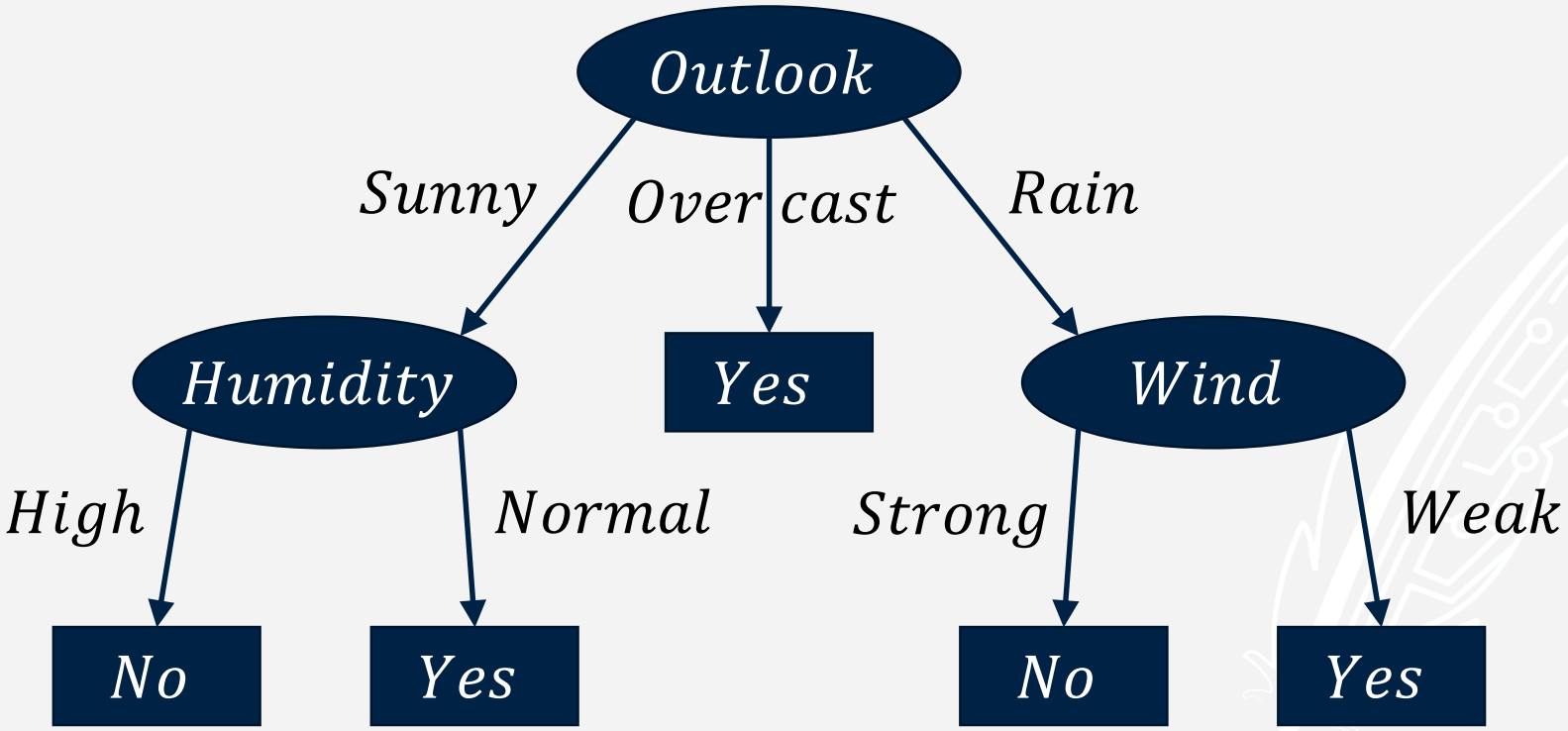
Decision Tree Induction

Stephen G. Ware

CSCI 4525 / 5525

Day	Outlook	Temperature	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	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



Decision Tree

Given a data set with some number of attributes and a class label, a decision tree is a directed tree such that:

- Each non-leaf node represents an attribute.
- Each non-leaf node has an outgoing edge labeled with one possible value of that attribute.
- Each leaf node represents a value of the class label.

Decision Tree Classification

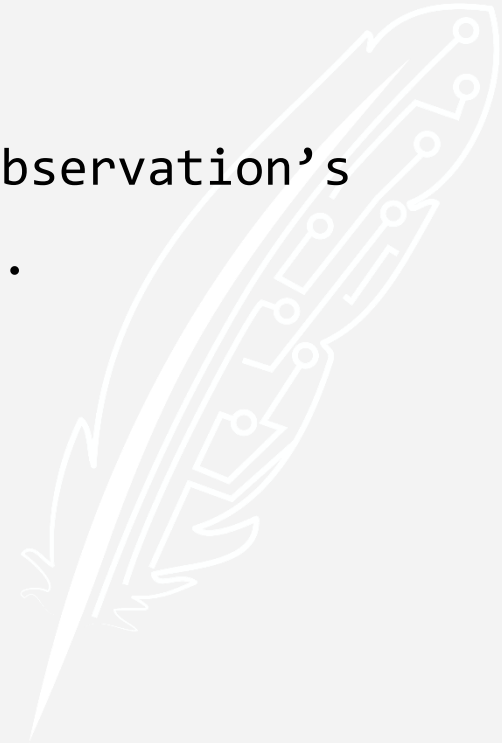
To classify an observation using a decision tree:

Start at the root node.

Until a leaf node is reached:

    Follow the edge corresponding to the observation's value for the current node's attribute.

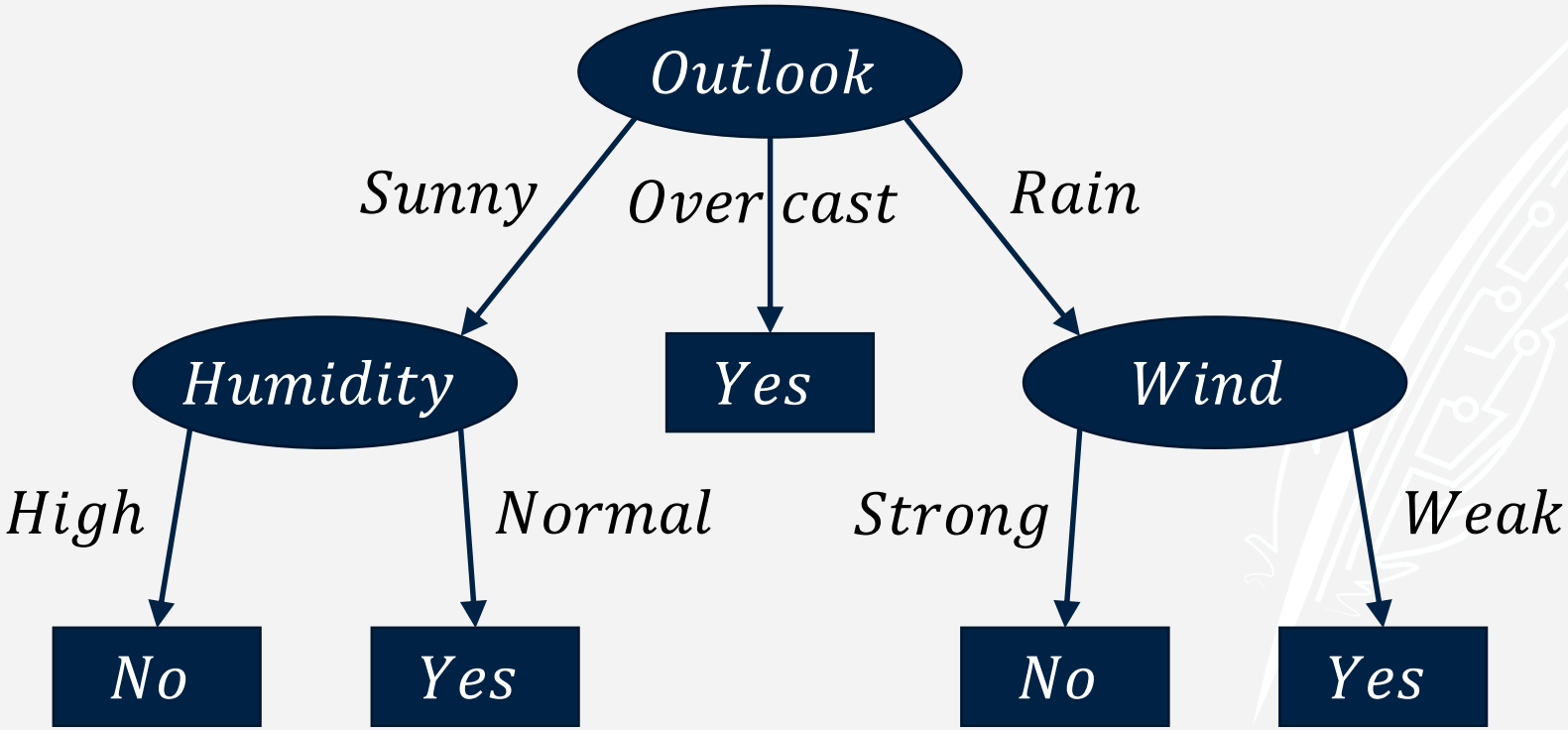
Return the value of the leaf node.



Decision Tree

Classify:

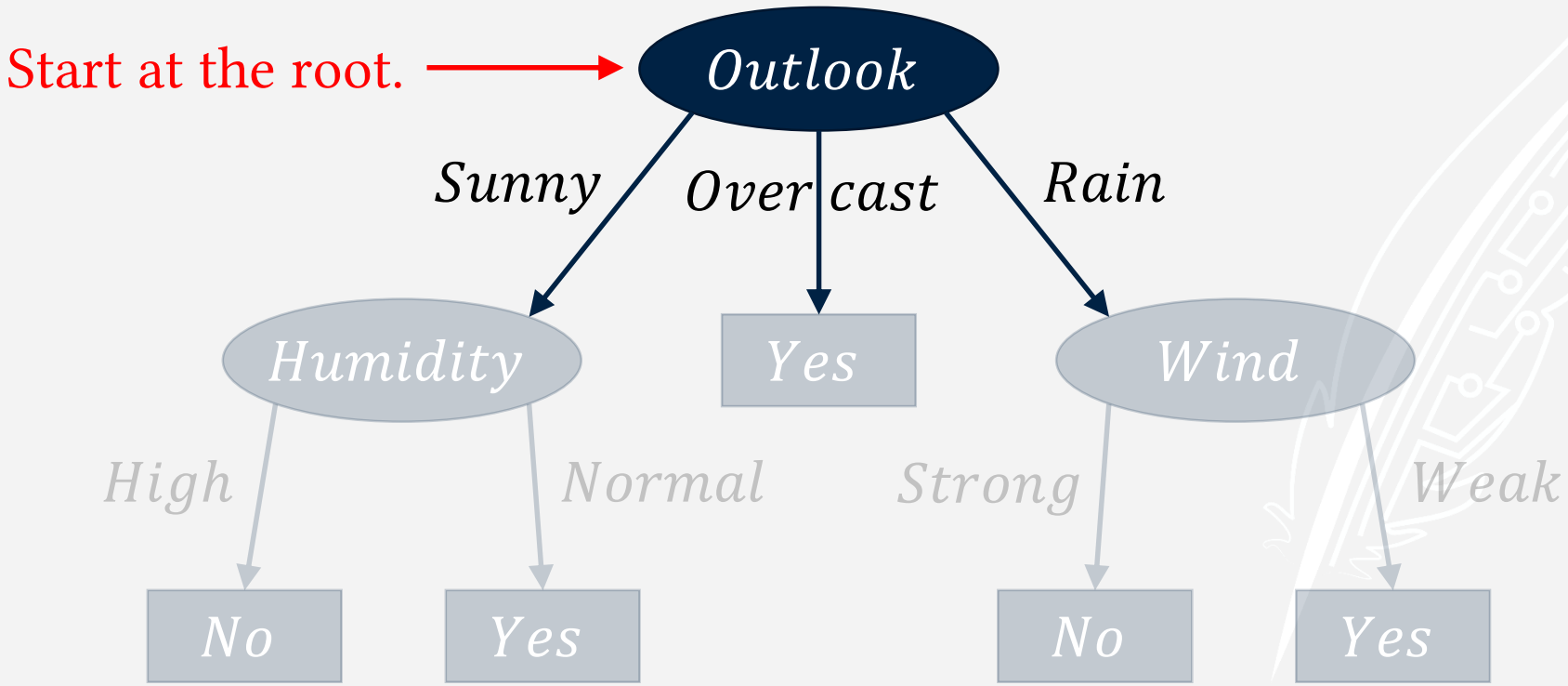
Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Mild	Normal	Strong	?



Decision Tree

Classify:

Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Mild	Normal	Strong	?

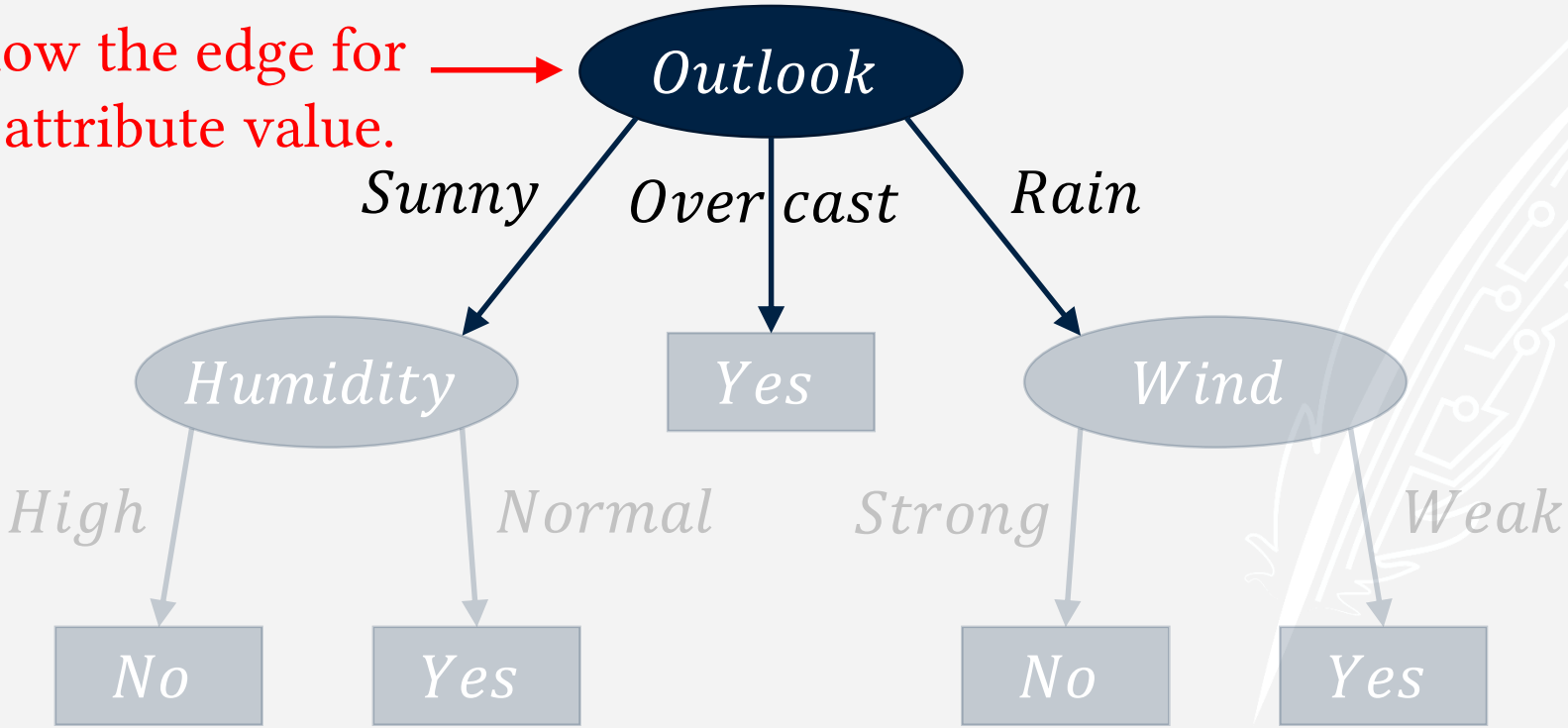


Decision Tree

Classify:

Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Mild	Normal	Strong	?

Follow the edge for the attribute value.

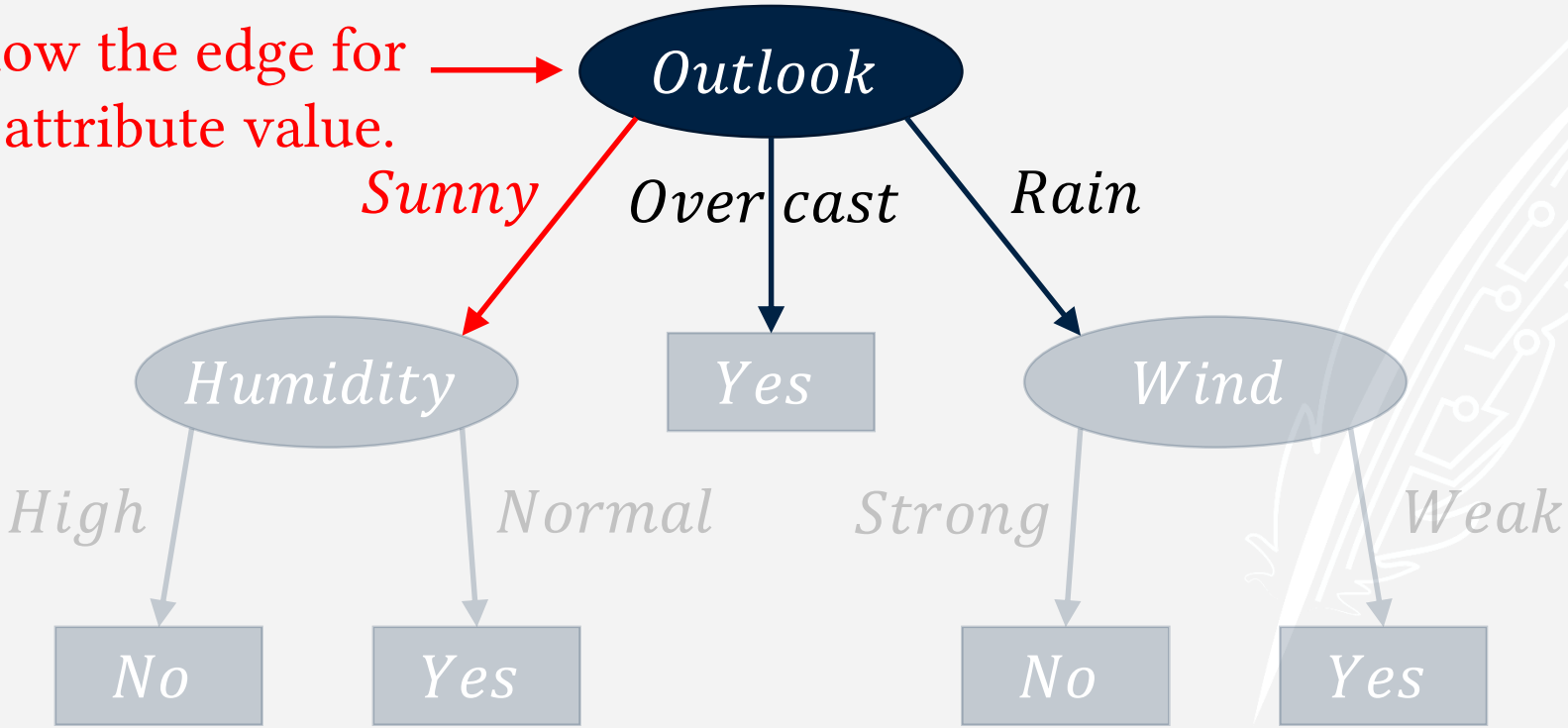


Decision Tree

Classify:

Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Mild	Normal	Strong	?

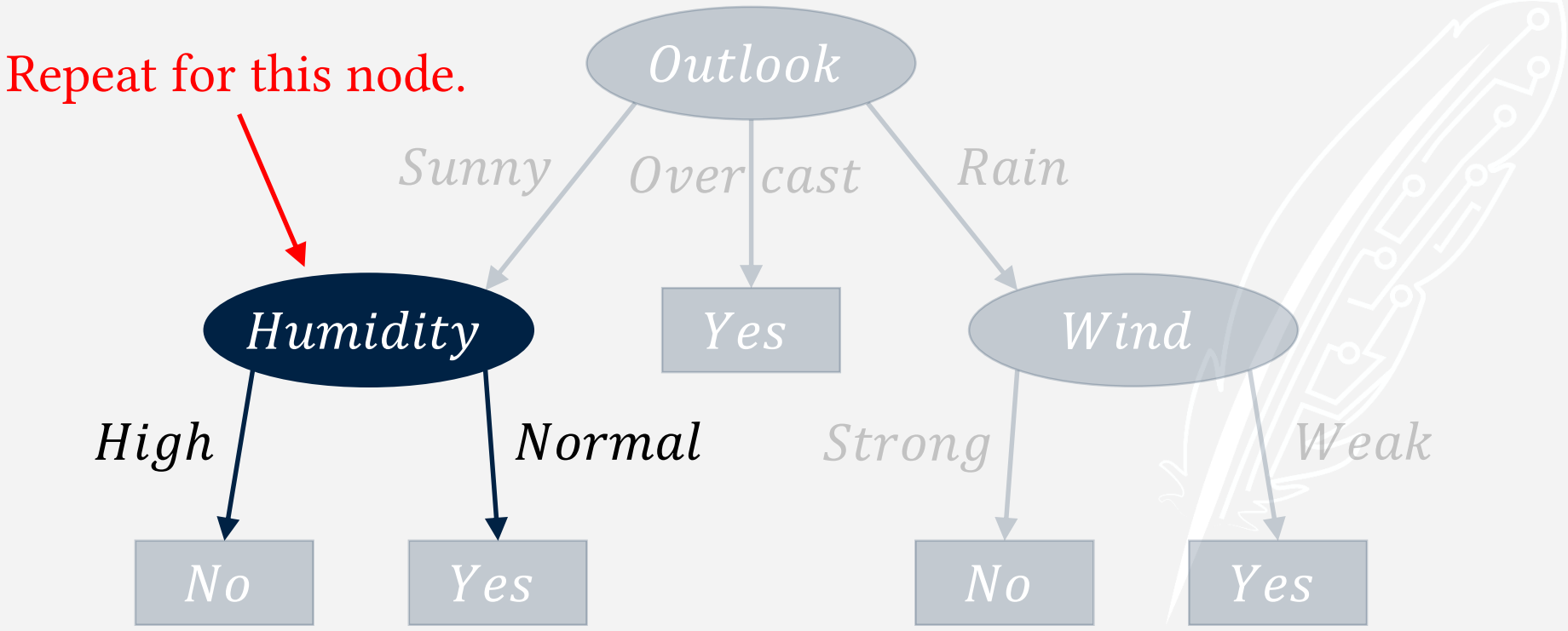
Follow the edge for the attribute value.



Decision Tree

Classify:

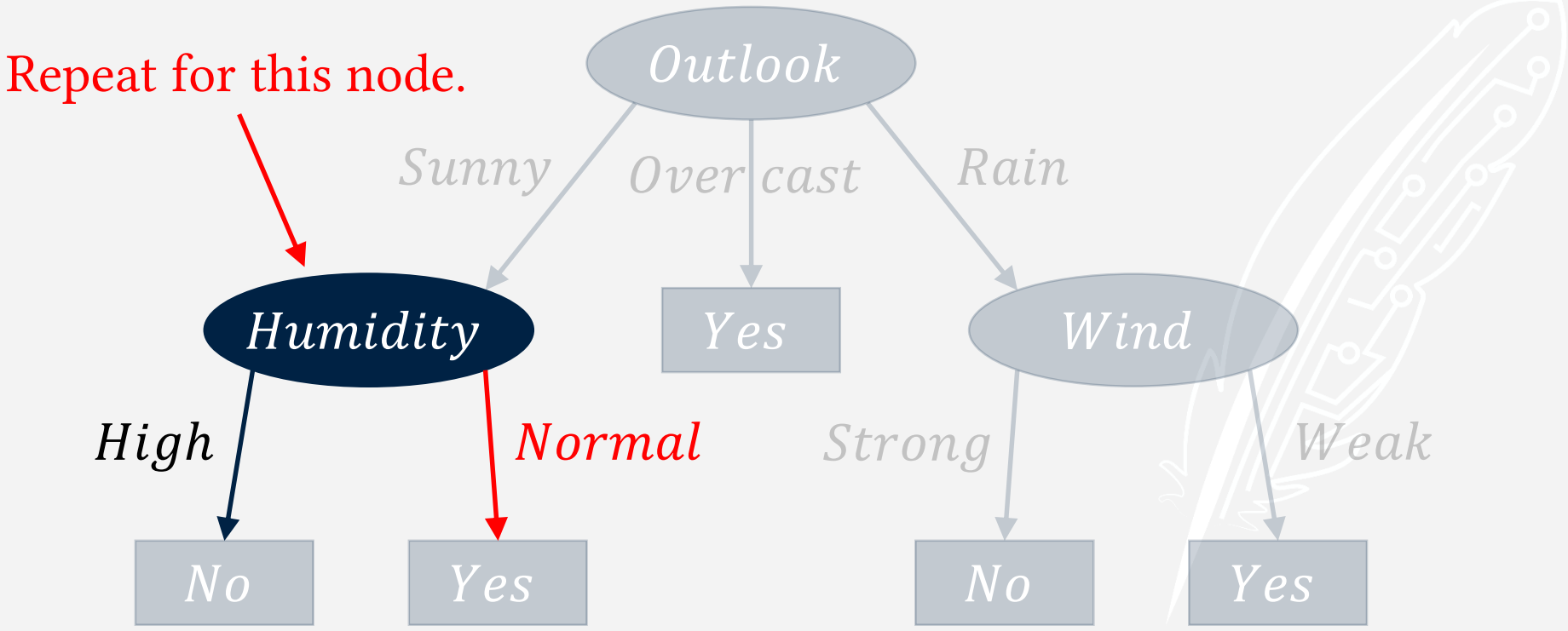
Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Mild	Normal	Strong	?



Decision Tree

Classify:

Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Mild	Normal	Strong	?

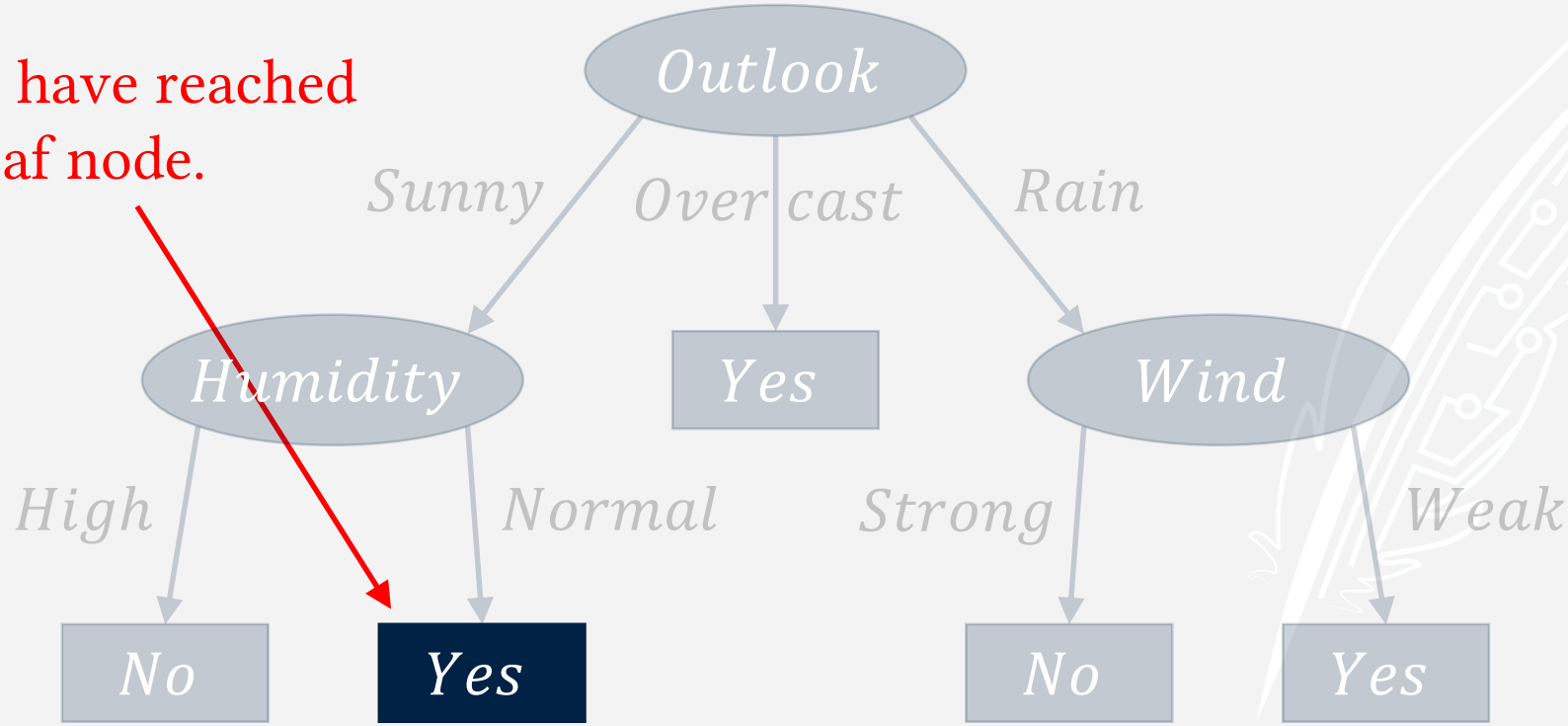


Decision Tree

Classify:

Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Mild	Normal	Strong	?

We have reached a leaf node.

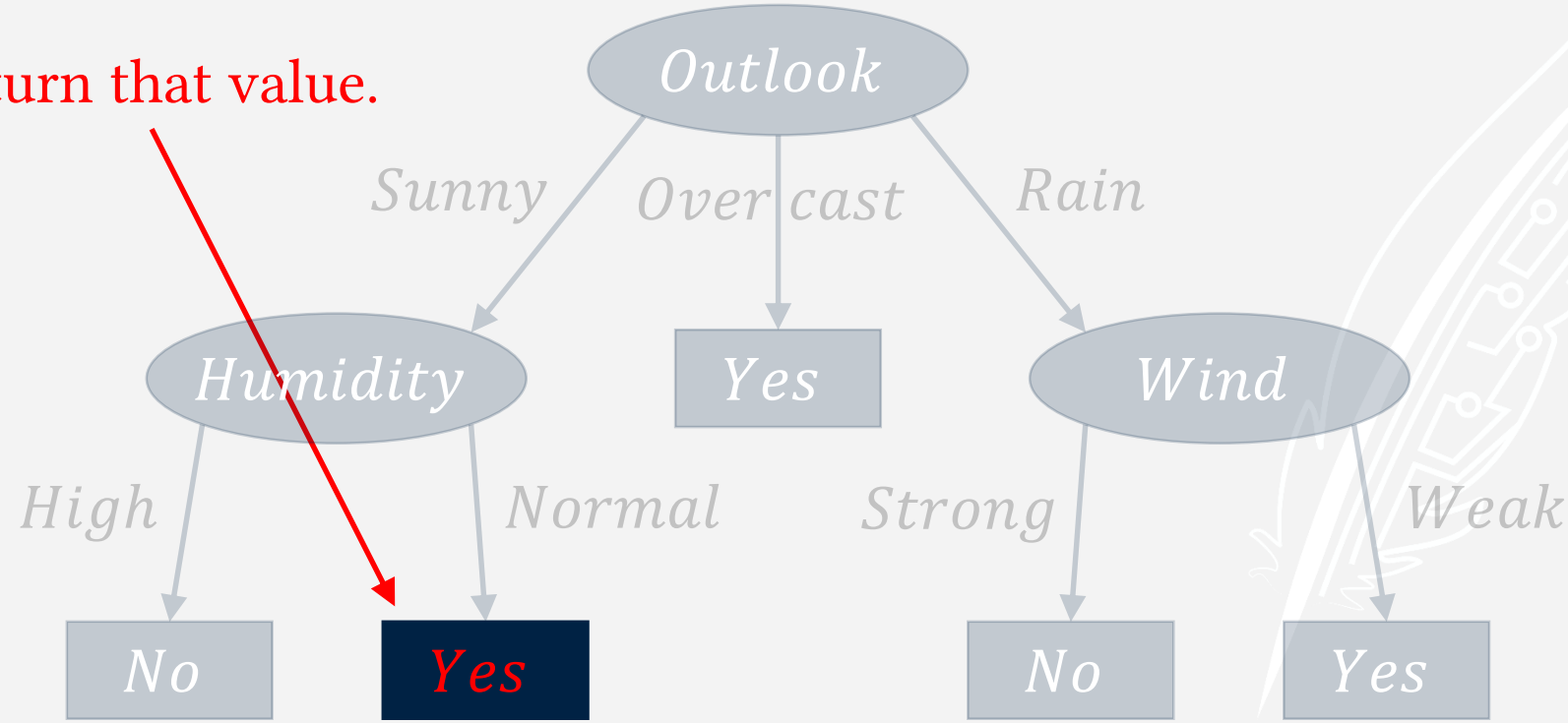


Decision Tree

Classify:

Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Mild	Normal	Strong	Yes

Return that value.



Decision Tree Induction

Decision trees can be learned from training data.

We should aim to:

- Minimize the depth of the tree (i.e. minimize the time it takes to classify something using the tree).
- Minimize the time it takes to build the tree.

Space of Trees

Consider a data set with n Boolean attributes.

How many different decision trees exist? $> 2^{2^n}$

This is a very large space to search! So, rather than search it, we will use a greedy algorithm to construct a decision tree which is usually very accurate and usually very small.

Space of Trees

Consider a data set with n Boolean attributes.

How many different decision trees exist? $> 2^{2^n}$

This is a very large space to search! So, rather than search it, we will use a greedy algorithm to construct a decision tree which is usually very accurate and usually very small.

Note: Not greedy search. There will be no search; we will simply construct a single tree using a greedy method.

Decision Tree Induction

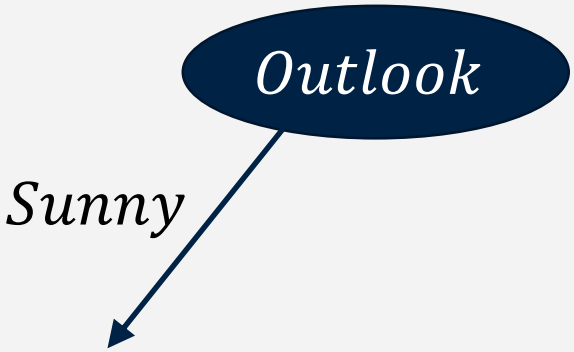
```
Input: a table of examples.
Output: a decision tree.

If all examples have the same class label:
    Create a leaf node with that class label.
If there are no more attributes to choose from:
    Create a leaf node with the majority class label.
Else:
    Choose an attribute.  Create a non-leaf node for it.
    Split the table into smaller tables for each value of the attribute.
    Recursively call this algorithm for each child + smaller table.
```

Do all examples have the same class label?
No.
Choose an attribute.
Create a non-leaf node.

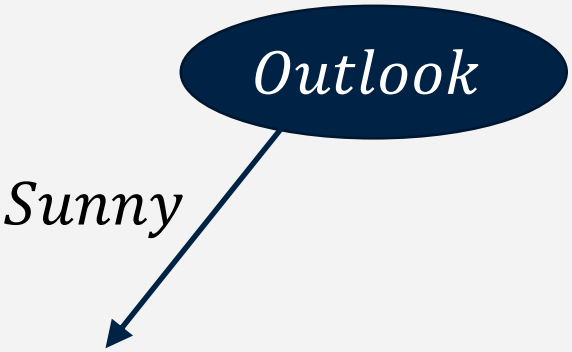
Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	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

Recursively call the algorithm for the branch of the tree corresponding to Outlook = Sunny.



Day	Outlook	Temperature	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	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

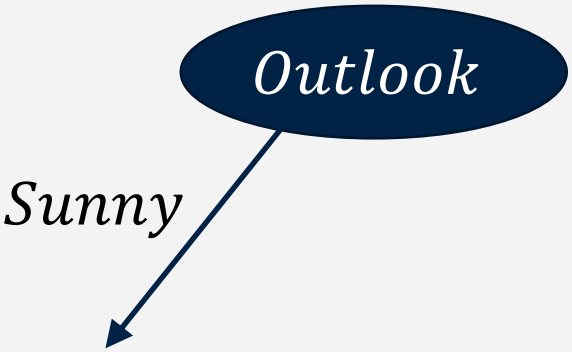
The table for this branch contains only those examples whose Outlook was Sunny.



Day	Outlook	Temperature	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	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

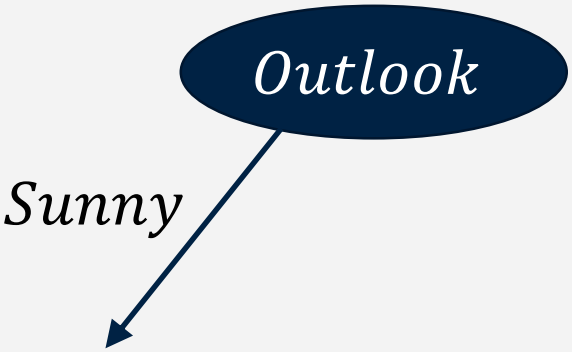
The table for this branch contains only those examples whose Outlook was Sunny.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

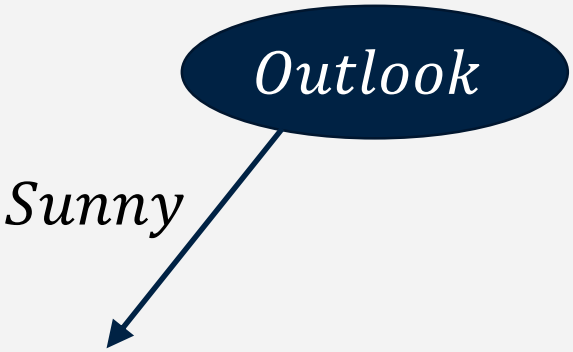


This means the Outlook column is no longer needed.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

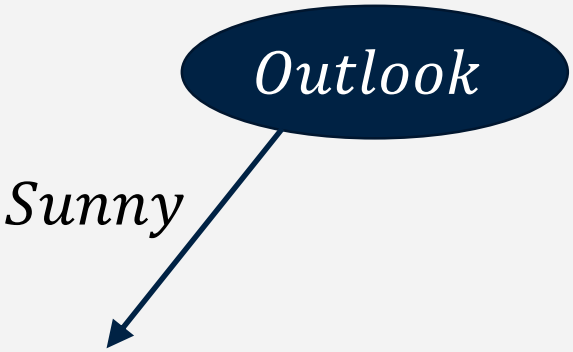


This means the Outlook column is no longer needed.



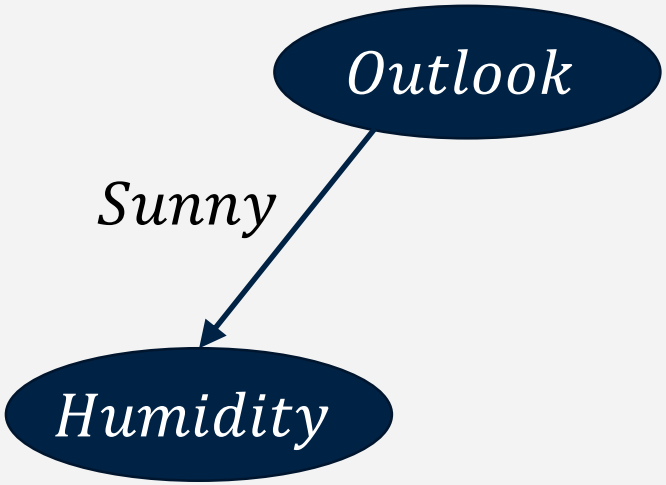
Day	Temperature	Humidity	Wind	Play Tennis?
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

Not all examples have the same class label, so we choose another attribute and create a non-leaf node.



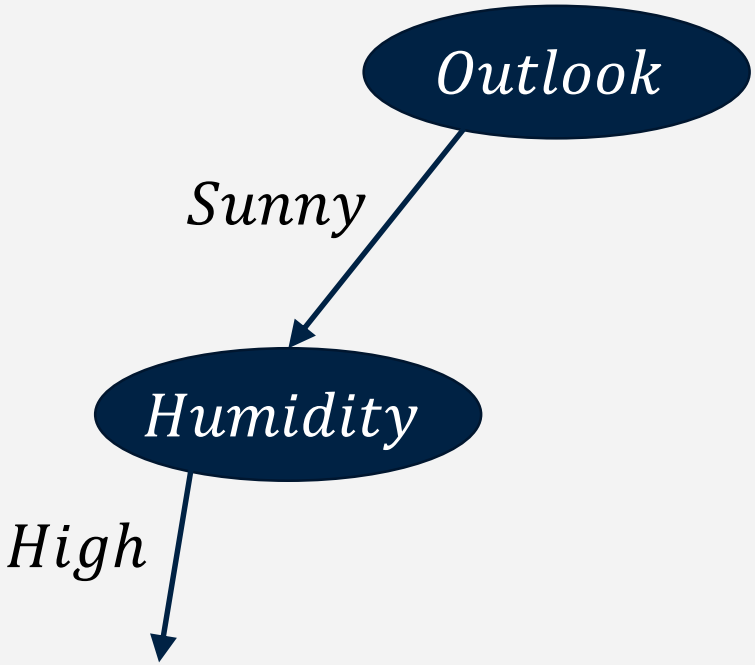
Day	Temperature	Humidity	Wind	Play Tennis?
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

Not all examples have the same class label, so we choose another attribute and create a non-leaf node.



Day	Temperature	Humidity	Wind	Play Tennis?
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

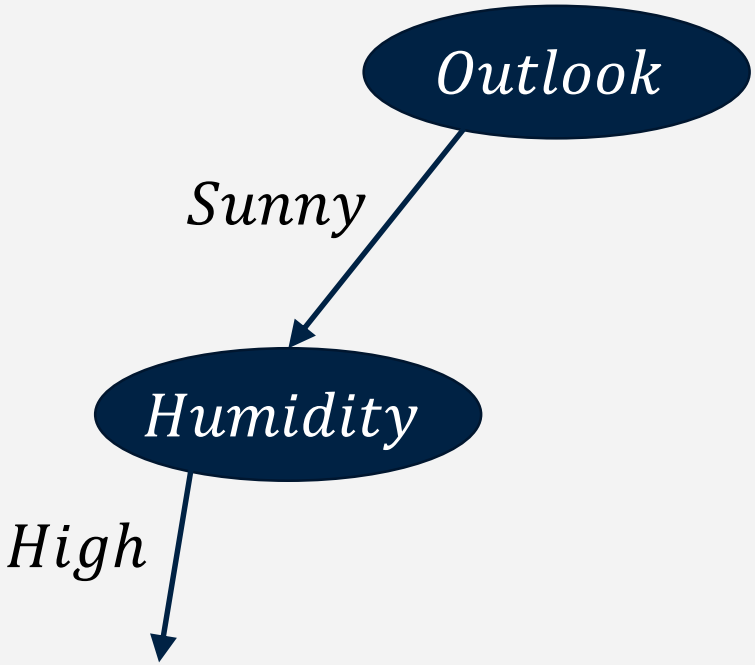
Recursively call the algorithm for the branch of the tree corresponding to Humidity = High.



Day	Temperature	Humidity	Wind	Play Tennis?
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

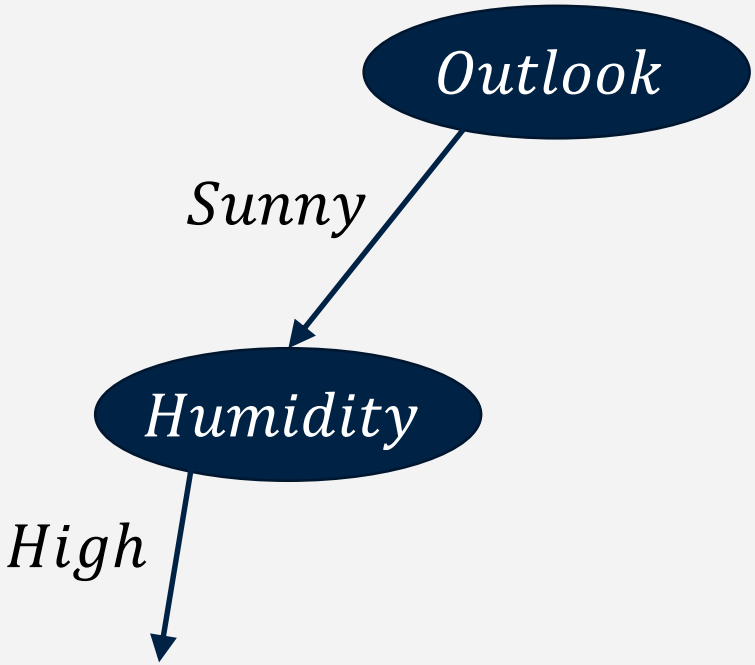
The table for this branch contains only those examples whose Humidity was High.

Day	Temperature	Humidity	Wind	Play Tennis?
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes



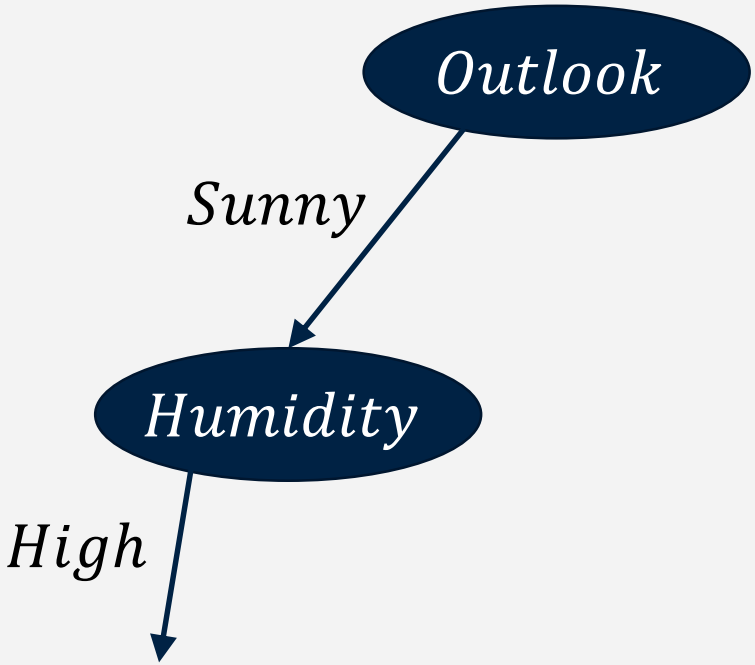
The table for this branch contains only those examples whose Humidity was High.

Day	Temperature	Humidity	Wind	Play Tennis?
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No



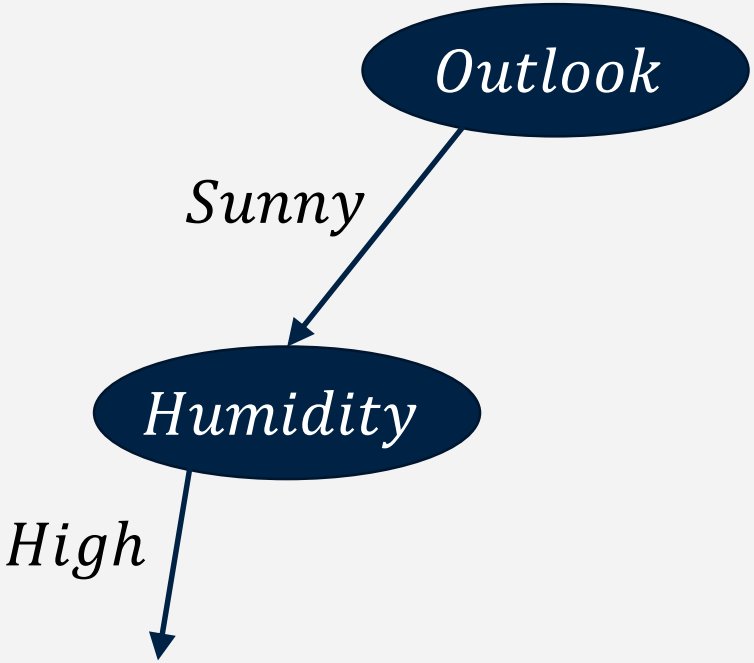
This means the Humidity column is no longer needed.

Day	Temperature	Humidity	Wind	Play Tennis?
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No



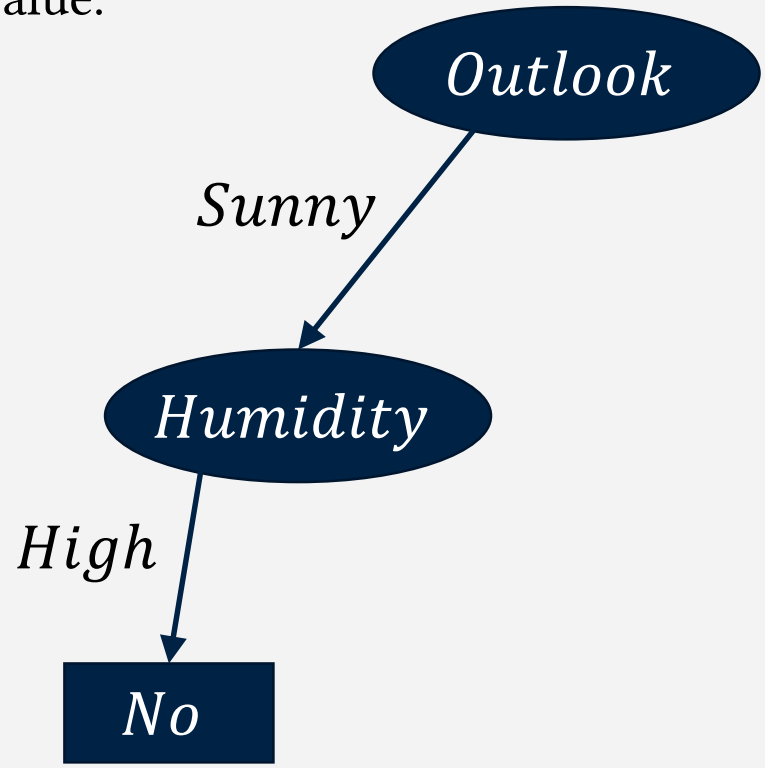
This means the Humidity column is no longer needed.

Day	Temperature	Wind	Play Tennis?
D1	Hot	Weak	No
D2	Hot	Strong	No
D8	Mild	Weak	No

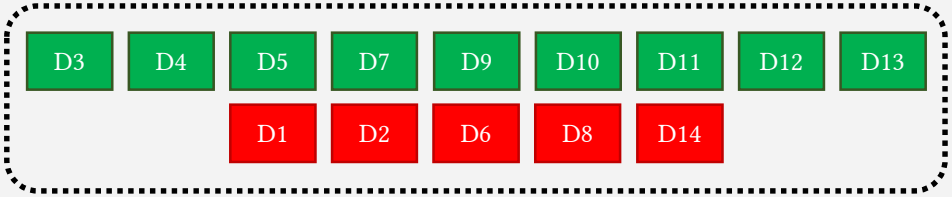


All the remaining examples have the same class label of “No.” Create a leaf node with that value.

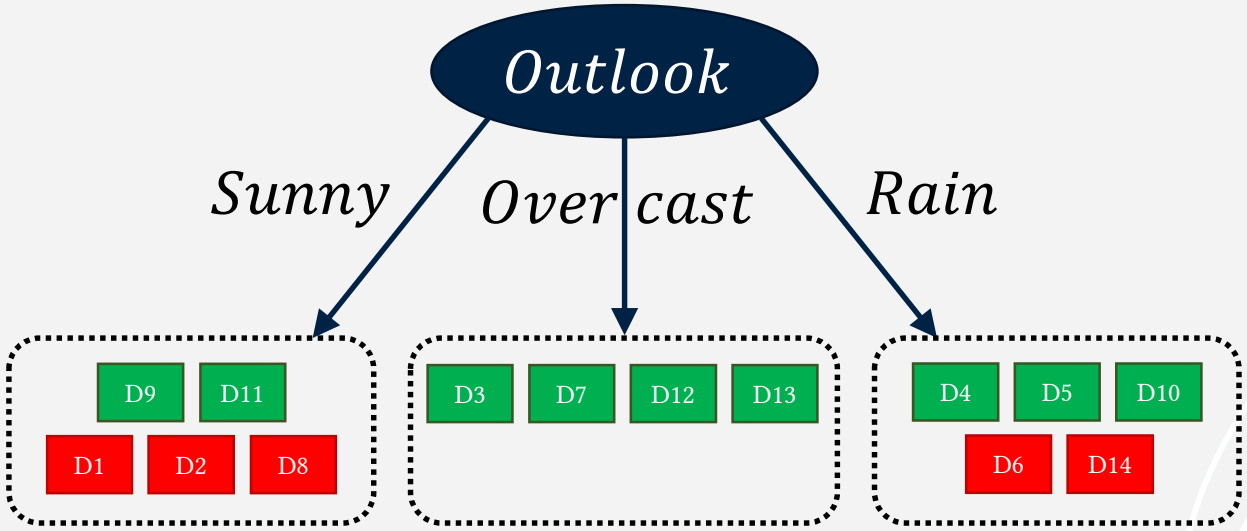
Day	Temperature	Wind	Play Tennis?
D1	Hot	Weak	No
D2	Hot	Strong	No
D8	Mild	Weak	No



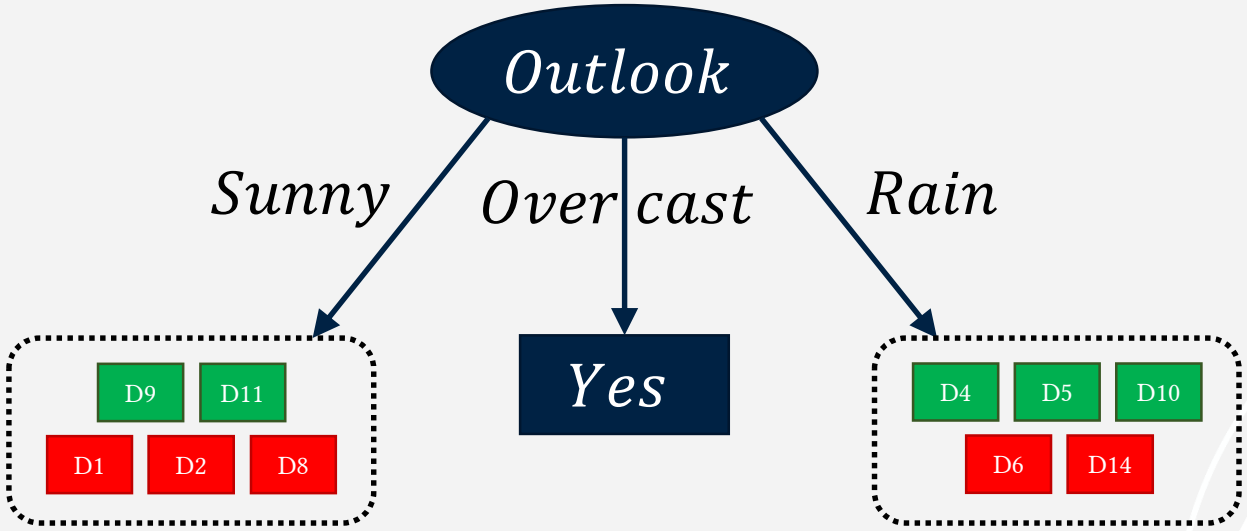
Decision Tree Induction



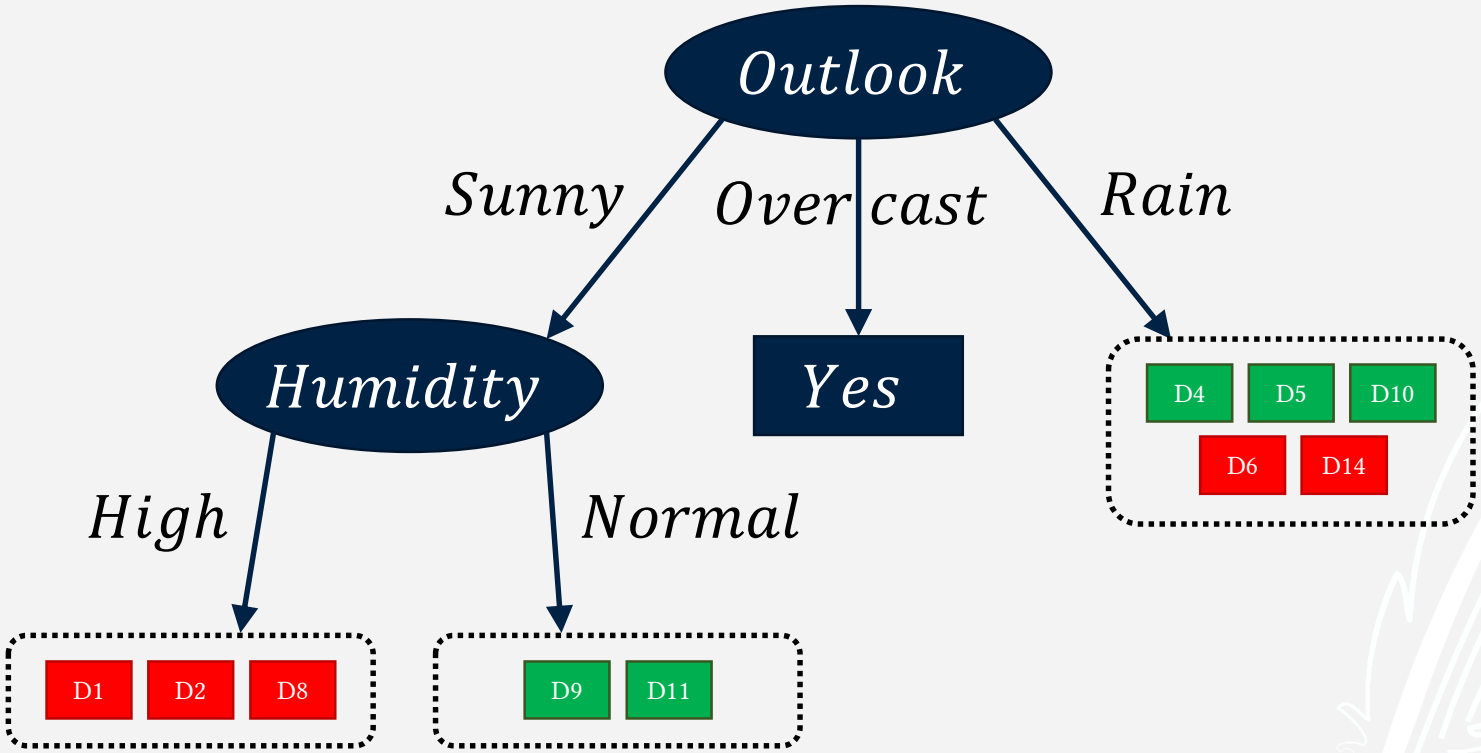
Decision Tree Induction



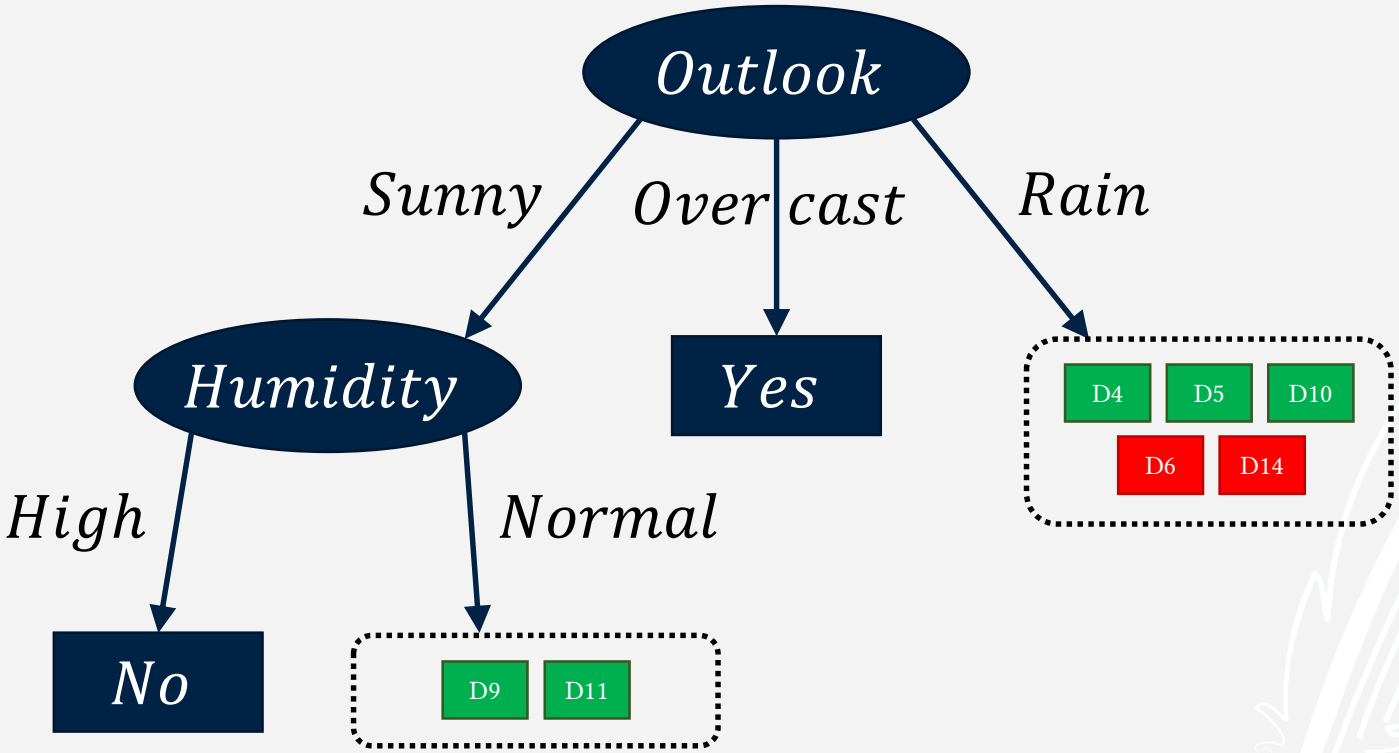
Decision Tree Induction



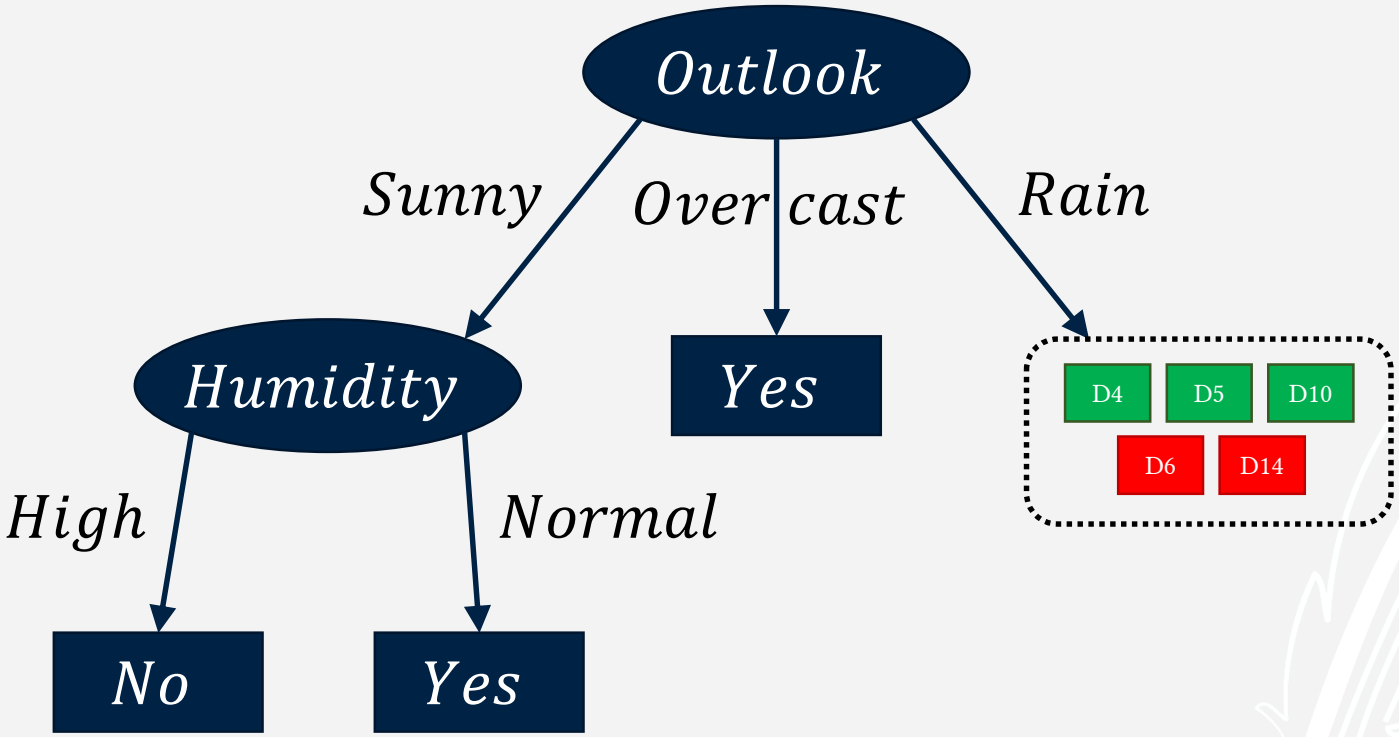
Decision Tree Induction



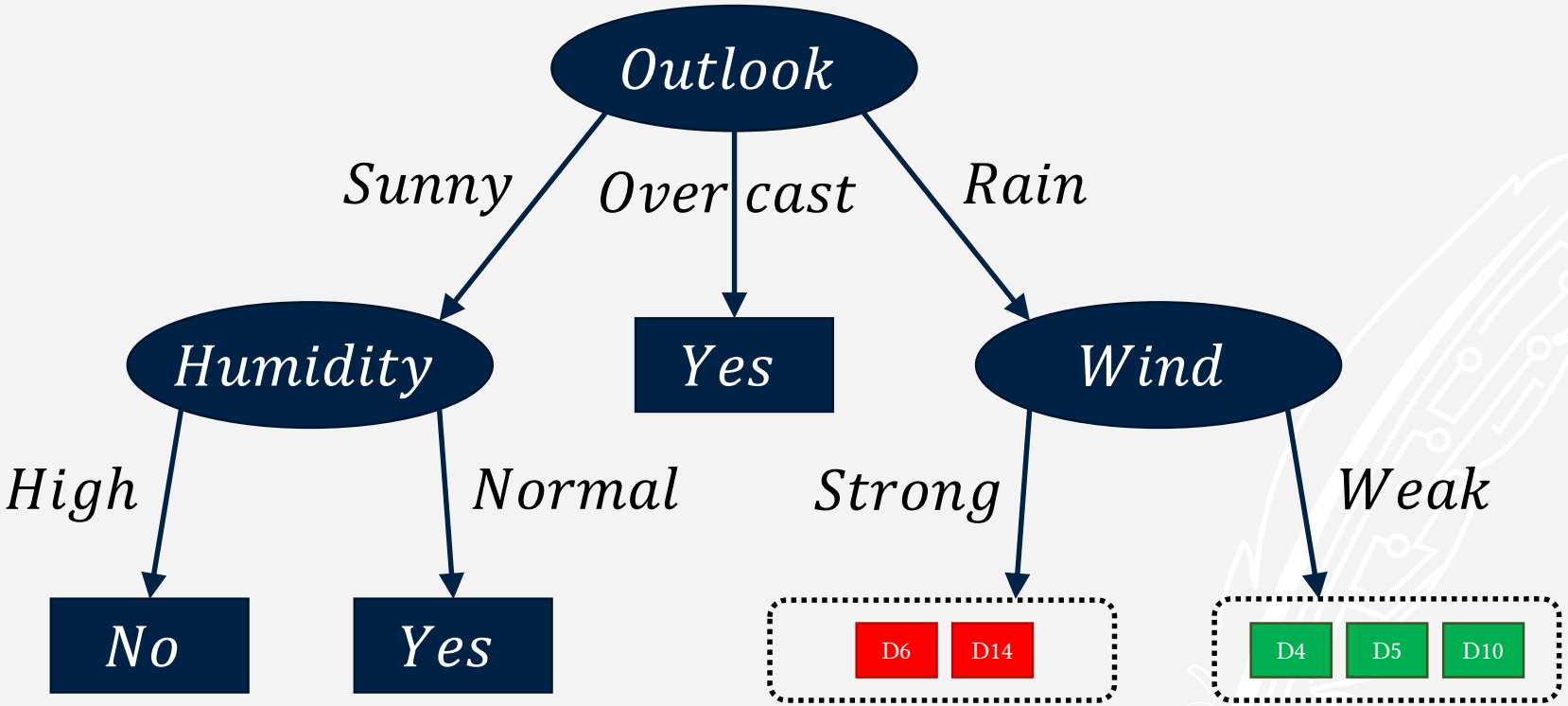
Decision Tree Induction



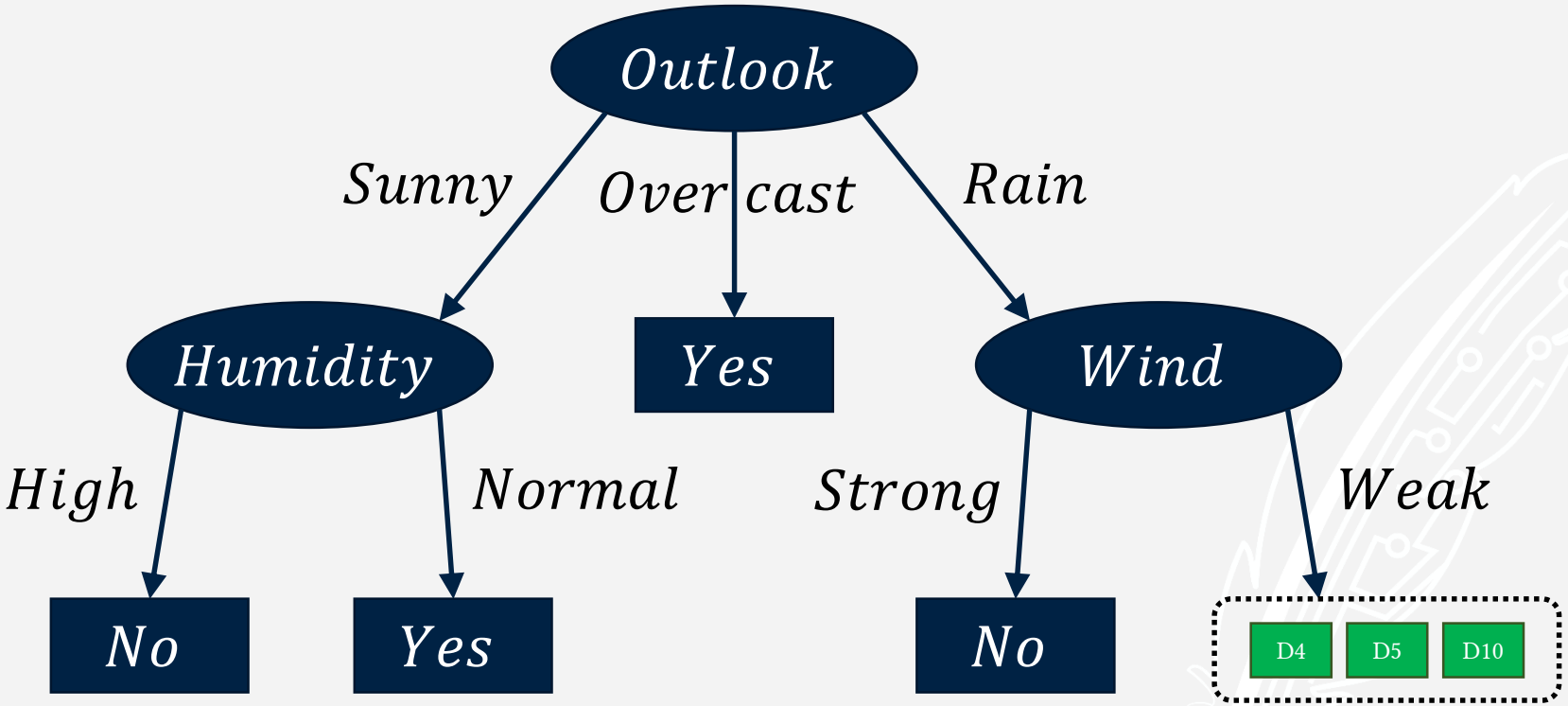
Decision Tree Induction



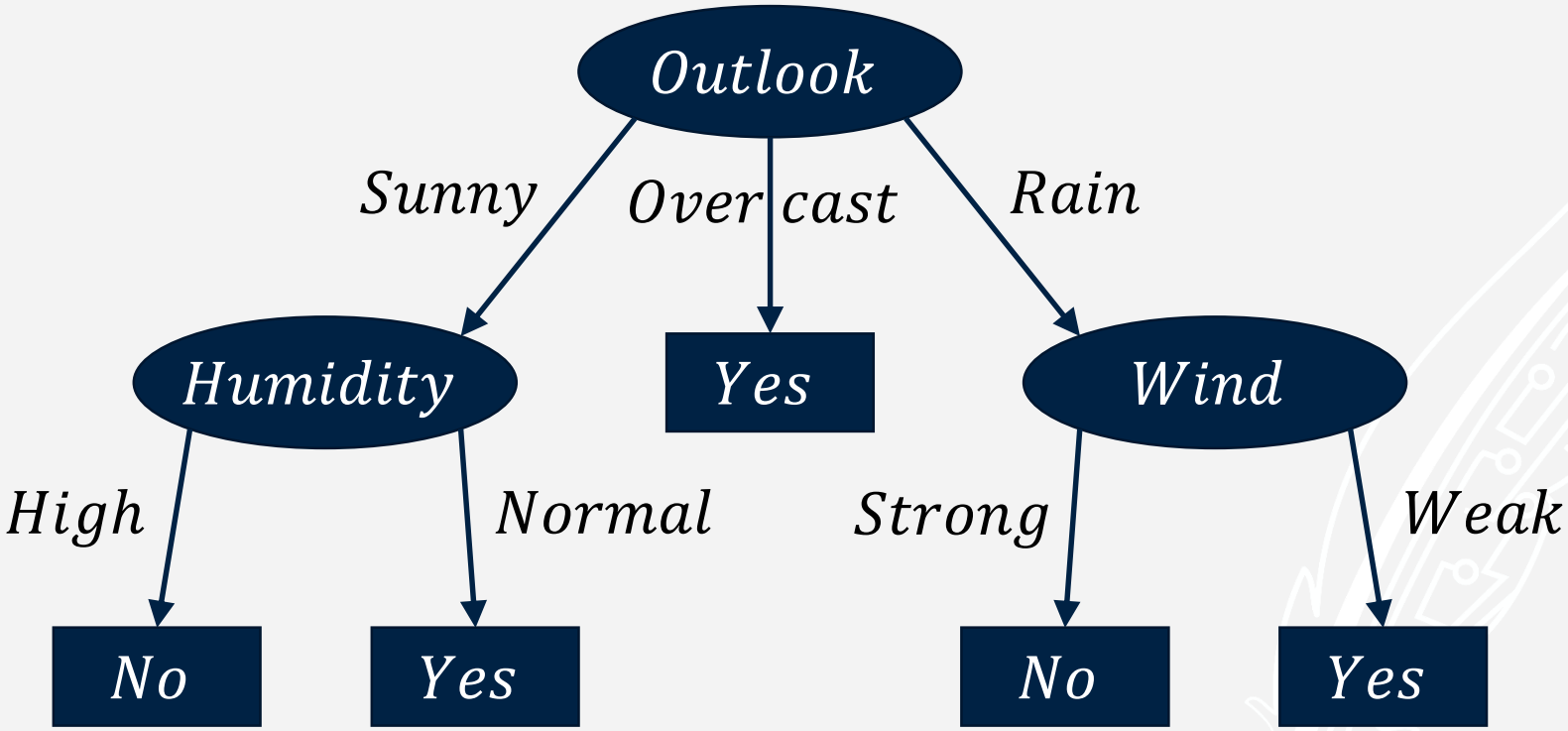
Decision Tree Induction



Decision Tree Induction



Decision Tree Induction



Decision Tree Induction

```
Input: a table of examples.
Output: a decision tree.

If all examples have the same class label:
    Create a leaf node with that class label.
If there are no more attributes to choose from:
    Create a leaf node with the majority class label.
Else:
    Choose an attribute.  Create a non-leaf node for it.
    Split the table into smaller tables for each value of the attribute.
    Recursively call this algorithm for each child + smaller table.
```

Decision Tree Induction

Input: a table of examples.

Output: a decision tree.

If all examples have the same class label:

    Create a leaf node with that class label.

If there are no more attributes to choose from:

    Create a leaf node with the majority class label.

Else:

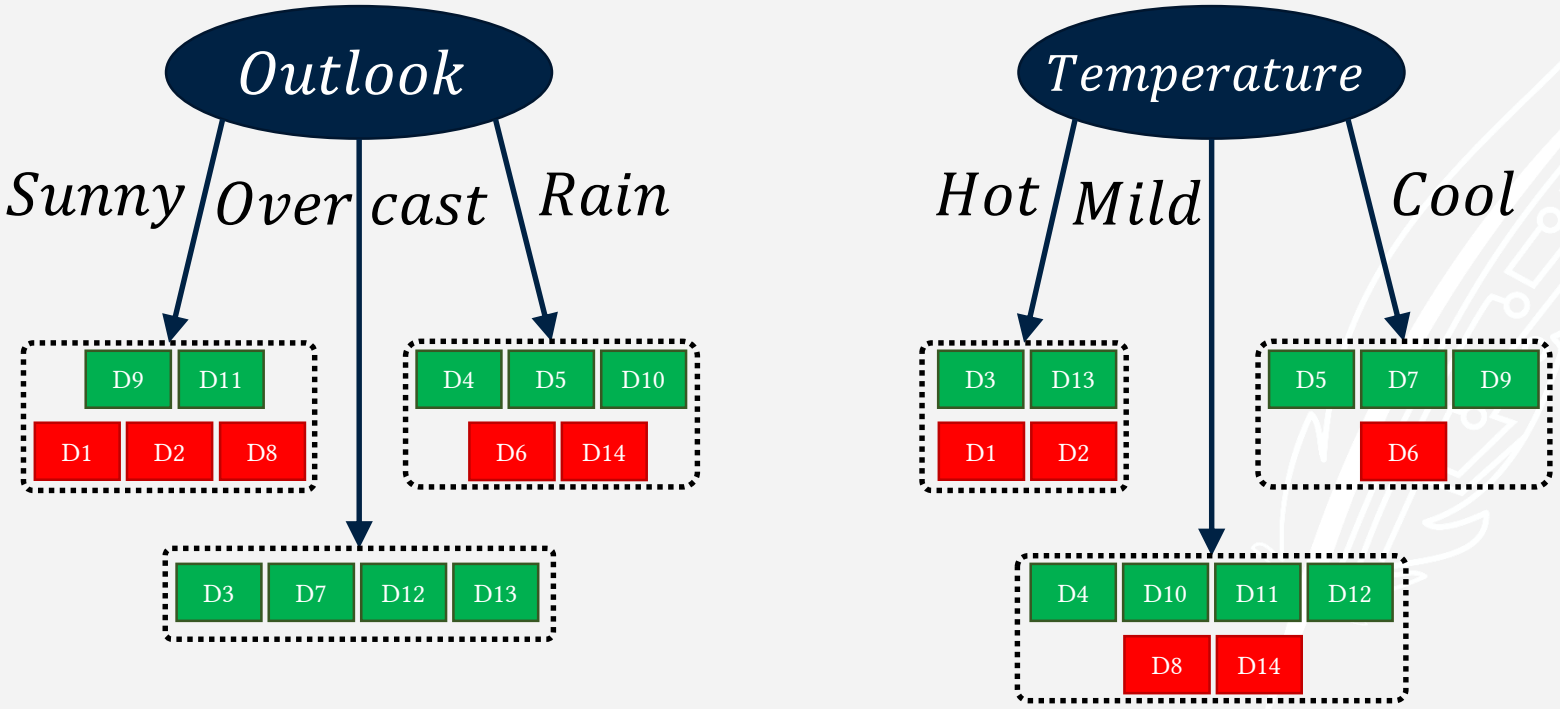
    Choose an attribute. Create a non-leaf node for it.

    Split the table into smaller tables for each value of the attribute.

    Recursively call this algorithm for each child + smaller table.

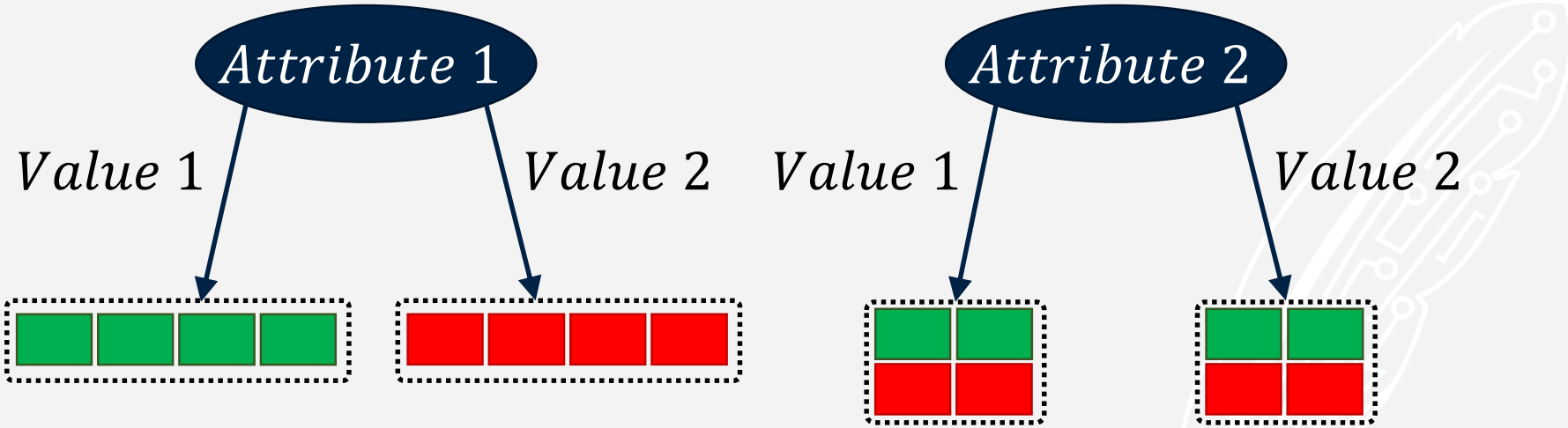
Choosing an Attribute

Which node makes a better root?



Choosing an Attribute

Which node makes a better root?



Choosing an Attribute

When a group contains mostly examples with the same class label, we say that it has **high purity** or low entropy.

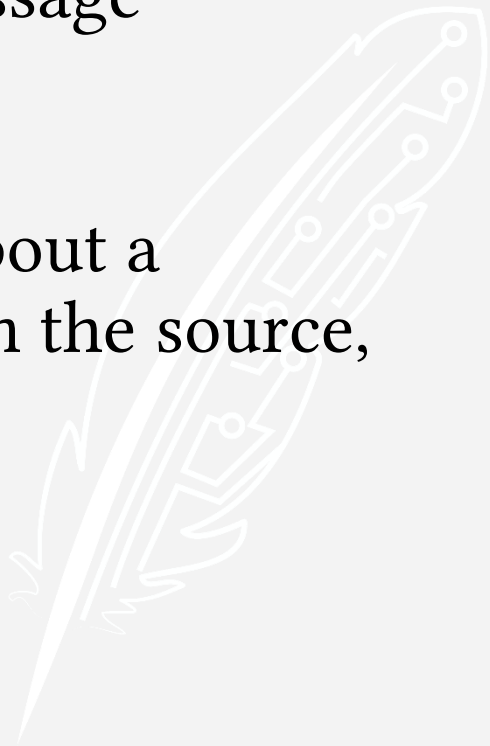
When a group contains about even numbers of each class label, we say that it has low purity or **high entropy**.

We want to choose attributes which divide the group of examples into pure (i.e. low entropy) groups.

Entropy

In the field of Information Theory, **entropy** is a measure of how much information a message contains.

Entropy characterizes our uncertainty about a source of information. The more random the source, the more entropy it has.



Entropy

How many bits are needed to communicate a message?

- The results of a fair coin flip require 1 bit to communicate, so it has 1 bit of entropy.
- The roll of a fair 4-sided die requires 2 bits to communicate, so it has 2 bits of entropy.
- A two-headed coin has no uncertainty. No bits are needed to communicate its value, and thus it has 0 bits of entropy.

Entropy

- If we have an unfair coin which comes up heads 99% of the time, it must have between 0 and 1 bits of entropy (with a value closer to 0 than 1).

In other words, the less likely an event is to happen (e.g. tails on this unfair coin), the more information that event provides when it occurs.

Entropy

For some random variable V with possible values $\{v_1, v_2, \ldots, v_n\}$, we can express entropy in bits as:

$$entropy(V) = \sum_{v \in V} -P(v) \log_2 P(v)$$

Entropy

For some random variable V with possible values $\{v_1, v_2, \ldots, v_n\}$, we can express entropy in bits as:

$$entropy(V) = \sum_{v \in V} -P(v) \log_2 P(v)$$

For every possible value of V

Entropy

For some random variable V with possible values $\{v_1, v_2, \ldots, v_n\}$, we can express entropy in bits as:

$$entropy(V) = \sum_{v \in V} -P(v) \log_2 P(v)$$

The probability that $V = v$

Entropy

For some random variable V with possible values $\{v_1, v_2, \ldots, v_n\}$, we can express entropy in bits as:

$$entropy(V) = \sum_{v \in V} -P(v) \log_2 P(v)$$

Because we are dealing with binary bits

Entropy

For some random variable V with possible values $\{v_1, v_2, \ldots, v_n\}$, we can express entropy in bits as:

$$entropy(V) = \sum_{v \in V} -P(v) \log_2 P(v)$$

Remember that the log of a number between 0 and 1 is a negative number

Entropy

For some random variable V with possible values $\{v_1, v_2, \ldots, v_n\}$, we can express entropy in bits as:

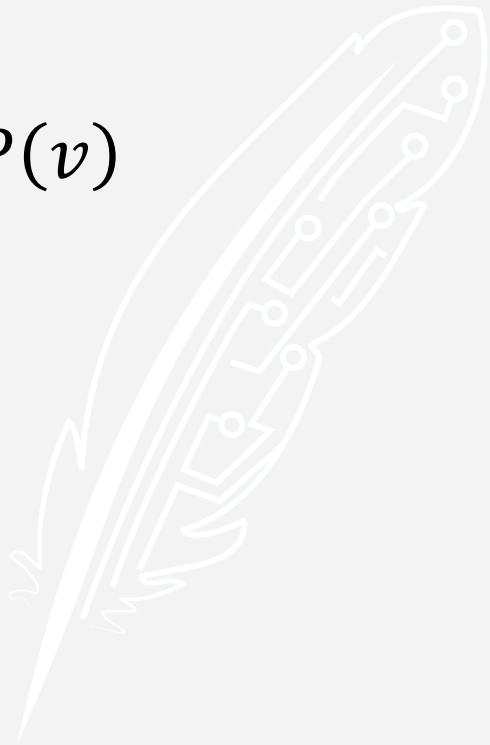
$$entropy(V) = \sum_{v \in V} -P(v) \log_2 P(v)$$

So we multiply the expression by -1.

Entropy

How many bits of entropy has a fair coin?

$$entropy(V) = \sum_{v \in V} -P(v) \log_2 P(v)$$



Entropy

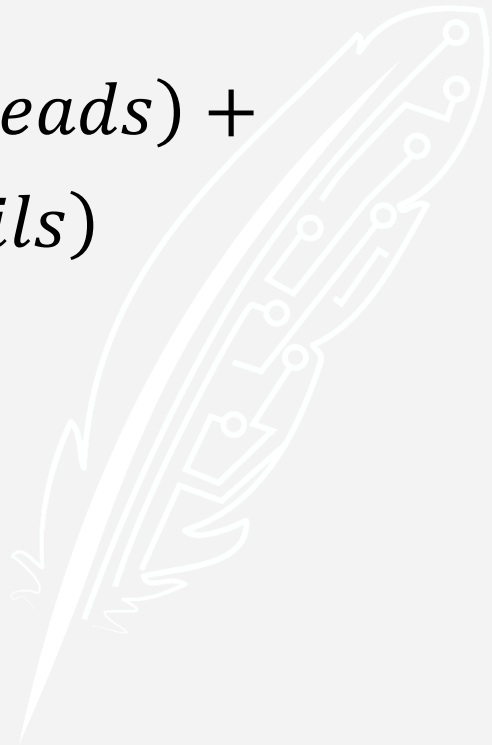
How many bits of entropy has a fair coin?

$$entropy(V) = \sum_{v \in \{heads, tails\}} -P(v) \log_2 P(v)$$

Entropy

How many bits of entropy has a fair coin?

$$entropy(V) = -P(heads)\log_2 P(heads) + \\ -P(tails)\log_2 P(tails)$$



Entropy

How many bits of entropy has a fair coin?

$$entropy(V) = -0.5 \log_2 0.5 + -0.5 \log_2 0.5$$

Entropy

How many bits of entropy has a fair coin?

$$entropy(V) = -0.5 \cdot -1 + -0.5 \cdot -1$$

Entropy

How many bits of entropy has a fair coin?

$$entropy(V) = 0.5 + 0.5$$

Entropy

How many bits of entropy has a fair coin?

$$entropy(V) = 1$$

Entropy

How many bits of entropy has a two-headed coin?

$$entropy(V) = \sum_{v \in \{heads\}} -P(heads) \log_2 P(heads)$$

Entropy

How many bits of entropy has a two-headed coin?

$$entropy(V) = -1 \log_2 1$$

Entropy

How many bits of entropy has a two-headed coin?

$$entropy(V) = -1 \cdot 0$$

Entropy

How many bits of entropy has a two-headed coin?

$$entropy(V) = 0$$

Entropy

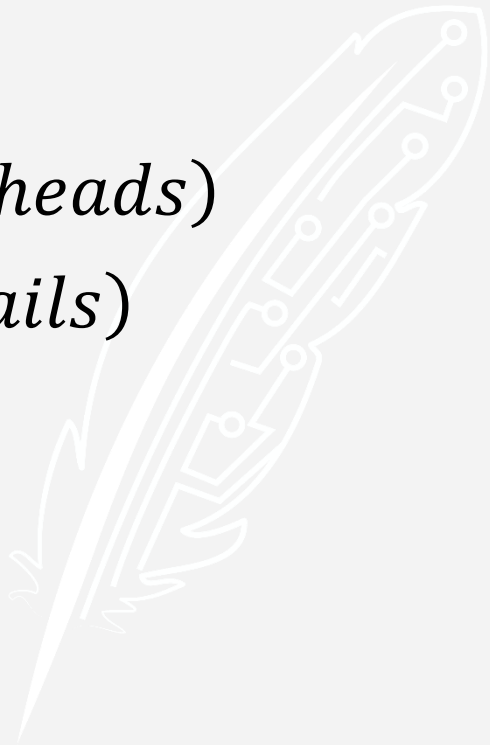
How many bits of entropy has an unfair coin that comes up heads 99% of the time?

$$entropy(V) = \sum_{v \in \{heads, tails\}} -P(v) \log_2 P(v)$$

Entropy

How many bits of entropy has an unfair coin that comes up heads 99% of the time?

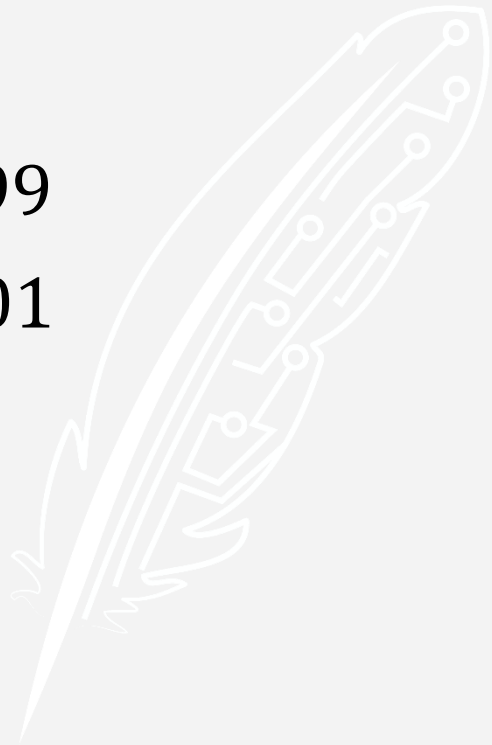
$$entropy(V) = -P(heads)\log_2 P(heads) \\ -P(tails)\log_2 P(tails)$$



Entropy

How many bits of entropy has an unfair coin that comes up heads 99% of the time?

$$entropy(V) = -0.99 \log_2 0.99 \\ -0.01 \log_2 0.01$$



Entropy

How many bits of entropy has an unfair coin that comes up heads 99% of the time?

$$entropy(V) \approx 0.08$$

Choosing an Attribute

When deciding which attribute to split on, the goal is to choose the one which will divide the remaining examples into the most pure groups.

Choosing an Attribute

We can think of the class variable as a random variable (in this case, one with two possible values, “yes” and “no”).

If the current group of examples has 50% “yes” and 50% “no,” then we are very uncertain how to classify new examples. We might as well just flip a coin and use that to classify new examples. Such a group has high entropy.

Choosing an Attribute

We can think of the class variable as a random variable (in this case, one with two possible values, “yes” and “no”).

However, if the current group of examples has only “yes” or only “no,” then we are 100% confident how to classify new examples. Such a group has no entropy.

Information Gain

When choosing an attribute, we should choose the one that splits the examples into groups with the lowest entropy.

By reducing entropy in the group of examples, we gain information.

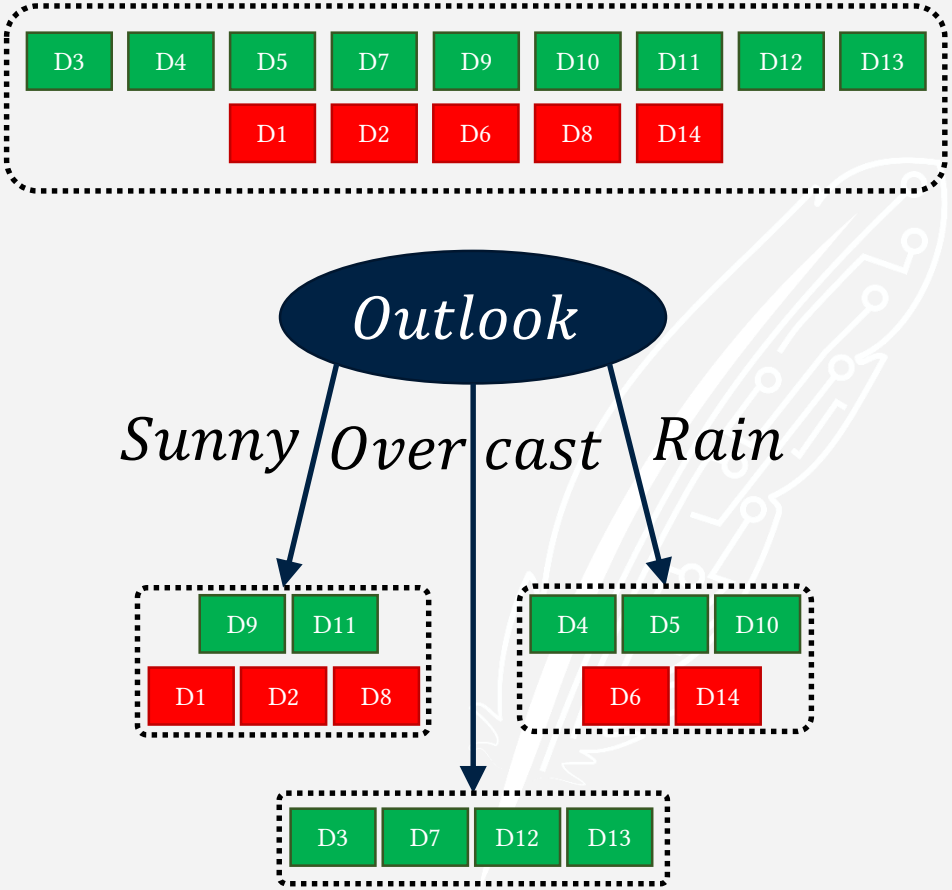
Information Gain

Given a set of candidate nodes to add to the decision tree (i.e. attributes to split on), the **information gain** of a node can be expressed as the entropy of that node minus the weighted average entropy of its children.

Information Gain

Information gain is the entropy of the parent minus the weighted average entropy of the children.

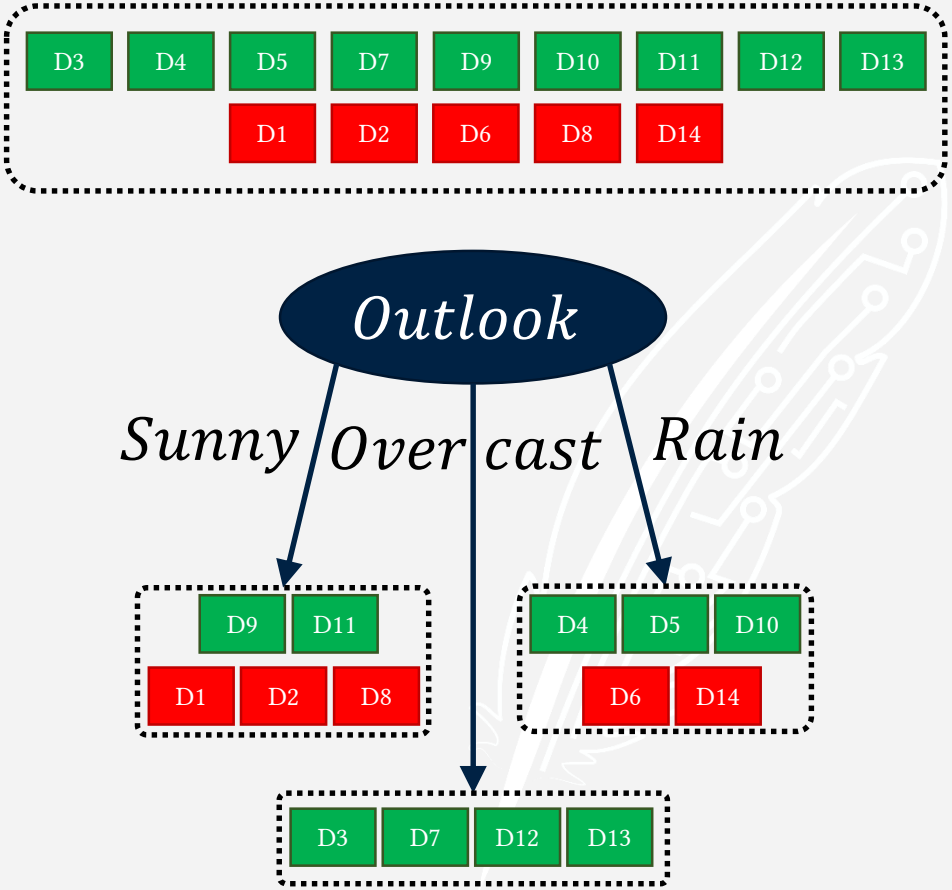
$$entropy(parent) = \sum_{v \in \{yes, no\}} -P(v) \log_2 P(v)$$



Information Gain

Information gain is the entropy of the parent minus the weighted average entropy of the children.

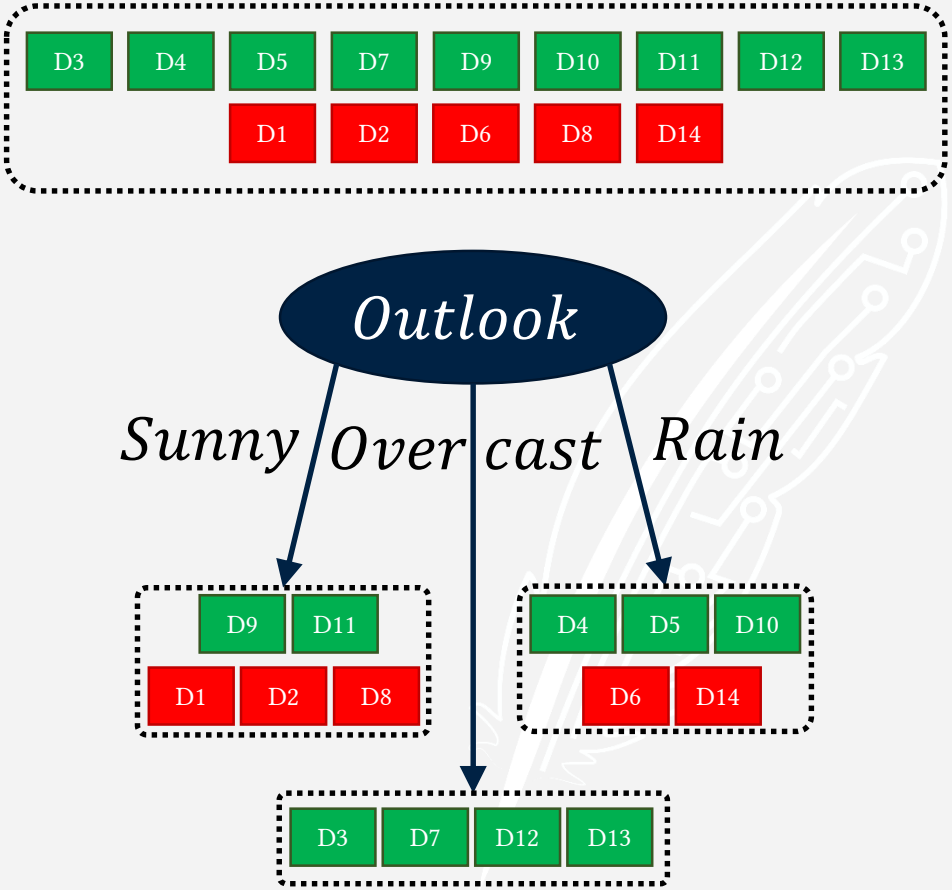
$$entropy(parent) = -P(yes)\log_2 P(yes) + -P(no)\log_2 P(no)$$



Information Gain

Information gain is the entropy of the parent minus the weighted average entropy of the children.

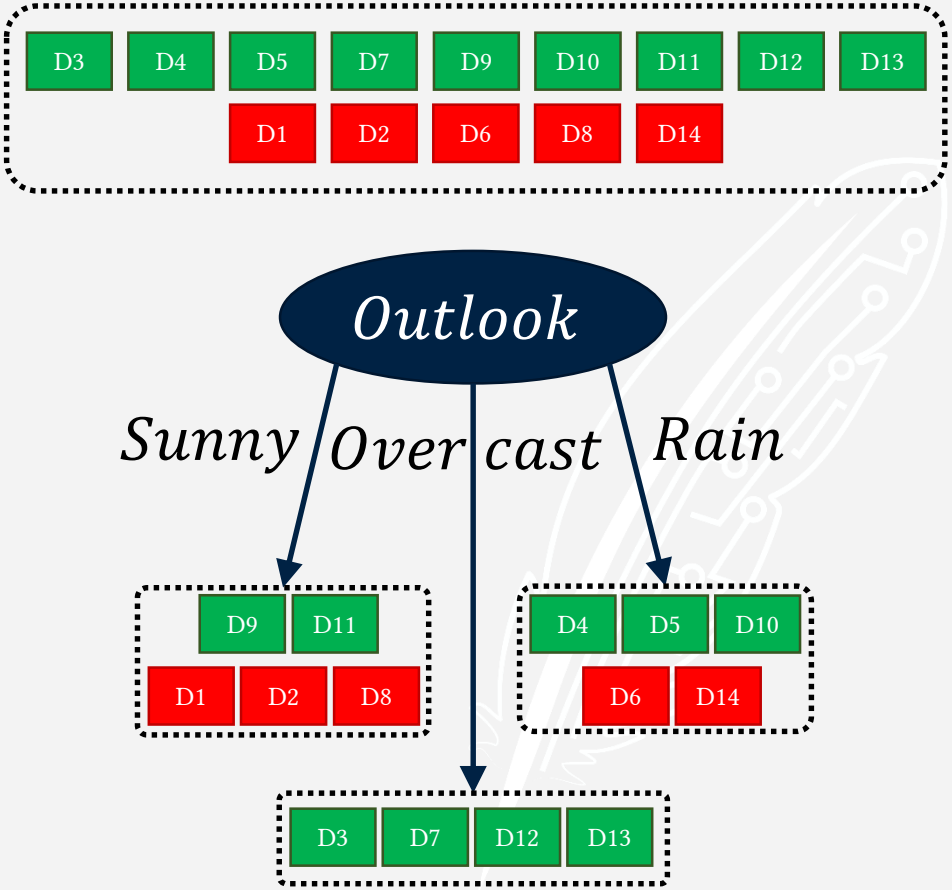
$$entropy(parent) = -\frac{9}{14}\log_2\frac{9}{14} + -\frac{5}{14}\log_2\frac{5}{14}$$



Information Gain

Information gain is the entropy of the parent minus the weighted average entropy of the children.

$$entropy(parent) \approx 0.94$$

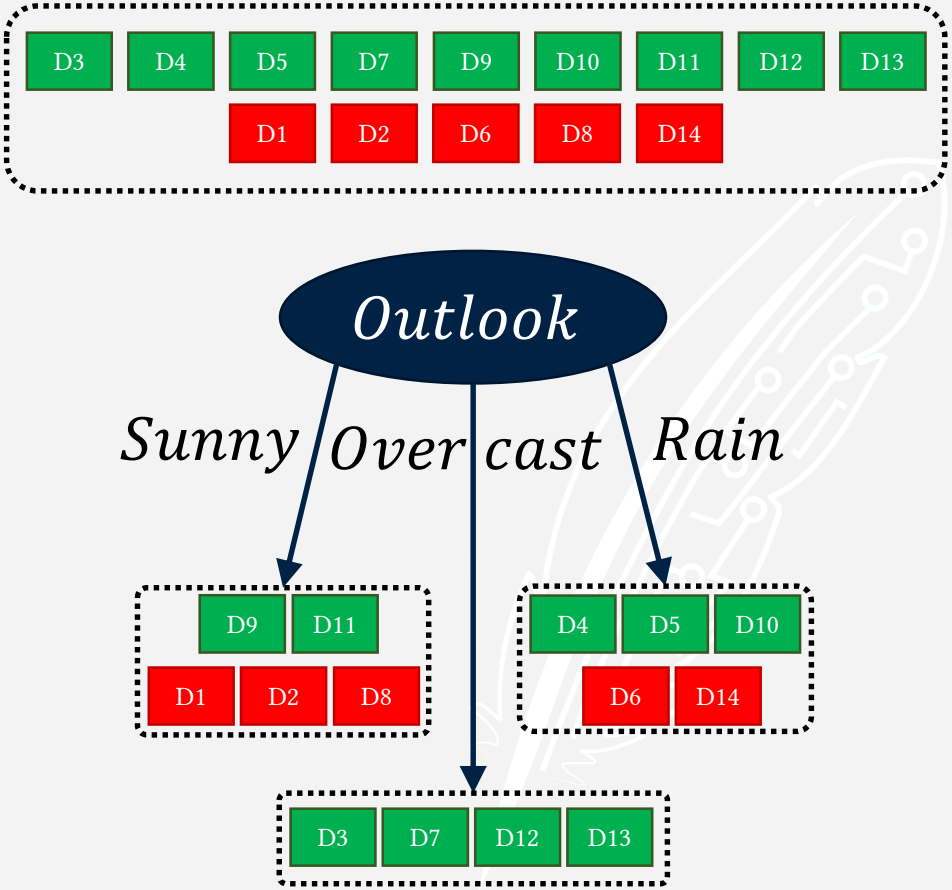


Information Gain

Information gain is the entropy of the parent minus the weighted average entropy of the children.

$$entropy(parent) \approx 0.94$$

$$entropy_{avg}(child) = \begin{bmatrix} 5/14 \cdot entropy(sunny) + \\ 4/14 \cdot entropy(over) + \\ 5/14 \cdot entropy(rain) \end{bmatrix}$$

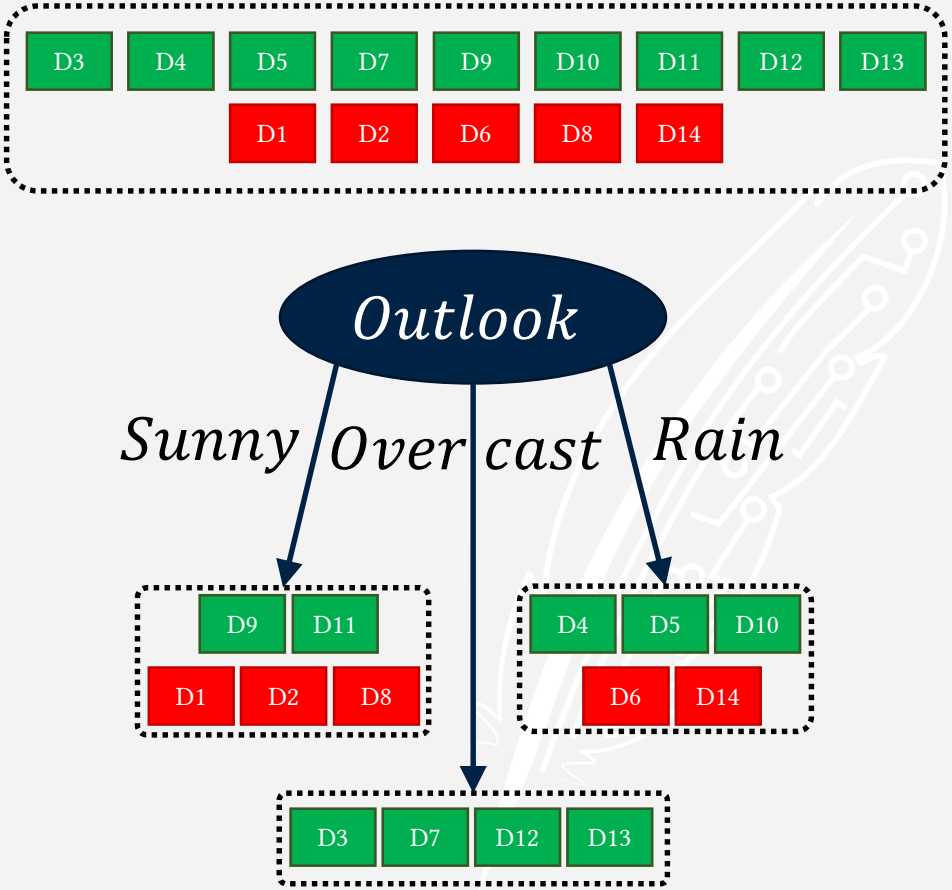


Information Gain

Information gain is the entropy of the parent minus the weighted average entropy of the children.

$$entropy(parent) \approx 0.94$$

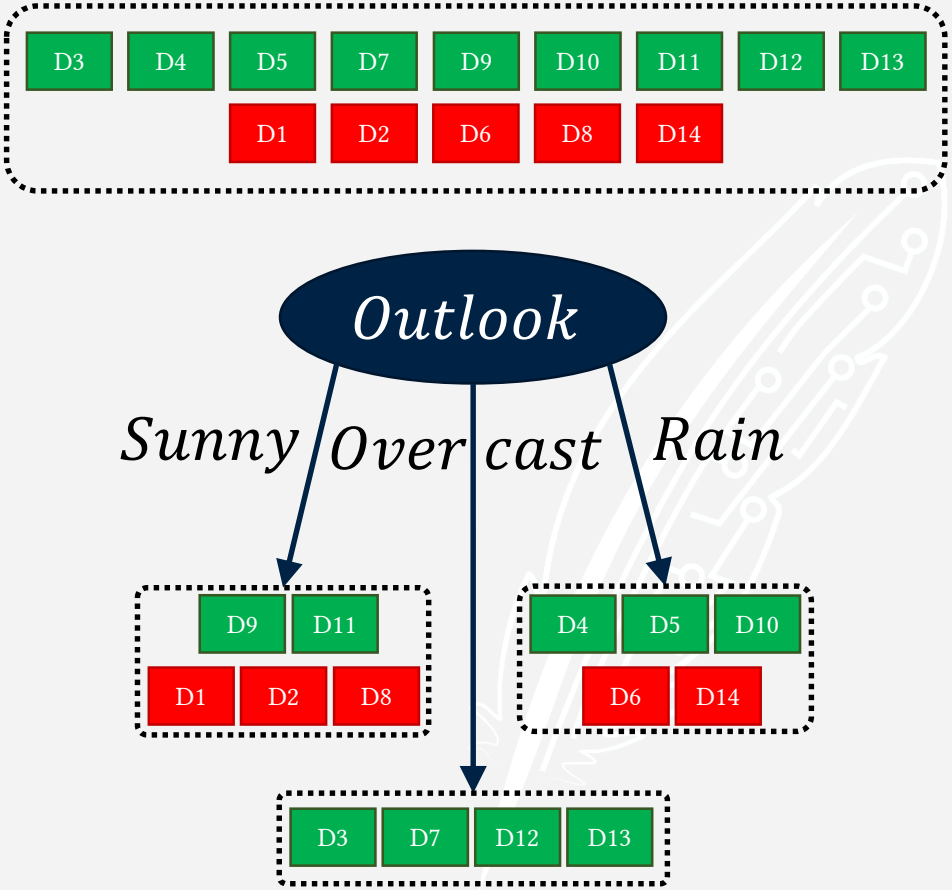
$$entropy_{avg}(child) = \begin{bmatrix} 5/14 \cdot 0.97 + \\ 4/14 \cdot 0.00 + \\ 5/14 \cdot 0.97 \end{bmatrix}$$



Information Gain

Information gain is the entropy of the parent minus the weighted average entropy of the children.

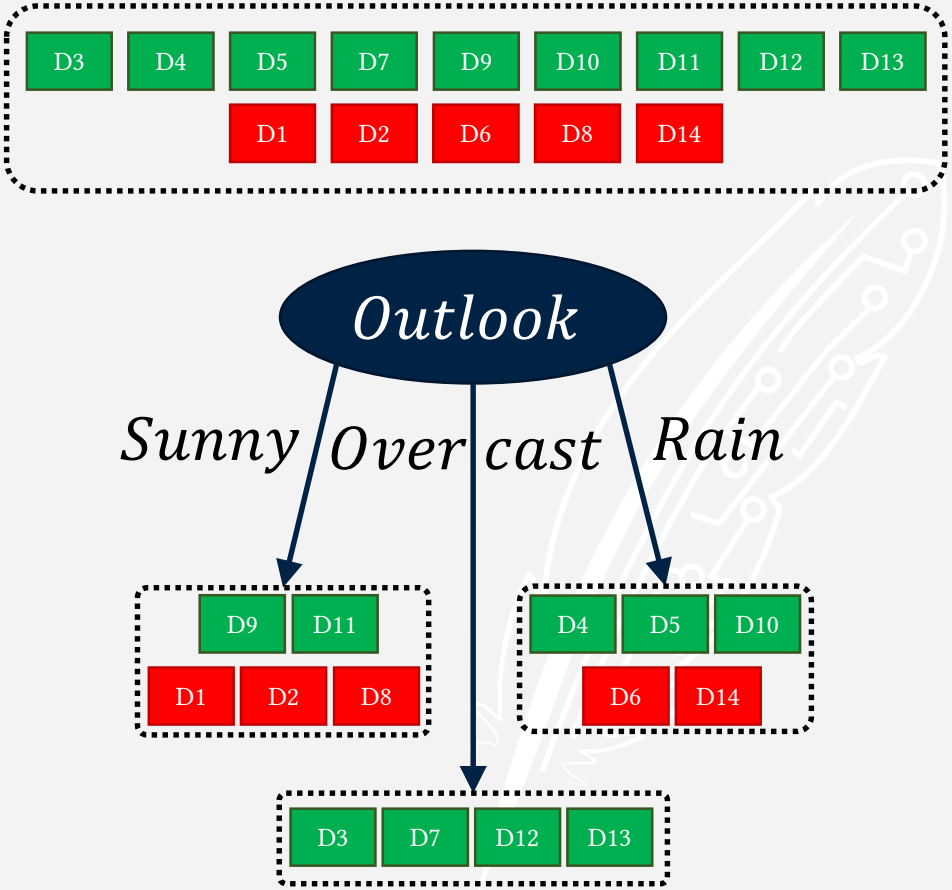
$$entropy(parent) \approx 0.94$$
$$entropy_{avg}(child) \approx 0.69$$



Information Gain

Information gain is the entropy of the parent minus the weighted average entropy of the children.

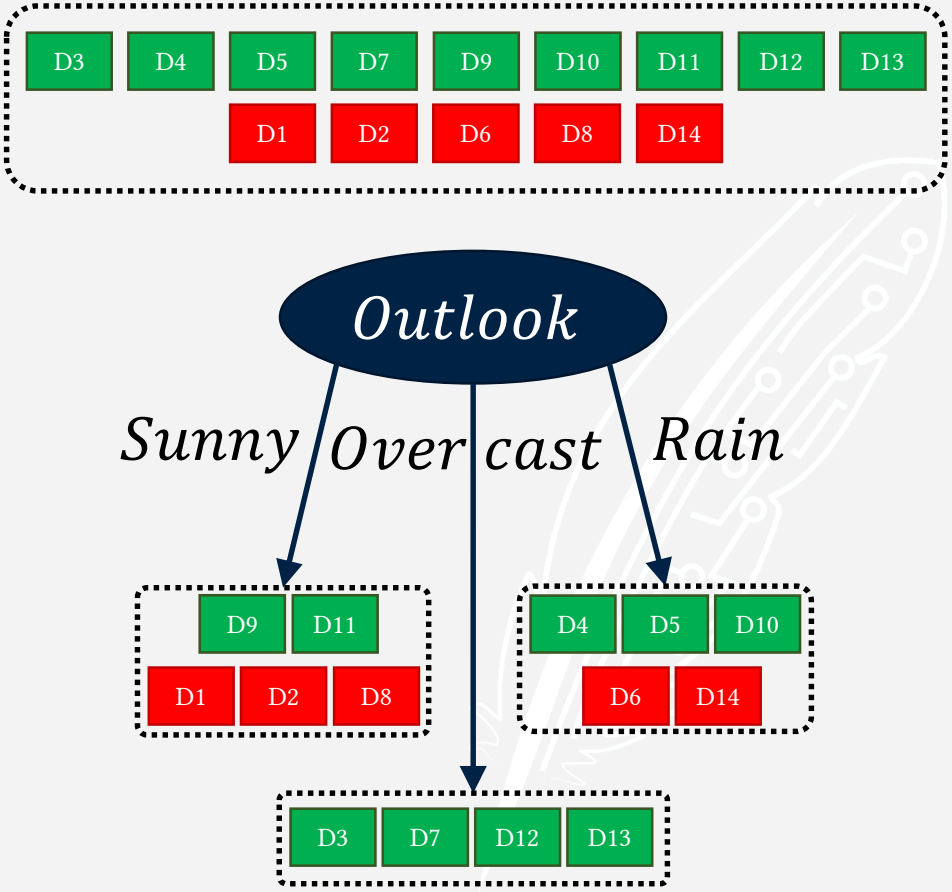
$$entropy(parent) \approx 0.94$$
$$entropy_{avg}(child) \approx 0.69$$
$$gain \approx 0.94 - 0.69$$



Information Gain

Information gain is the entropy of the parent minus the weighted average entropy of the children.

$$entropy(parent) \approx 0.94$$
$$entropy_{avg}(child) \approx 0.69$$
$$gain \approx 0.25$$



Information Gain

Calculate information gain for each attribute.
Which node will be the best root?

$gain \approx 0.246$ $gain \approx 0.029$ $gain \approx 0.151$ $gain \approx 0.048$

Information Gain

Calculate information gain for each attribute.
Which node will be the best root?

Outlook *Temperature* *Humidity* *Wind*

$gain \approx 0.246$ $gain \approx 0.029$ $gain \approx 0.151$ $gain \approx 0.048$

Decision Tree Induction

Input: a table of examples.

Output: a decision tree.

If all examples have the same class label:

    Create a leaf node with that class label.

If there are no more attributes to choose from:

    Create a leaf node with the majority class label.

Else:

    Choose an attribute. Create a non-leaf node for it.

    Split the table into smaller tables for each value of the attribute.

    Recursively call this algorithm for each child + smaller table.

Decision Tree Induction

Input: a table of examples.

Output: a decision tree.

If all examples have the same class label:

    Create a leaf node with that class label.

If there are no more attributes to choose from:

    Create a leaf node with the majority class label.

Else:

    Choose the attribute with highest information gain.

    Create a non-leaf node for it.

    Split the table into smaller tables for each value of the attribute.

    Recursively call this algorithm for each child + smaller table.

Pruning

Decision trees, like all supervised machine learning tools, are prone to overfitting.

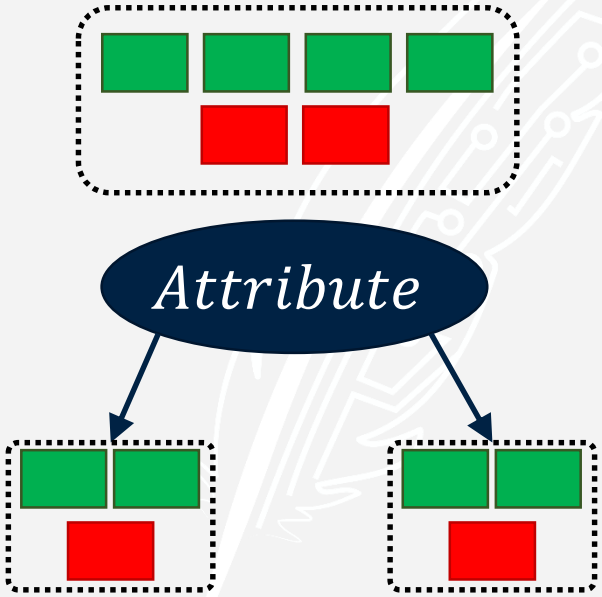
This can be corrected by pruning the tree after it has been constructed.

Pruned trees are often smaller, making them easier to understand. They usually also generalize better.

Pruning

The goal is to remove attribute nodes which only exist because of noise in the training data.

For such nodes, information gain is close to 0.



χ^2 Pruning

χ^2 pruning uses the χ^2 test for statistical significance to determine if a node should be pruned.

Given some observed distribution, we assume that it occurred at random (the null hypothesis). A statistical significance test returns a p-value, which indicates the probability that the observed distribution would occur by chance.

For p-values below a low threshold (usually 0.05), we reject the null hypothesis and assume the observed distribution did not occur by chance.

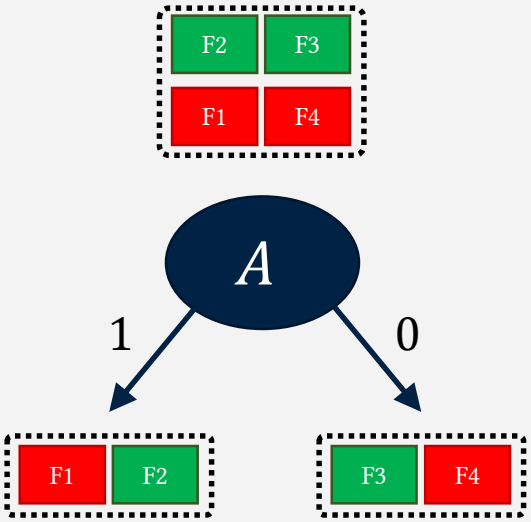
Need for Pruning?

Question: Since pruning removes nodes with low information gain, why not just avoid these nodes when building the tree in the first place?

Answer: Because this does not allow the tree to recognize meaningful combinations of attributes.

Perhaps the tree splits poorly on A1, and this allows it to then split well on A2. Such combinations would be missed.

Build then Prune Example



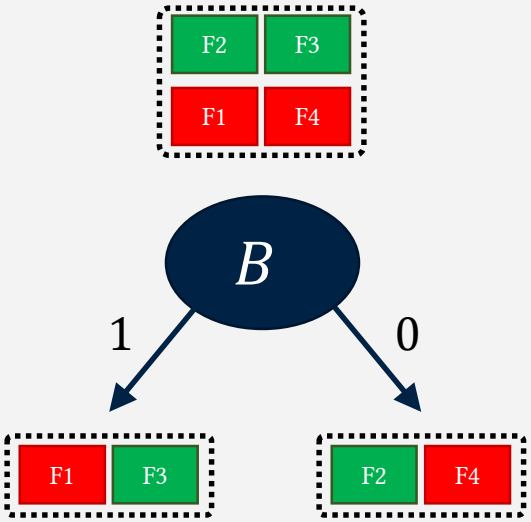
Index	A	B	A xor B
F1	1	1	0
F2	1	0	1
F3	0	1	1
F4	0	0	0

Consider the XOR function.

Would splitting on A be a good split?

No, because the resulting groups both have high entropy.

Build then Prune Example



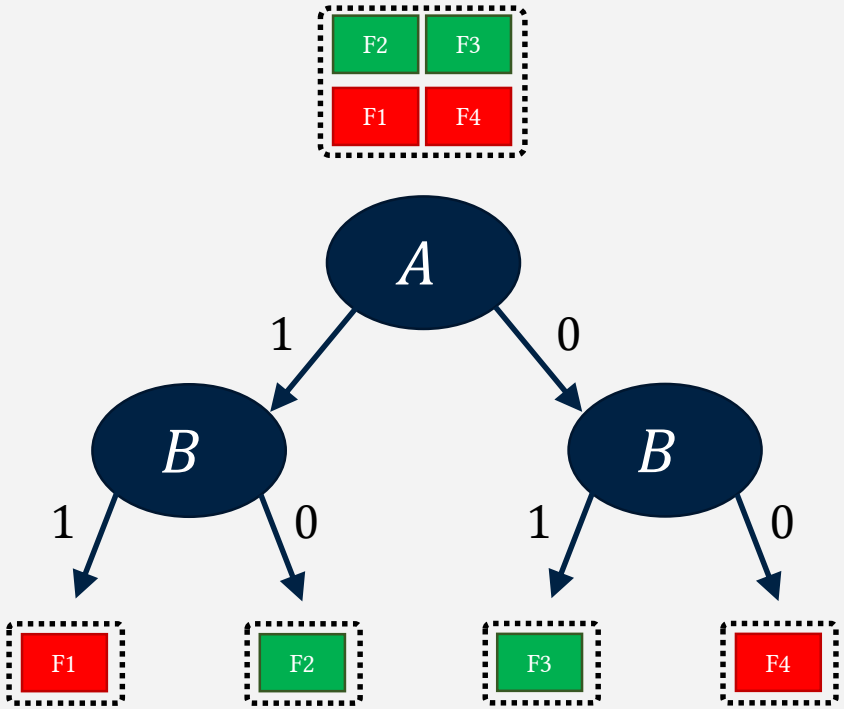
Index	A	B	A xor B
F1	1	1	0
F2	1	0	1
F3	0	1	1
F4	0	0	0

What about splitting on B?

Still no, for the same reason.

Does that mean there are no good splits we can make?

Build then Prune Example



Index	A	B	A xor B
F1	1	1	0
F2	1	0	1
F3	0	1	1
F4	0	0	0

We can still build a perfectly accurate decision tree, but sometimes we have to make suboptimal decisions early which enable optimal decisions later.