UNSW SYDNEY | Australia's Global University

COMP9321

Data Services Engineering

Term 1, 2019

Week 6 Lecture 2

Supervised Learning

COMP9321 2019T1

Decision Tree

Supervised Learning

Background

- Decision trees have a long history in machine learning
- The first popular algorithm dates back to 1979
- Very popular in many real world problems
  - Intuitive to understand
  - Easy to build

Motivation

- How do people make decisions?
  - Consider a variety of factors
  - Follow a logical path of checks

- An Example
  - Should I eat at this restaurant?
    - If there is no wait
      - Yes
  - If there is short wait and I am hungry
    - Yes
  - Else
    - No

Advantages

Handling of categorical variables
Handling of missing values and unknown lables
Detection of nonlinear relationships
Visualization and interpretation in decision trees

Decision Tree

Goal: Categorization

Given an event, predict is category. Examples:

- Who won a given ball game?
- How should we file a given email?
- What word sense was intended for a given occurrence of a word?

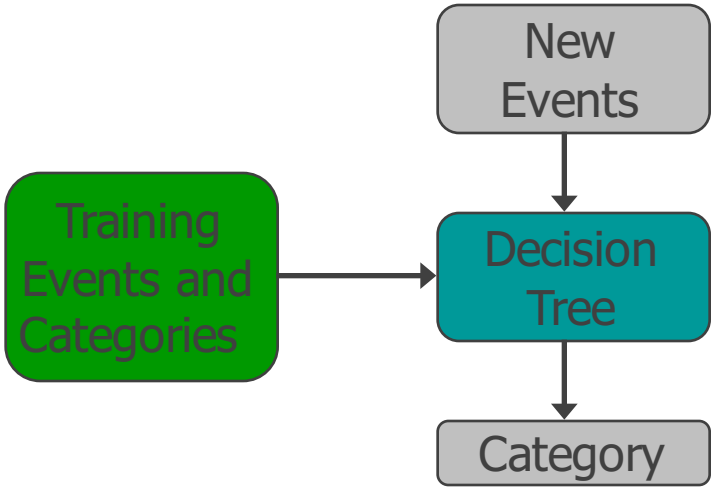
Event = list of features. Examples:

- Ball game: Which players were on offense?
- Email: Who sent the email?
- Disambiguation: What was the preceding word?

Introduction

Use a decision tree to predict categories for new events.

Use training data to build the decision tree.



Introduction

A decision tree has 2 kinds of nodes

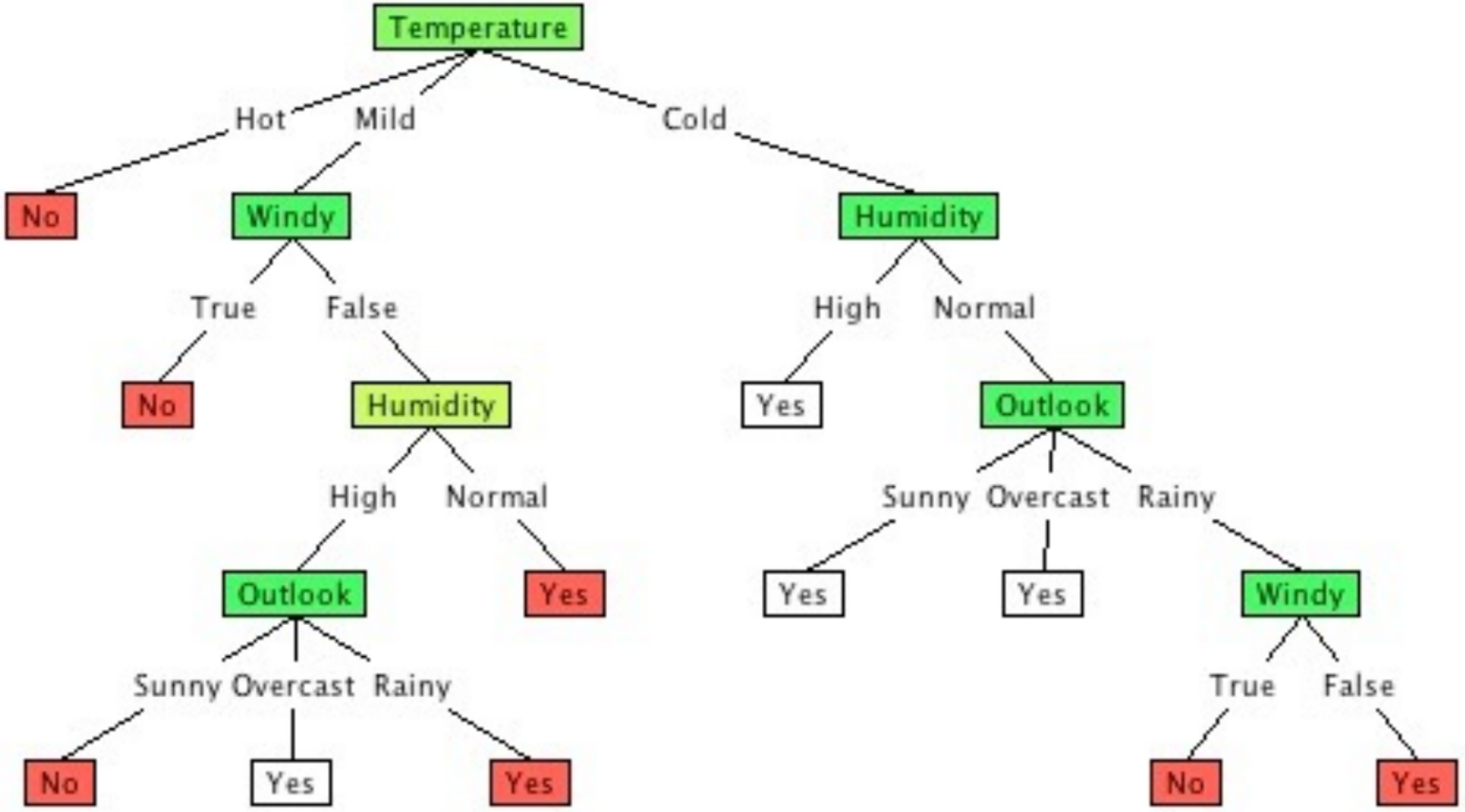
1. Each leaf node has a class label, determined by majority vote of training examples reaching that leaf.
2. Each internal node is a question on features. It branches out according to the answers.

Decision Tree for Play Tennis

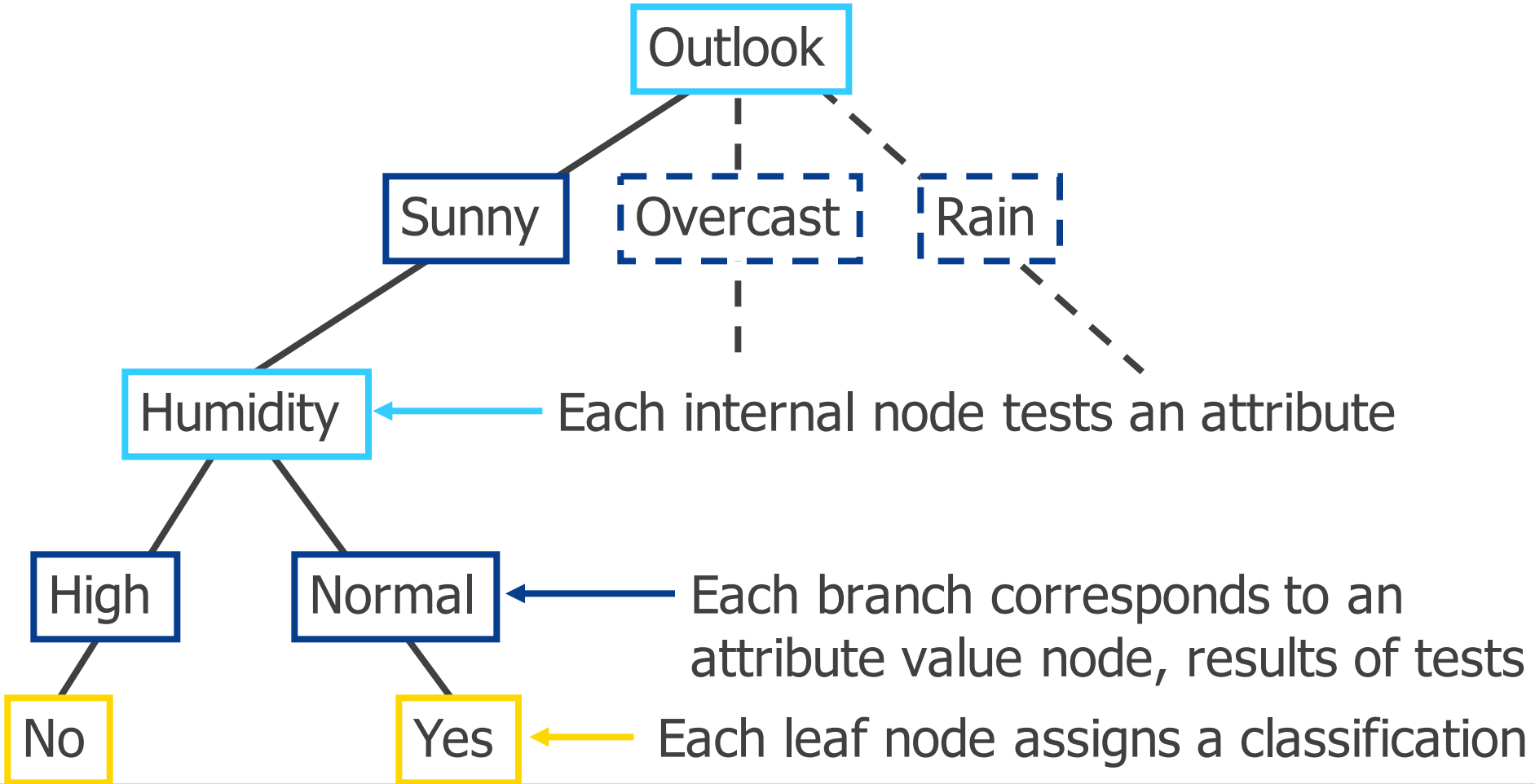
Play Tennis	Outlook	Temperature	Humidity	Windy
No	Sunny	Hot	High	No
No	Sunny	Hot	High	Yes
Yes	Overcast	Hot	High	No
Yes	Rainy	Mild	High	No
Yes	Rainy	Cold	Normal	No

If temperature is not hot → Play
If outlook is overcast → Play
Otherwise → Don't play tennis

Decision Tree for Play Tennis



Decision Tree for PlayTennis - Structure



Function Class

What functions can decision trees model?

Non-linear: very powerful function class

A decision tree can encode any Boolean function

Proof

- Given a truth table for a function
- Construct a path in the tree for each row of the table
- Given a row as input, follow that path to the desired leaf (output)

Problem: exponentially large trees!

Trimming the Tree

Can we produce smaller decision trees for functions?

- Yes (Possible)

Key decision factors

Counter examples

- Parity function: Return 1 on even inputs, 0 on odd inputs
- Majority function: Return 1 if more than half of inputs are 1

Bias-Variance Trade-off

- Bias: Representation Power of Decision Trees
- Variance: require a sample size exponential in depth

What Makes a Good Tree

Small tree:

- Occam's razor: a guideline to help us choose. Simpler is better
- Avoids over-fitting

What Makes a Good Tree

- Occam's razor: a guideline to help us choose. Simpler is better

What Makes a Good Tree

- Occam's razor: a guideline to help us choose. Simpler is better

Prank explanation requires:	**Licking explanation requires:**
1. Human (not observed)	1. Cat (observed)
2. Ability to enter house (unknown)	2. Licking (observed)
3. Motive to play prank (unknown)	
4. Leaving no other trace (observed)	

https://www.youtube.com/watch?v=AQNxNeQ9cxw

What Makes a Good Tree

Small tree:

- Occam's razor: a guideline to help us choose. Simpler is better
- Avoids over-fitting

A decision tree may be human readable, but not use human logic!

How do we build small trees that accurately capture data?

Learning an optimal decision tree is computationally intractable

Greedy Algorithm

We can get good trees by simple greedy algorithms

Adjustments are usually to fix greedy selection problems

Recursive:

1. Select the “best” variable, and generate child nodes: One for each possible value;
2. Partition samples using the possible values, and assign these subsets of samples to the child nodes;
3. Repeat for each child node until all samples associated with a node that are either all positive or all negative.

Variable Selection

The best variable for partition

- The most informative variable
- Select the variable that is most informative about the labels
- Classification error?

Example: $\mathbb{P}(Y = 1|X_1 = 1) = 0.75$ v.s. $\mathbb{P}(Y = 1|X_2 = 1) = 0.55$

Which to Choose?

Information Theory

The quantification of information

Founded by Claude Shannon

Basic concepts:

Entropy: $H(X) = -\sum_x \mathbb{P}(X = x) \log \mathbb{P}(X = x)$

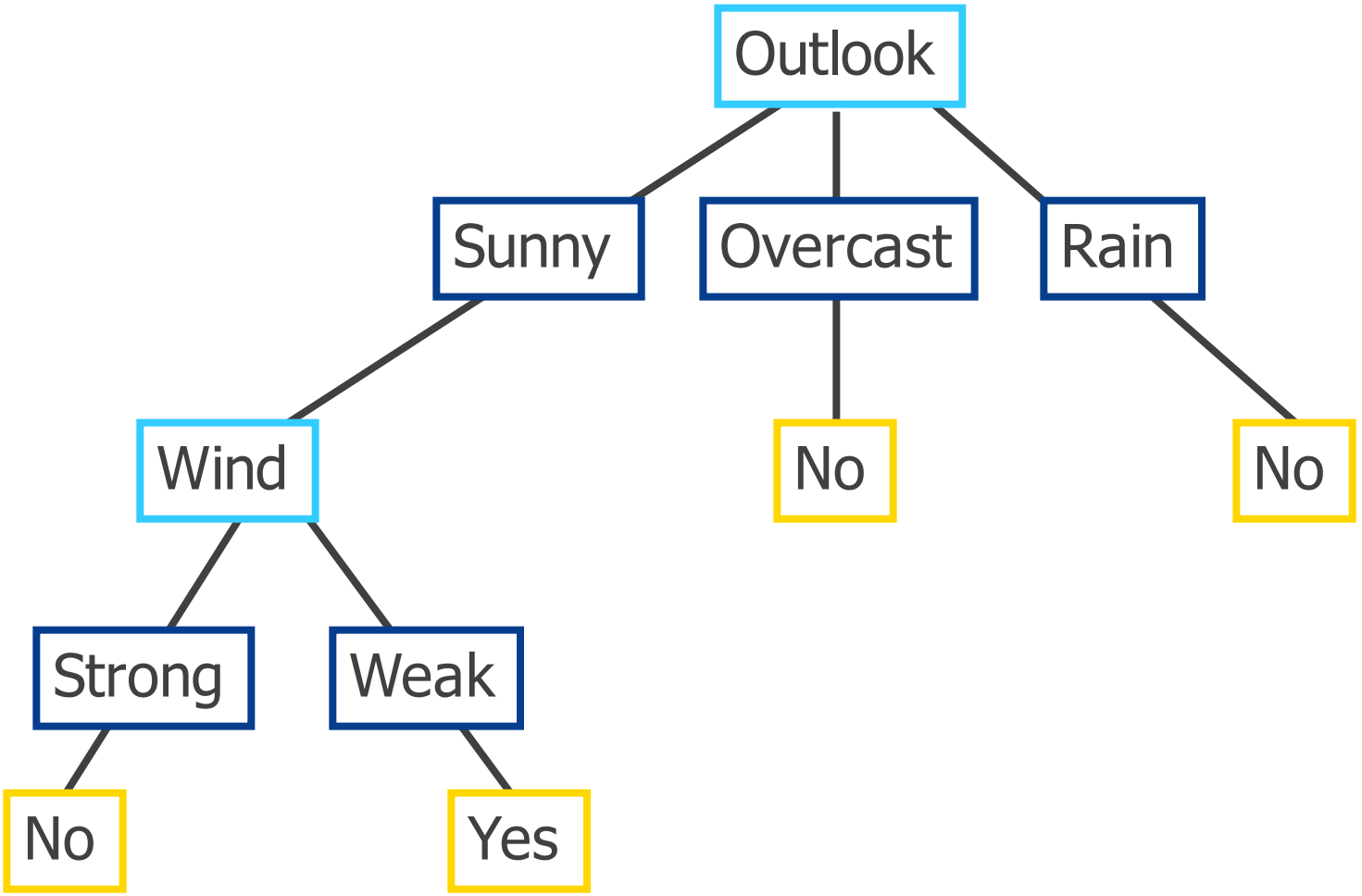
Conditional Entropy: $H(Y|X) = \sum_x \mathbb{P}(X = x) H(Y|X = x)$

Information Gain: $\mathsf{IG(Y|X)} = \mathsf{H(Y)} - \mathsf{H(Y|X)}$

Select the variable with the highest information gain

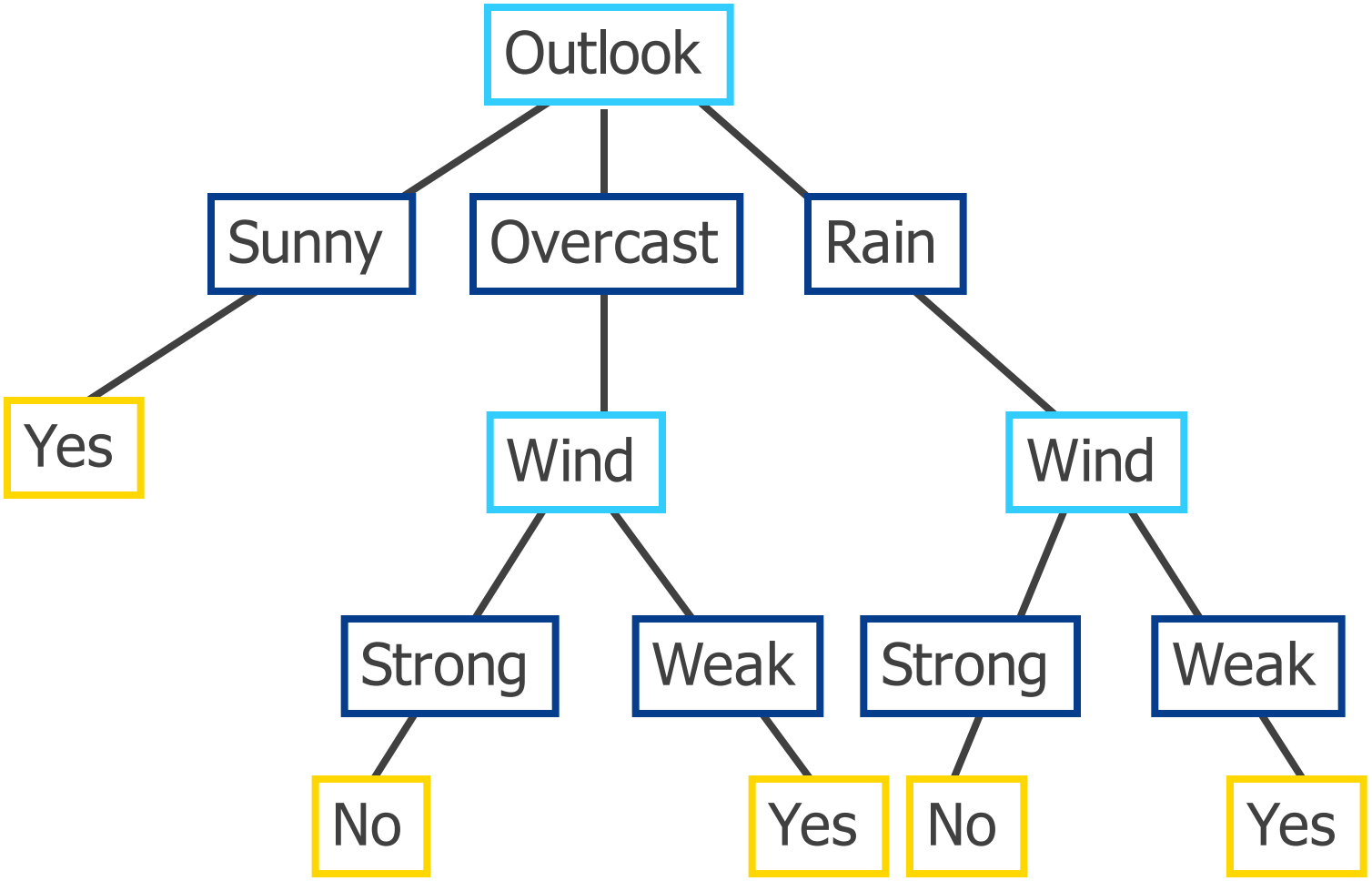
Decision Tree for Conjunction

Outlook=Sunny ∧ Wind=Weak



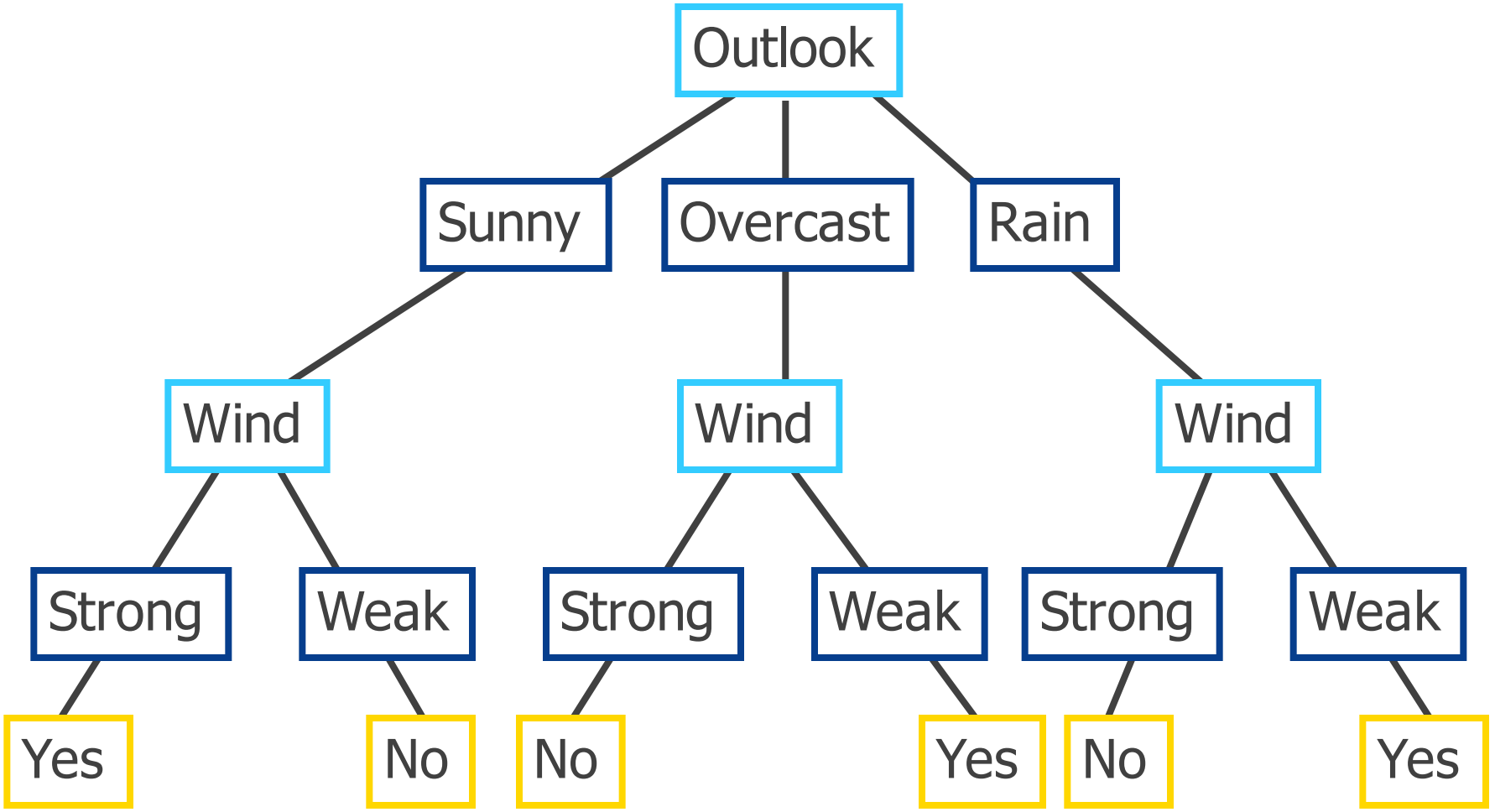
Decision Tree for Disjunction

Outlook=Sunny ∨ Wind=Weak



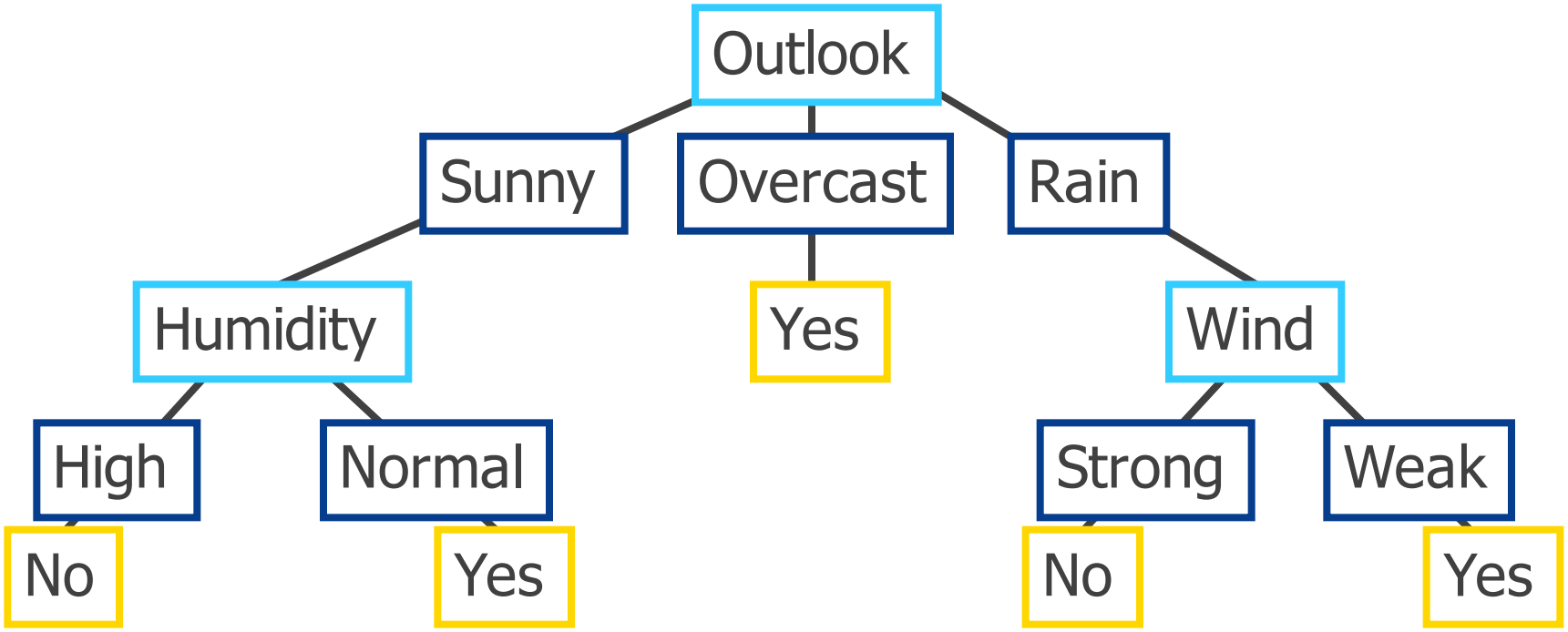
Decision Tree for XOR

Outlook=Sunny XOR Wind=Weak



Decision Tree

- decision trees represent disjunctions of conjunctions



(Outlook=Sunny ∧ Humidity=Normal)
∨ (Outlook=Overcast)
∨ (Outlook=Rain ∧ Wind=Weak)

When to consider Decision Trees

Instances describable by attribute-value pairs

Target function is discrete valued

Disjunctive hypothesis may be required

Possibly noisy training data

Missing attribute values

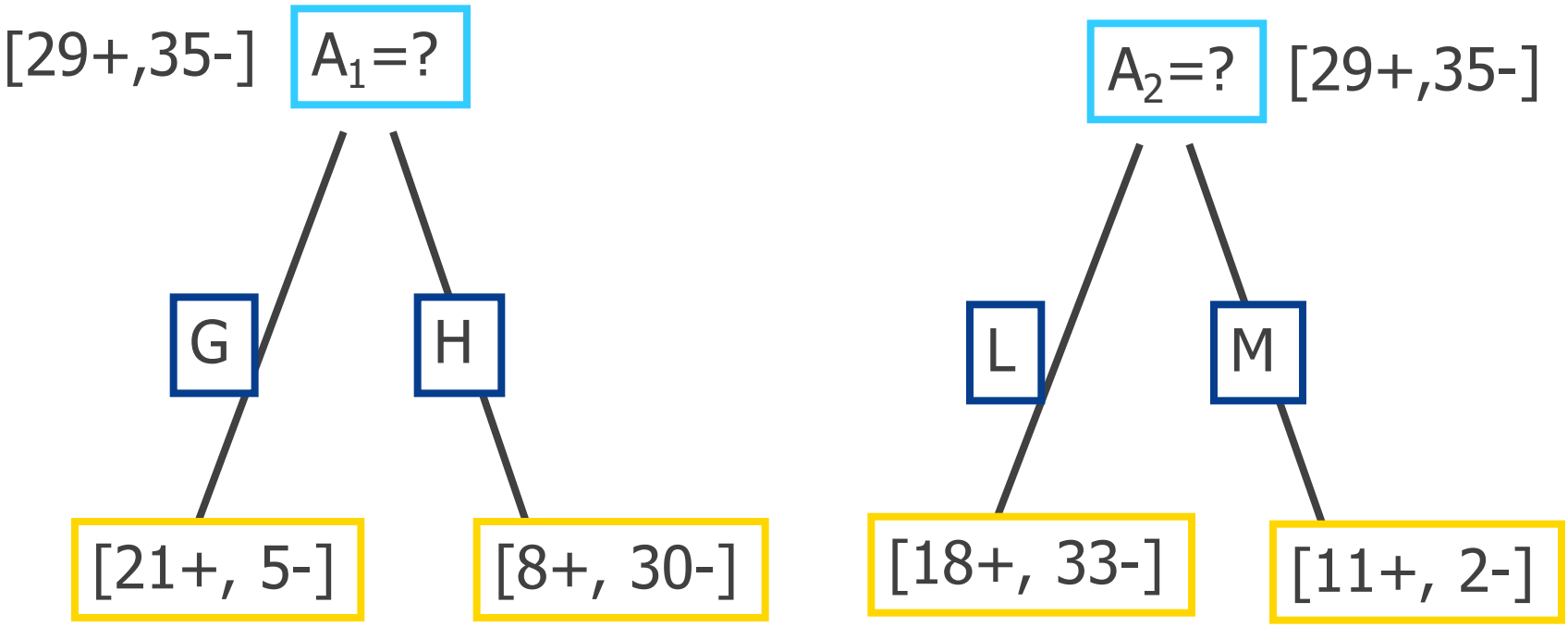
Examples:

- Medical diagnosis
- Credit risk analysis
- Object classification for robot manipulator (Tan 1993)

Top-Down Induction of Decision Trees ID3

1. A ← the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A create new descendant
4. Sort training examples to leaf node according to the attribute value of the branch
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.

Which attribute is best?



Entropy

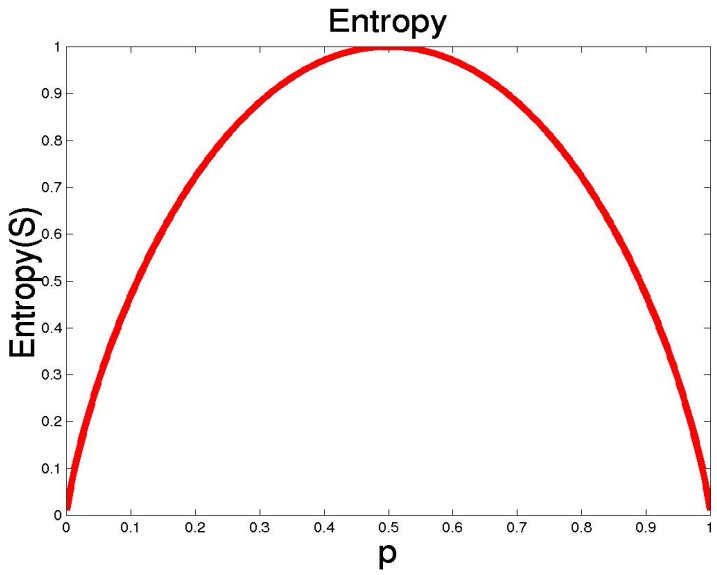
S is a sample of training examples

p_+ is the proportion of positive examples

p_- is the proportion of negative examples

Entropy measures the impurity of S

- Entropy(S) = $-p+ \log_2 p_+ - p_- \log_2 p_-$



Entropy

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

Why?

Information theory optimal length code assign

$-\log_2 p$ bits to messages having probability p.

So the expected number of bits to encode

(+ or -) of random member of S:

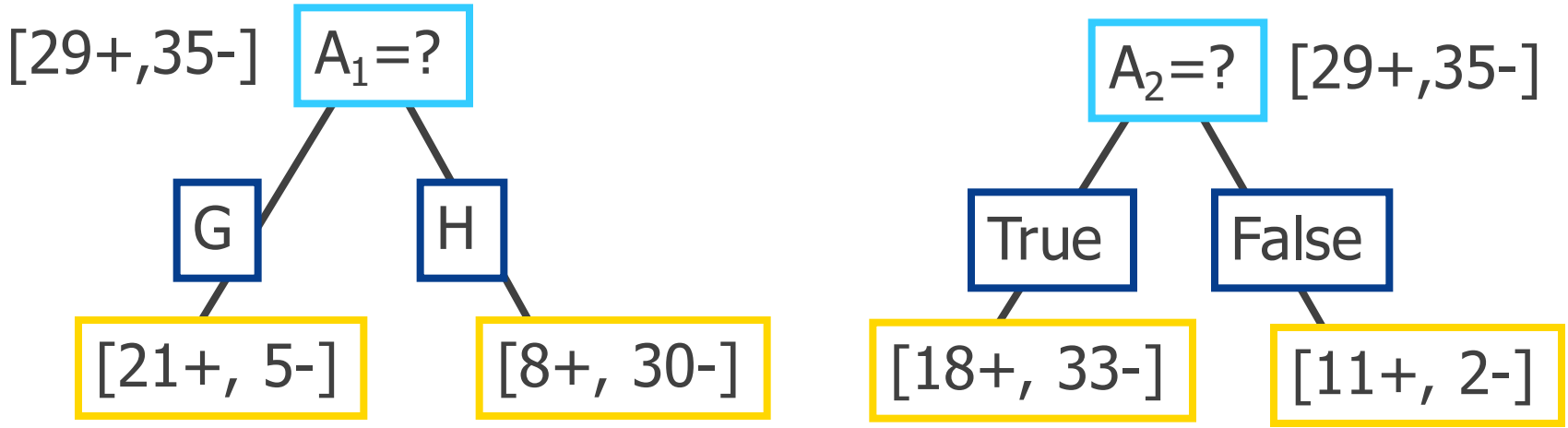
- $-p_+ \log_2 p_+ - p_- \log_2 p_-$

Information Gain (S=E)

Gain(S,A): expected reduction in entropy due to sorting S on attribute A

Entropy([29+,35-]) = -29/64 log2 29/64 – 35/64 log2 35/64
= 0.99

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



Information Gain

Entropy([21+,5-]) = 0.71

Entropy([8+,30-]) = 0.74

Gain(S,A1)=Entropy(S)

-26/64*Entropy([21+,5-])

-38/64*Entropy([8+,30-])

=0.27

Entropy([18+,33-]) = 0.94

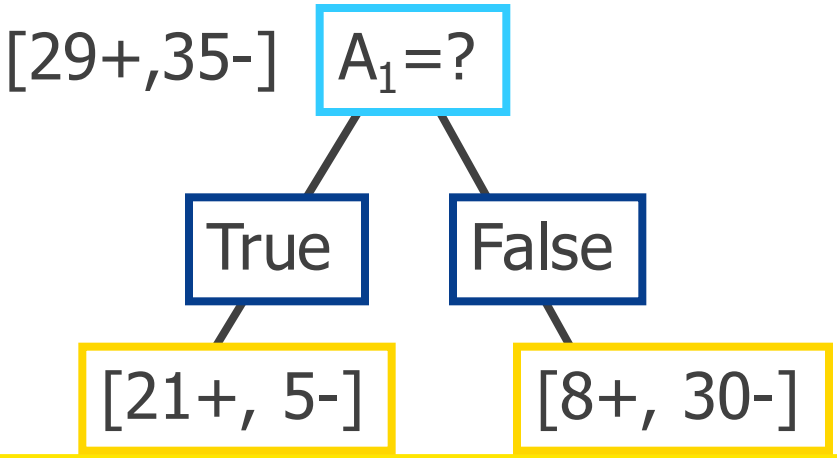
Entropy([11+,2-]) = 0.62

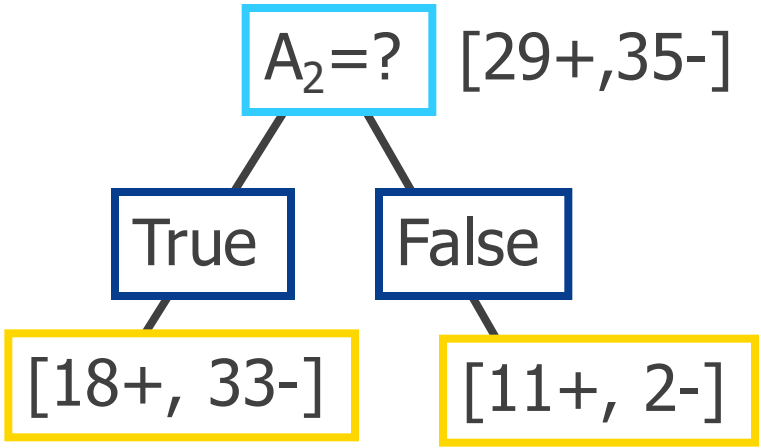
Gain(S,A2)=Entropy(S)

-51/64*Entropy([18+,33-])

-13/64*Entropy([11+,2-])

=0.12

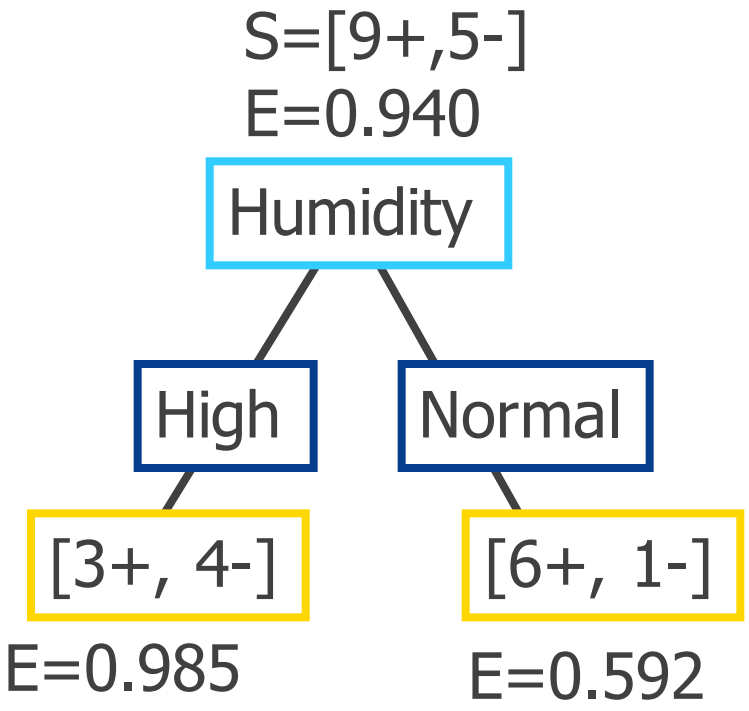


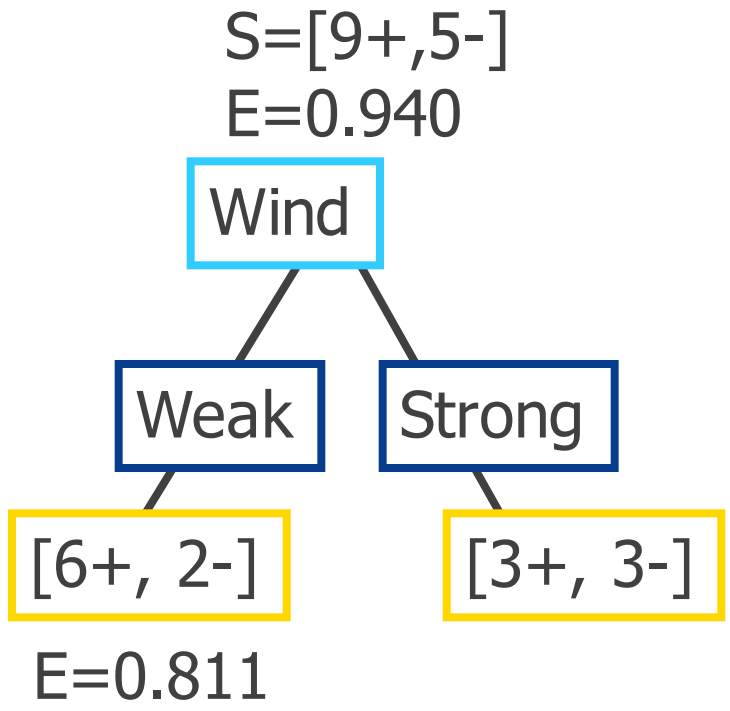


Training Examples

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Selecting the Next Attribute



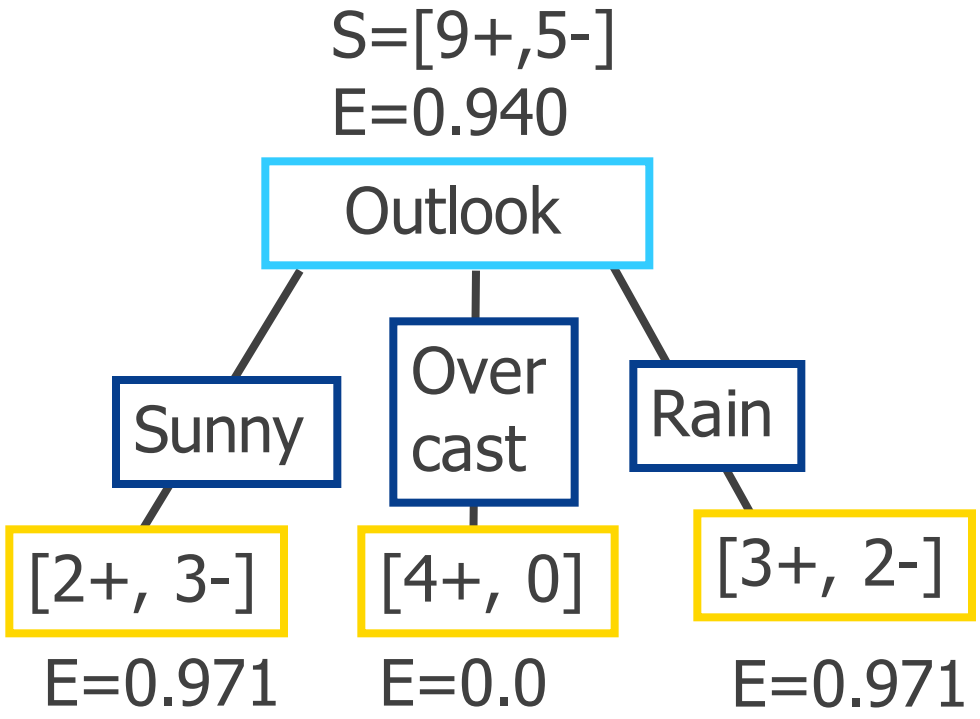


Gain(S,Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151

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

Humidity provides greater info. gain than Wind, w.r.t target classification.

Selecting the Next Attribute



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

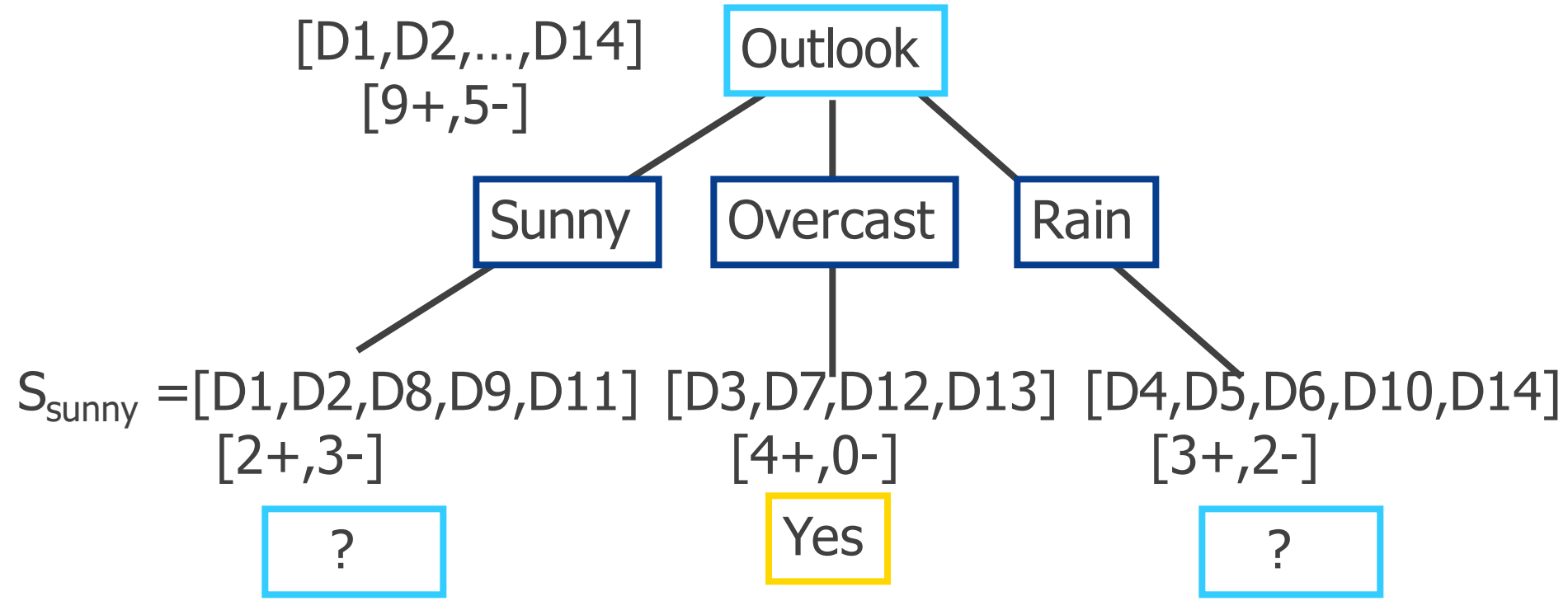
Selecting the Next Attribute

The information gain values for the 4 attributes are:

- Gain(S,Outlook) =0.247
- Gain(S,Humidity) =0.151
- Gain(S,Wind) =0.048
- Gain(S,Temperature) =0.029

where S denotes the collection of training examples

ID3 Algorithm

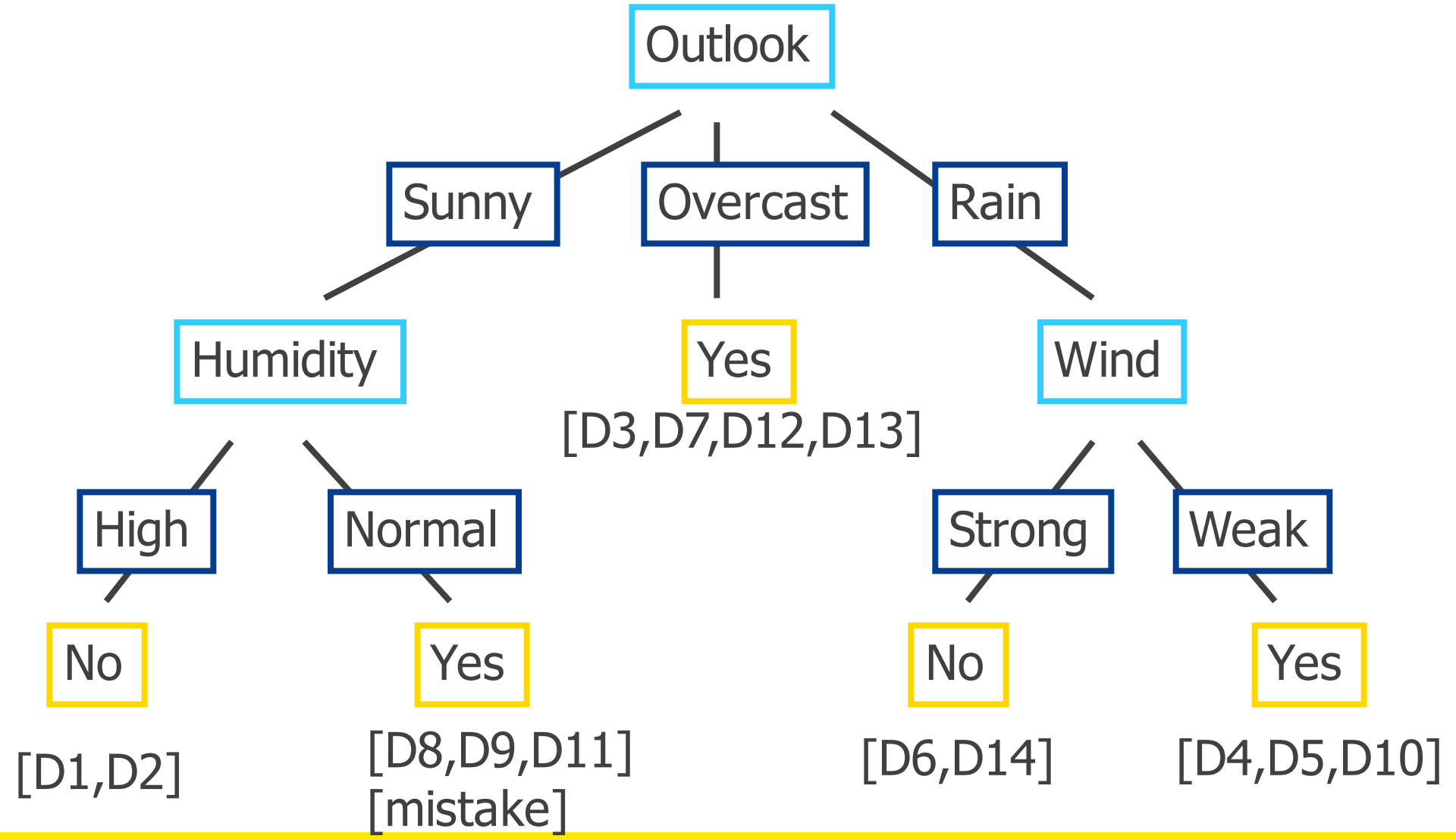


Gain(S_{sunny}, Humidity)=0.970-(3/5)0.0 – 2/5(0.0) = 0.970

Gain(S_{sunny}, Temp.)=0.970-(2/5)0.0 –2/5(1.0)-(1/5)0.0 = 0.570

Gain(S_{sunny}, Wind)=0.970= -(2/5)1.0 – 3/5(0.918) = 0.019

ID3 Algorithm



Occam's Razor

"If two theories explain the facts equally weel, then the simpler theory is to be preferred"

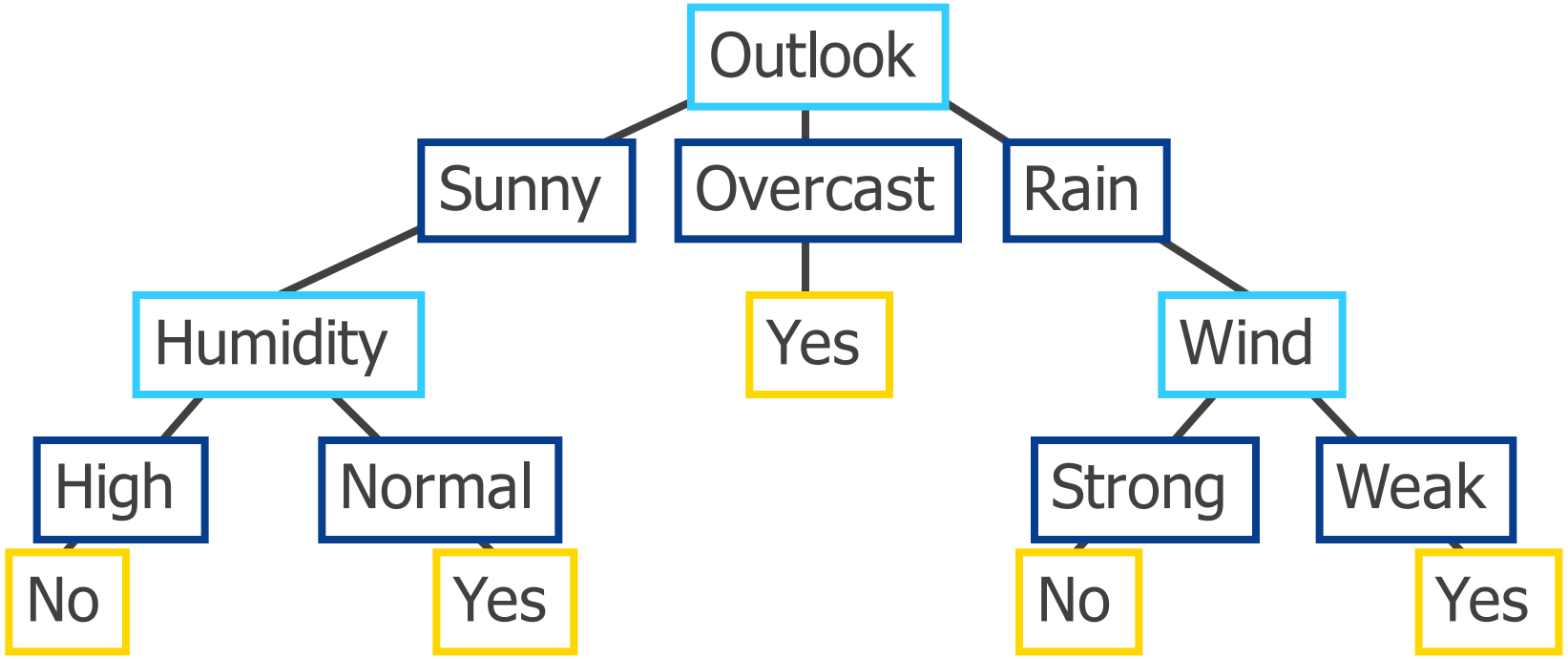
Arguments in favor:

- Fewer short hypotheses than long hypotheses
- A short hypothesis that fits the data is unlikely to be a coincidence
- A long hypothesis that fits the data might be a coincidence

Arguments opposed:

- There are many ways to define small sets of hypotheses

Converting a Tree to Rules



R_1: If (Outlook=Sunny) ∧ (Humidity=High) Then PlayTennis=No
R_2: If (Outlook=Sunny) ∧ (Humidity=Normal) Then PlayTennis=Yes
R_3: If (Outlook=Overcast) Then PlayTennis=Yes
R_4: If (Outlook=Rain) ∧ (Wind=Strong) Then PlayTennis=No
R_5: If (Outlook=Rain) ∧ (Wind=Weak) Then PlayTennis=Yes

Continuous Valued Attributes

Create a discrete attribute to test continuous

Temperature = 24.50C

(Temperature > 20.00C) = {true, false}

Where to set the threshold?

Temperature	15^{0}C	18^{0}C	19^{0}C	22^{0}C	24^{0}C	27^{0}C
PlayTennis	No	No	Yes	Yes	Yes	No

Unknown Attribute Values

What if some examples have missing values of A?

Use training example anyway sort through tree

If node n tests A, assign most common value of A among other examples sorted to node n.

Assign most common value of A among other examples with same target value

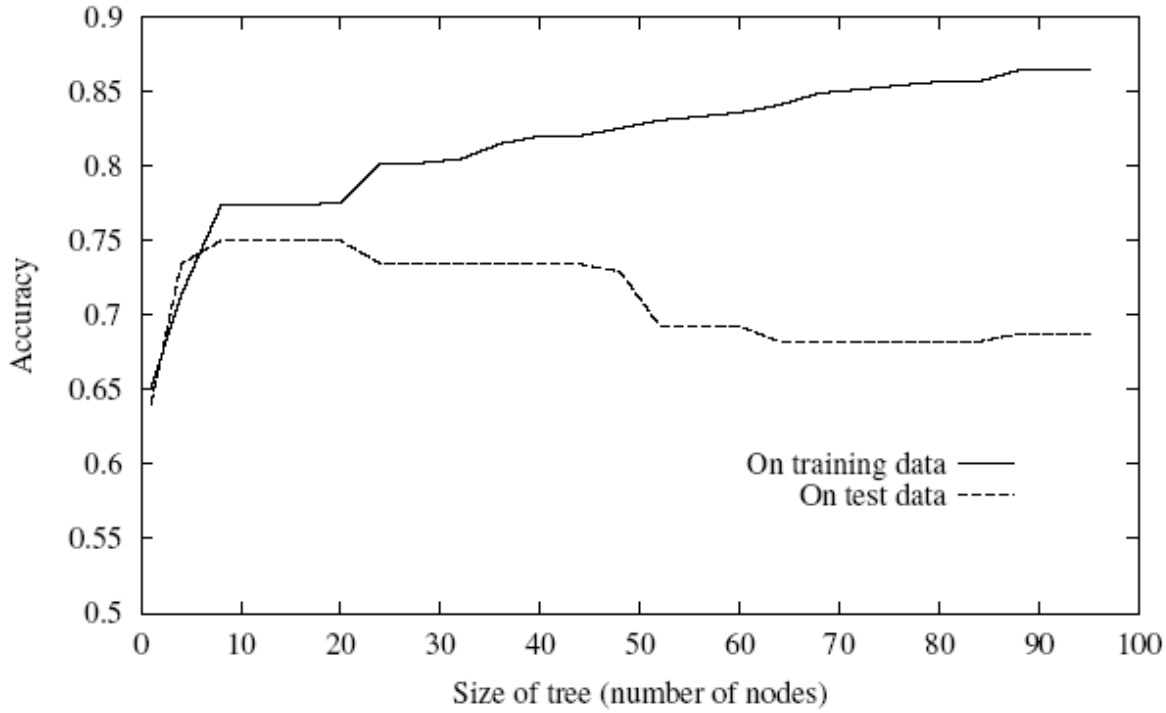
Assign probability pi to each possible value vi of A

- Assign fraction pi of example to each descendant in tree

Classify new examples in the same fashion

Overfitting

One of the biggest problems with decision trees is Overfitting



Avoid Overfitting

stop growing when split not statistically significant

grow full tree, then post-prune

Select “best” tree:

measure performance over training data

measure performance over separate validation data set

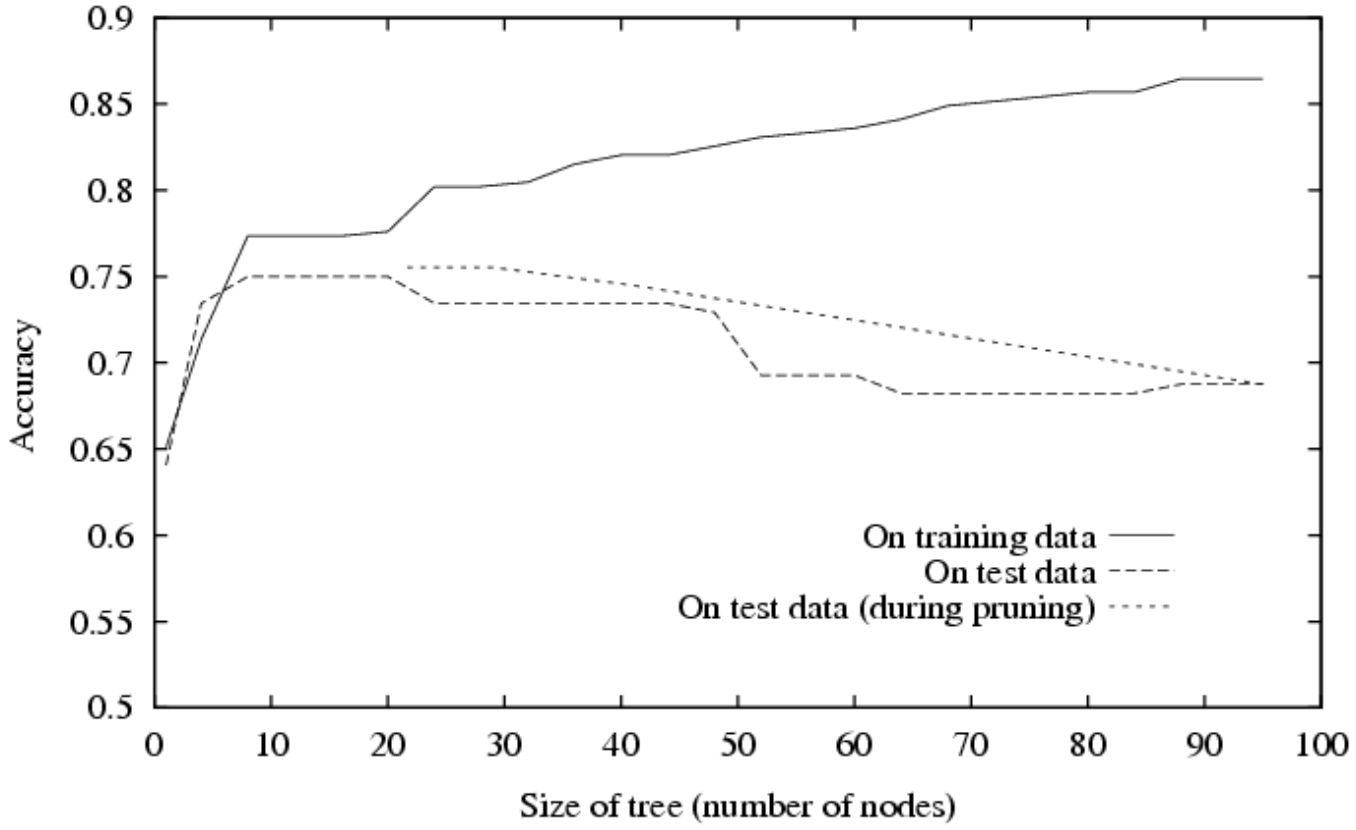
min(|tree|+|misclassifications(tree)|)

Avoid Overfitting

Idea 1: Stop growing the tree when the RSS doesn't drop by more than a threshold with any new cut.

Idea 2: Prune a large tree from the leaves to the root. I Weakest link pruning:

Effect of Reduced Error Pruning



Cross-Validation

Idea 3: Find the optimal subtree by cross validation. → There are too many possibilities – harder than best subsets!

Estimate the accuracy of an hypothesis induced by a supervised learning algorithm

Predict the accuracy of an hypothesis over future unseen instances

Select the optimal hypothesis from a given set of alternative hypotheses

- Pruning decision trees
- Model selection
- Feature selection

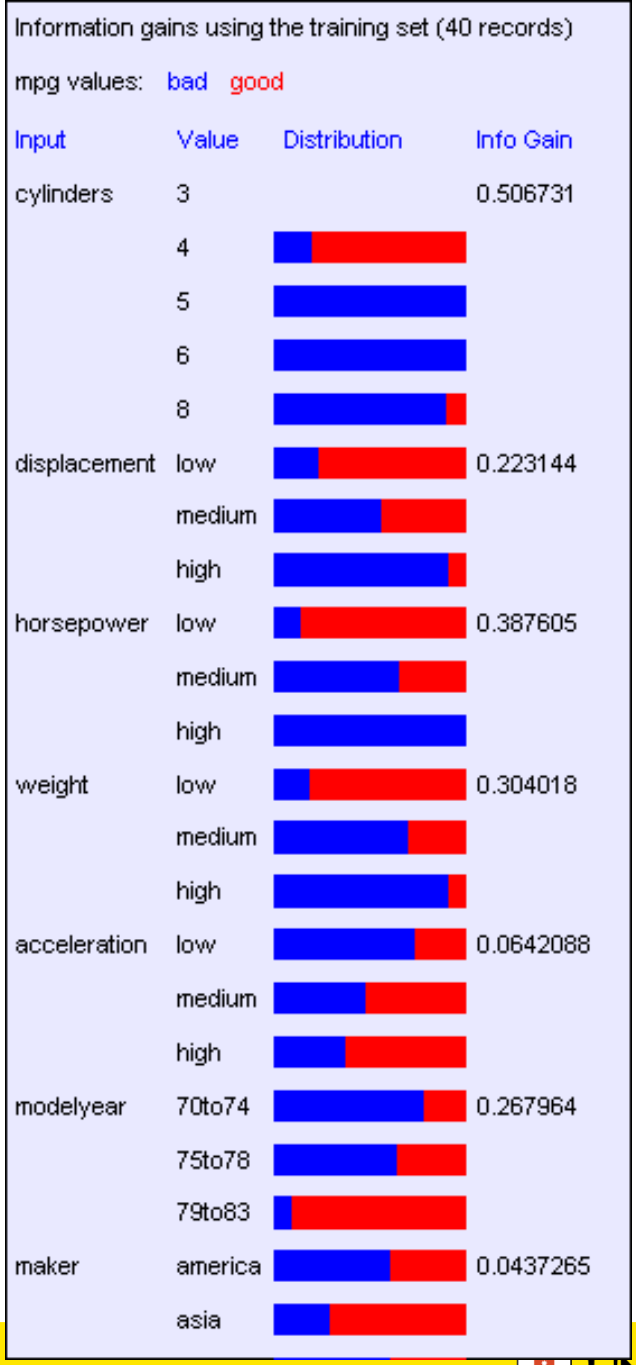
Combining multiple classifiers (bagging/boosting)

Automobile Miles-per-gallon prediction

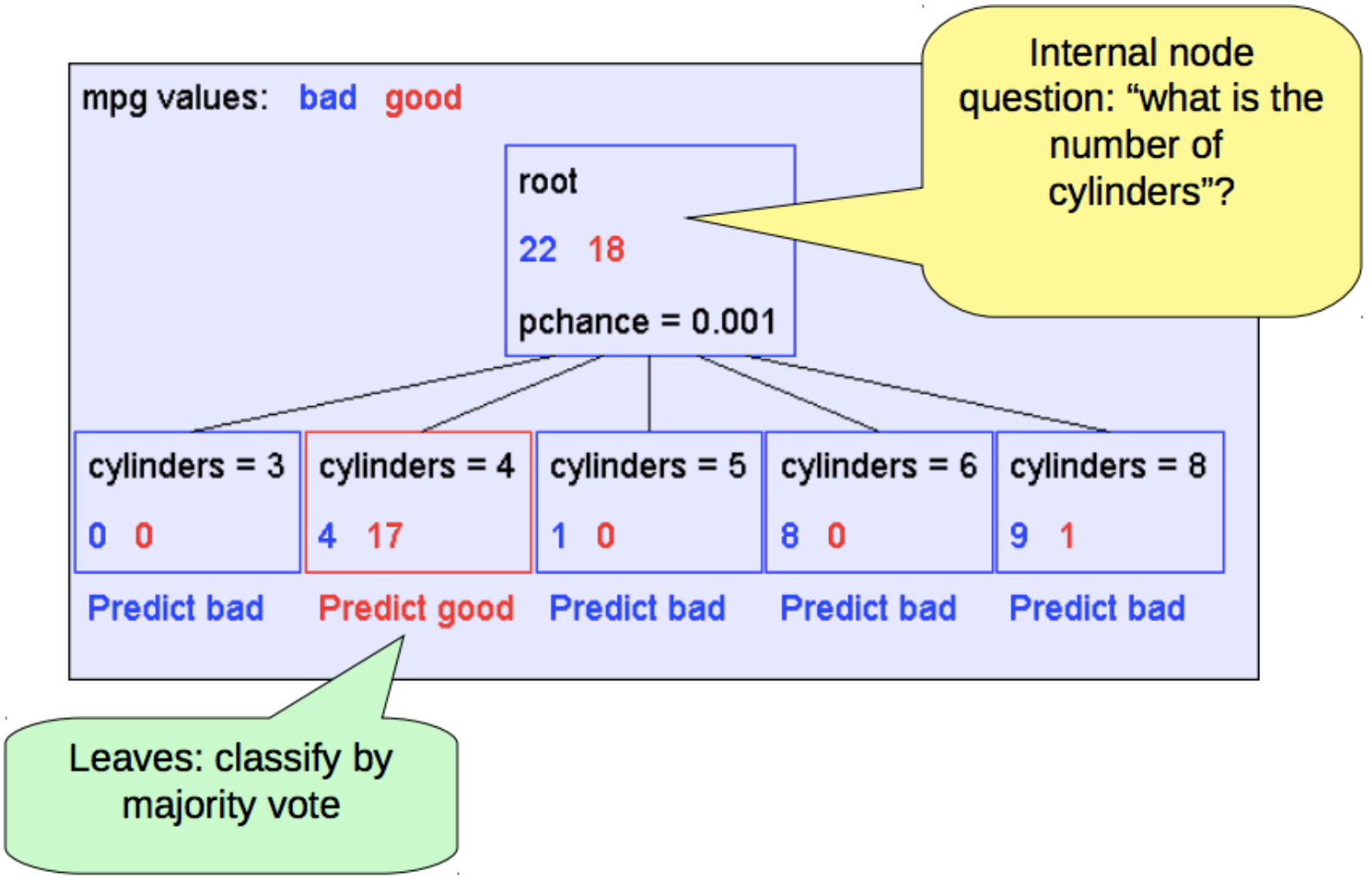
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe

Look at all the information gains…

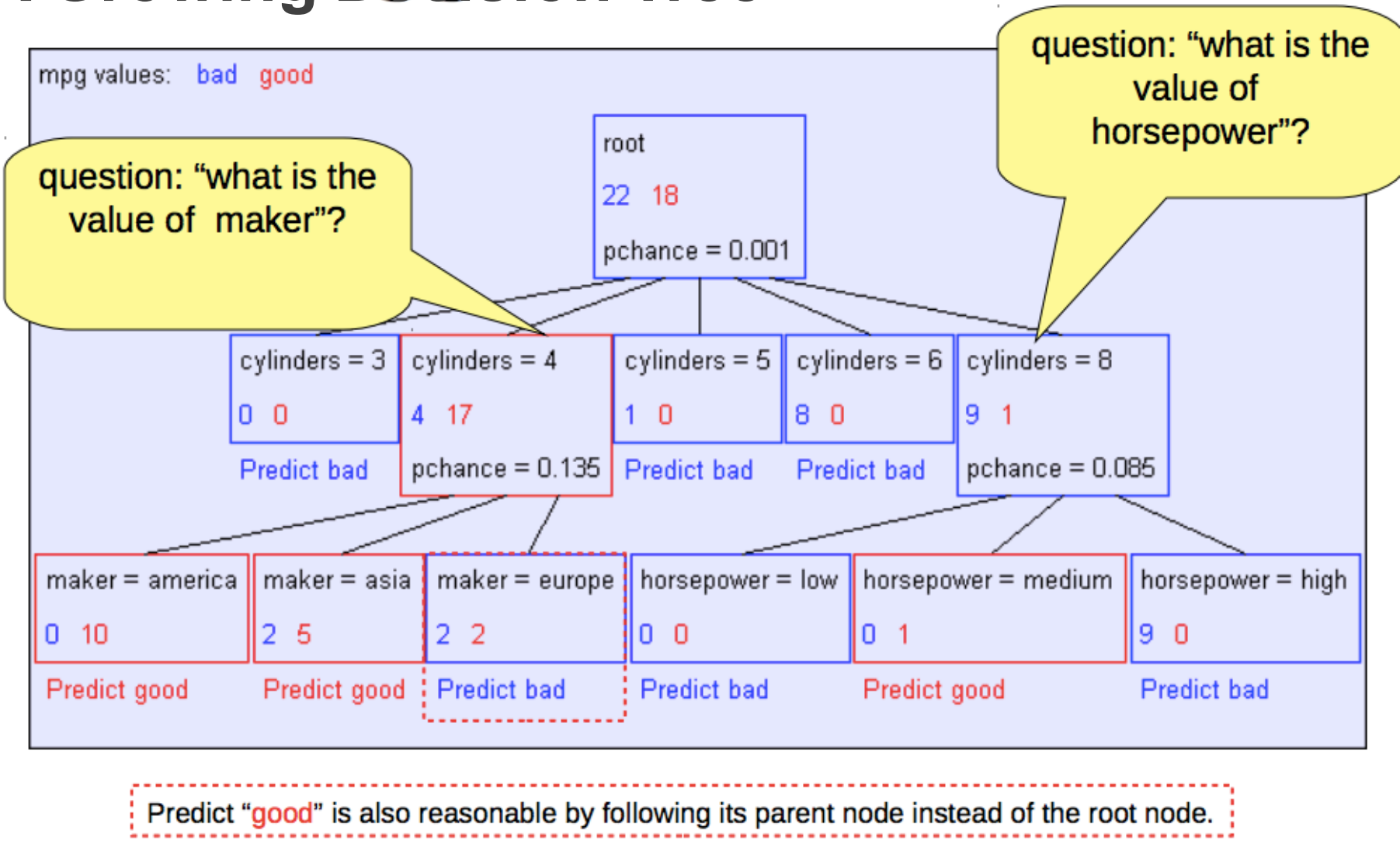
Suppose we want to predict MPG.



A Small Decision Tree

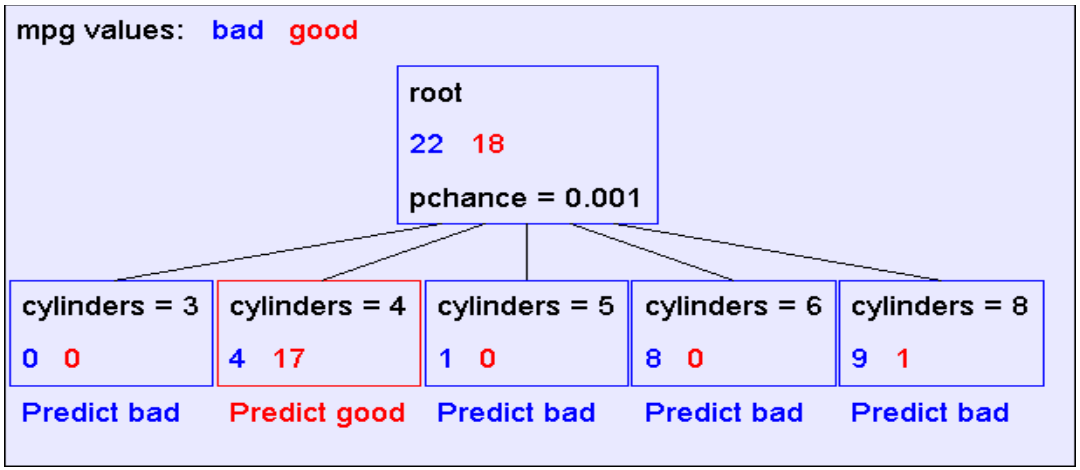


A Growing Decision Tree

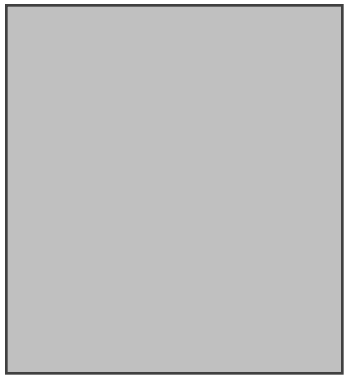


Predict "good" is also reasonable by following its parent node instead of the root node.

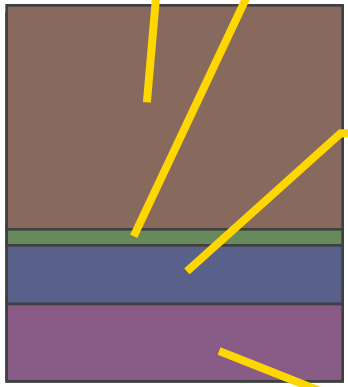
Recursion Step



Take the Original Dataset..



And partition it according to the value of the attribute we split on



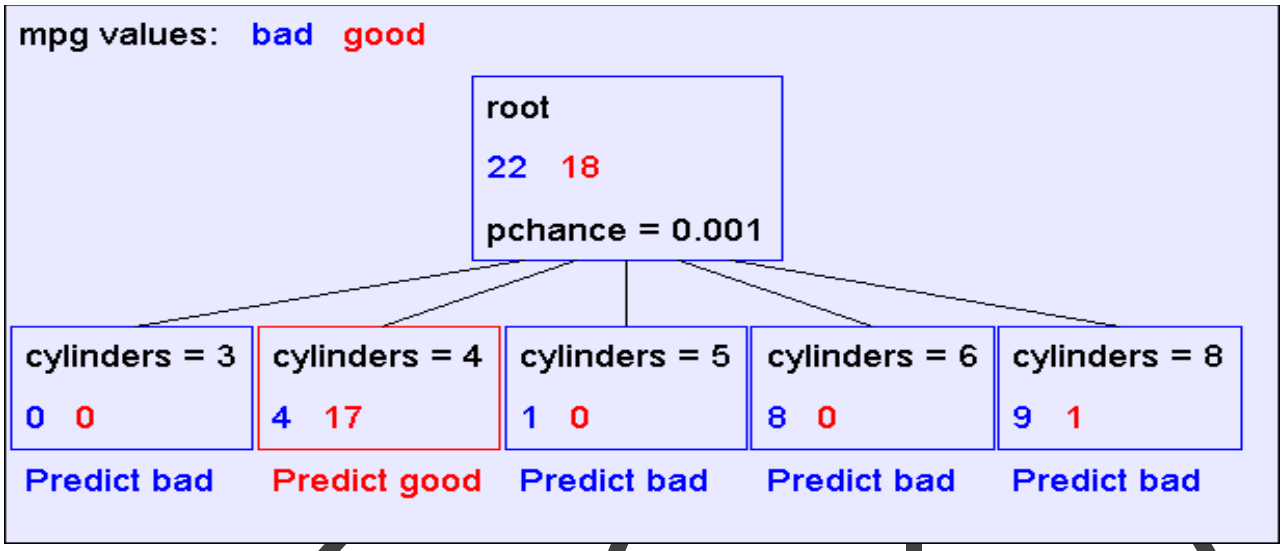
Records in which cylinders = 4

Records in which cylinders = 5

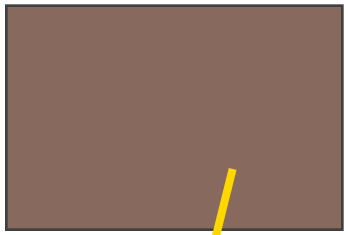
Records in which cylinders = 6

Records in which cylinders = 8

Recursion Step



Build tree from These records..



Records in which cylinders = 4

Build tree from These records..

Records in which cylinders = 5

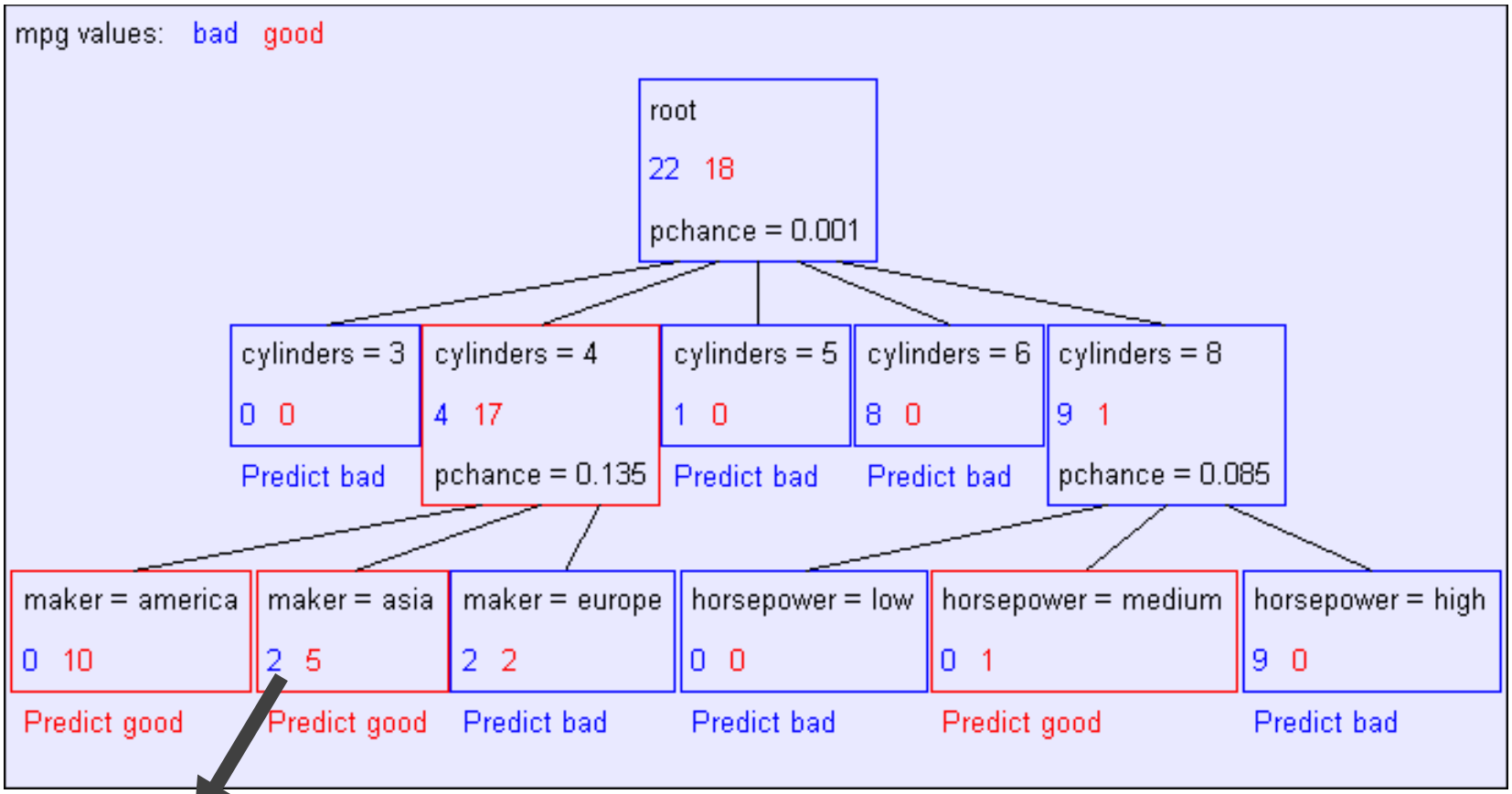
Build tree from These records..

Records in which cylinders = 6

Build tree from These records..

Records in which cylinders = 8

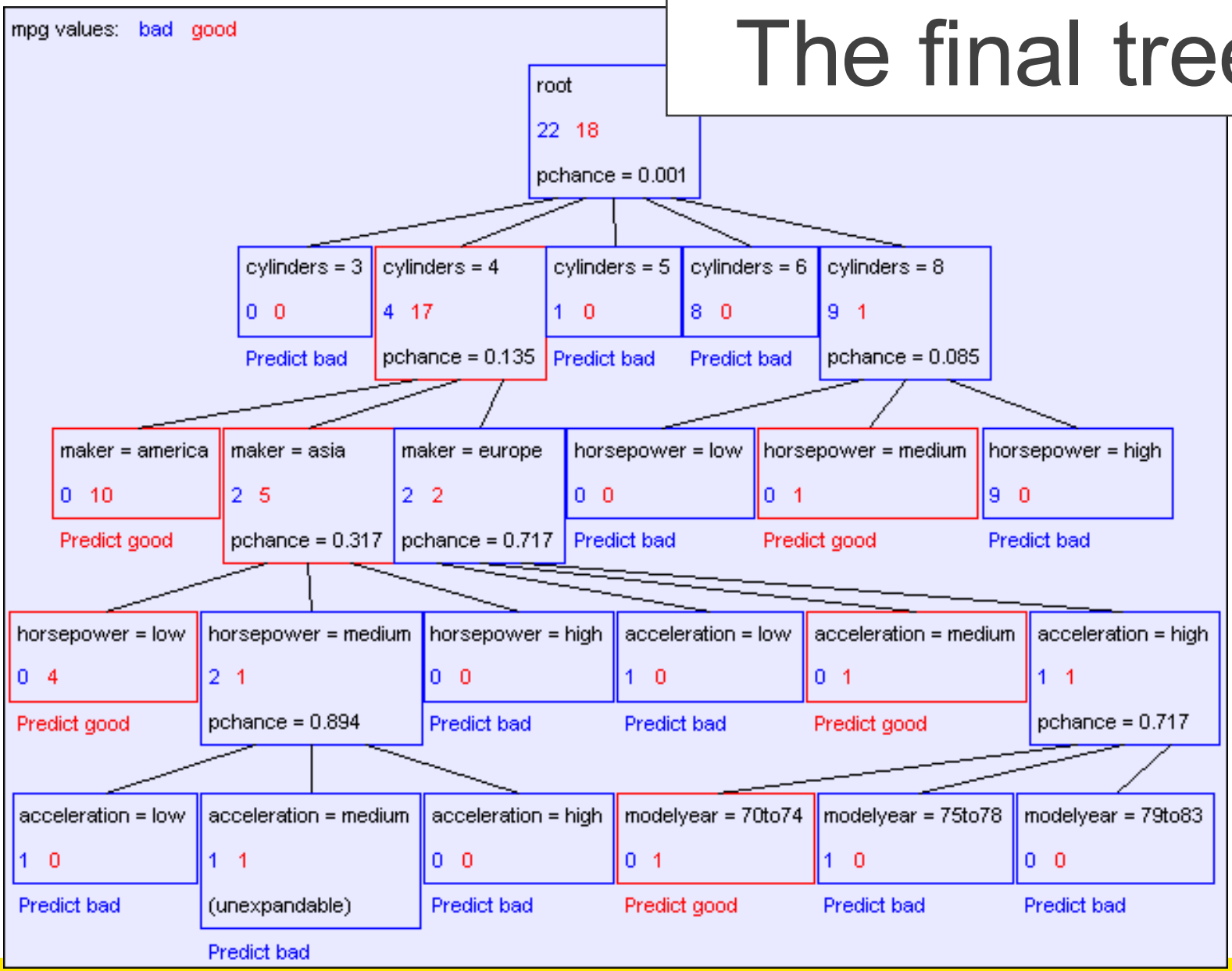
Second level of tree



Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

The final tree



Significance of Split

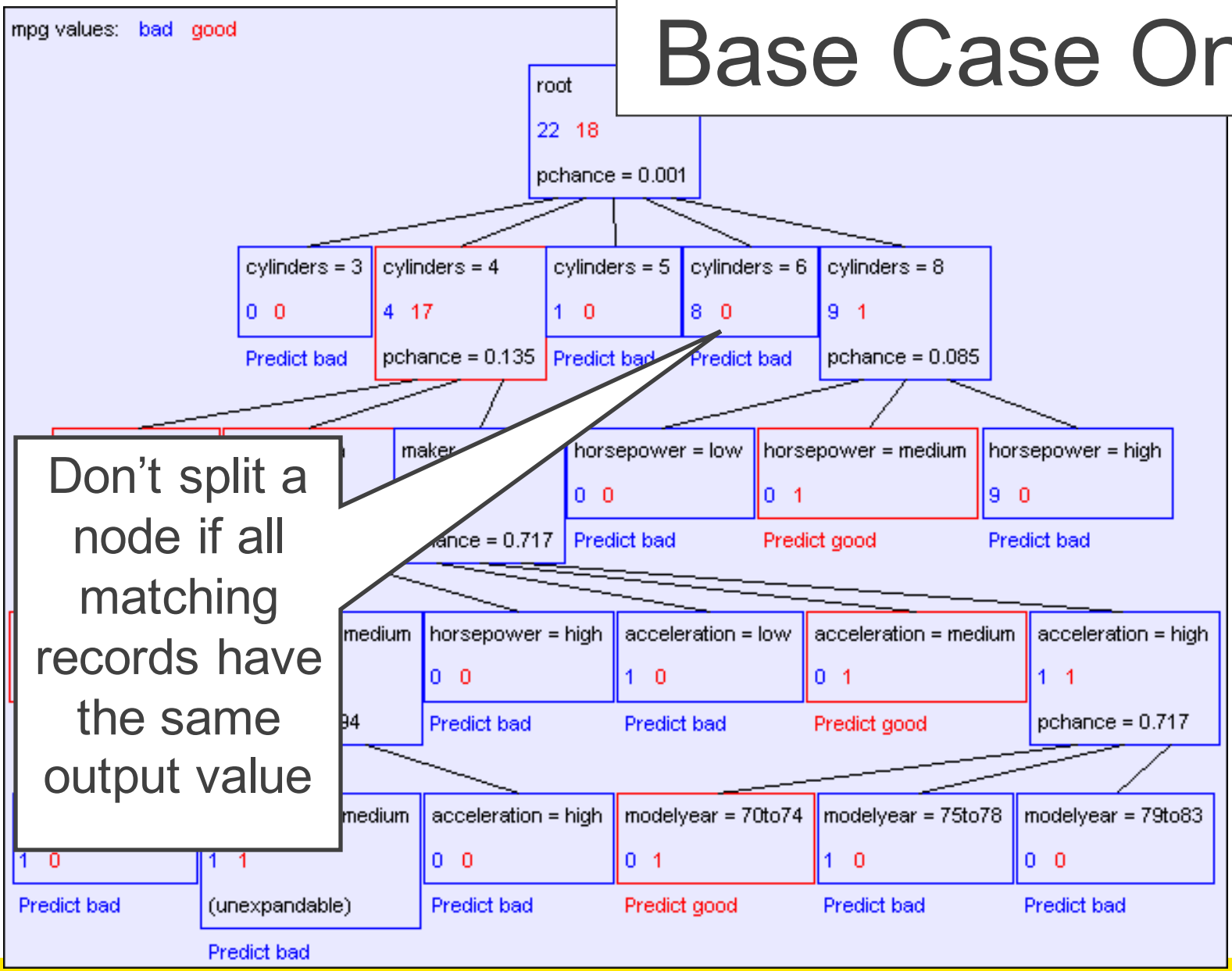
Starting with:

- Three cars with 4 cylinders, from Asia, with medium HP
- 2 bad MPG
- 1 good MPG

Probably shouldn't split if the counts are so small they could be due to chance A chi-squared test can tell us how likely it is that deviations from a perfect split are due to chance

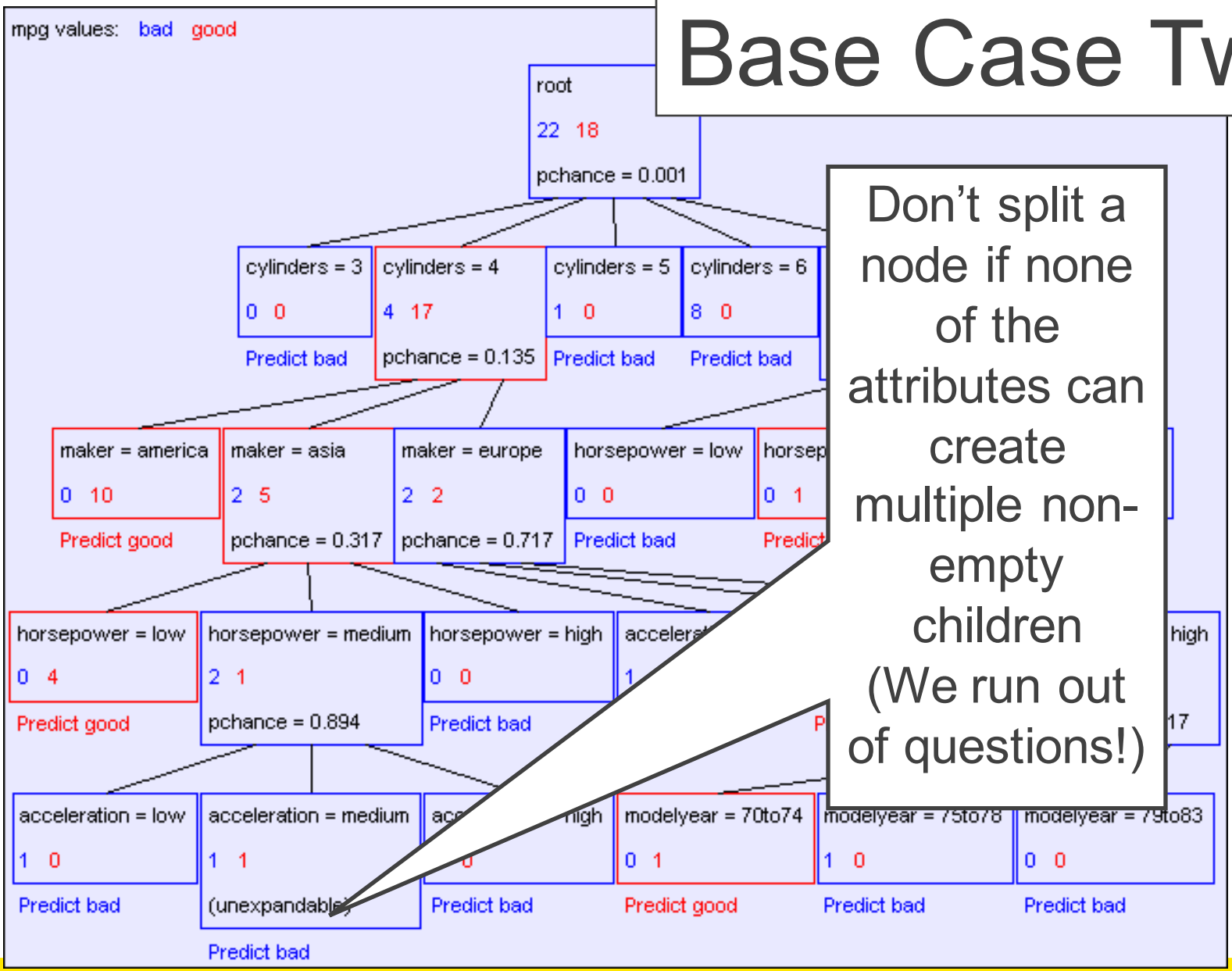
Each split will have a significance value, pchance

Base Case One

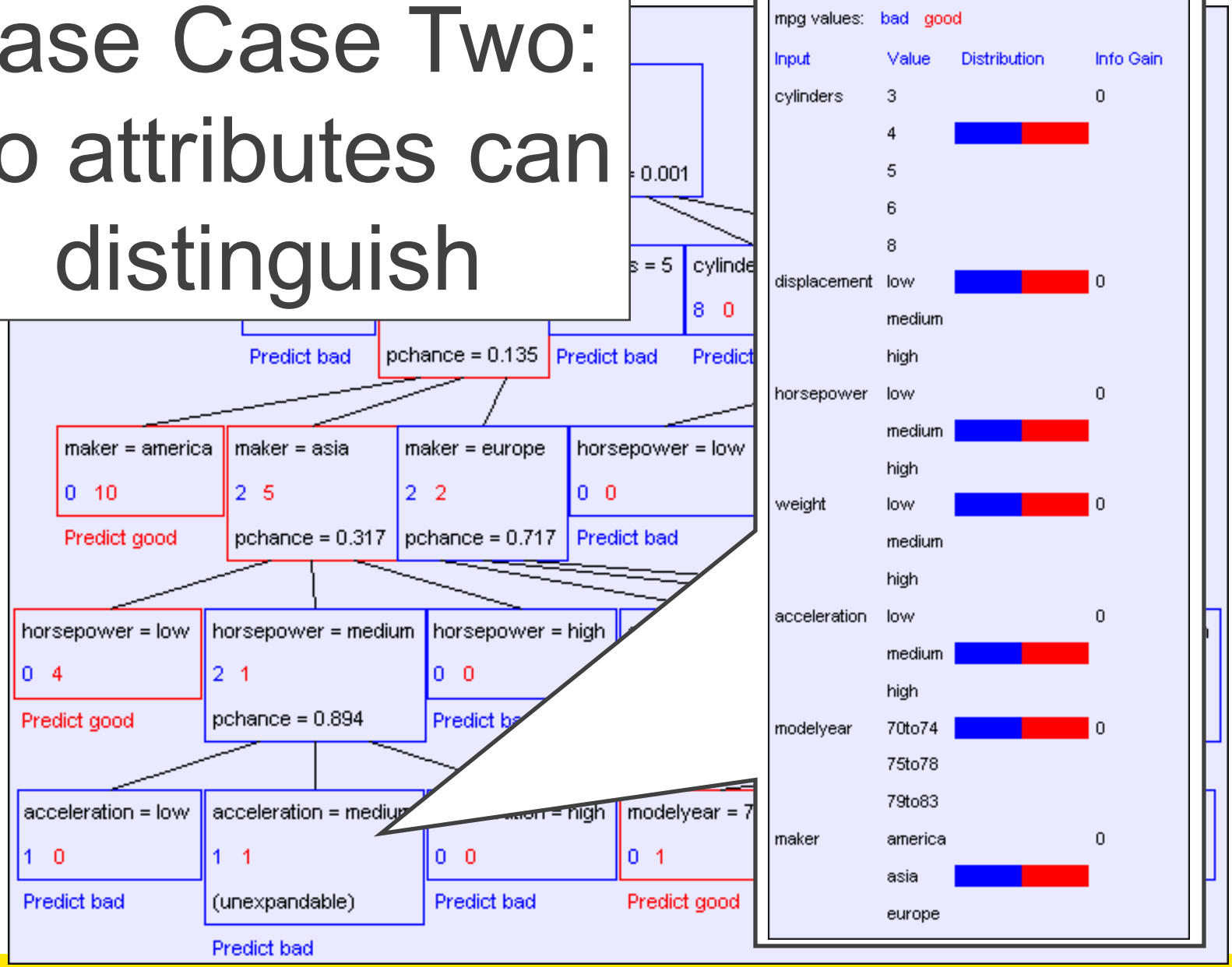


Don't split a node if all matching records have the same output value

Base Case Two



Base Case Two: No attributes can distinguish



Base Cases

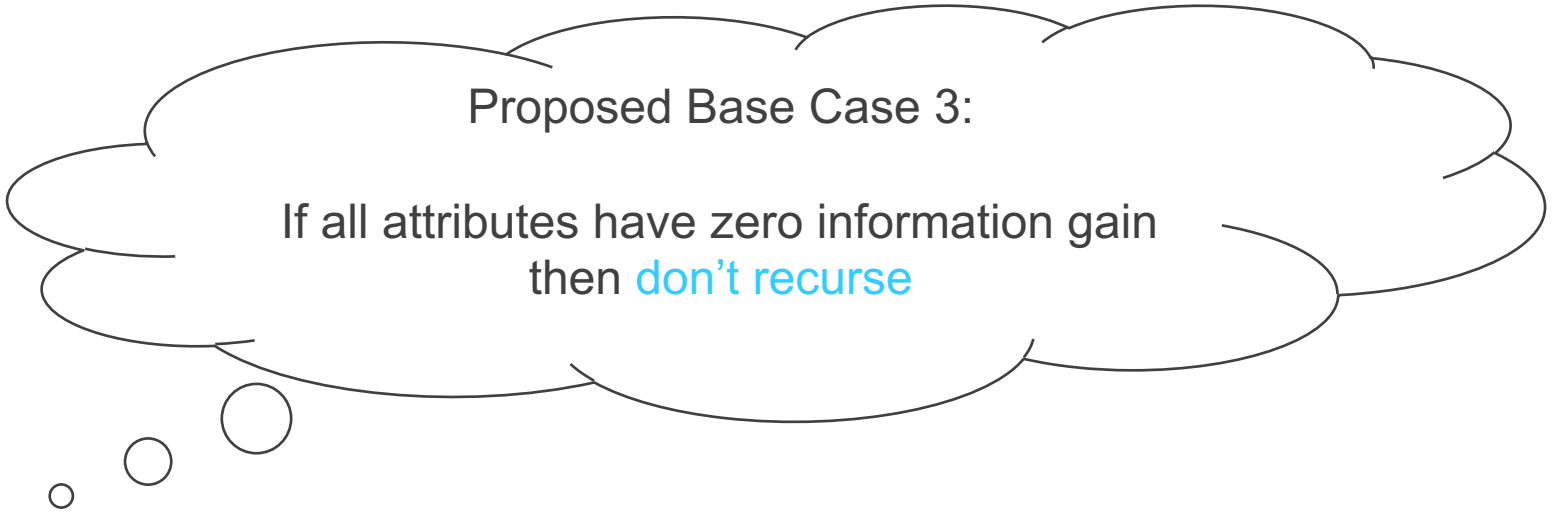
Base Case One: If all records in current data subset have the same output then don't recurse

Base Case Two: If all records have exactly the same set of input attributes then don't recurse

Base Cases: An idea

Base Case One: If all records in current data subset have the same output then don't recurse

Base Case Two: If all records have exactly the same set of input attributes then don't recurse



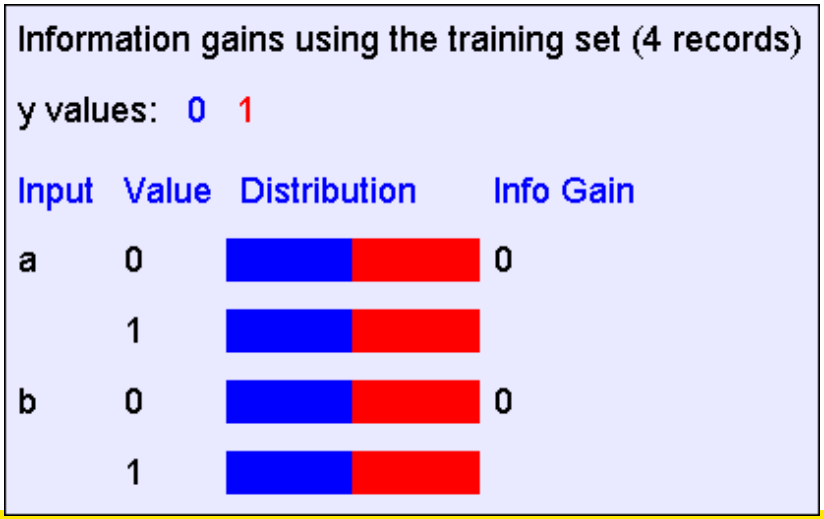
- *Is this a good idea?*

The problem with Base Case 3

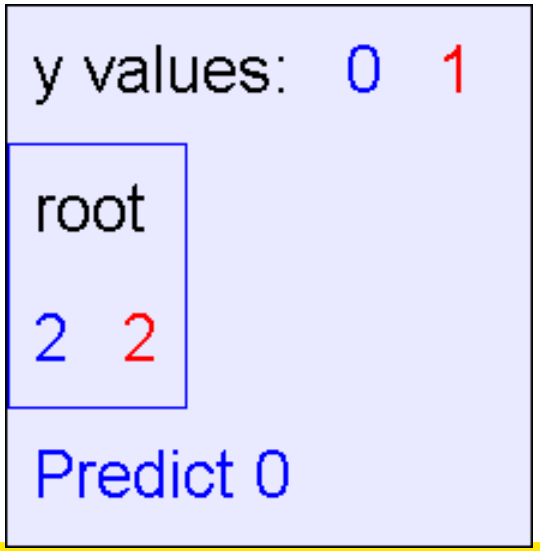
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

y = a XOR b

The information gains:



The resulting decision tree:

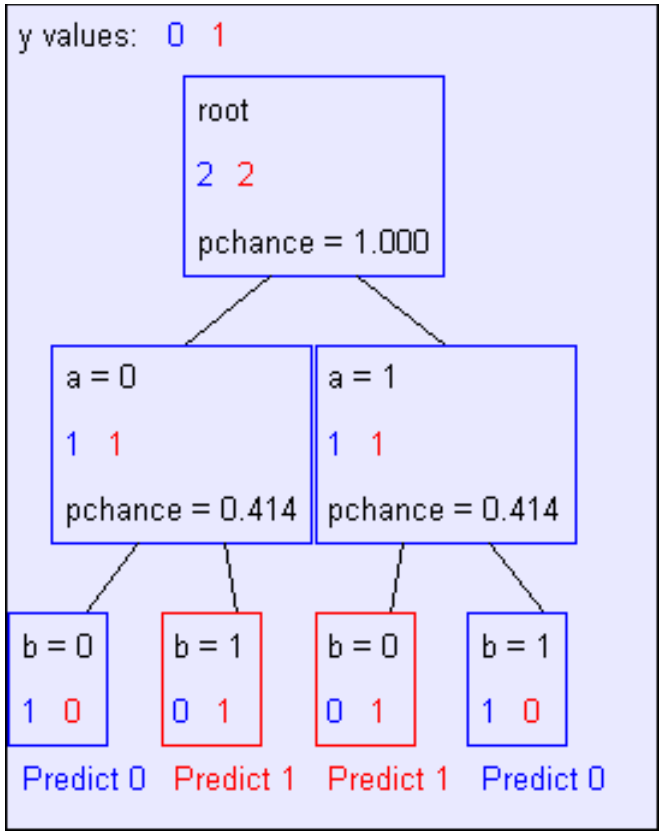


If we omit Base Case 3:

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

y = a XOR b

The resulting decision tree:



Basic Decision Tree Building Summarized

BuildTree(DataSet,Output)

If all output values are the same in DataSet, return a leaf node that says “predict this unique output”

If all input values are the same, return a leaf node that says “predict the majority output”

Else find attribute X with highest Info Gain

Suppose X has nX distinct values

- Create and return a non-leaf node with nX children.
- The i’th child should be built by calling
  - BuildTree(DSi,Output)
  - Where DSi built consists of all those records in DataSet for which X = ith distinct value of X.

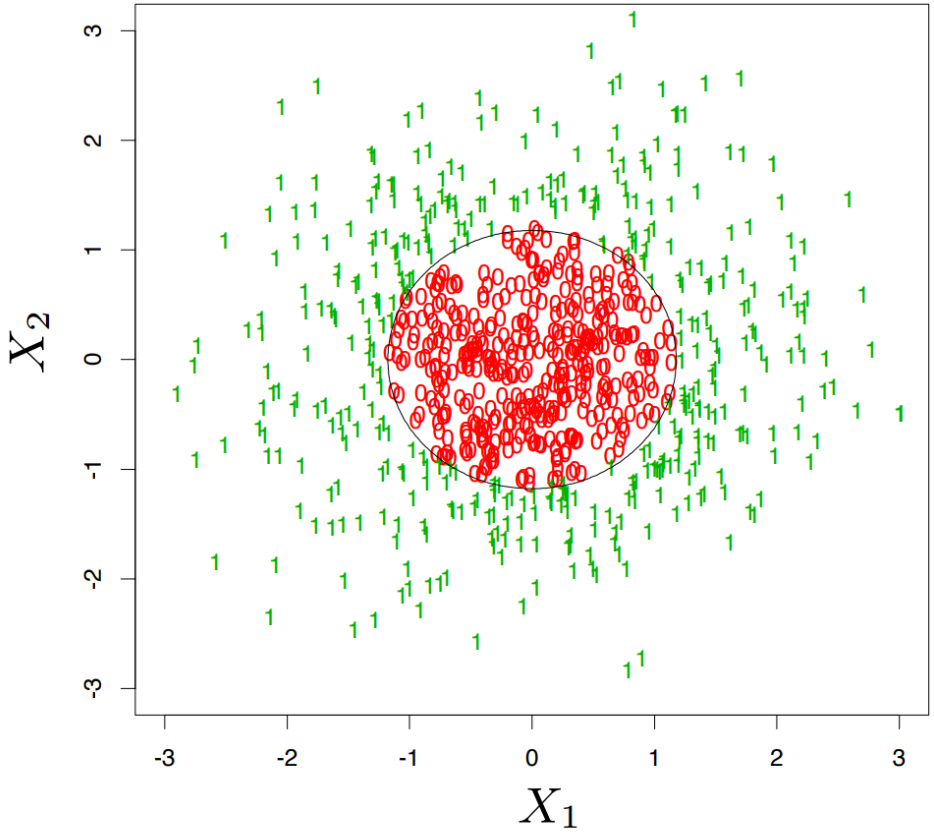
A Quick Summary of Basic Decision Tree

- Can encode any function
- Top-down learning (not perfect!)
- Information gain
- Bottom-up pruning/Cross validation to prevent overfitting

Random Forests

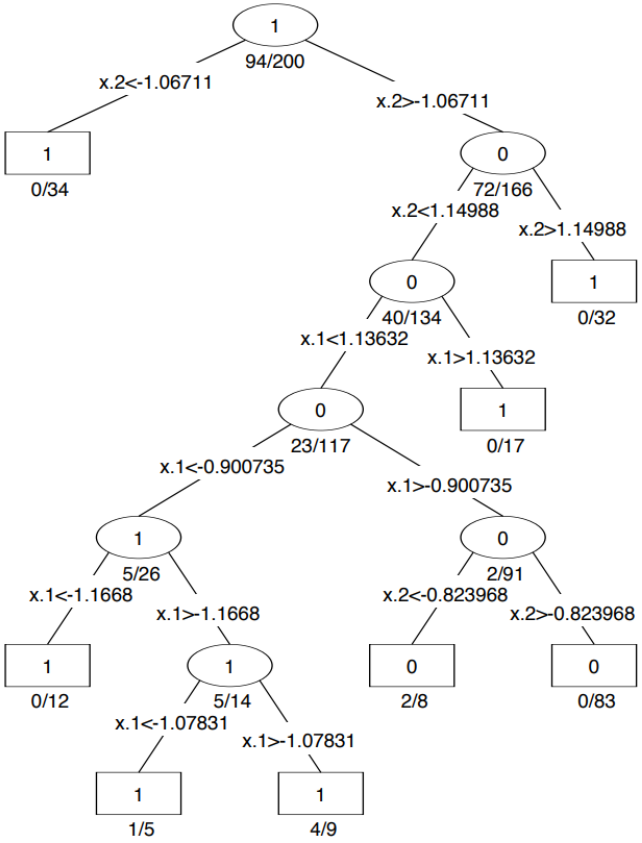
Supervised Learning

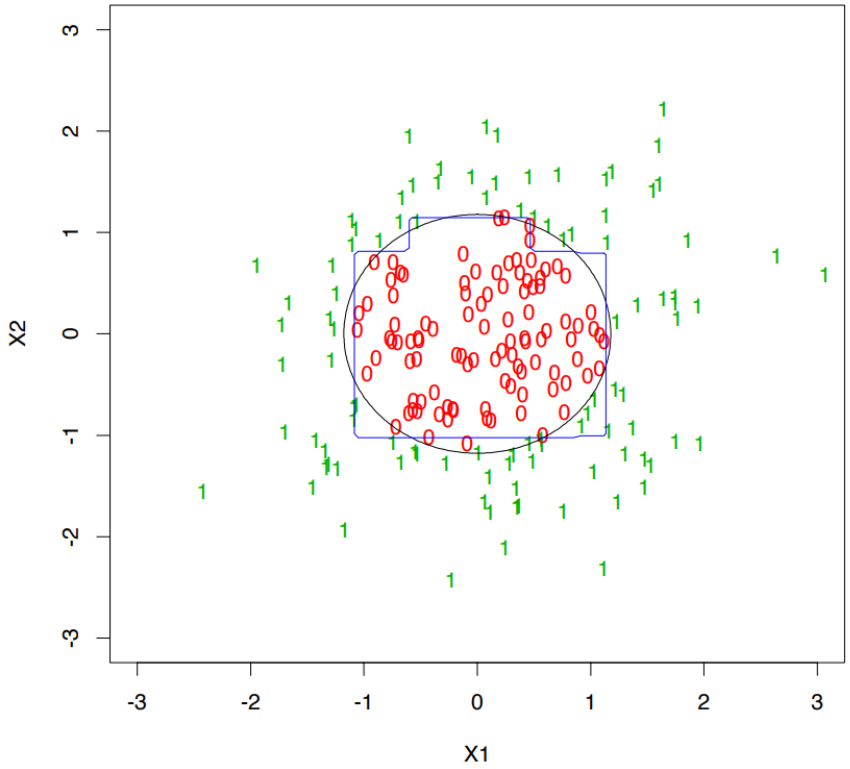
Toy Example



Nonlinear separable data.
Optimal decision boundary: $X_1^2 + X_2^2 = 1$

Toy Example





Sample size: 200
7 branching nodes; 6 layers.
Classification error: 7.3% when d = 2; > 30% when d = 10

Ensemble Learning

Decision trees can be simple, but often overfitting and of high variance

Two minds working together can often achieve better results.

- It is a well-known statistical intuition that averaging measurements can lead to a more stable and reliable

Combination of models: model ensembles

Powerful, but with increased algorithmic and model complexity

Model Averaging

Bagging (Breiman, 1996): Fit many large trees to bootstrap-resampled versions of the training data, and classify by majority vote.

Boosting (Freund & Shapire, 1996): Fit many large or small trees to reweighted versions of the training data. Classify by weighted majority vote.

Random Forests (Breiman 1999): Fancier version of bagging. In general Boosting>Random Forests>Bagging>Single Tree

Bagging

Bagging, short for 'bootstrap aggregating', is a simple but highly effective ensemble method that creates diverse models on different random samples of the original dataset.

These samples are taken uniformly with replacement and are known as bootstrap samples.

Bagging

Bagging is the application of the Bootstrap procedure to a high-variance machine learning algorithm, typically decision trees.

Decision trees are sensitive to the specific data on which they are trained. If the training data is changed (e.g. a tree is trained on a subset of the training data) the resulting decision tree can be quite different and in turn the predictions can be quite different.

Random forest

Definition

- Collection of unpruned CARTs
- Rule to combine individual tree decisions

Purpose

- Improve prediction accuracy

Principle

- Encouraging diversity among the tree

Solution: randomness

- Bagging
- Random decision trees (rCART)

A Forest Model

Parallel, bagged ensemble of the tree models

Strong predictive power, but lower interpretability

Many trees created using subsets of the data and averaged together

More hypermeters than decisions control the model complexity

- Number of trees
- Sampling rate
- Number of variables to try

Bagging

Advantages:
Technique of ensemble learning...
- ... to avoid over-fitting
  - Important since trees are unpruned
- ... to improve stability and accuracy

Two steps
- Bootstrap sample set
- Aggregation

Bootstrap

L: original learning set composed of p samples

Generate K learning sets Lk...

... composed of q samples, $q \leq p$,...

... obtained by uniform sampling with replacement from L

In consequences, Lk may contain repeated samples

Random forest: q = p

Asymptotic proportion of unique samples in Lk = 100 (1 - 1/e) ~ 63%

→ The remaining samples can be used for testing

Aggregation

Learning

For each Lk, one classifier Ck (rCART) is learned

Prediction

S: a new sample

Aggregation = majority vote among the K predictions/votes Ck(S)

Bagging/Bootstrap Aggregation

Motivation: Average a given procedure over many samples to reduce the variance.

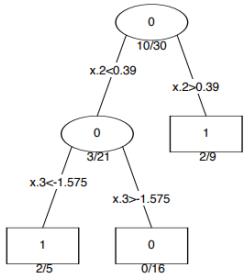
- Given a classifier $C(S, x)$, based on our training data S, producing a predicted class label at input point x.
- To bag C, we draw bootstrap samples $S_1, ..., S_B$ each of size N with replacement from the training data. Then

$$\widehat{C}_{\text{Bag}} = \text{Mojarity Vote}\{C(S_b, x)\}_{b=1}^{B}.$$

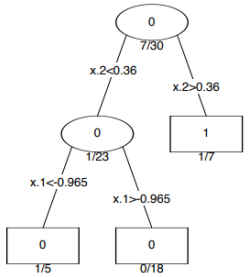
- Bagging can dramatically reduce the variance of unstable procedures (like trees), leading to improved prediction.
- All simple structures in a tree are lost.

Bagging/Bootstrap Aggregation

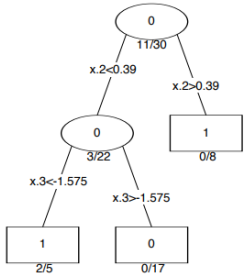
Original Tree



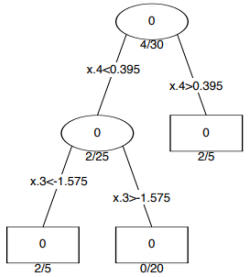
Bootstrap Tree 1



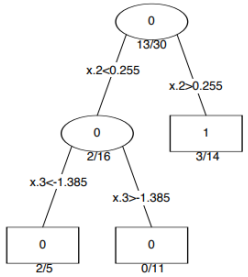
Bootstrap Tree 2



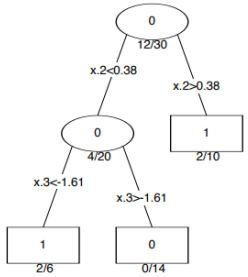
Bootstrap Tree 3



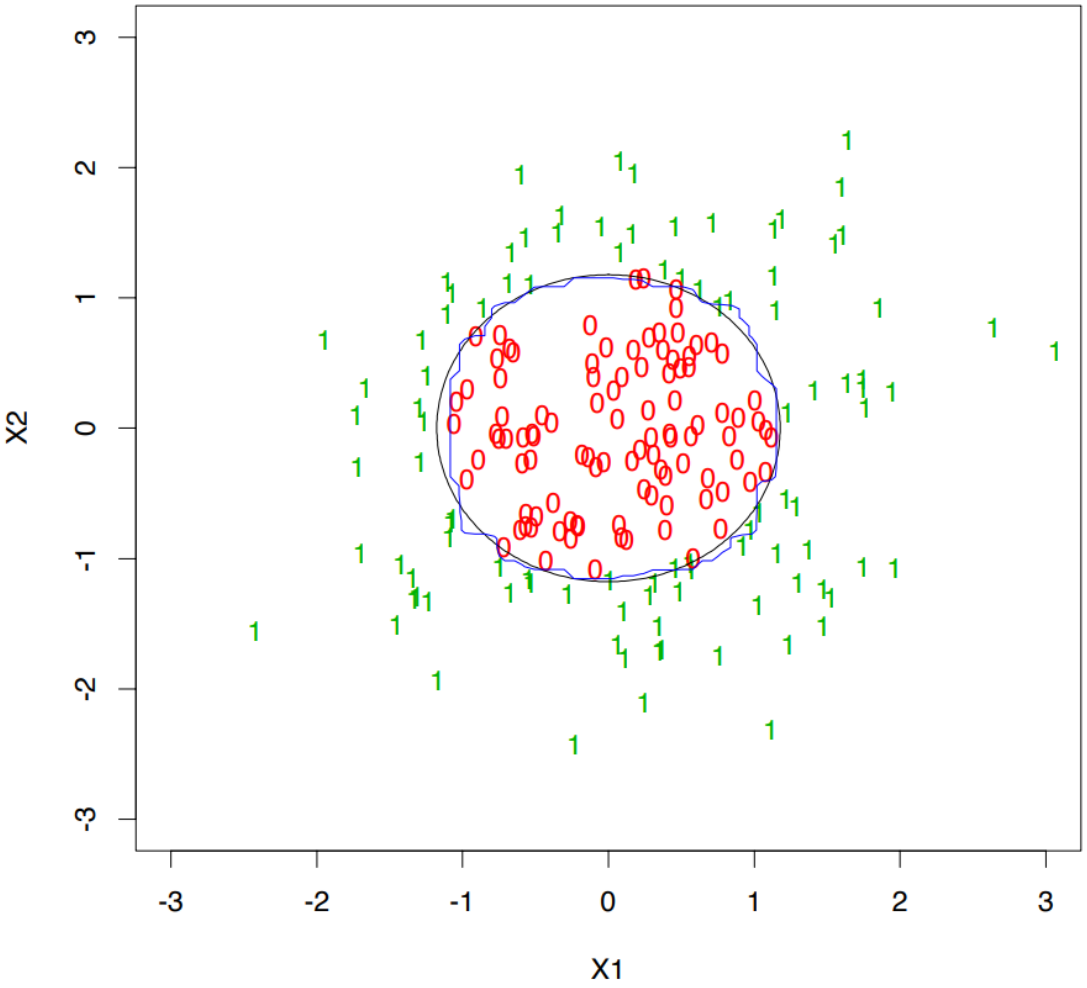
Bootstrap Tree 4



Bootstrap Tree 5



Bagging Decision Trees



A smoother decision boundary.

Classification error: 3.2% (Single deeper tree 7.3%).

Random Forest

A problem with decision trees like CART is that they are greedy. They choose which variable to split on using a greedy algorithm that minimizes error. As such, even with Bagging, the decision trees can have a lot of structural similarities and in turn have high correlation in their predictions.

Combining predictions from multiple models in ensembles works better if the predictions from the sub-models are uncorrelated or at best weakly correlated.

Random forest changes the algorithm for the way that the sub-trees are learned so that the resulting predictions from all of the subtrees have less correlation.

Random Forest Model

Parallel, bagged ensemble of the tree models

Strong predictive power, but lower interpretability

Many trees created using subsets of the data and averaged together

More hypermeters than decisions control the model complexity

- Number of trees
- Sampling rate
- Number of variables to try

Random Forest

Bagging features and samples simultaneously:

- At each tree split, a random sample of m features is drawn, and only those m features are considered for splitting. Typically $m = \sqrt{d}$ or $\log_2 d$, where d is the number of features
- For each tree grown on a bootstrap sample, the error rate for observations left out of the bootstrap sample is monitored. This is called the "out-of-bag" error rate.
- random forests tries to improve on bagging by "de-correlating" the trees. Each tree has the same expectation.

Random decision tree

All labeled samples initially assigned to root node

N ← root node

With node N do

Find the feature F among a random subset of features + threshold value T...

... that split the samples assigned to N into 2 subsets S_{left} and S_{right}...

... so as to maximize the label purity within these subsets

Assign (F,T) to N

If S_{left} and S_{right} too small to be splitted

Attach child leaf nodes L_{left} and L_{right} to N

Tag the leaves with the most present label in S_{left} and S_{right}, resp.

else

Attach child nodes N_{left} and N_{right} to N

Assign S_{left} and S_{right} to them, resp.

Repeat procedure for $N = N_{left}$ and $N = N_{right}$

Random subset of features

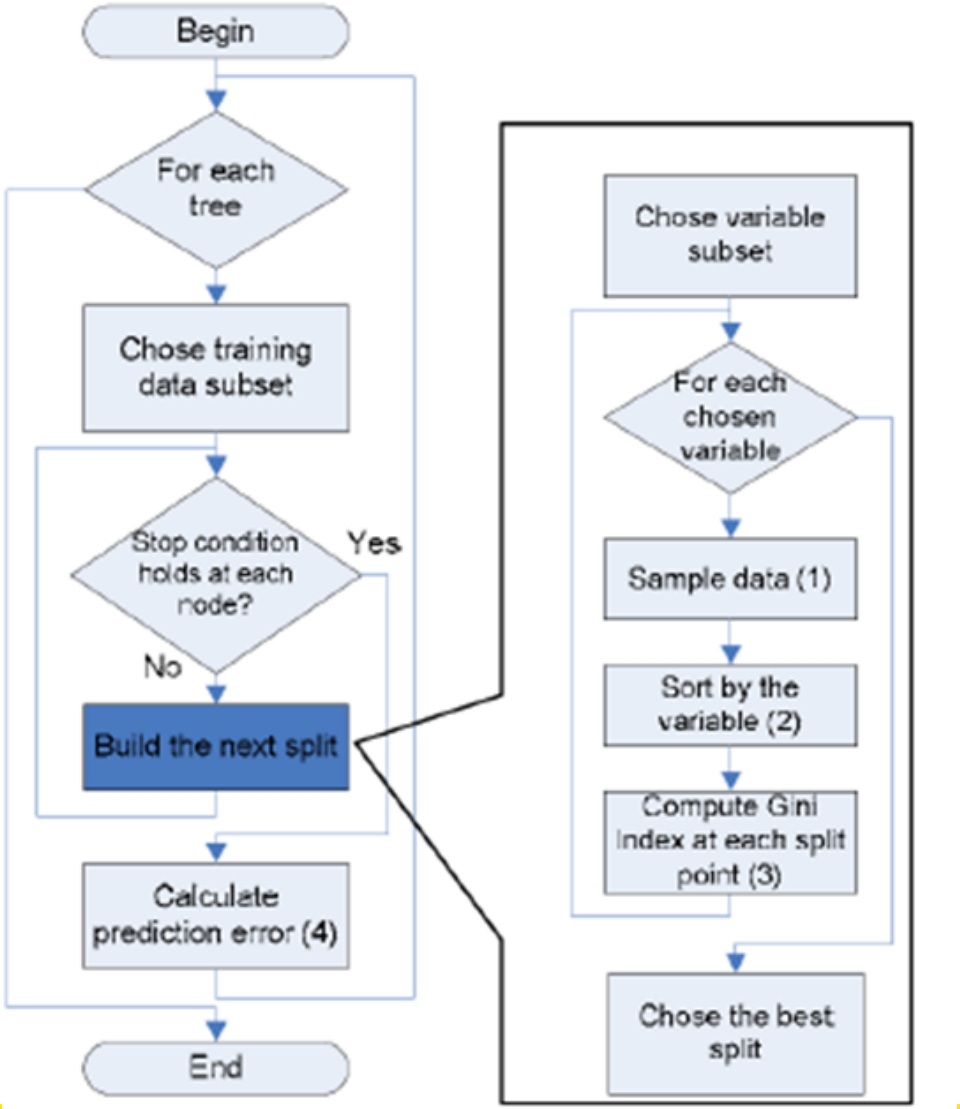
Random drawing repeated at each node

For D-dimensional samples, typical subset size = round(sqrt(D)) (also round(log2(x)))

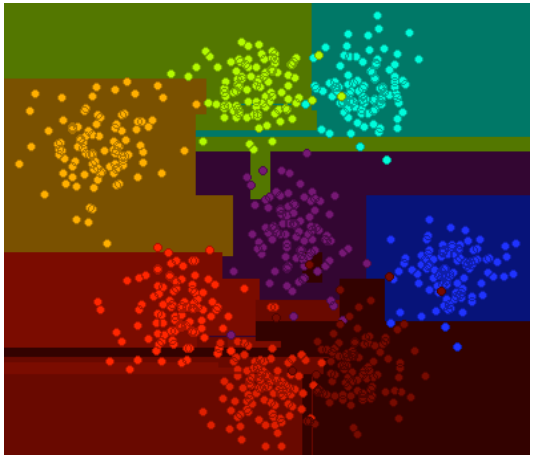
→ Increases diversity among the rCARTs + reduces computational load

Typical purity: Gini index

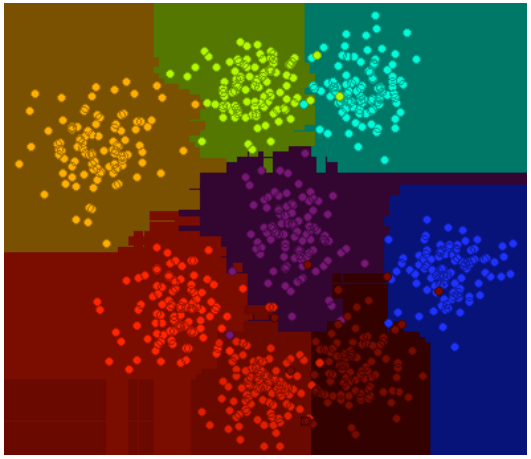
Random decision tree



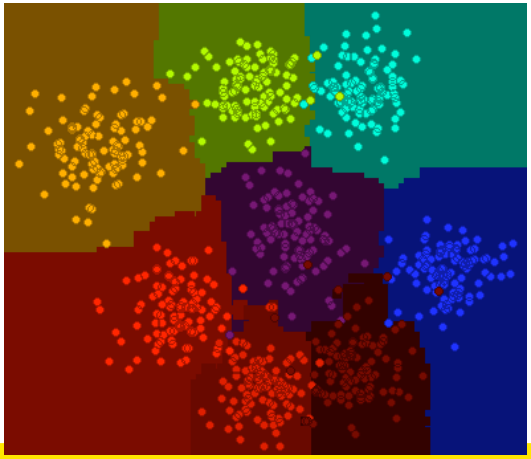
Illustration



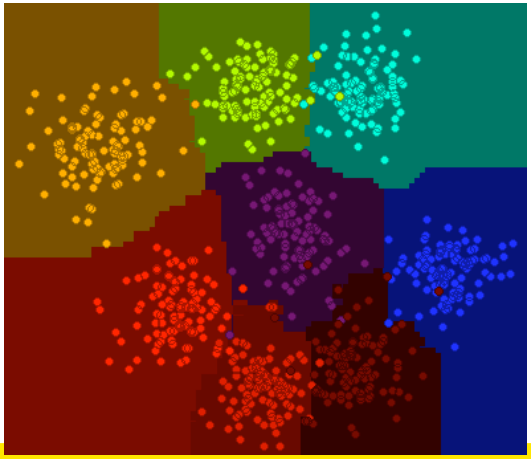
1 rCART



10 rCARTs



100 rCARTs



500 rCARTs

Features and Advantages

It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.

It runs efficiently on large databases.

It can handle thousands of input variables without variable deletion.

Features and Advantages

It gives estimates of what variables are important in the classification.

It generates an internal unbiased estimate of the generalization error as the forest building progresses.

It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.

Features and Advantages

It has methods for balancing error in class population unbalanced data sets.

Generated forests can be saved for future use on other data.

Prototypes are computed that give information about the relation between the variables and the classification.

Features and Advantages

It computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data.

The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.

It offers an experimental method for detecting variable interactions.