mathbf{0.9108}}$$

This difference between the entropy of the system before and after the split into subsets is called the ***gain***:

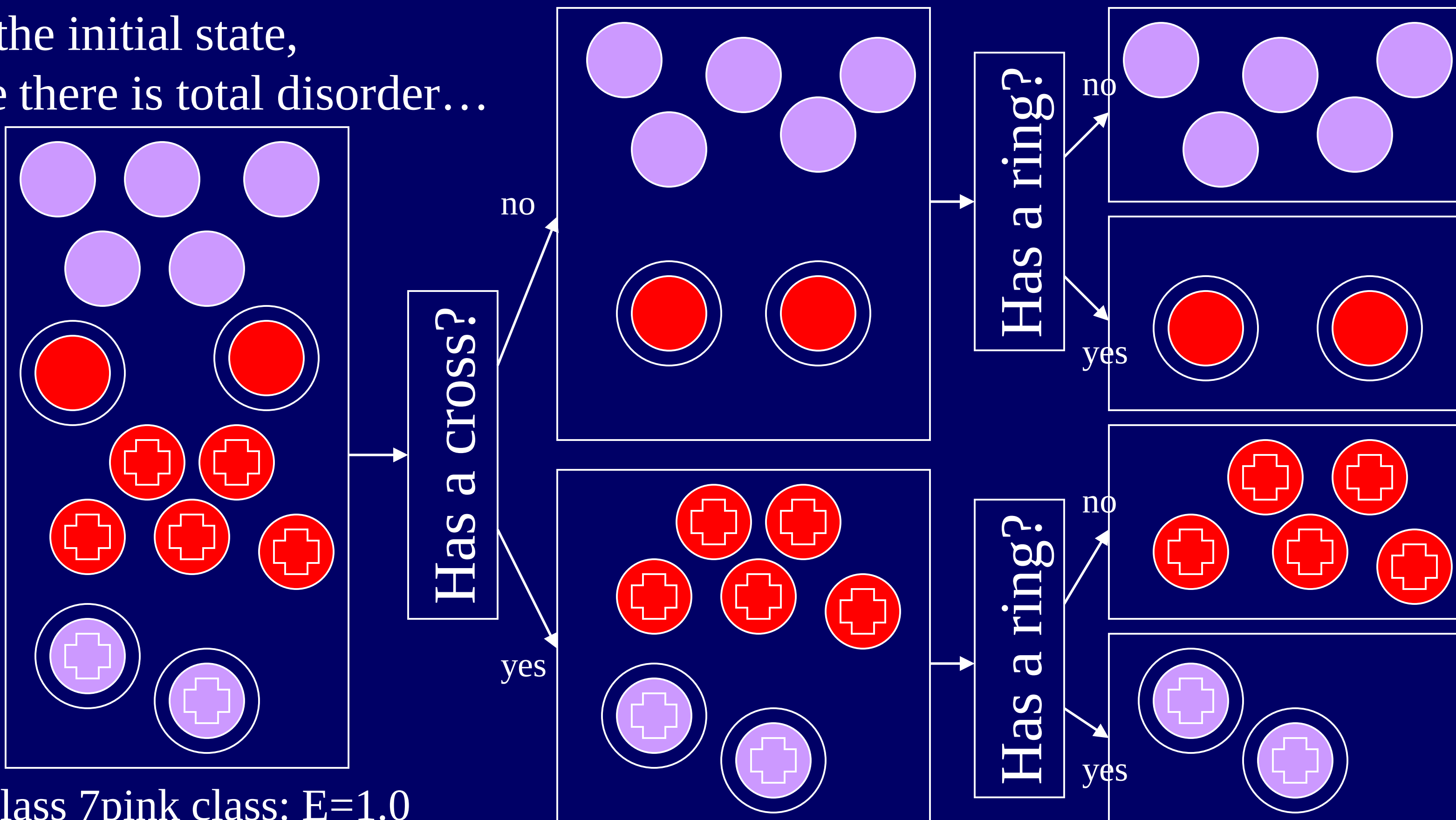
$$\text{Gain}(S, A) = \overbrace{\text{Entropy}(S)}^{\text{E(before)}} - \overbrace{\sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)}^{\text{E(afterwards)}}$$

$$\text{Gain(S,Temperature)} = \quad 0.940 \quad - \quad 0.9108 \quad = \underline{\mathbf{0.029}}$$

Decreasing Entropy

From the initial state,
Where there is total disorder…

…to the final state where all subsets contain a single class



7red class 7pink class: E=1.0

Both subsets
E=-2/7log2/7 –5/7log5/7

All subset: E=0.0

Tabulating the Possibilities

Attribute=value	|+|	|-|	E	E after dividing by attribute A	Gain
Outlook=sunny	2	3	-2/5 log 2/5 – 3/5 log 3/5 = 0.9709	0.6935	0.2467
Outlook=o'cast	4	0	-4/4 log 4/4 – 0/4 log 0/4 = 0.0		
Outlook=rain	3	2	-3/5 log 3/5 – 2/5 log 2/5 = 0.9709		
Temp'=hot	2	2	-2/2 log 2/2 – 2/2 log 2/2 = 1.0	0.9111	0.0292
Temp'=mild	4	2	-4/6 log 4/6 – 2/6 log 2/6 = 0.9183		
Temp'=cool	3	1	-3/4 log 3/4 – 1/4 log 1/4 = 0.8112		
Etc...					

… etc

This shows the entropy calculations…

Table continued…

E for each subset of A	Weight by proportion of total	E after A is the sum of the weighted values	Gain = (E before dividing by A) – (E after A)
-2/5 log 2/5 – 3/5 log 3/5 = **0.9709**	0.9709 x 5/14 = **0.34675**	0.6935	0.2467
-4/4 log 4/4 – 0/4 log 0/4 = **0.0**	0.0 x 4/14 = **0.0**		
-3/5 log 3/5 – 2/5 log 2/5 = **0.9709**	0.9709 x 5/14 = **0.34675**		
-2/2 log 2/2 – 2/2 log 2/2 = **1.0**	1.0 x 4/14 = **0.2857**	0.9111	0.0292
-4/6 log 4/6 – 2/6 log 2/6 = **0.9183**	0.9183 x 6/14 = **0.3935**		
-3/4 log 3/4 – 1/4 log 1/4 = **0.8112**	0.8112 x 4/14 = **0.2317**		

…and this shows the gain calculations

Gain

- We calculate the gain for all the attributes.
- Then we see which of them will bring more ‘order’ to the set of examples.

- Gain(S,Outlook) = **0.246**
- Gain(S,Humidity) = 0.151
- Gain(S,Wind) = 0.048
- Gain(S, Temp’) = 0.029

- The first node in the tree should be the one with the highest value, i.e. ‘*Outlook*’.

ID3 (Decision Tree Algorithm)

- ID3 was the first proper decision tree algorithm to use this mechanism:

  Building a decision tree with ID3 algorithm

  1. *Select the attribute with the most gain*
  2. *Create the subsets for each value of the attribute*
  3. *For each subset*
     1. *if not all the elements of the subset belongs to same class repeat the steps 1-3 for the subset*

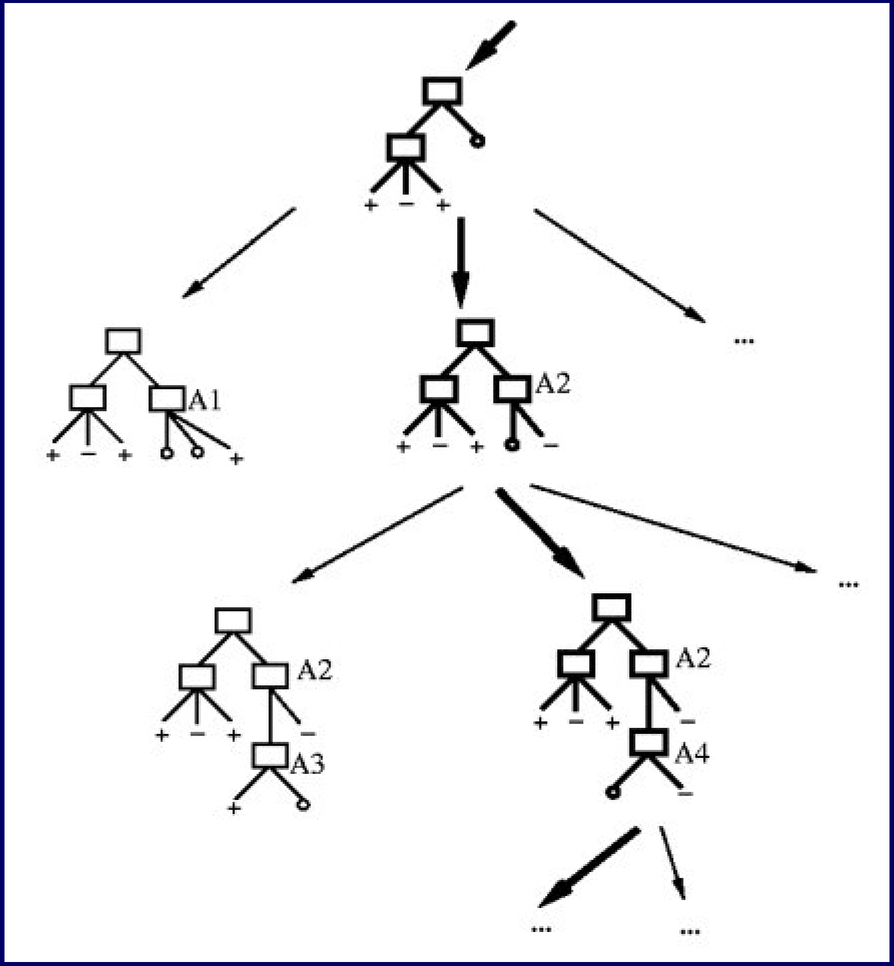
- Or in more detail...

ID3 (Decision Tree Algorithm)

- <u>Function DecisionTtreeLearner(*Examples, TargetClass, Attributes)*</u>

- create a *Root node for the tree*
- **if** all *Examples are positive,* **return** the single-node tree *Root, with label = Yes*
- **if** all *Examples are negative,* **return** the single-node tree *Root, with label = No*
- **if** *Attributes list is empty,*
    - **return** the single-node tree *Root, with label = most common value of TargetClass in Examples*
- **else**
    - *A* = the attribute from *Attributes with the highest information gain with respect to Examples*
    - Make *A the decision attribute for Root*
    - **for** each possible value *v of A:*
        - add a new tree branch below *Root, corresponding to the test A = v*
        - let *Examplesv be the subset of Examples that have value v for attribute A*
        - **if** *Examplesv is empty* **then**
            - add a leaf node below this new branch with label = most common value of *TargetClass in Examples*
        - **else**
            - add the subtree DTL(*Examplesv, TargetClass, Attributes - { A })*
        - **end if**
- **end**
- **return** *Root*

Hypothesis Space

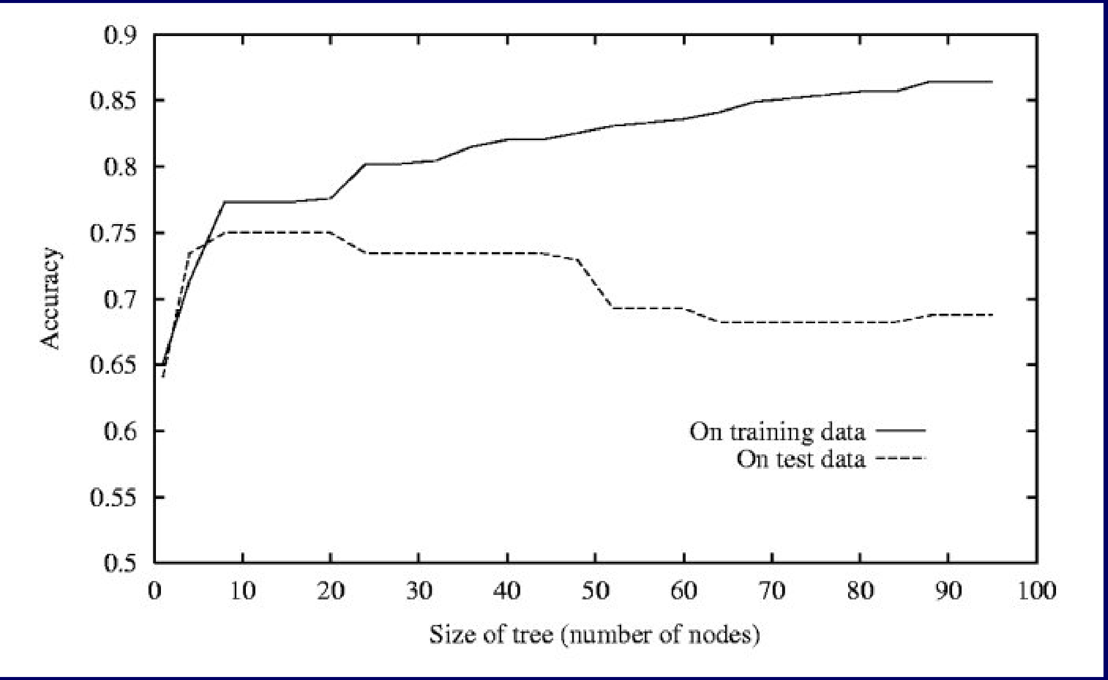


ID3 is a heuristic to select a tree in top-down fashion.

It's a form of Best-first search!

The Problem of Overfitting

- Trees may grow to include ***irrelevant attributes*** (e.g. color of dice).
- ***Noise*** may add ***spurious nodes*** to the tree.
- This can cause ***overfitting*** of the ***training data*** relative to ***test data.***



Hypothesis *H* **overfits** the data if there exists *H'* with greater error than *H*, over training examples, but less error than *H* over entire distribution of instances.

Fixing Overfitting

Two approaches to pruning

Prepruning: Stop growing tree during the training when it is determined that there is not enough data to make reliable choices.

Postpruning: Grow whole tree but then remove the branches that do not contribute good overall performance.

Determining which branch to prune

Cross-validation: Reserve some of the training data(validation set) to evaluate the performance of the tree.

Statistical testing: Use a statistical test (chi-square) to determine if going below a branch is likely to produces an improvement behind the training set (a good explanation of chi-square is at. http://www.physics.csbsju.edu/stats/chi-square.html)

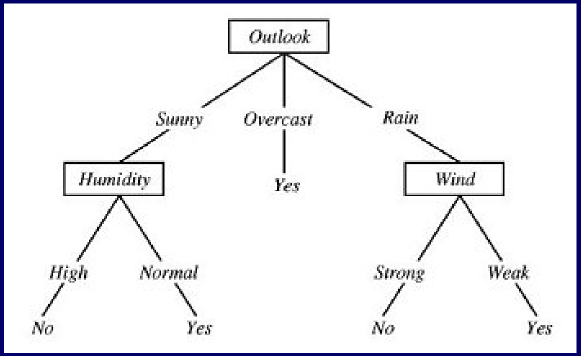
Minimum Description Length: Use an explicit measure (MDL) to determine if the additional complexity of the hypothesis is less complex than just remembering any exceptions.

Reduced-Error Pruning

Reduced-error pruning

1. divide data into **training set** and **validation set**
2. build a decision tree using the training set, allowing overfitting to occur
3. for each node in tree
   - consider effect of removing subtree rooted at node, making it a leaf, and assigning it majority classification for examples at that node
   - note performance of pruned tree over validation set
4. remove node that most improves accuracy over validation set
5. repeat steps 3 and 4 until further pruning is harmful

Rule Post-Pruning



Rule post-pruning

•infer decision tree, allowing overfitting to occur
•convert tree to a set of rules (one for each path in tree)
•prune (generalize) each rule by removing any preconditions (*i.e., attribute tests) that result in improving its accuracy over the validation set*
•sort pruned rules by accuracy, and consider them in this order when classifying subsequent instances

•**IF (Outlook = Sunny) ^ (Humidity = High) THEN PlayTennis = No**
•Try removing **(Outlook = Sunny)** condition or **(Humidity = High)** condition from the rule and select whichever pruning step leads to the biggest improvement in accuracy on the validation set (or else neither if no improvement results).

•**advantages of rule representation:**
•each path's constraints can be pruned independently of other paths in the tree
•removes distinction between attribute tests that occur near the root and those that occur later in the tree
•converting to rules improves readability