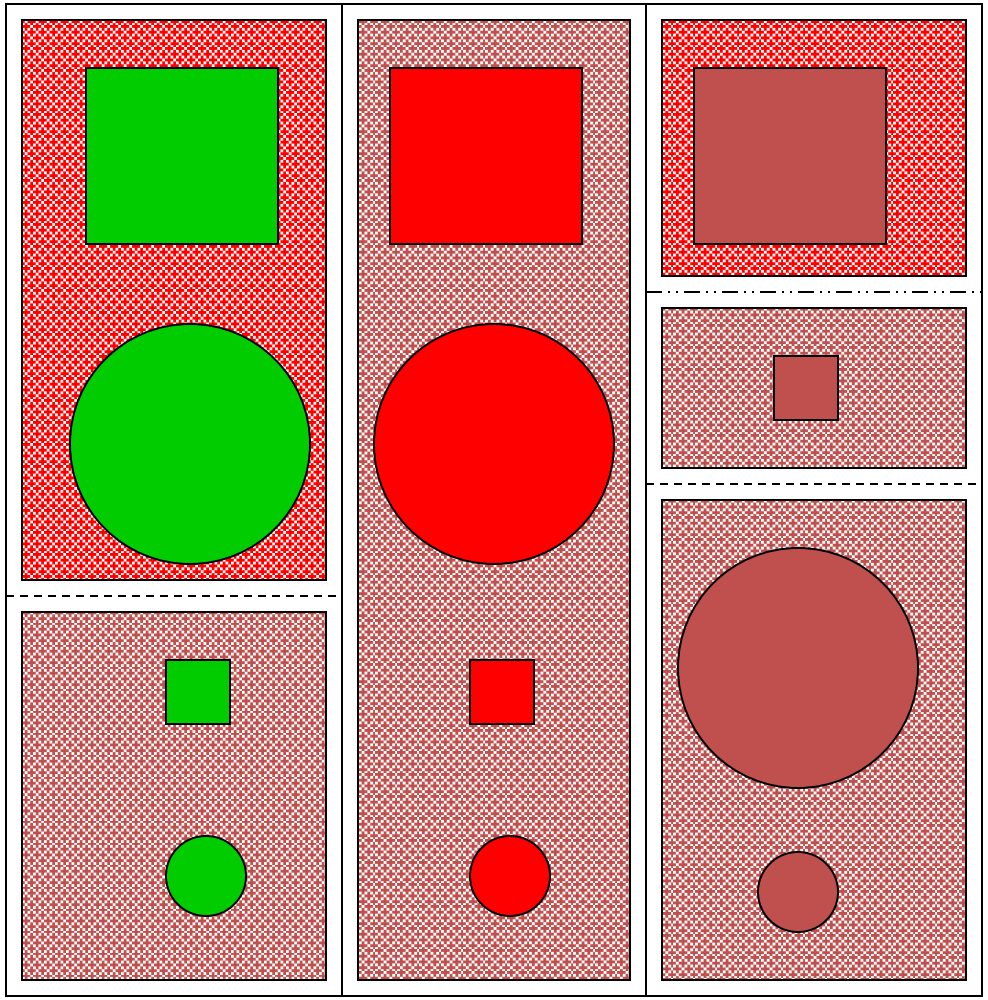
DECISION TREES & RANDOM FORESTS

Decision Trees (DTs)

- **Supervised** learning method used for **classification** and **regression**
- Given a set of training tuples, learn model to predict one value from the others
  - Learned value typically a class (e.g., goodRisk)
- Resulting model is simple to understand, interpret, visualize, and apply
- One of the oldest ML algorithms, but still useful for many problems

Learning a Concept

The red groups are **negative** examples, blue **positive**



Shape Attributes

- **Size**: large, small
- **Color**: red, green, blue
- **Shape**: square, circle

Task

Classify new object with a size, color & shape as positive or negative

Training data

Attribute to be learned

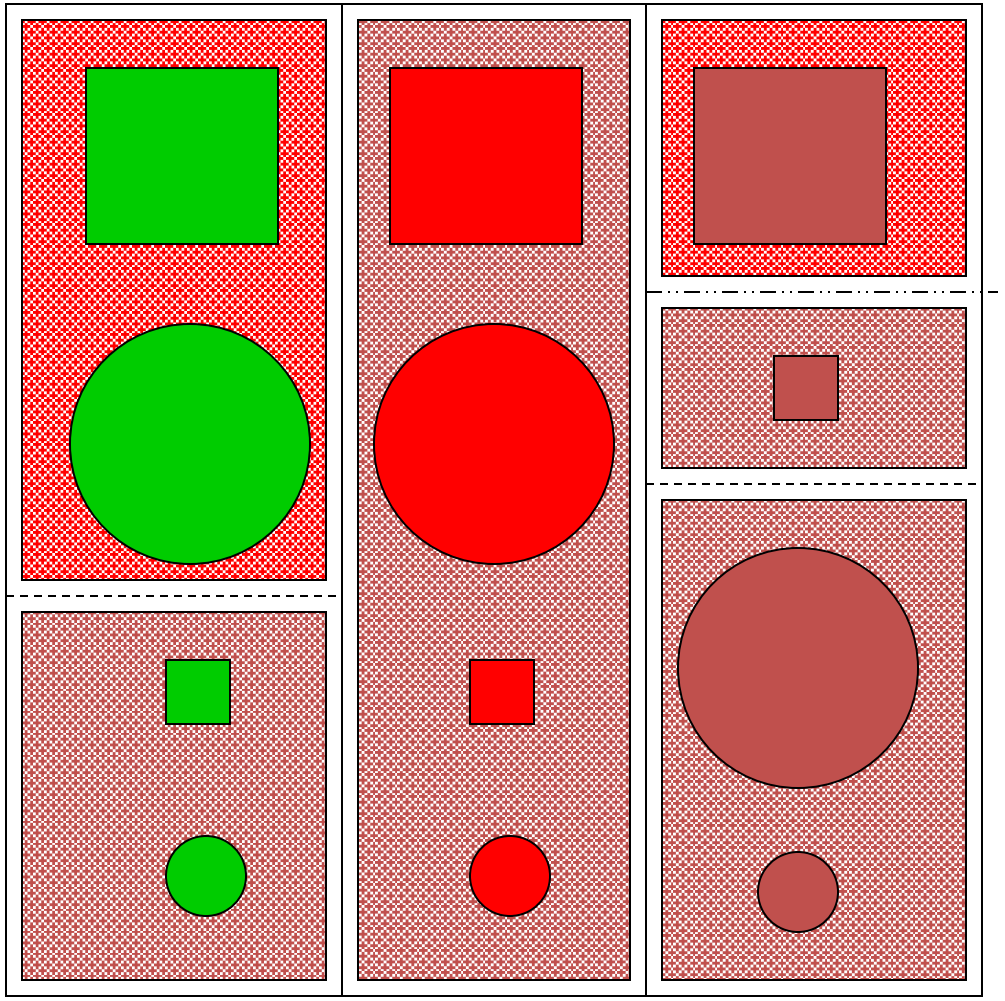
Size	Color	Shape	class
Large	Green	Square	Negative
Large	Green	Circle	Negative
Small	Green	Square	Positive
Small	Green	Circle	Positive
Large	Red	Square	Positive
Large	Red	Circle	Positive
Small	Red	Square	Positive
Small	Red	Circle	Positive
Large	Blue	Square	Negative
Small	Blue	Square	Positive
Large	Blue	Circle	Positive
Small	Blue	Circle	Positive

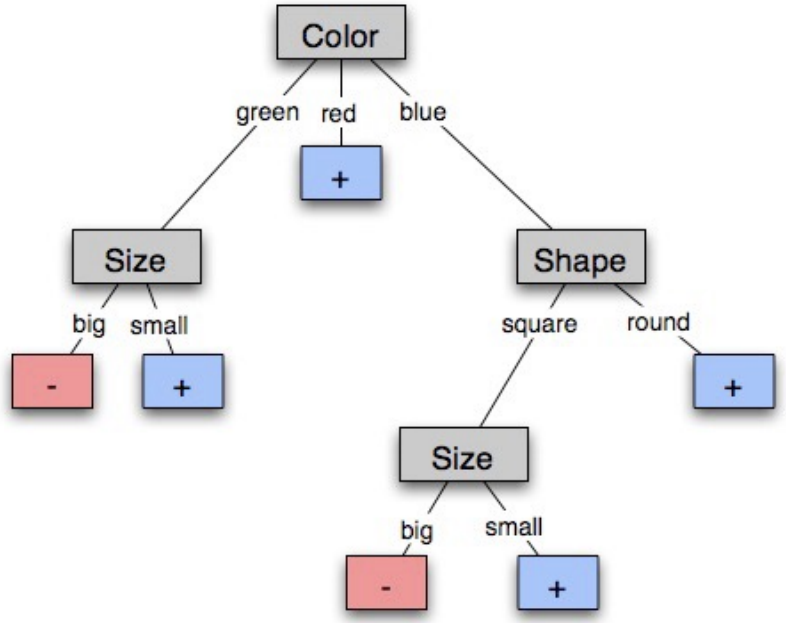
attributes

example instances

A decision tree-induced partition

The red groups are negative examples, blue positive

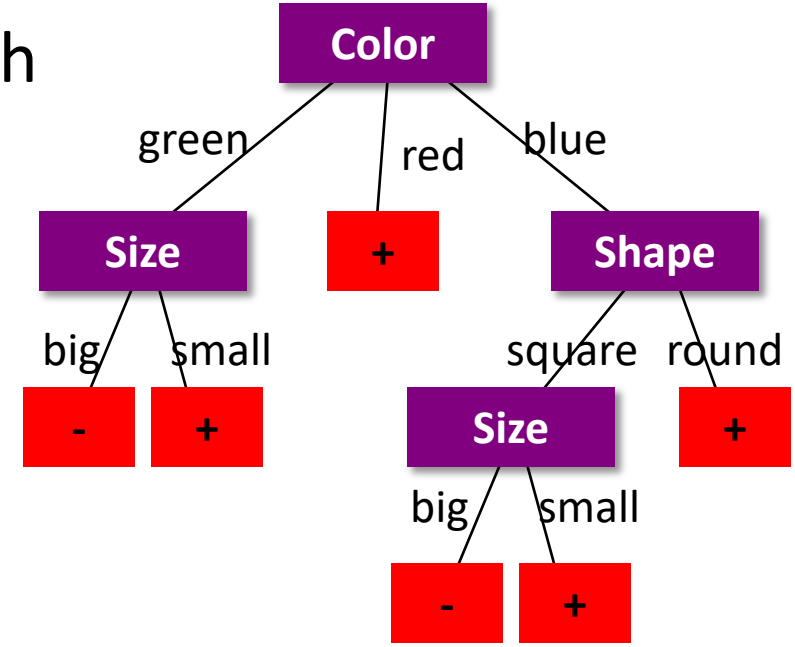




Negative things are big, green shapes and big, blue squares

Learning decision trees

- Goal: Build **decision tree** to classify examples as positive or negative instances of concept using supervised learning from training data
- A **decision tree** is a tree in which
  - **non-leaf nodes** have an attribute (feature)
  - **leaf nodes** have a classification (+ or -)
  - **arcs** have a possible value of its attribute

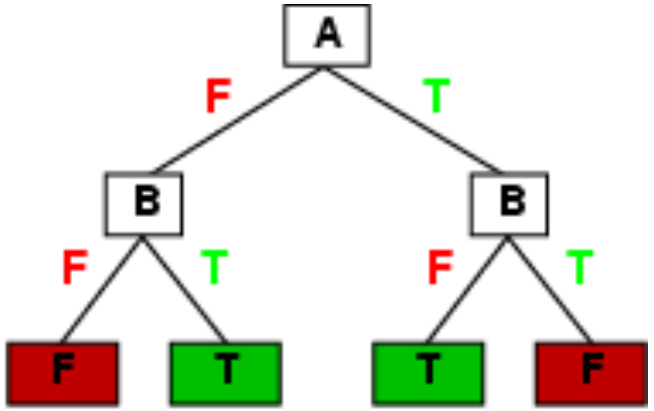


- Generalization: allow for >2 classes
  - e.g., classify stocks as {sell, hold, buy}

Expressiveness of Decision Trees

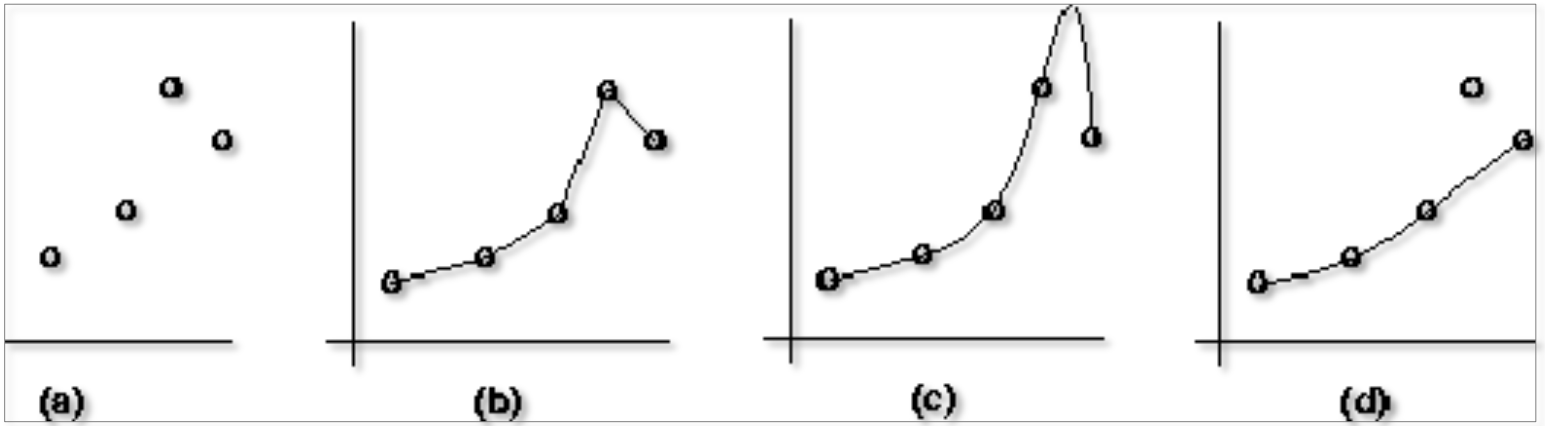
- Can express any function of input attributes, e.g., for Boolean functions, truth table row → path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- There's a consistent decision tree for any training set with one path to leaf for each example, but it probably won't **generalize** to new examples
- Prefer more **compact** decision trees

Inductive learning and bias



- Suppose that we want to learn a function **f(x) = y** and we're given sample (x,y) pairs, as in figure (a)
- Can make several hypotheses about f, e.g.: (b), (c) & (d)
- Preference reveals learning technique **bias**, e.g.:
  - prefer piece-wise linear functions (b)
  - prefer a smooth function (c)
  - prefer a simpler function and treat outliers as noise (d)

Preference bias: Occam's Razor

- William of Ockham (1285-1347)
  - *non sunt multiplicanda entia praeter necessitatem*
  - entities are not to be multiplied beyond necessity
- **Simplest** consistent explanation is the best
- **Smaller** decision trees correctly classifying training examples preferred over larger ones
- Finding **the** smallest decision tree is NP-hard, so we use algorithms that find reasonably small ones

Issues

- It's like 20 questions
- We can generate many decision trees depending on what attributes we ask about and in what order
- How do we decide?
- What makes one decision tree better than another: number of nodes? number of leaves? maximum depth?

ID3 / C4.5 / J48 Algorithm

- Greedy algorithm for decision tree construction developed by Ross Quinlan 1987-1993
- Top-down construction of tree by recursively selecting ***best attribute*** to use at current node
  - Once attribute selected for current node, generate child nodes, one for each possible attribute value
  - Partition examples using values of attribute, & assign these subsets of examples to the child nodes
  - Repeat for each child node until examples associated with a node are all positive or negative

Choosing best attribute

- **Key problem:** choose attribute to split given set of examples
- Possibilities for choosing attribute:
  - **Random:** Select one at random
  - **Least-values:** one with smallest # of possible values
  - **Most-values:** one with largest # of possible values
  - **Max-gain:** one with largest expected ***information gain***
  - **Gini impurity:** one with smallest gini impurity value
- The last two measure the **homogeneity** of the target variable within the subsets
- The ID3 and C4.5 algorithms uses **max-gain**

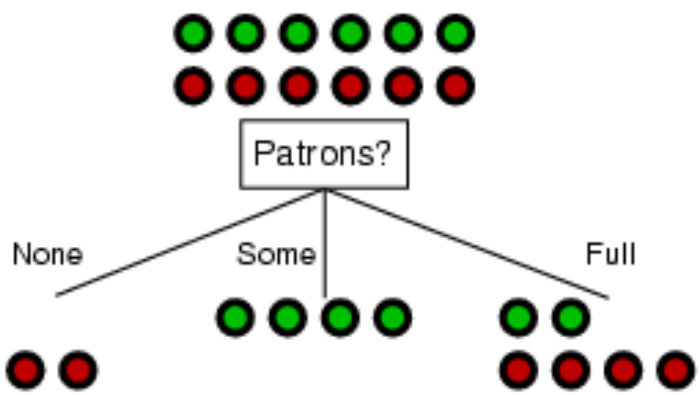
A Simple Example

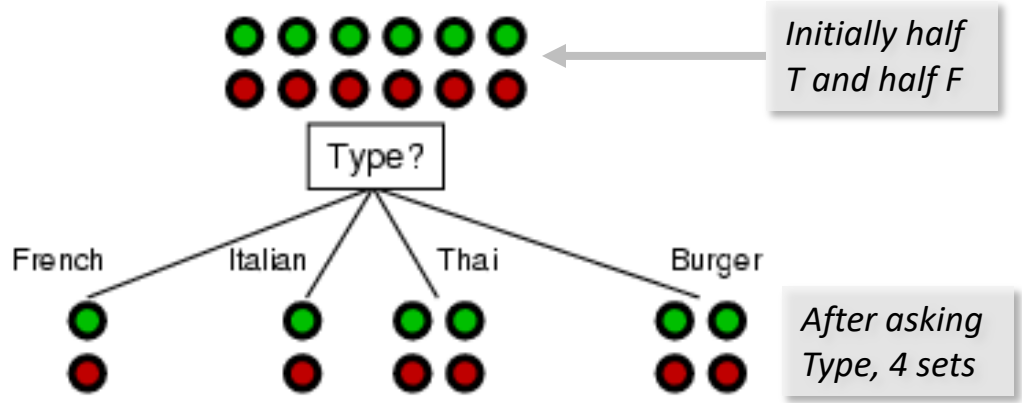
For this data, is it better to start the tree by asking about the restaurant **type** or its current **number of patrons**?

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Choosing an attribute

Idea: good attribute choice splits examples into subsets that are as close to *all of one type* as possible, e.g., for binary attributes, all T or all F

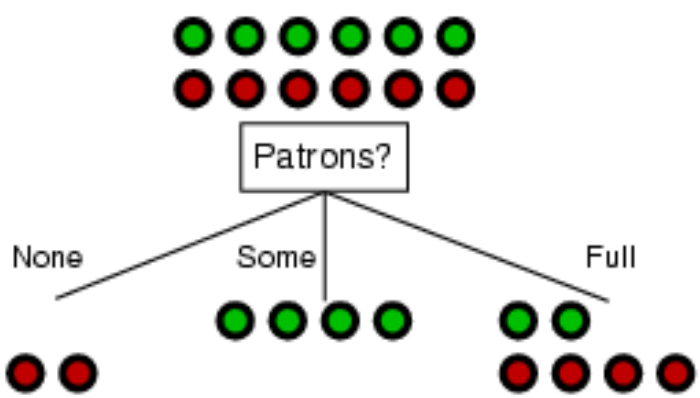


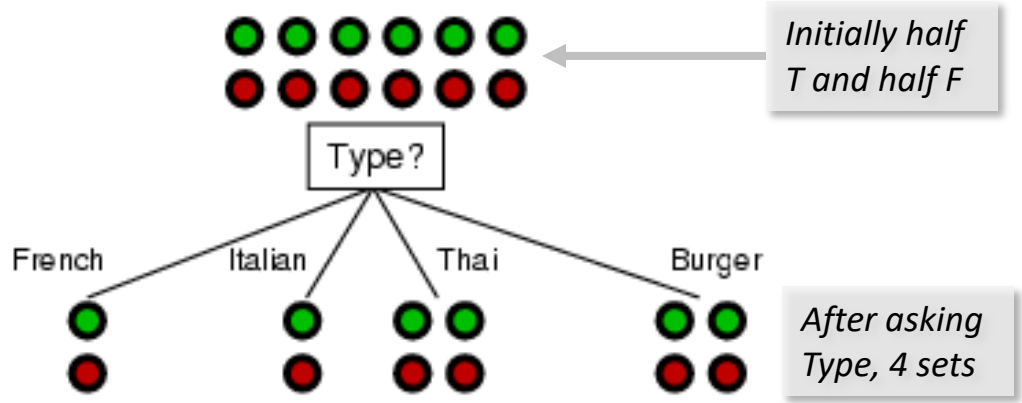


Which is better: asking about *Patrons* or *Type?*

Choosing an attribute

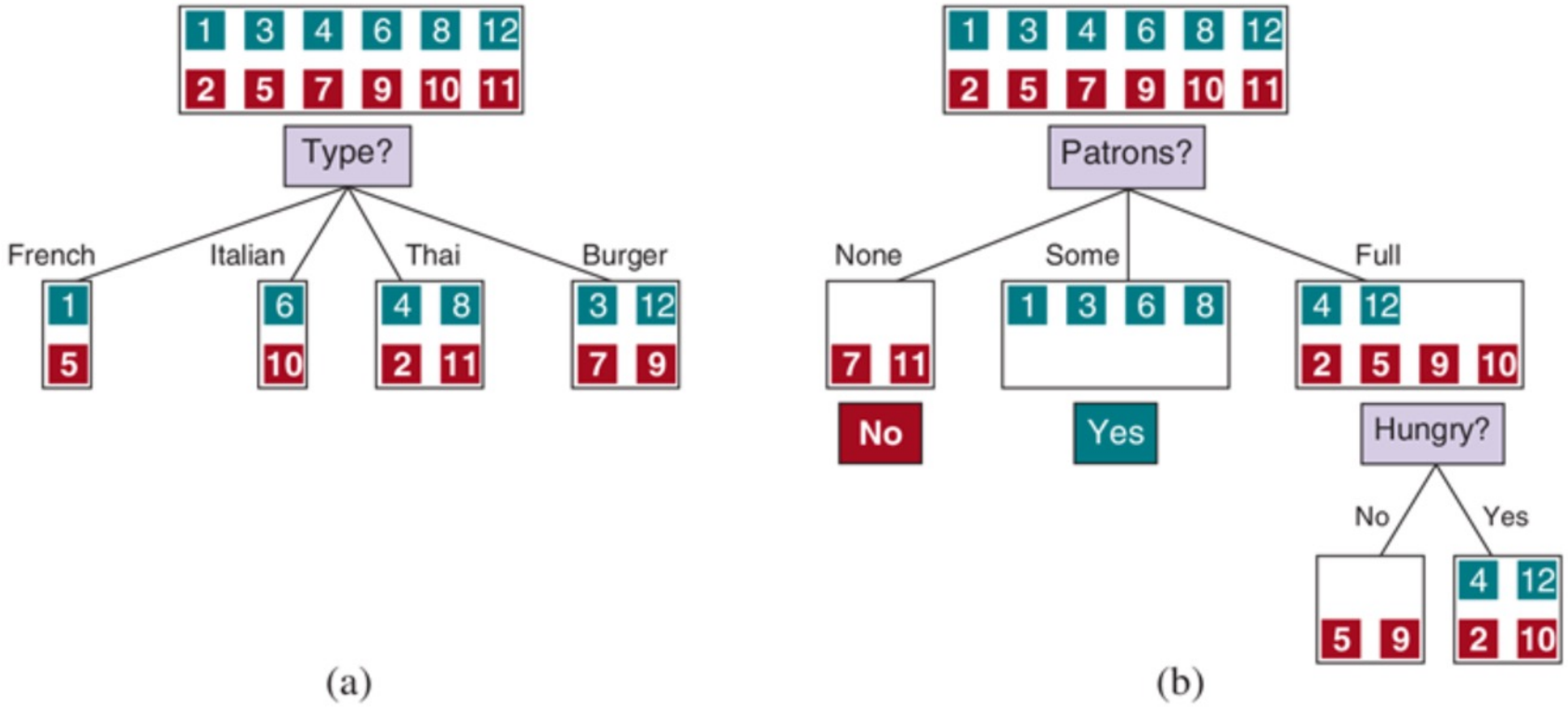
Idea: good attribute choice splits examples into subsets that are as close to *all of one type* as possible, e.g., for binary attributes, all T or all F





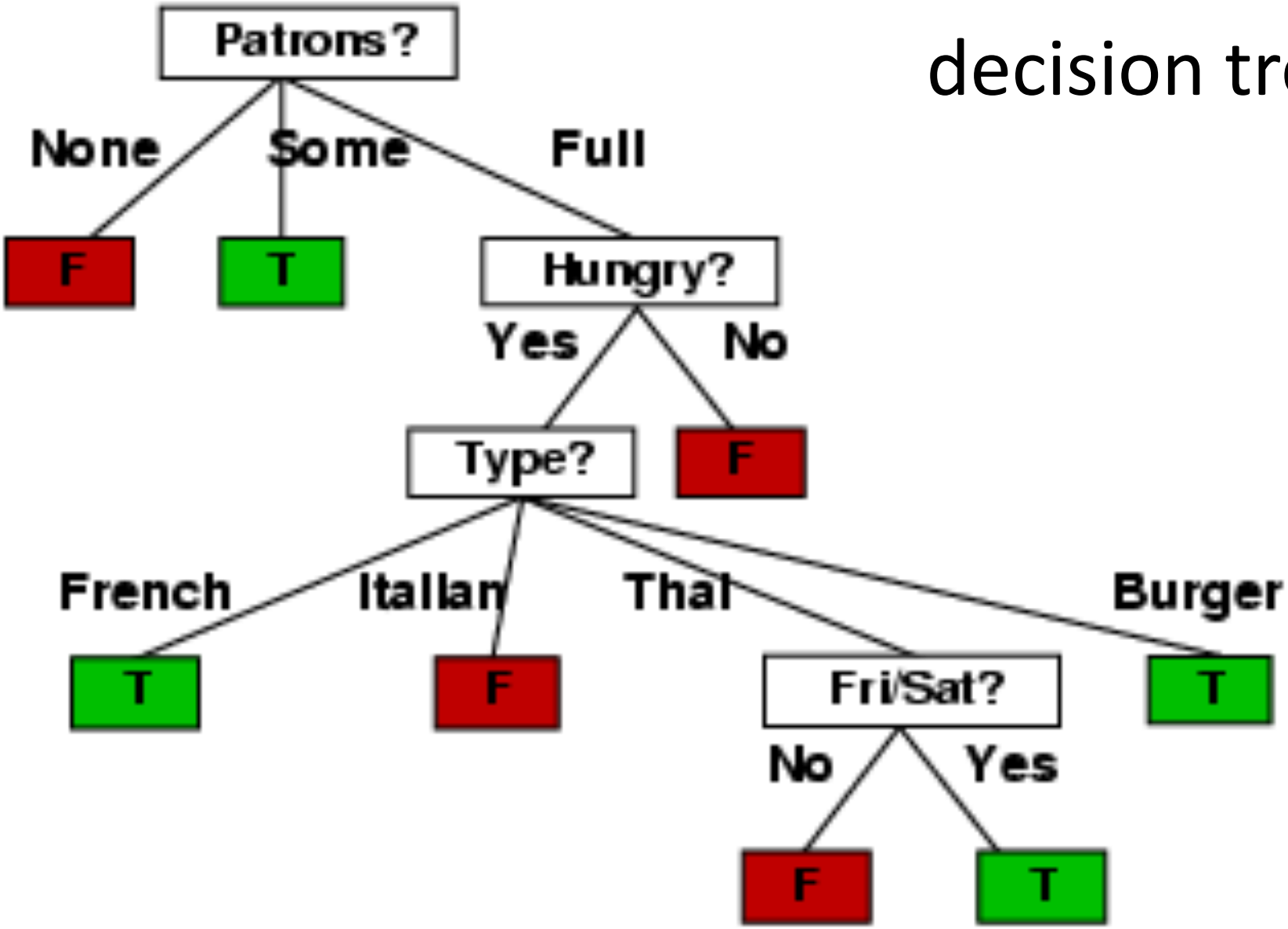
- **Patrons:** for six examples we know the answer and for six we can predict with probability 0.67
- **Type:** our prediction is no better than chance (0.50)

Choosing Patrons yields more information

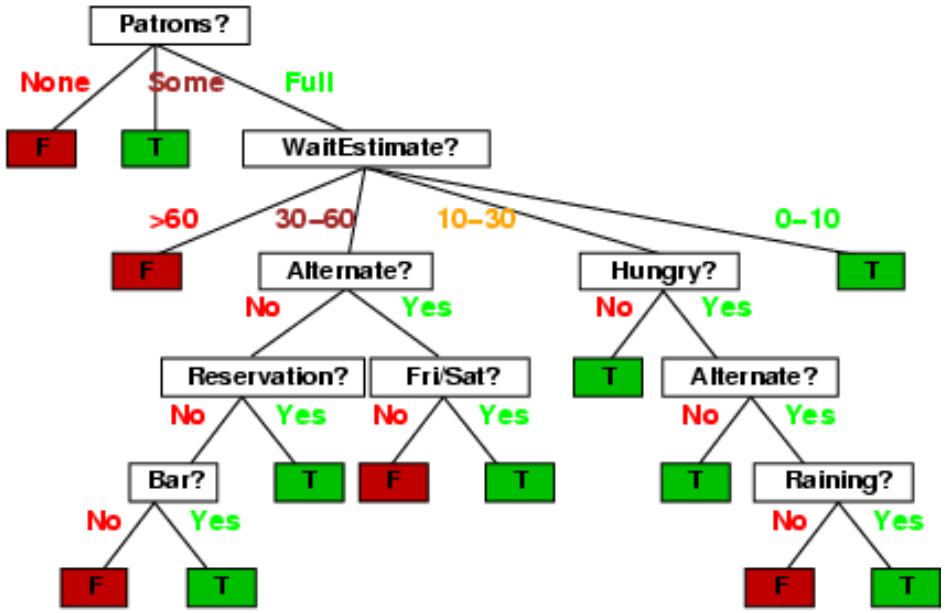


The ID3 algorithm used this to decide what attribute to ask about next when building a decision tree

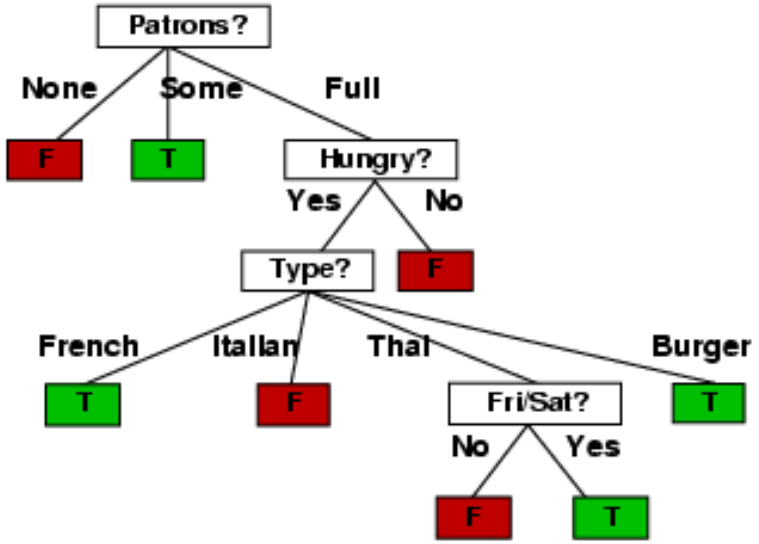
ID3-induced decision tree



Compare the two Decision Trees



Human-generated decision tree



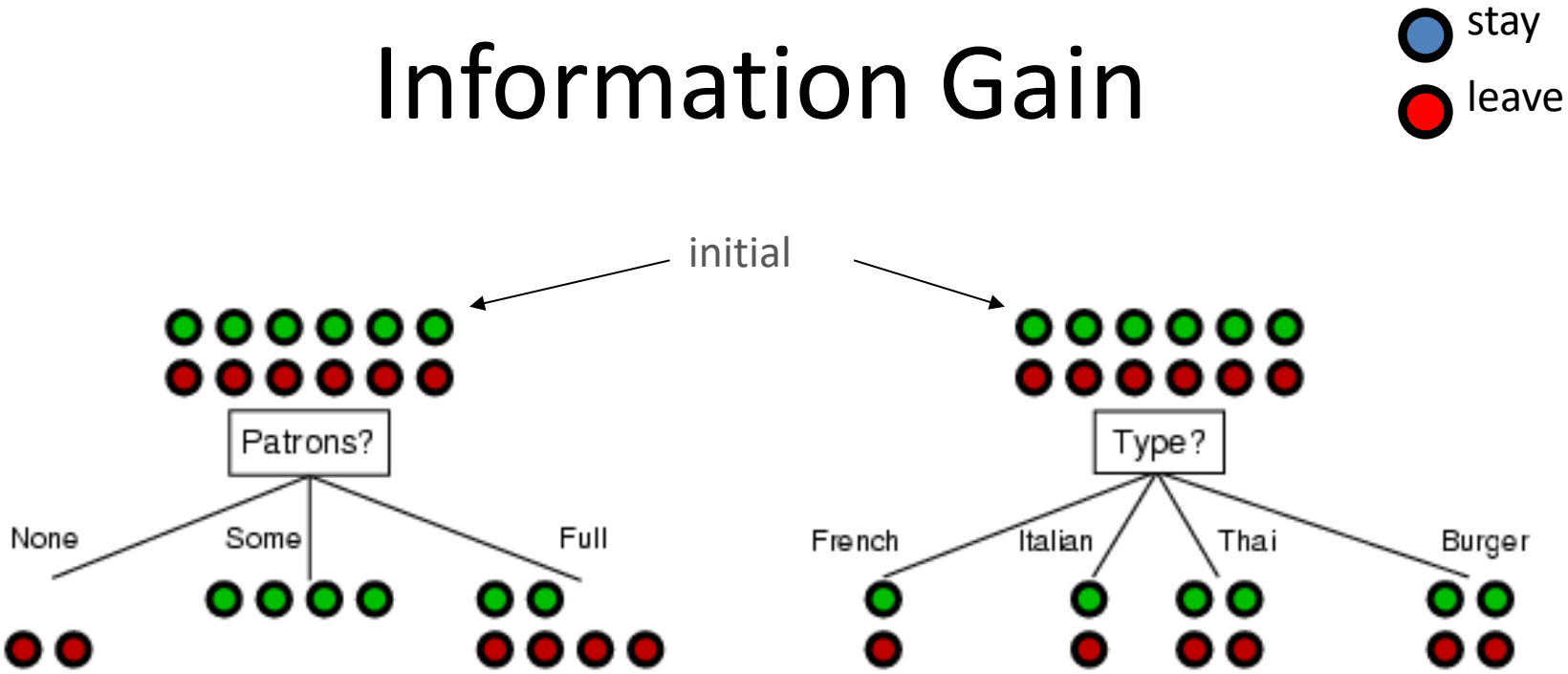
ID3-generated decision tree

- Intuitively, ID3 tree looks better: it's shallower and has fewer nodes
- ID3 uses **information theory** to decide which question is best to ask next

Information gain in knowing an attribute

- **Gain(X,T) = Info(T) - Info(X,T)** is difference of
  - **Info(T):** info needed to identify T's class
  - **Info(X,T):** info needed to identify T's class after attribute X's value known
- This is gain in information due to knowing value of attribute X
- Used to rank attributes and build DT where each node uses attribute with greatest gain of those not yet considered in path from root
- goal: **create small DTs** to minimize questions

Information Gain



- Initially half of examples are stay and half **leave**
- After knowing Type?, still half are stay and half **leave**
  We are no wiser for knowing Type ☹
- After knowing Patrons?, we know the class for six and know a likely class for the other six
  We've learned something, but need more info if Patrons=Full 😊

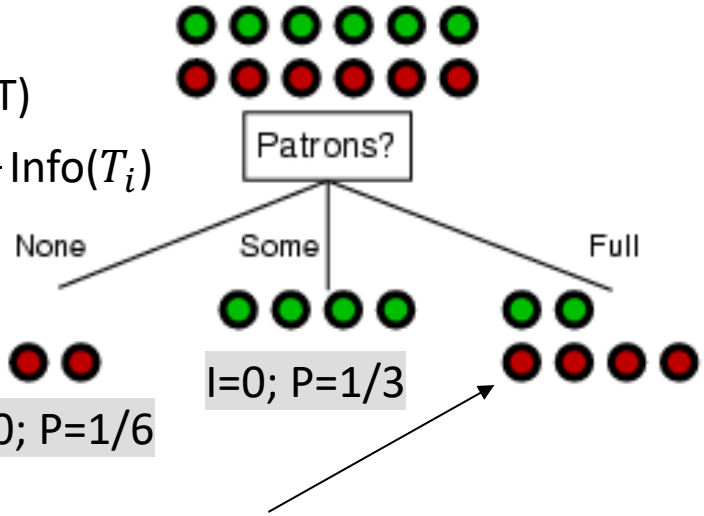
Information Gain

stay
leave

I = Info(T)
$= -\sum_c \widehat{p_c} \log_2 \widehat{p_c}$

I = -(.5*$\log_2$(.5) + .5*$\log_2$(.5)) = 0.5+0.5 => **1.0**

Info(X, T)
$= \sum_i \frac{|T_i|}{|T|} \text{Info}(T_i)$

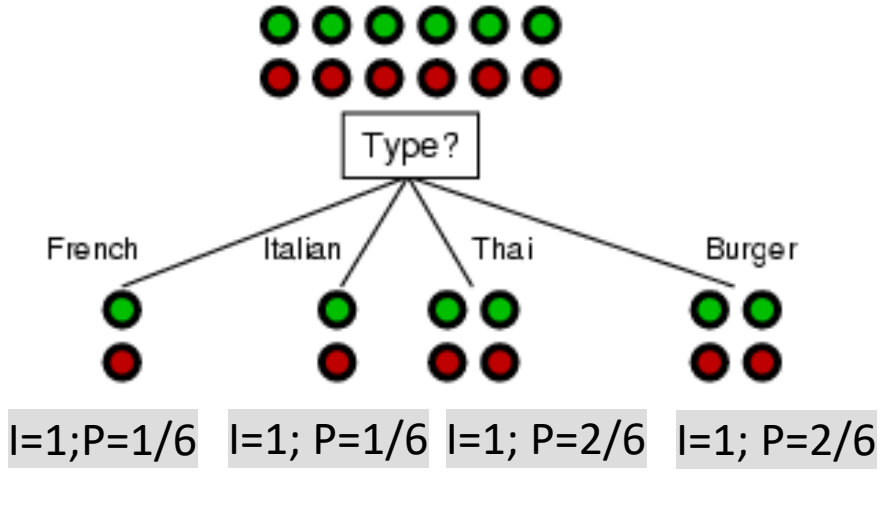


I=0; P=1/6

I=0; P=1/3

I=-(1/3*$\log_2$(1/3)+2/3*$\log_2$(2/3)),
P=6/12=1/2 => 0.91/2 = 0.46

Information gain = 1 - 0.46 => **0.54**



I=1;P=1/6

I=1; P=1/6

I=1; P=2/6

I=1; P=2/6

I = 6/6*1 => **1.0**

Information gain = 1 - 1 => **0.0**

- **Information gain** for asking **Patrons = 0.54**, for asking **Type = 0**
- Note: If only one of the N categories has any instances, the information entropy is always 0

Extensions of ID3

- Using other selection metric gain ratios, e.g., gini impurity metric
- Handle real-valued data
- Noisy data and overfitting
- Generation of rules
- Setting parameters
- Cross-validation for experimental validation of performance
- **C4.5:** extension of ID3 accounting for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, etc.

Real-valued data?

- Many ML systems work only on **nominal data**
- We often classify data into one of 4 basic types:
  - **Nominal data** is named, e.g., representing restaurant type as Thai, French, Italian, Burger
  - **Ordinal data** has a well-defined sequence: small, medium, large
  - **Discrete data** is easily represented by integers
  - **Continuous data** is captured by **real** numbers
- There are others, like intervals: age 0-3, 3-5, …
- Handling some types may need new techniques

Real-valued => Nominal Data

For ML systems that expect nominal data:

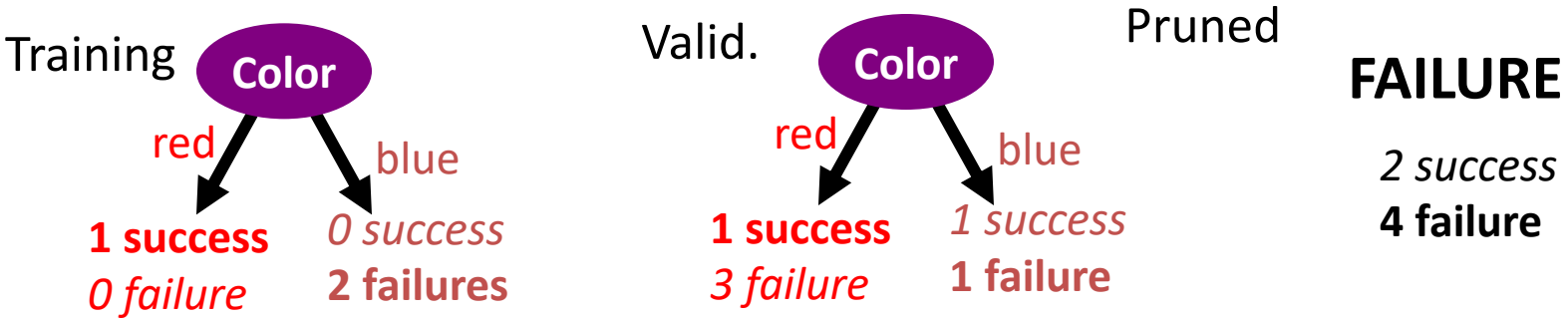
- Select thresholds defining intervals so each becomes a discrete value of attribute
- Use heuristics: e.g., always divide into quartiles
- Use domain knowledge: e.g., divide age into infant (0-2), toddler (2-5), school-aged (5-8)
- Or treat this as another learning problem
  - Try different ways to discretize continuous variable; see which yield better results w.r.t. some metric
  - E.g., try midpoint between every pair of values

Avoiding Overfitting

- Remove obviously **irrelevant features**
  - E.g., remove ‘year observed’, ‘month observed’, ‘day observed’, ‘observer name’ from the attributes used
- Get **more training data**
- **Pruning** lower nodes in a decision tree
  - E.g., if info. gain of best attribute at a node is below a threshold, stop and make this node a leaf rather than generating children nodes

Pruning decision trees

- Pruning a decision tree is done by replacing a whole subtree by a leaf node
- Replacement takes place if the expected error rate in the subtree is greater than in the single leaf, e.g.,
  - **Training data:** 1 training red success and 2 training blue failures
  - **Validation data**: 3 red failures and one blue success
  - Consider replacing subtree by a single node indicating failure
- After replacement, only 2 errors instead of 4



Ensembles

Key Idea: “Wisdom of the crowd“

groups of people can often make better decisions than individuals

Apply this to ML

Learn multiple classifiers and combine their predictions

Combining Multiple Classifiers by Voting

Train several classifiers and take majority of predictions

For regression use mean or median of the predictions

For ranking and collective classification use some form of averaging

A common family of approaches is called bagging

Bagging: Split the Data

Q: What can go wrong with option 1?

Option 1: Split the data into K pieces and train a classifier on each

Bagging: Split the Data

Option 1: Split the data into K pieces and train a classifier on each

Q: What can go wrong with option 1?

A: Small sample → poor performance

Bagging: Split the Data

Option 1: Split the data into K pieces and train a classifier on each

Q: What can go wrong with option 1?

A: Small sample → poor performance

Option 2: Bootstrap aggregation (bagging) resampling

Bagging: Split the Data

Option 1: Split the data into K pieces and train a classifier on each

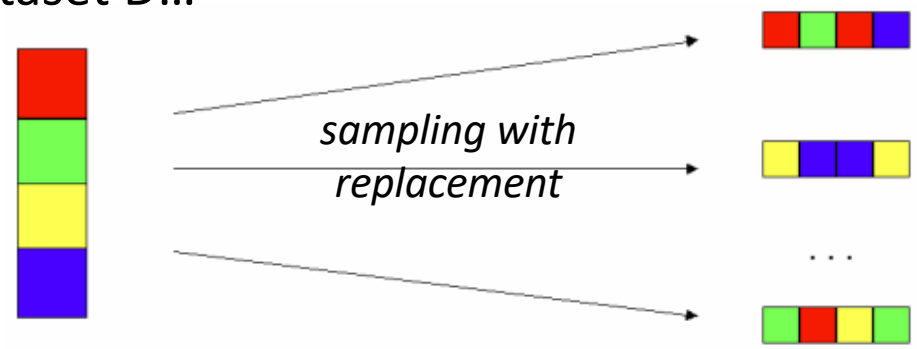
Q: What can go wrong with option 1?

A: Small sample → poor performance

Option 2: Bootstrap aggregation (bagging) resampling

Obtain datasets $D_1, D_2, \ldots, D_N$ using bootstrap resampling from D

Given a dataset D…



get new datasets $\hat{D}$ by random sampling with replacement from D

Courtesy Hamed Pirsiavash

Bagging: Split the Data

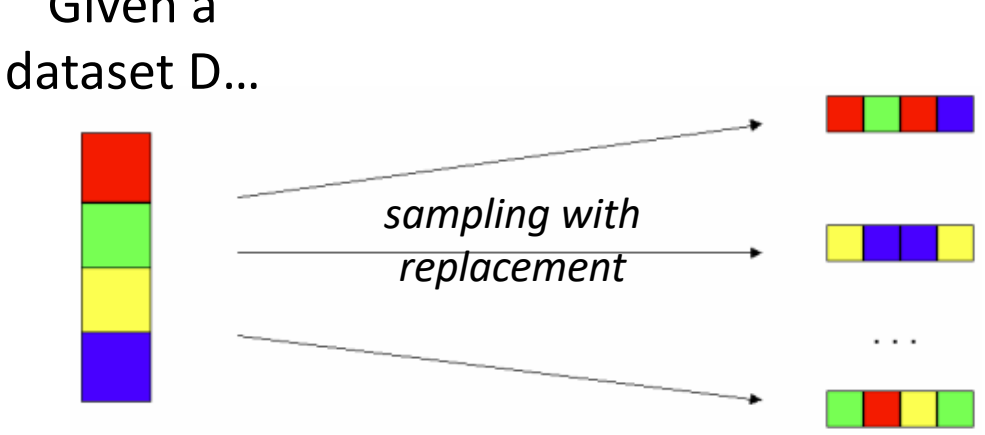
Option 1: Split the data into K pieces and train a classifier on each

Q: What can go wrong with option 1?

A: Small sample → poor performance

Option 2: Bootstrap aggregation (bagging) resampling

- Obtain datasets $D_1, D_2, \ldots, D_N$ using bootstrap resampling from D
- Train classifiers on each dataset and average their predictions



Courtesy Hamed Pirsiavash

Bagging Decision Trees

How would it work?

Bagging Decision Trees

How would it work?

Bootstrap sample S samples $\{(X_1, Y_1), \ldots, (X_S, Y_S)\}$

Train a tree t_s on (X_s, Y_s)

At test time: $\hat{y} = \text{avg}(t_1(x), \ldots t_S(x))$

Random Forests

Bagging trees with one modification

At each split point, choose a **random subset of features** of size **k** and pick the best among these

Train decision trees of depth **d**

Average results from multiple randomly trained trees

Q: What's the difference between bagging decision trees and random forests?

Courtesy Hamed Pirsiavash

Random Forests

Bagging trees with one modification

At each split point, choose a random subset of features of size **k** and pick the best among these

Train decision trees of depth **d**

Average results from multiple randomly trained trees

Q: What's the difference between bagging decision trees and random forests?

A: Bagging → highly correlated trees (reuse good features)

Courtesy Hamed Pirsiavash

Summary: decision tree learning

- Still widely used learning methods in practice for problems with relatively **few features**
- Strengths
  - **Fast and easy** to implement
  - **Simple model:** translate to a set of rules
  - **Useful: e**mpirically valid in many commercial products
  - **Robust:** handles noisy data
  - **Explainable:** easy for people to understand
- Weaknesses
  - Large decision trees may be hard to understand
  - Requires fixed-length feature vectors
  - Non-incremental, adding one new feature requires rebuilding entire tree

ADDED INFORMATION

Information theory 101

- For **n equally probable** possible messages or data values, each has probability **1/n**
- Def: Information of a message is $-\log_2(p) = \log_2(n)$

  e.g., with 16 messages, then log(16) = 4 and we need **4 bits** to identify/send each message
- What if the messages are not equally likely?
- For **probability distribution P** $(p_1, p_2 \ldots p_n)$ for n messages, its information (*H* or *information entropy*) is:

$$\mathbf{I(P)} = -(p_1 * \log(p_1) + p_2 * \log(p_2) + .. + p_n * \log(p_n))$$

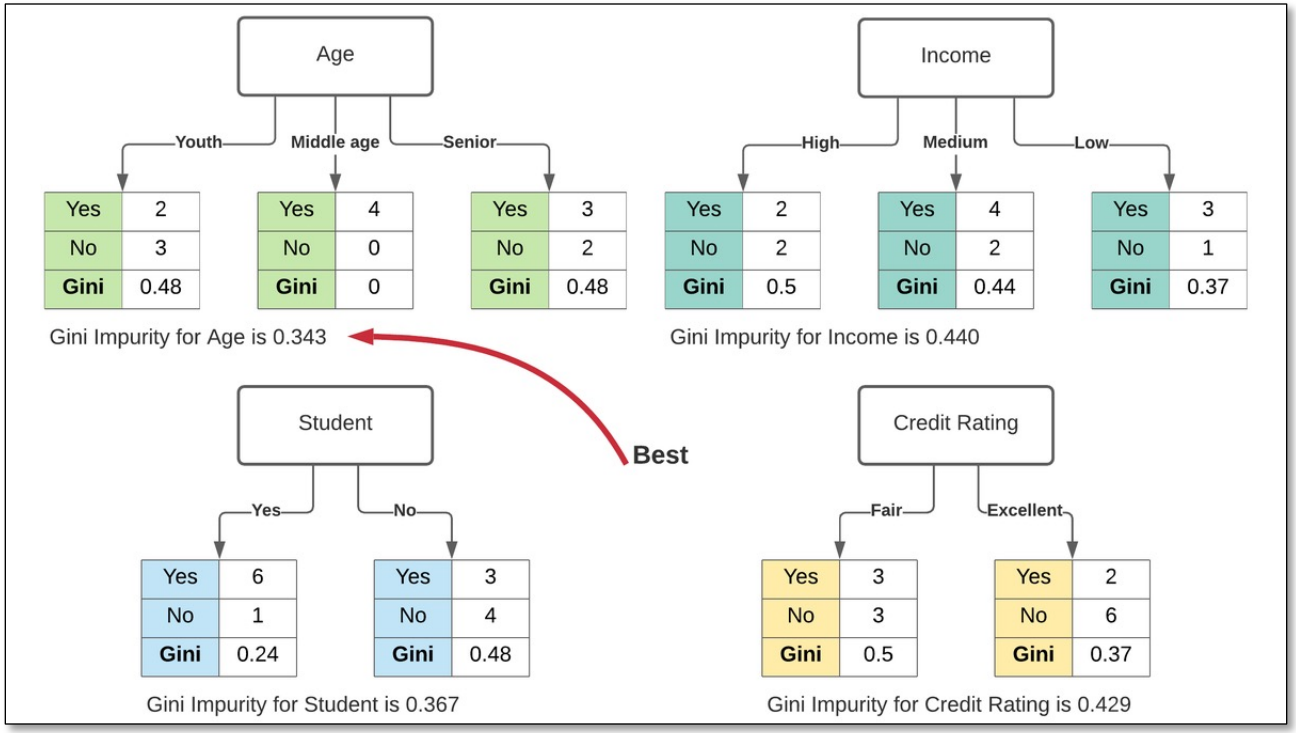
Information entropy of a distribution

$I(P) = -(p_1 * \log(p_1) + p_2 * \log(p_2) + .. + p_n * \log(p_n))$

- Examples:
  - If P is (0.5, 0.5) then I(P) = -(0.5*1 + 0.5*1) = **1**
  - If P is (1, 0) then I(P) = 1*0 + 0*log(0) = **0**
- More **uniform probability** distribution, **greater its information**: more information is conveyed by a message telling you which event actually occurred
- Entropy is the **average number of bits/message** needed to represent a **stream** of messages

Gini Impurity Metric of a Dataset

- Number between 0-0.5, lower is better
- Indicates likelihood of new data item being misclassified if given random class label according to class distribution
- Very similar to information gain, slightly faster to compute



DT to decide if someone a good credit risk based on 4 properties