CS4710: Artificial Intelligence
Intro to Machine Learning

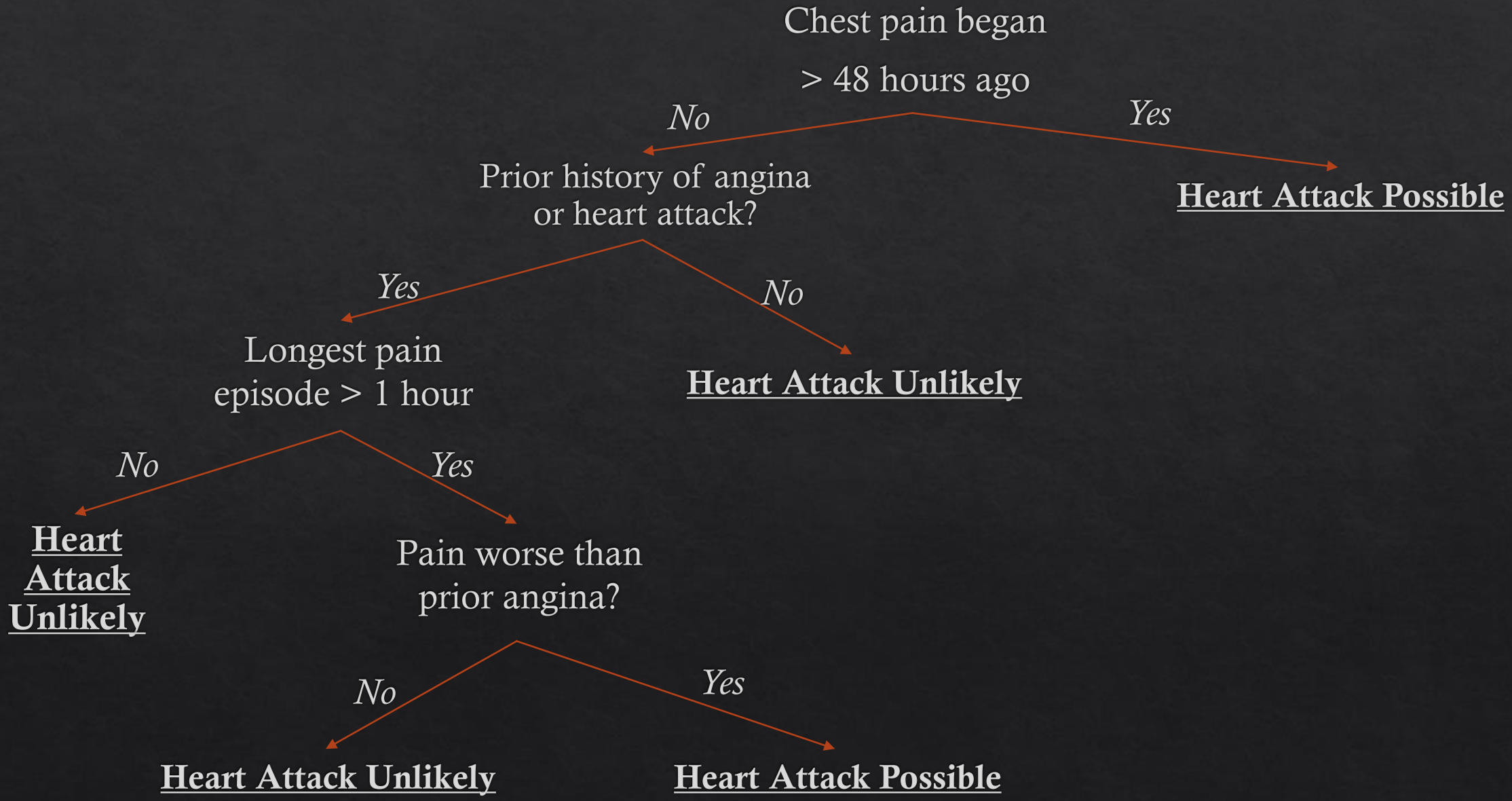
Intelligent beings (e.g., humans) are excellent at learning. How can we try to program systems to learn interesting new things?

Decision Tree Induction

Decision Trees

- A **Decision Tree** is a tree that when followed, provides a prediction of interest for a given data point
- Think of this as a flow chart
- Each node is a *feature* and based on the value of that feature, we recurse on the appropriate branch of the tree
- Leaf nodes are classifications of the data point

Example Decision Tree



Decision Tree Induction

- More common approach to learning
- Idea! Use the training set to build a decision tree
- Requirements:
  - Want the tree to be as small as possible

Similar Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	No	No	Yes	No	Yes
Jeff	No	Yes	No	Yes	No
Gail	Yes	No	Yes	Yes	Yes
Simon	No	Yes	Yes	Yes	No

Algorithm

SelectFeature(Examples){

    Pick *Feature* that best splits *Examples* into different result categories

    For each *Value* of *Feature*

        Find Subset *S* of *Examples* such that *Feature* == *Value*

        If all examples in *S* are in same result category

            Mark relevant node in the tree with that category

        Else

            Call *SelectFeature(S)*

}

Algorithm: Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	No	No	Yes	No	Yes
Jeff	No	Yes	No	Yes	No
Gail	Yes	No	Yes	Yes	Yes
Simon	No	Yes	Yes	Yes	No

SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

Mark relevant node in the tree with that category

Else

Call *SelectFeature(S)*

}

Algorithm: Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	No	No	Yes	No	Yes
Jeff	No	Yes	No	Yes	No
Gail	Yes	No	Yes	Yes	Yes
Simon	No	Yes	Yes	Yes	No

SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

Mark relevant node in the tree with that category

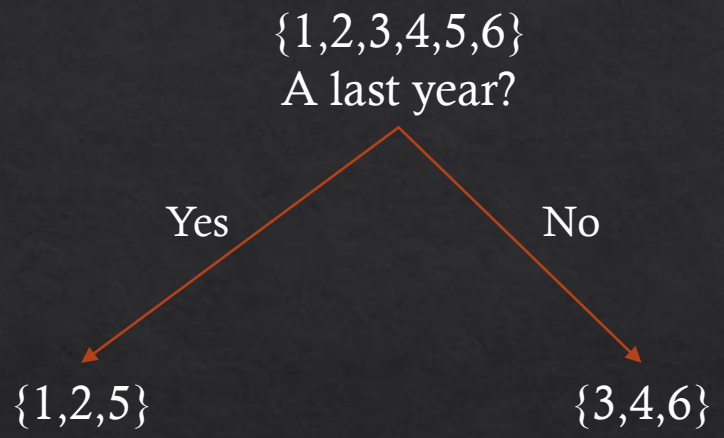
Else

Call *SelectFeature(S)*

}

Algorithm: Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	No	No	Yes	No	Yes
Jeff	No	Yes	No	Yes	No
Gail	Yes	No	Yes	Yes	Yes
Simon	No	Yes	Yes	Yes	No



SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

Mark relevant node in the tree with that category

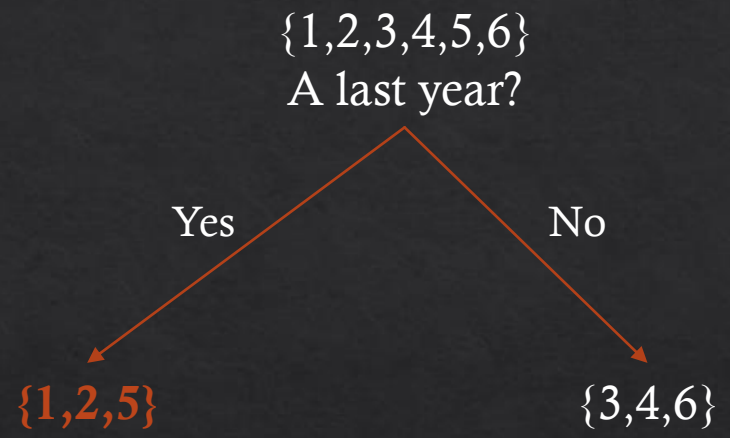
Else

Call *SelectFeature(S)*

}

Algorithm: Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	No	No	Yes	No	Yes
Jeff	No	Yes	No	Yes	No
Gail	Yes	No	Yes	Yes	Yes
Simon	No	Yes	Yes	Yes	No



SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

Mark relevant node in the tree with that category

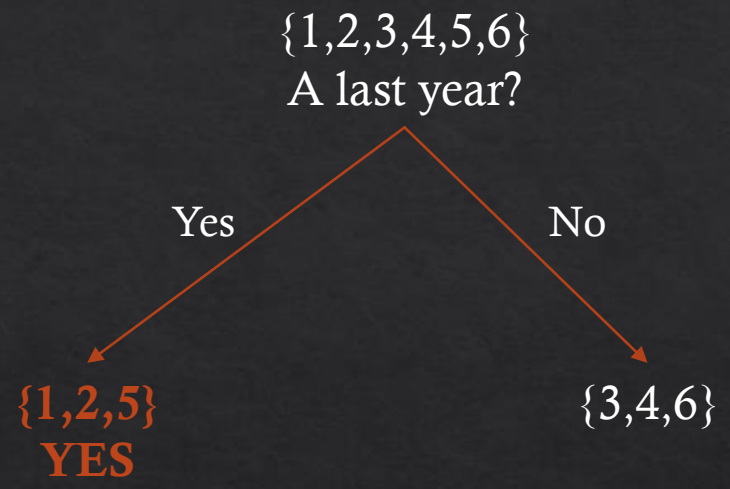
Else

Call *SelectFeature(S)*

}

Algorithm: Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	No	No	Yes	No	Yes
Jeff	No	Yes	No	Yes	No
Gail	Yes	No	Yes	Yes	Yes
Simon	No	Yes	Yes	Yes	No



SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

Mark relevant node in the tree with that category

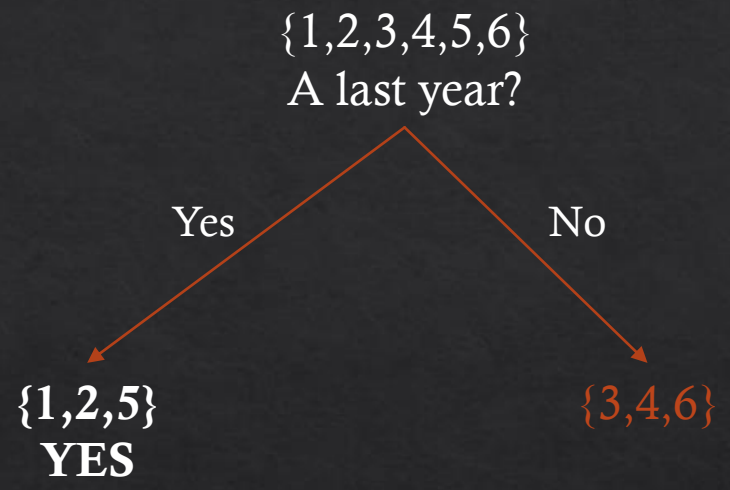
Else

Call *SelectFeature(S)*

}

Algorithm: Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	No	No	Yes	No	Yes
Jeff	No	Yes	No	Yes	No
Gail	Yes	No	Yes	Yes	Yes
Simon	No	Yes	Yes	Yes	No



SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

Mark relevant node in the tree with that category

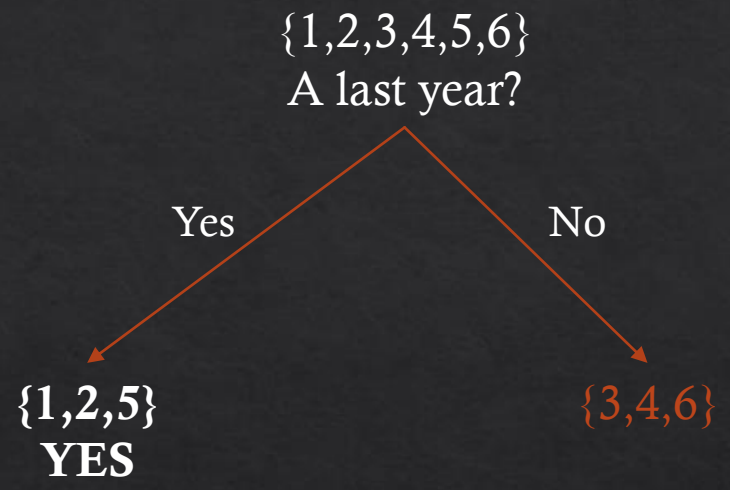
Else

Call *SelectFeature(S)*

}

Algorithm: Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	**No**	**No**	**Yes**	**No**	**Yes**
Jeff	**No**	**Yes**	**No**	**Yes**	**No**
Gail	Yes	No	Yes	Yes	Yes
Simon	**No**	**Yes**	**Yes**	**Yes**	**No**



SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

Mark relevant node in the tree with that category

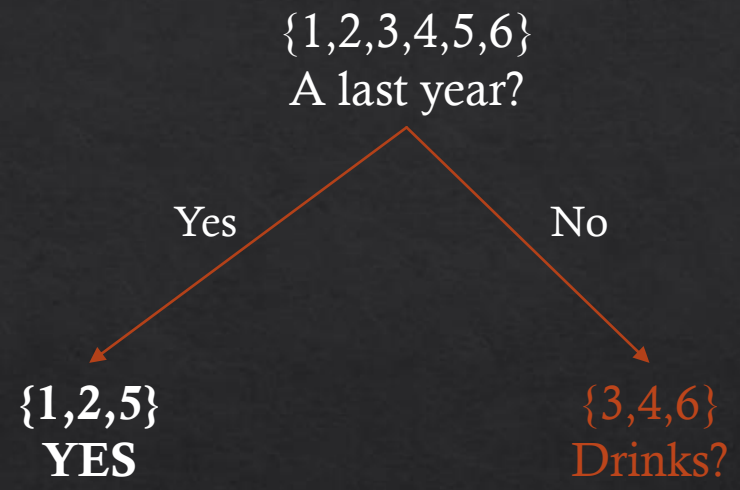
Else

Call *SelectFeature(S)*

}

Algorithm: Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	**No**	**No**	**Yes**	**No**	**Yes**
Jeff	**No**	**Yes**	**No**	**Yes**	**No**
Gail	Yes	No	Yes	Yes	Yes
Simon	**No**	**Yes**	**Yes**	**Yes**	**No**



SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

Mark relevant node in the tree with that category

Else

Call *SelectFeature(S)*

}

Algorithm: Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	**No**	**No**	**Yes**	**No**	**Yes**
Jeff	**No**	**Yes**	**No**	**Yes**	**No**
Gail	Yes	No	Yes	Yes	Yes
Simon	**No**	**Yes**	**Yes**	**Yes**	**No**

SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

Mark relevant node in the tree with that category

Else

Call *SelectFeature(S)*

}

Algorithm: Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	**No**	**No**	**Yes**	**No**	**Yes**
Jeff	**No**	**Yes**	**No**	**Yes**	**No**
Gail	Yes	No	Yes	Yes	Yes
Simon	**No**	**Yes**	**Yes**	**Yes**	**No**

SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

Mark relevant node in the tree with that category

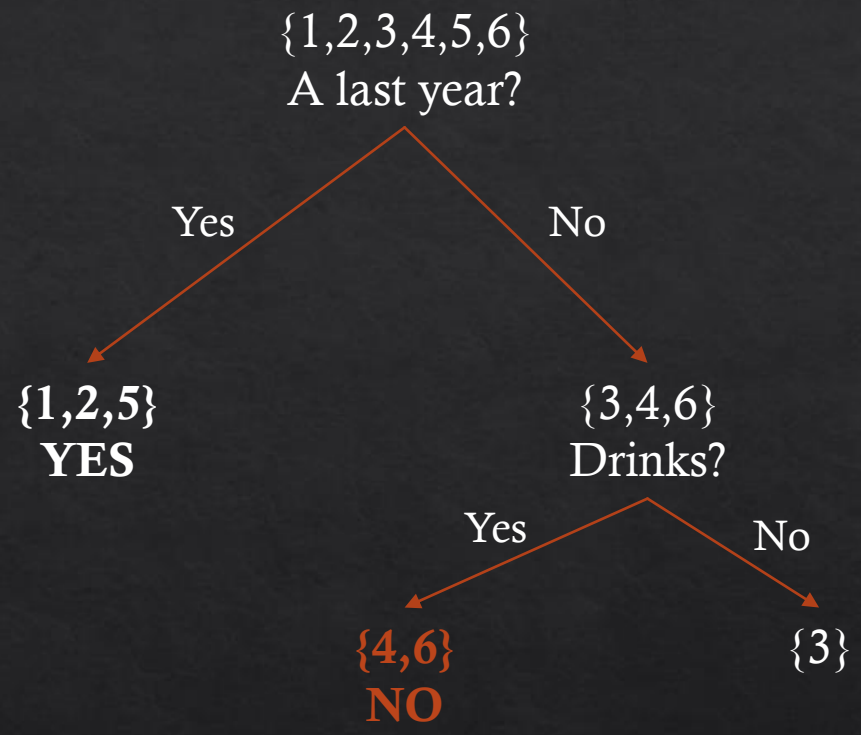
Else

Call *SelectFeature(S)*

}

Algorithm: Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	**No**	**No**	**Yes**	**No**	**Yes**
Jeff	**No**	**Yes**	**No**	**Yes**	**No**
Gail	Yes	No	Yes	Yes	Yes
Simon	**No**	**Yes**	**Yes**	**Yes**	**No**



SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

Mark relevant node in the tree with that category

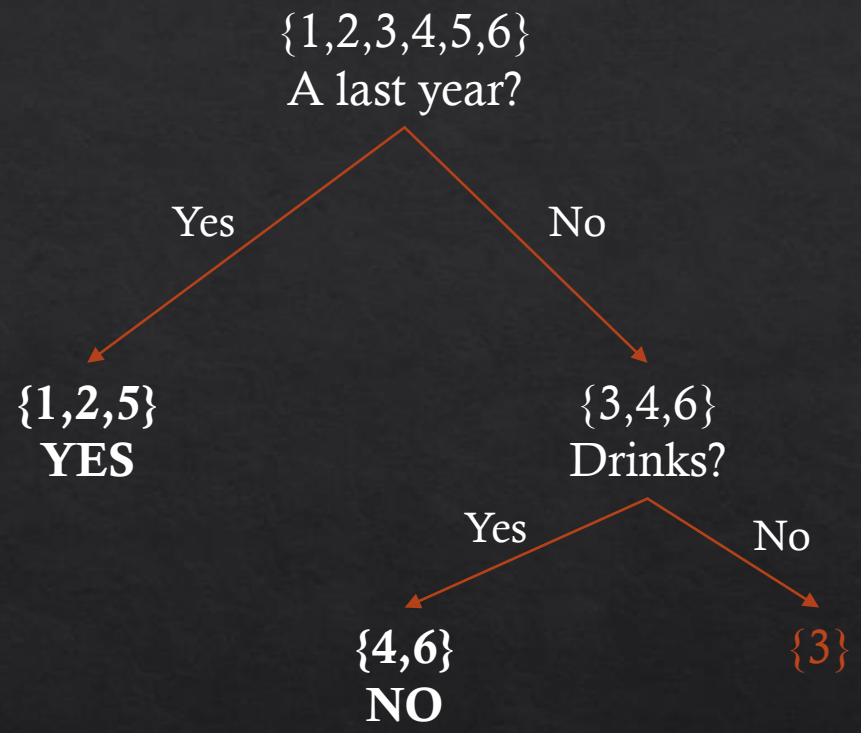
Else

Call *SelectFeature(S)*

}

Algorithm: Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	**No**	**No**	**Yes**	**No**	**Yes**
Jeff	**No**	**Yes**	**No**	**Yes**	**No**
Gail	Yes	No	Yes	Yes	Yes
Simon	**No**	**Yes**	**Yes**	**Yes**	**No**



SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

Mark relevant node in the tree with that category

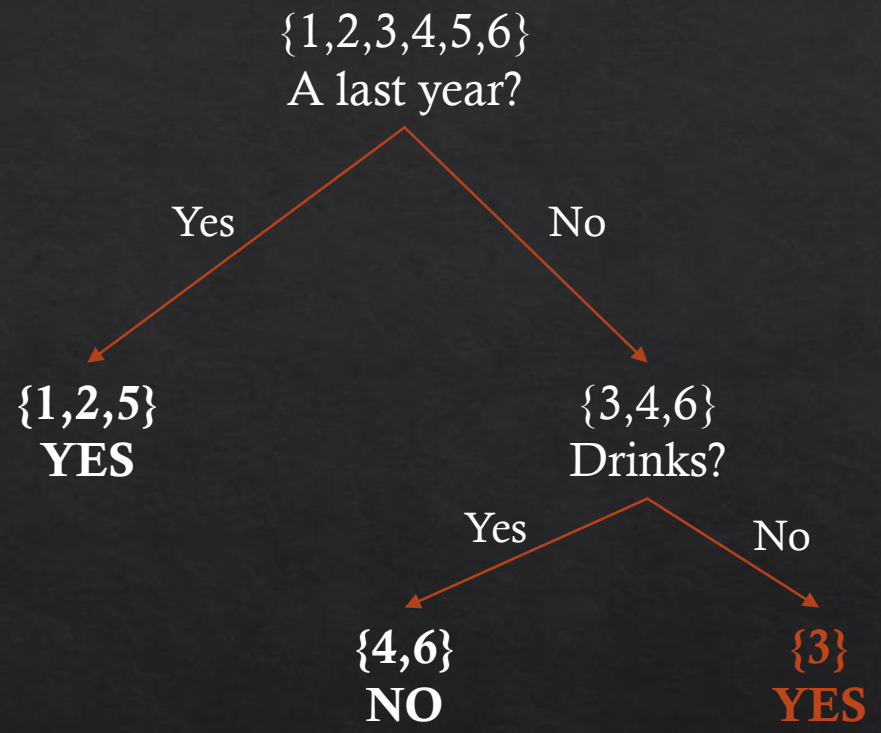
Else

Call *SelectFeature(S)*

}

Algorithm: Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	**No**	**No**	**Yes**	**No**	**Yes**
Jeff	**No**	**Yes**	**No**	**Yes**	**No**
Gail	Yes	No	Yes	Yes	Yes
Simon	**No**	**Yes**	**Yes**	**Yes**	**No**



SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

Mark relevant node in the tree with that category

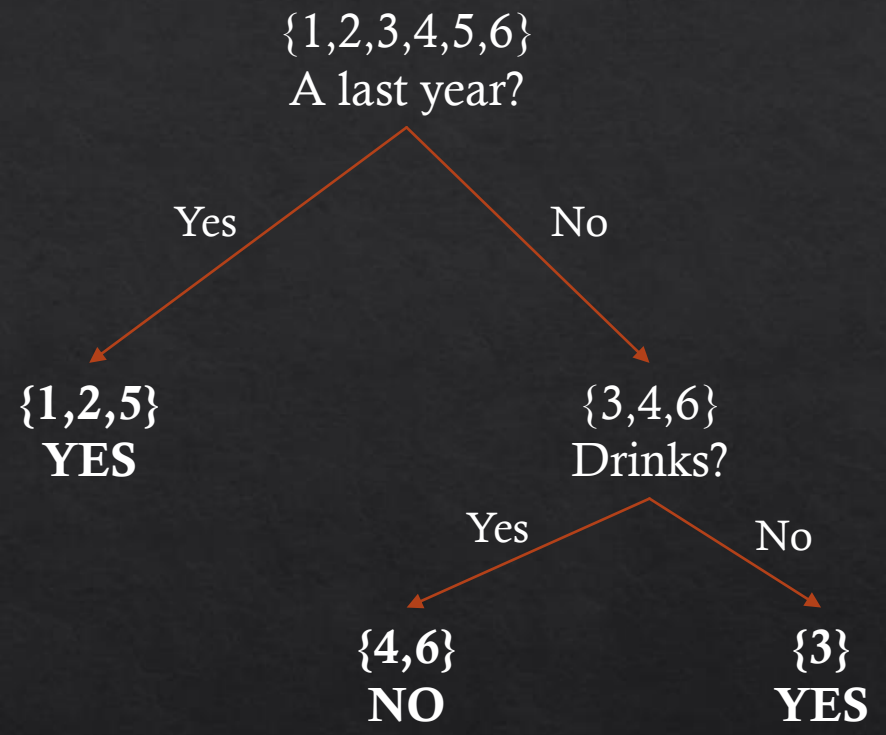
Else

Call *SelectFeature(S)*

}

Algorithm: Example

Student	A last year?	Male?	Works Hard?	Drinks?	A this year?
Richard	Yes	Yes	No	Yes	Yes
Allen	Yes	Yes	Yes	No	Yes
Alison	**No**	**No**	**Yes**	**No**	**Yes**
Jeff	**No**	**Yes**	**No**	**Yes**	**No**
Gail	Yes	No	Yes	Yes	Yes
Simon	**No**	**Yes**	**Yes**	**Yes**	**No**



SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

Mark relevant node in the tree with that category

Else

Call *SelectFeature(S)*

}

Decision Tree Induction

The ID3 Algorithm

ID3 Algorithm

SelectFeature(Examples){

Pick *Feature* that best splits *Examples* into different result categories ***(HOW TO DO THIS?)***

For each *Value* of *Feature*

Find Subset *S* of *Examples* such that *Feature* == *Value*

If all examples in *S* are in same result category

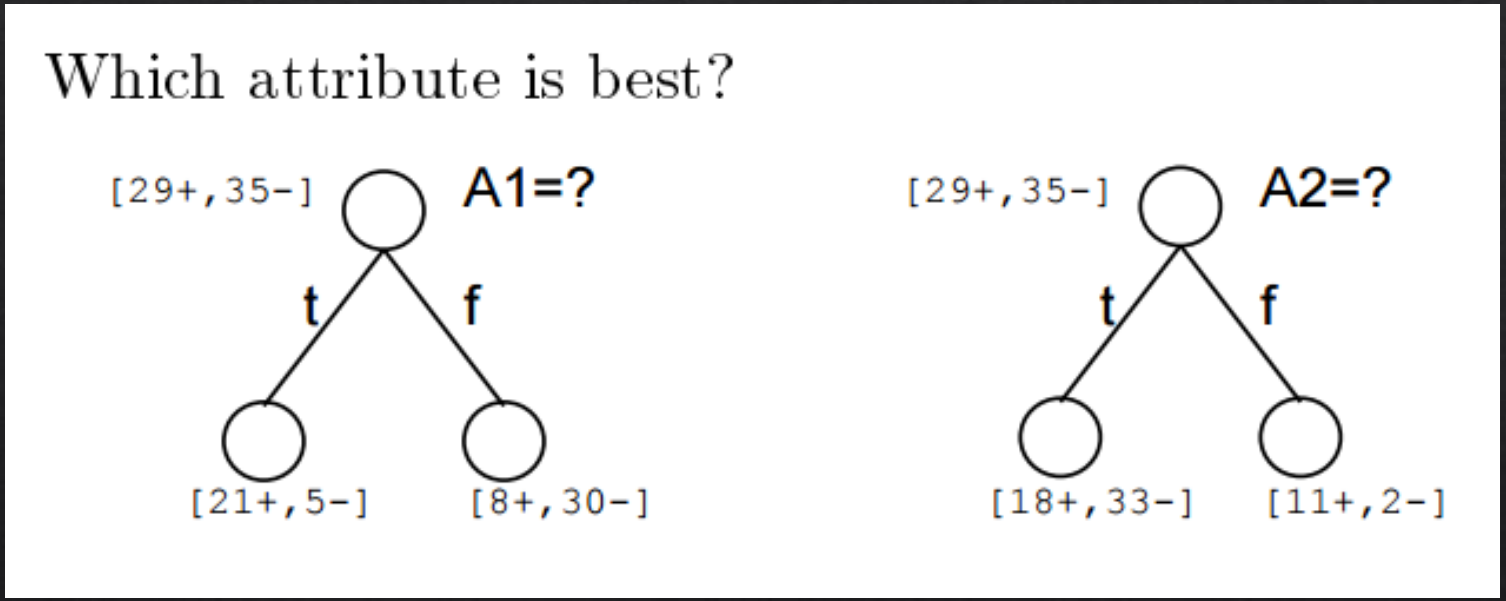
Mark relevant node in the tree with that category

Else

Call *SelectFeature(S)*

}

ID3 Algorithm: Which Attribute Best?



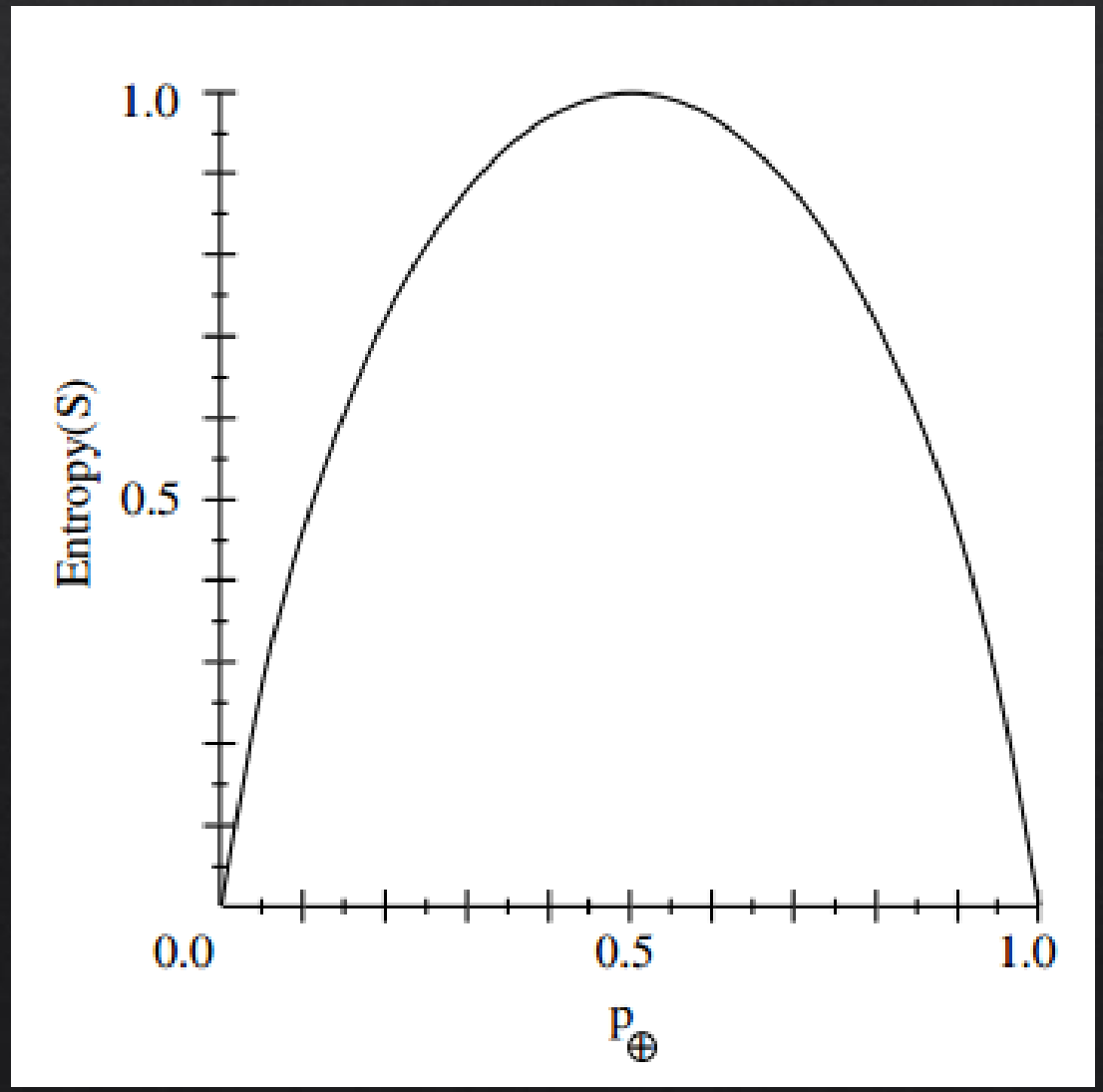
A1 and *A2* are the **features**

29+, 35- means **29 positive training examples** and **35 negative training examples**

T and *F* (on edges) are the **values** of the features (so Boolean features in this image)

…so which feature split is better?

ID3 Algorithm: Entropy



Suppose:

S is sample of training examples

P_+ is proportion of positive examples in S

P_- is proportion of negative examples in S

Entropy measure the impurity of S:

$$\text{Entropy}(S) = -P_+ * \log_2(P_+) - P_- * \log_2(P_-)$$

Entropy: Information Theory

Entropy(S) is the **expected number of bits** needed to encode class (+ or -) of randomly drawn member of S (under optimal, shortest-length code)

Think **Huffman Codes** here

Optimal length code assigns $-\log_2(p)$ bits to message having probability of *p*

So, over all possibilities we take a weighted average:

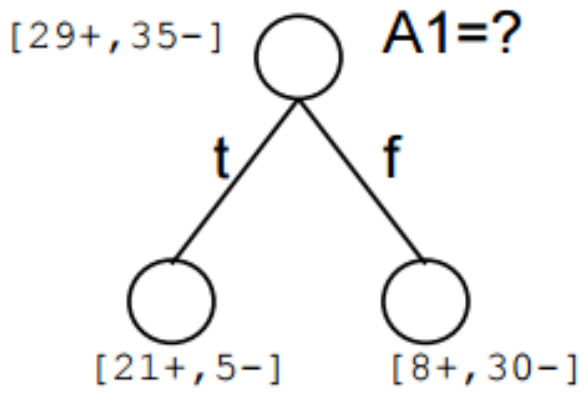
$$\text{Entropy}(S) = -P_+ * \log_2(P_+) - P_- * \log_2(P_-)$$

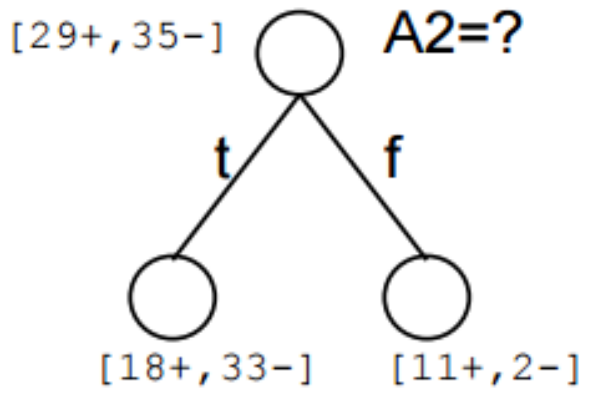
Information Gain

Information Gain is the amount the entropy has dropped given that you split on a specific feature at this step of the algorithm.

Gain(S,A) = expected **reduction in entropy** due to sorting on *A*

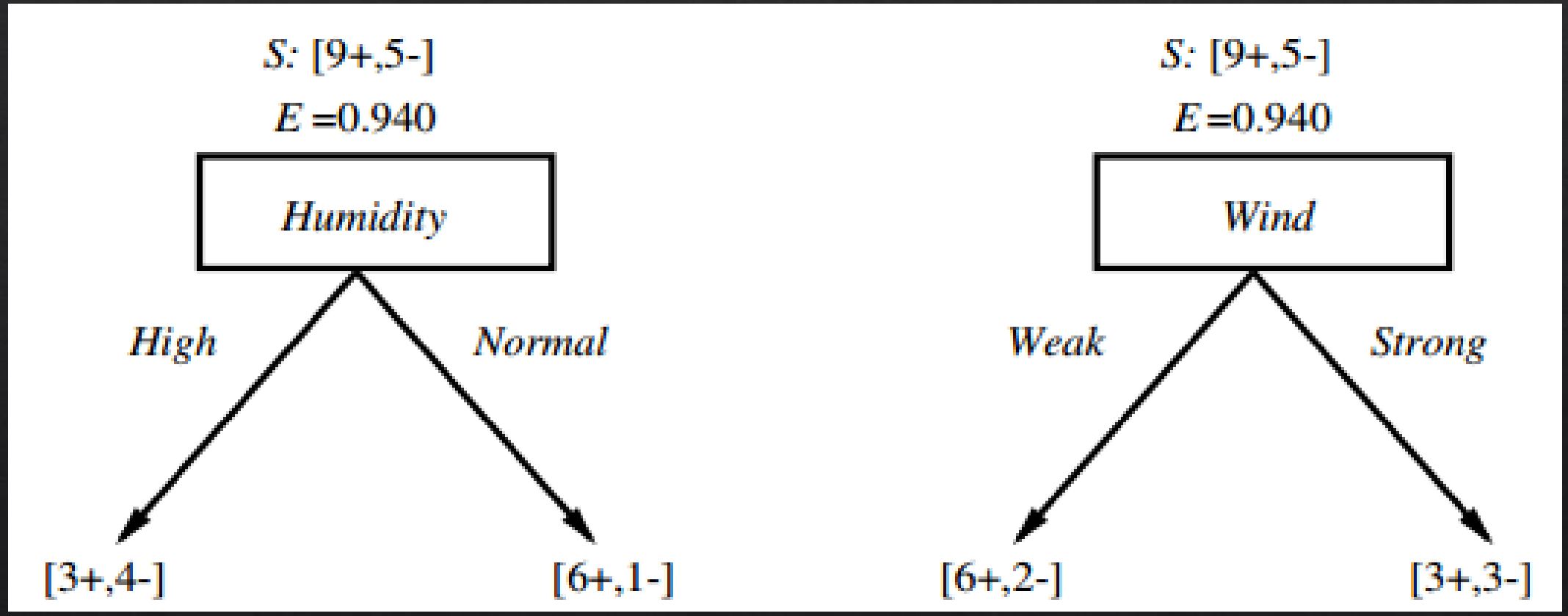
$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$



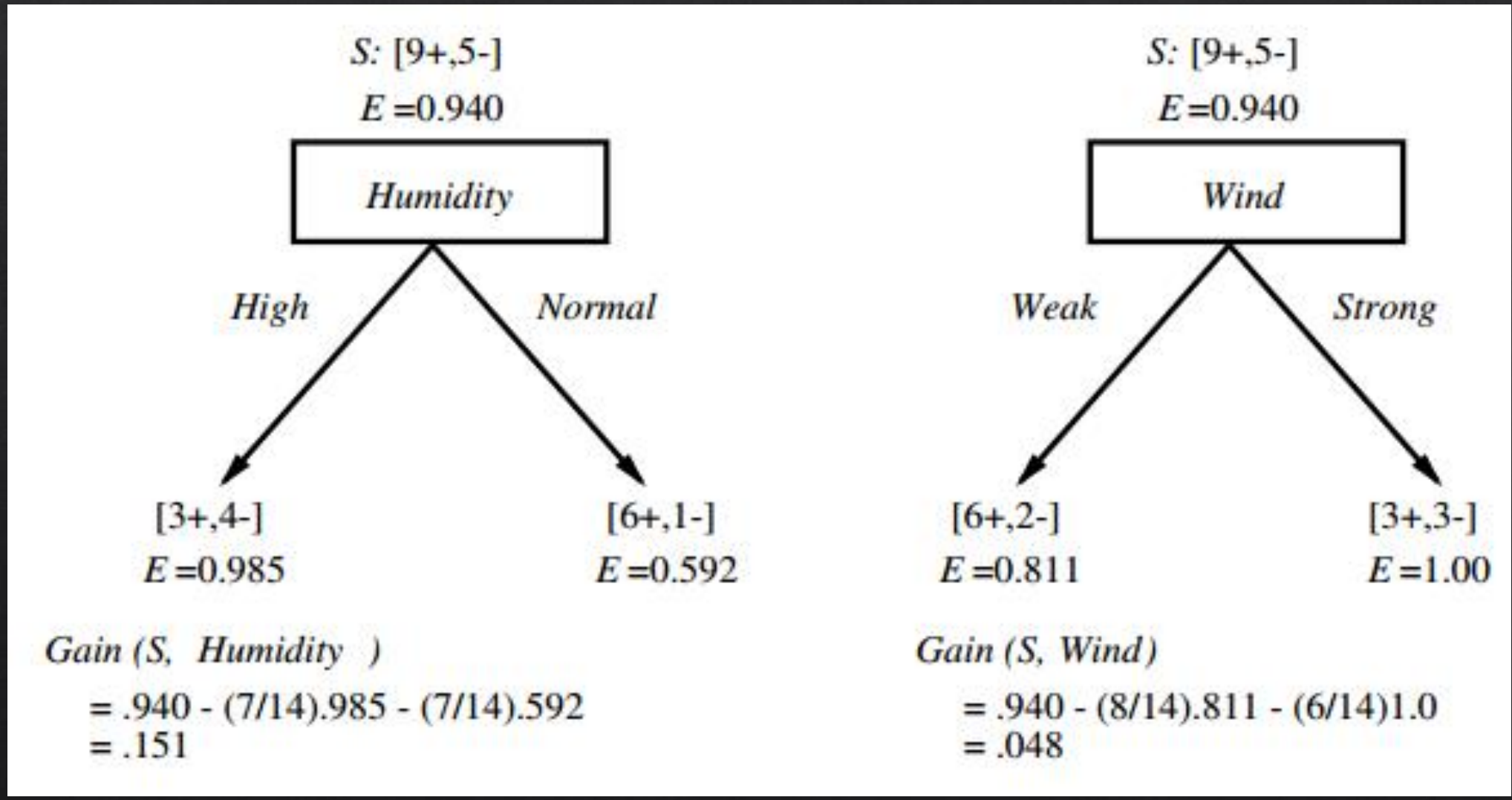


Which Attribute Better?

Try it on your own!



Which Attribute Better?

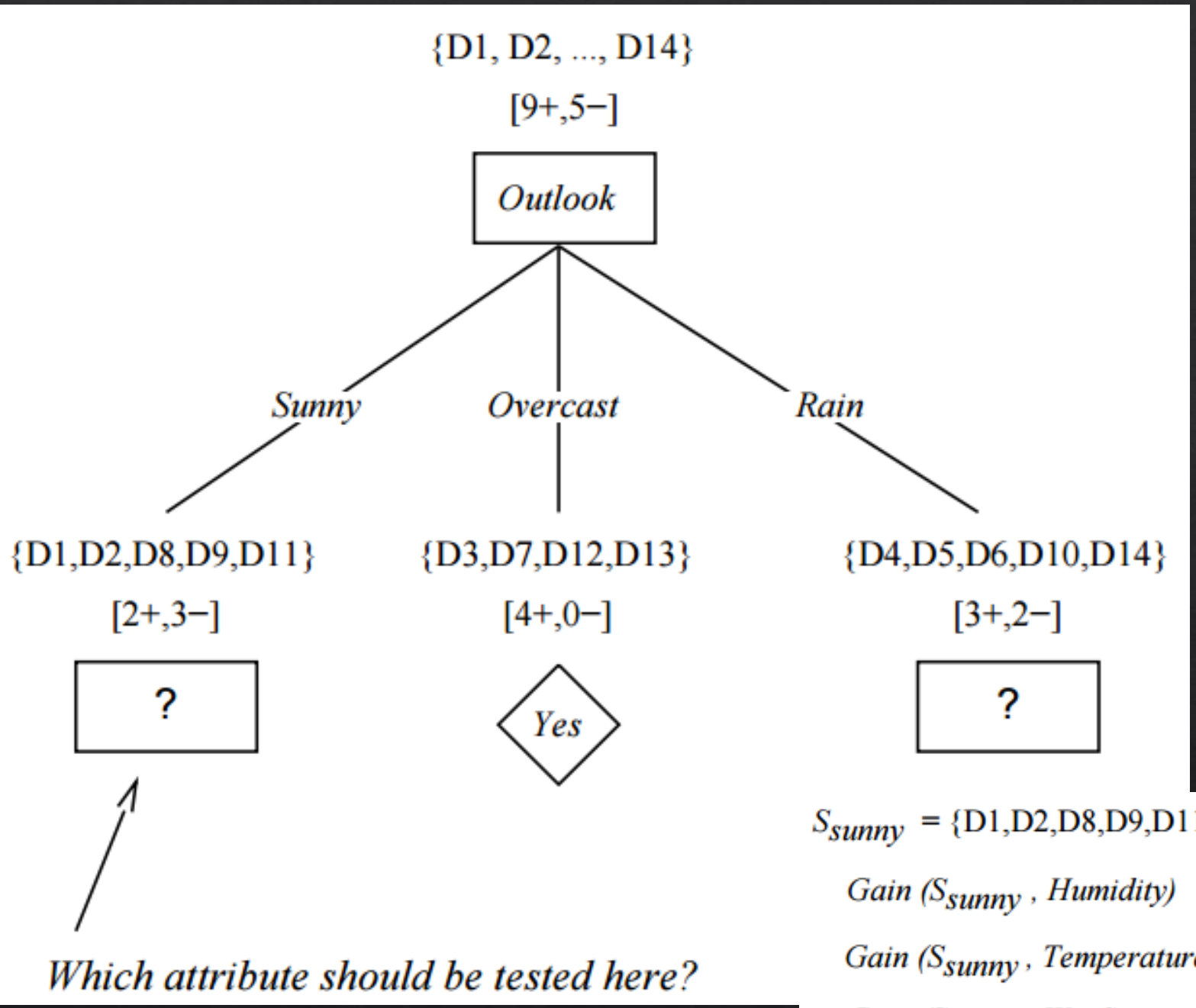


Gain (S, Humidity)

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

Gain (S, Wind)

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



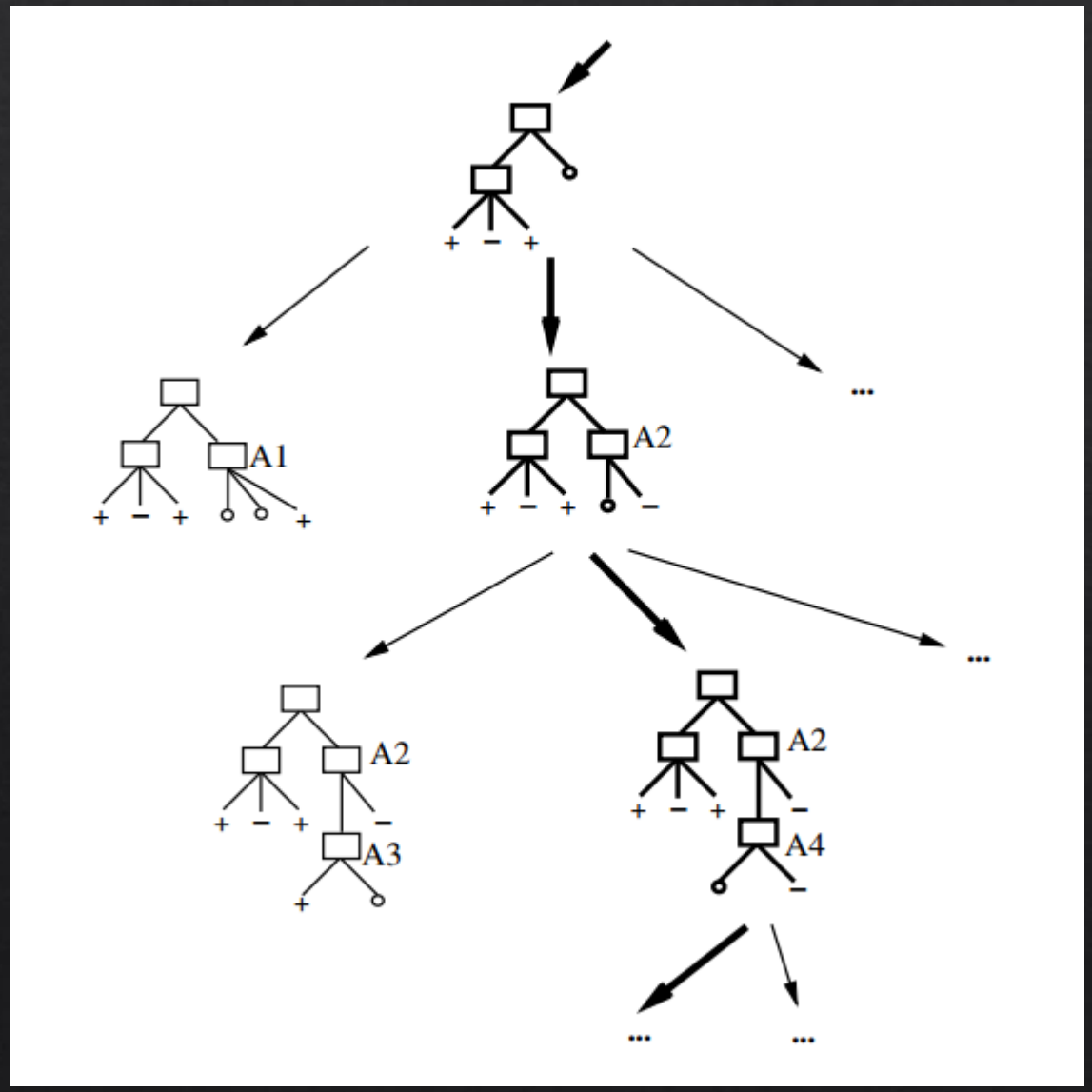
S_{sunny} = {D1,D2,D8,D9,D11}

Gain (S_{sunny}, *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

Gain (S_{sunny}, *Temperature*) = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

Gain (S_{sunny}, *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

Thinking of ID3 as Space-Search



Hypothesis space is **complete**

- The best decision tree MUST be in there

ID3 outputs a **specific decision tree** though

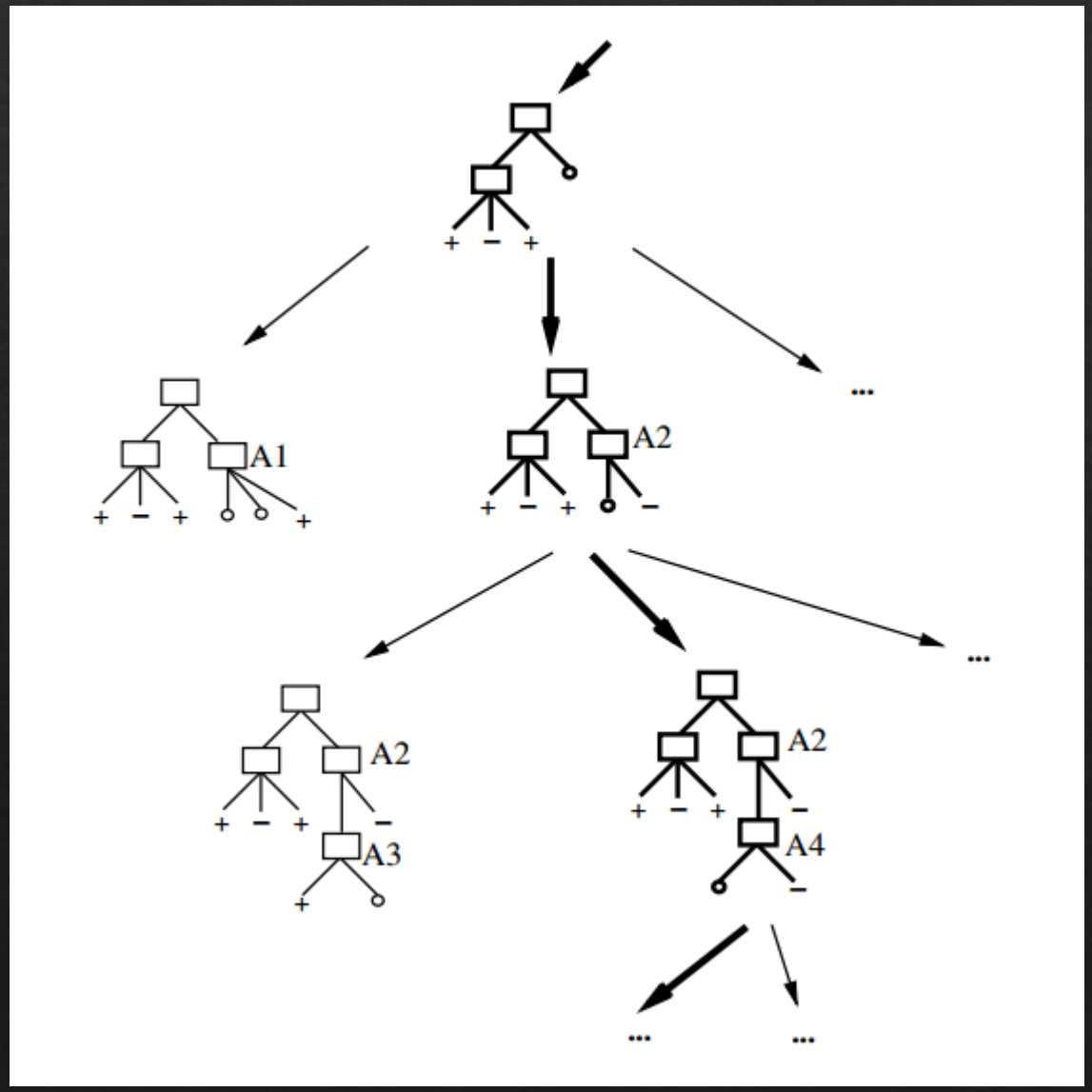
Is it the best one?

No back-tracking

Local-minima (so probably not optimal)

Search choices are **stochastic**, so pretty robust to noisy data

ID3 is Biased!

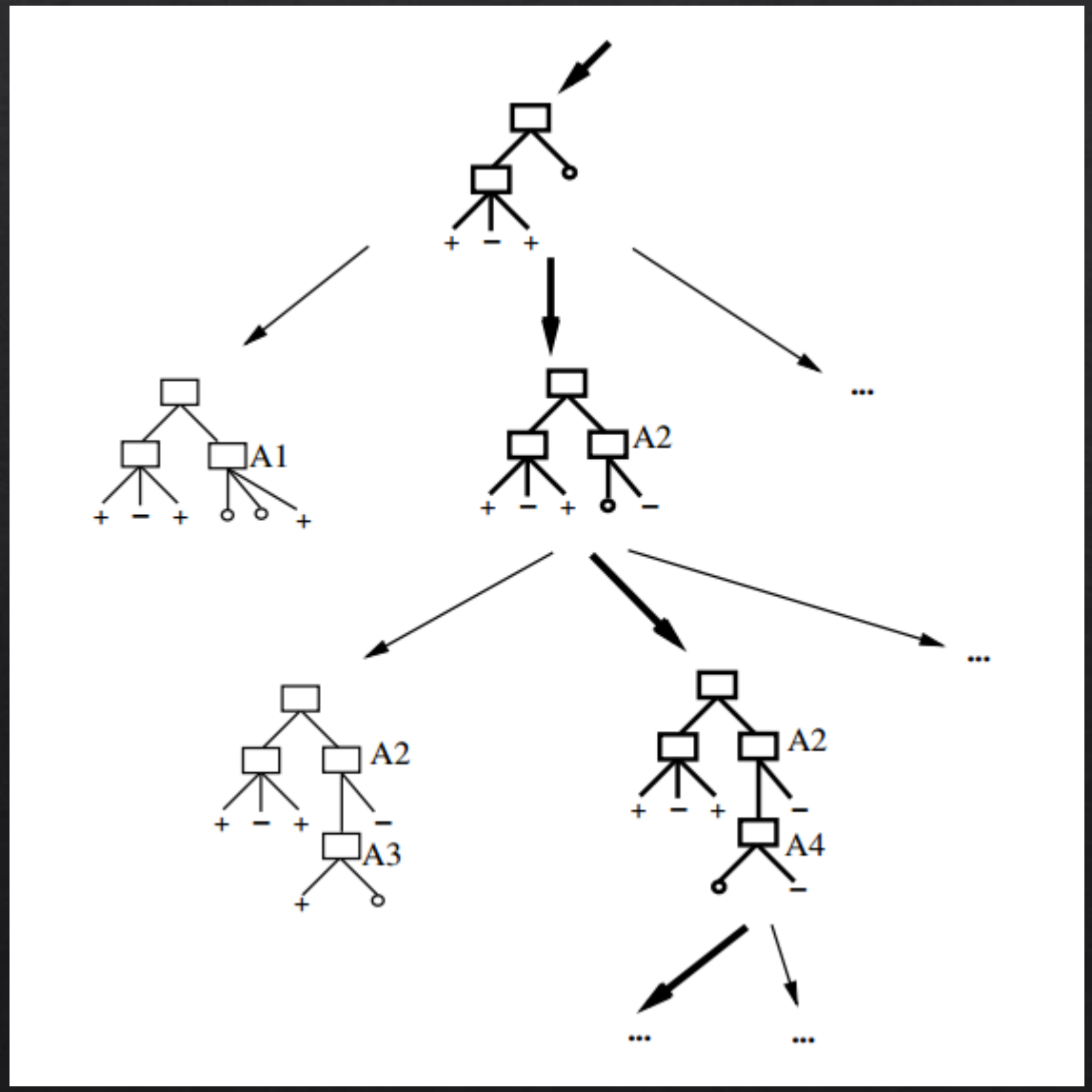


Preference for **shorter trees** with high information gains near the root. ID3 is a **greedy algorithm**.

Bias is a preference, not a restriction (i.e., defines the choice we make when searching, but whole tree is available)

Occam's Razor: Prefer the shortest hypothesis

ID3: Why Preference for Shorter Hypothesis?



Pros:

- Fewer short hypotheses
- Shorter = more general
- Longer hypothesis might be coincidence

Cons:

- There are many ways to define short hypotheses
- e.g., all trees with a prime number of nodes that use attributes beginning with letter Z

ID3: Overfitting

Overfitting is a problem in many AI algorithms in which learned rules infer noisy variations in data as real underlying differences.

Consider adding this example to earlier tree:

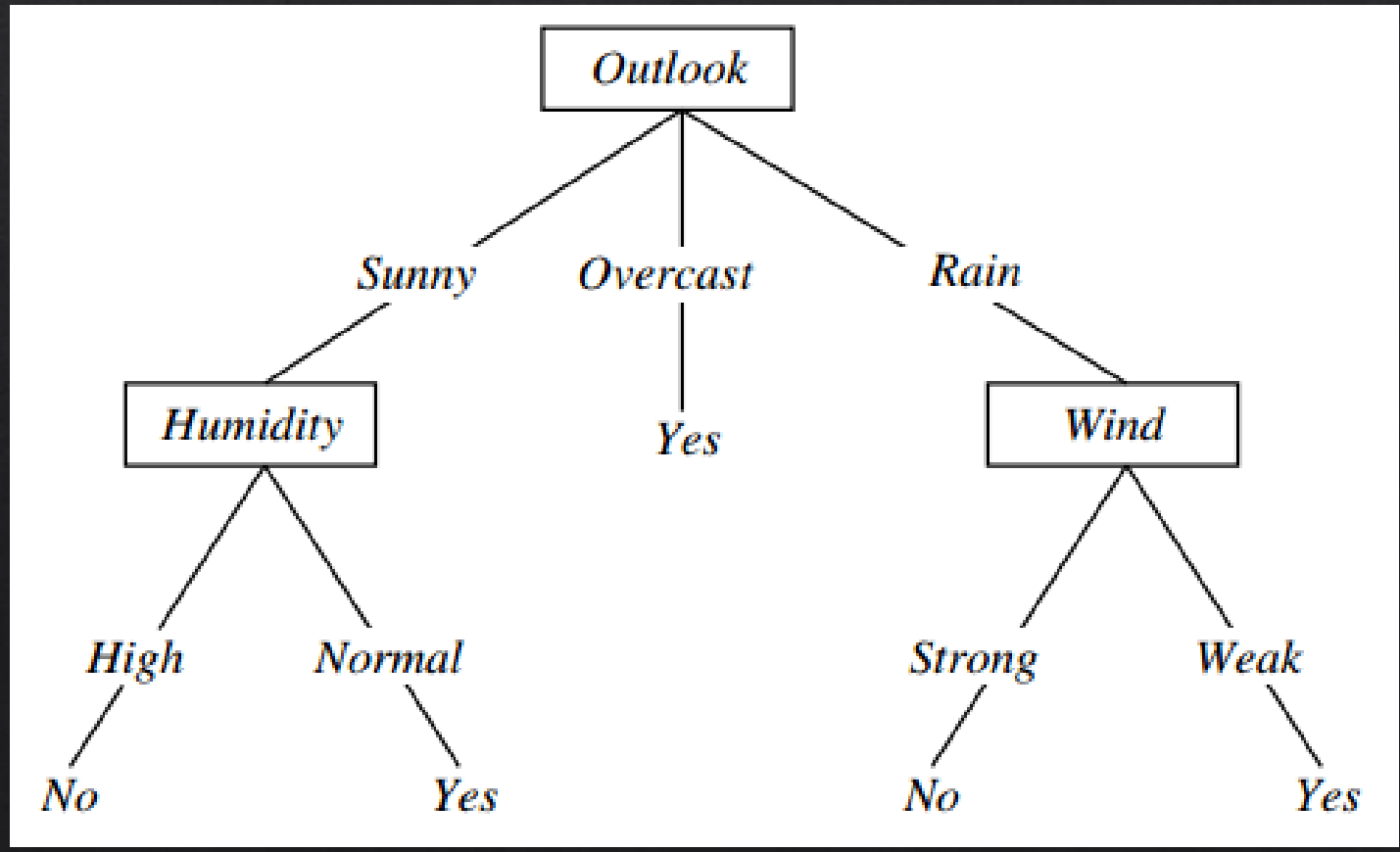
[Sunny, Hot,

Normal Humidity,

Strong Wind,

Play Tennis = NO]

What is the effect on this tree →



Defining Overfitting

Consider a hypothesis *h* over both training data and actual distribution:

Error over training data: *TrainError(h)* Error over entire distribution ***D***: *DError(h)*

Hypothesis *h* **overfits** the training data if there is another hypothesis h' such that:

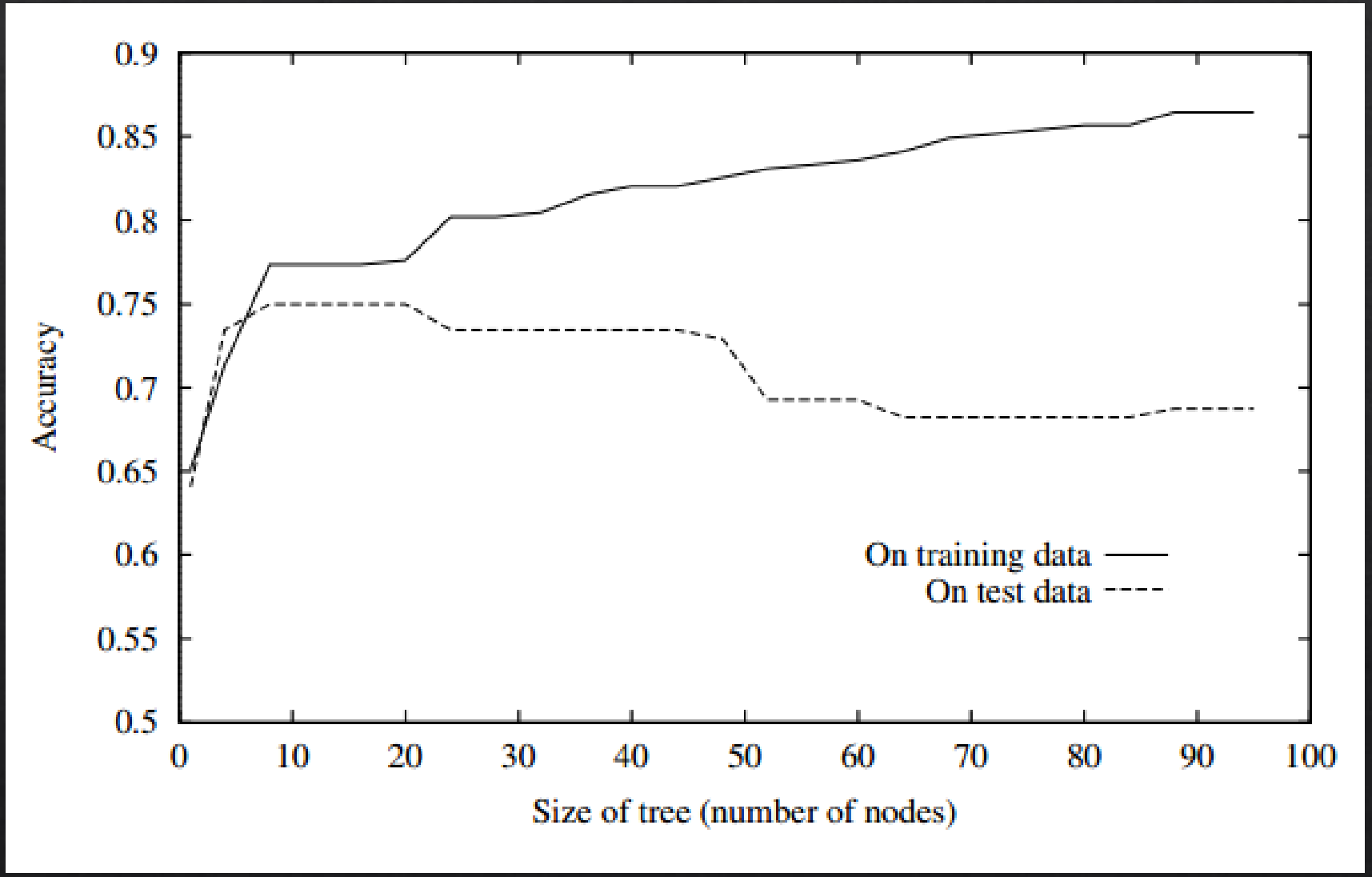
$$TrainError(h) < TrainError(h')$$

$$AND$$

$$DError(h) > DError(h')$$

Why does this make sense?

Defining Overfitting



Avoiding Overfitting

1. Stop growing when data split not statistically significant.
    1. E.g., 101 positive examples and 1 negative at a node. No need to split here.
2. Grow full tree, then post-prune

Additionally:

Split data into a **training set** and a **validation (or test) set**.

Build tree (train) on training set only. Validate on validation set only.

Greedily remove rules in tree until you improve accuracy on **validation set.**

Post-Pruning to fight Overfitting