Chapter 4 Machine Learning

COMP 6721 Introduction of AI

Russell & Norvig – Section 18.1 & 18.2

Supervised Learning Algorithms

- ***Linear Regression***
- ***Logistic Regression***
- Decision Tree
- Random Forest
- Naïve Bayes Classifier

Decision Tree

- Simplest, but most successful form of learning algorithm
- Very well-know algorithm is ID3 (Quinlan, 1987) and its successor C4.5
- Look for features that are very good indicators of the result, place these features (as questions) in nodes of the tree
- Split the examples so that those with different values for the chosen feature are in a different set
- Repeat the same process with another feature

*__ID3__ = Iterative Dichotomiser 3

ID3* / C4.5 Algorithm

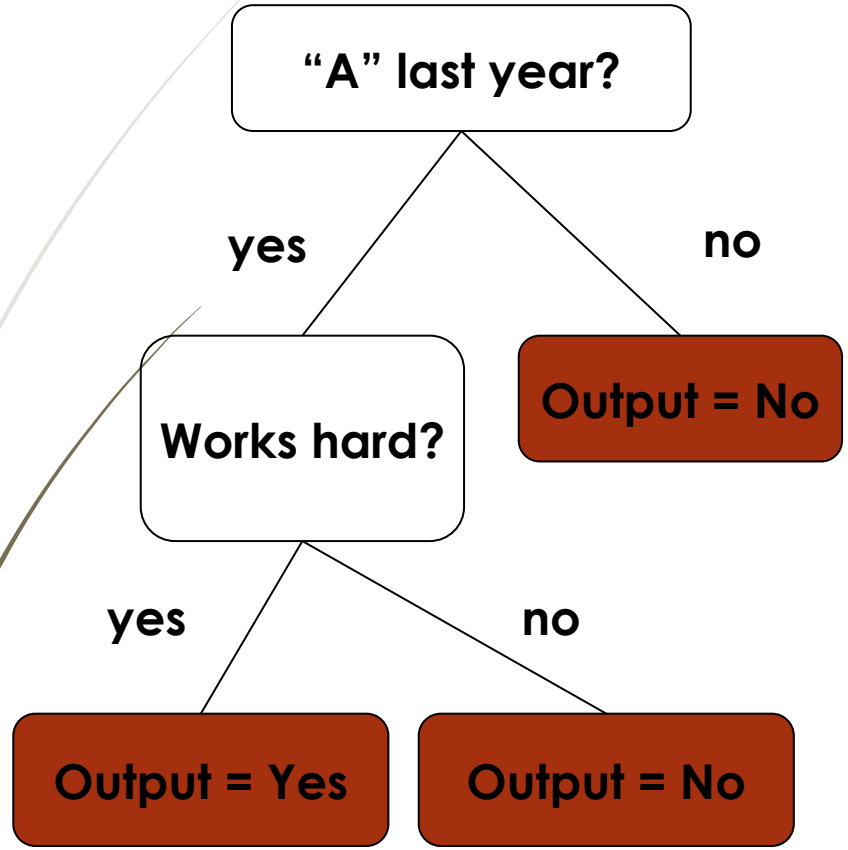
- Top-down construction of the decision tree
- Recursive selection of the "best feature" to use at the current node in the tree
  - Once the feature is selected for the current node, generate children nodes, one for each possible value of the selected attribute
  - Partition the examples using the possible values of this attribute, and assign these subsets of the examples to the appropriate child node
  - Repeat for each child node until all examples associated with a node are classified

Example 1

Information on last year's students to determine if a student will get an 'A' this year

	Features (X)				Output f(X)
Student	**'A' last year?**	**Black hair?**	**Works hard?**	**Drinks?**	**'A' this year?**
X1: Richard	Yes	Yes	No	Yes	**No**
X2: Alan	Yes	Yes	Yes	No	**Yes**
X3: Alison	No	No	Yes	No	**No**
X4: Jeff	No	Yes	No	Yes	**No**
X5: Gail	Yes	No	Yes	Yes	**Yes**
X6: Simon	No	Yes	Yes	Yes	**No**

Example 1



	Features				Output f(X)
Student	'A' last year?	Black hair?	Works hard?	Drinks?	**'A' this year?**
Richard	Yes	Yes	No	Yes	**No**
Alan	Yes	Yes	Yes	No	**Yes**
Alison	No	No	Yes	No	**No**
Jeff	No	Yes	No	Yes	**No**
Gail	Yes	No	Yes	Yes	**Yes**
Simon	No	Yes	Yes	Yes	**No**

Example 2 The Restaurant

- Goal: learn whether one should wait for a table
- Attributes
  - Alternate: another suitable restaurant nearby
  - Bar: comfortable bar for waiting
  - Fri/Sat: true on Fridays and Saturdays
  - Hungry: whether one is hungry
  - Patrons: how many people are present (none, some, full)
  - Price: price range ($, $$, $$$)
  - Raining: raining outside
  - Reservation: reservation made
  - Type: kind of restaurant (French, Italian, Thai, Burger)
  - WaitEstimate: estimated wait by host (0-10 mins, 10-30, 30-60, >60)

Example 2 The Restaurant

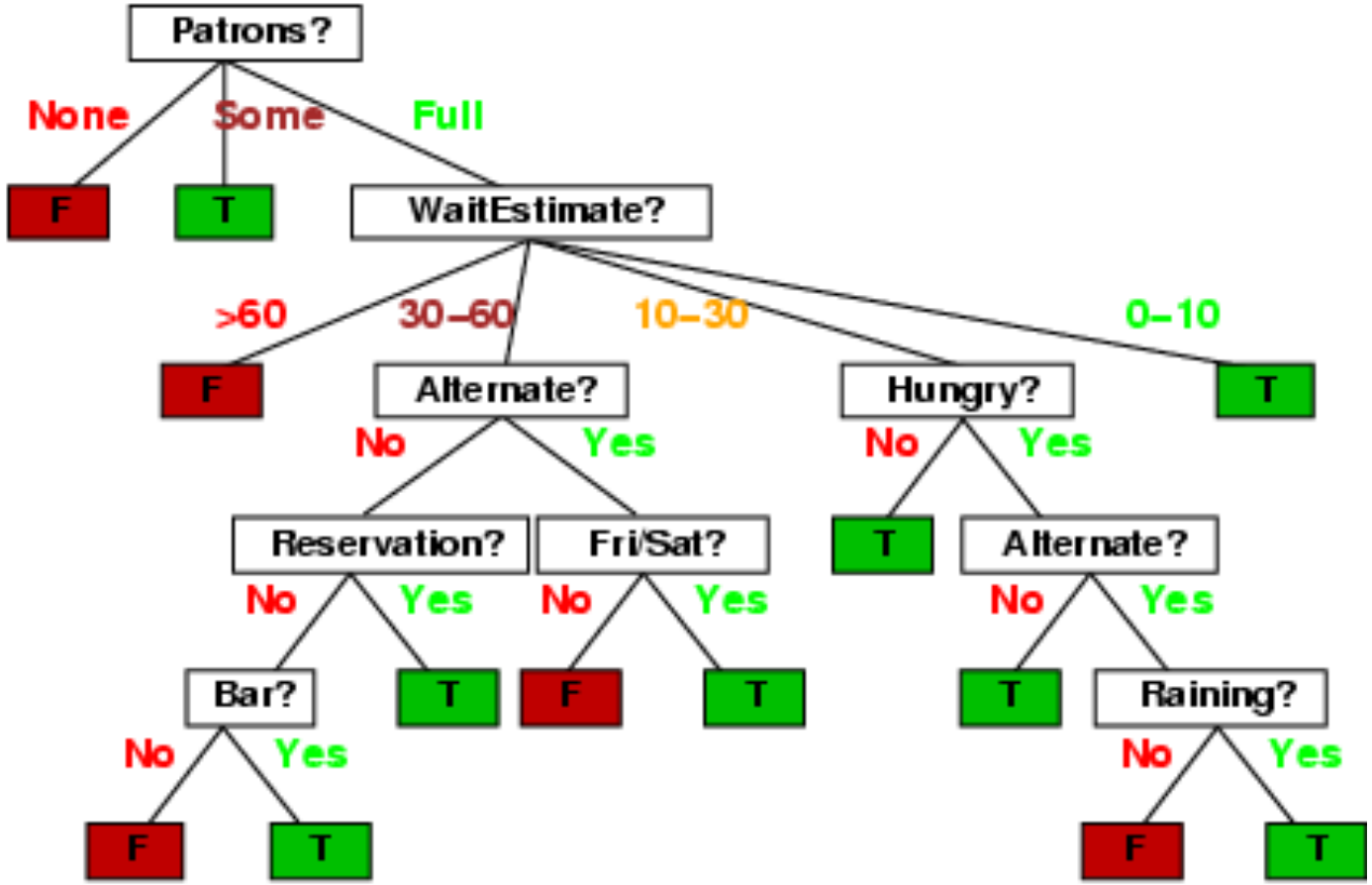
- Training Data

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

A First Decision Tree

Is it the best decision tree we can build?

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



Ockham's Razer Principle

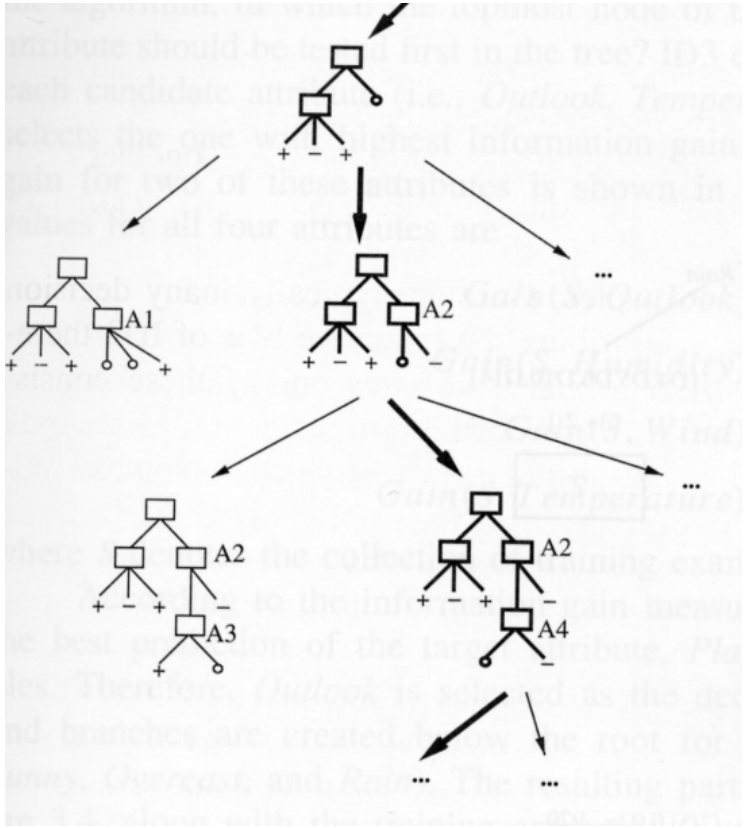
It is vain to do more than can be done with less… Entities should not be multiplied beyond necessity. [Ockham, 1324]

- In other words… always favor the simplest answer that correctly fits the training data
- i.e. the smallest tree on average
- This type of assumption is called **inductive bias**
  - inductive bias = making a choice beyond what the training instances contain

Finding the Best Tree

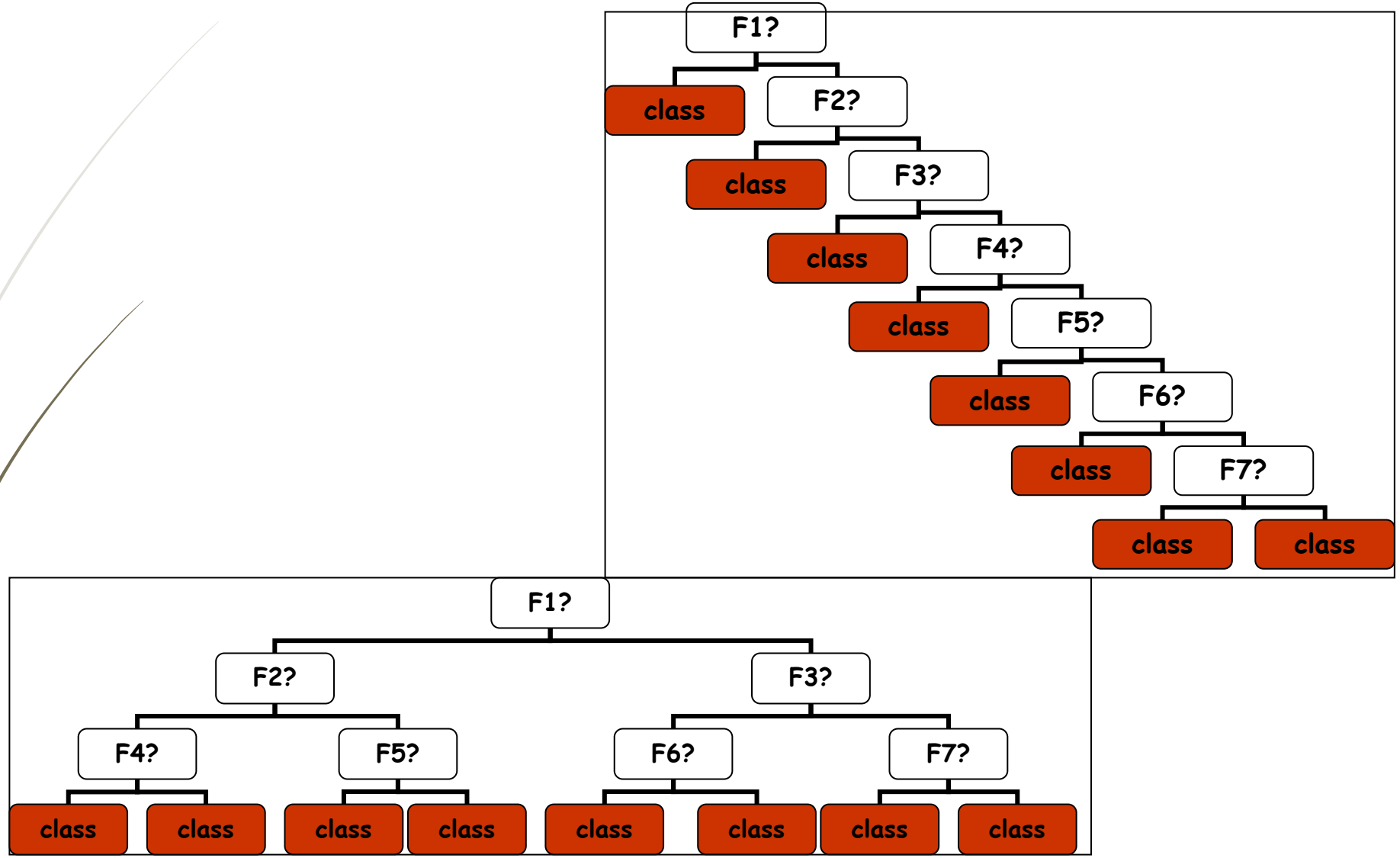
- can be seen as searching the space of all possible decision trees
- Inductive bias: prefer shorter trees on average
- how?
- search the space of all decision trees
  - always pick the next attribute to split the data based on its "discriminating power" (information gain)
  - in effect, steepest ascent hill-climbing search where heuristic is information gain

empty tree



complete tree

Which Tree is the Best ?

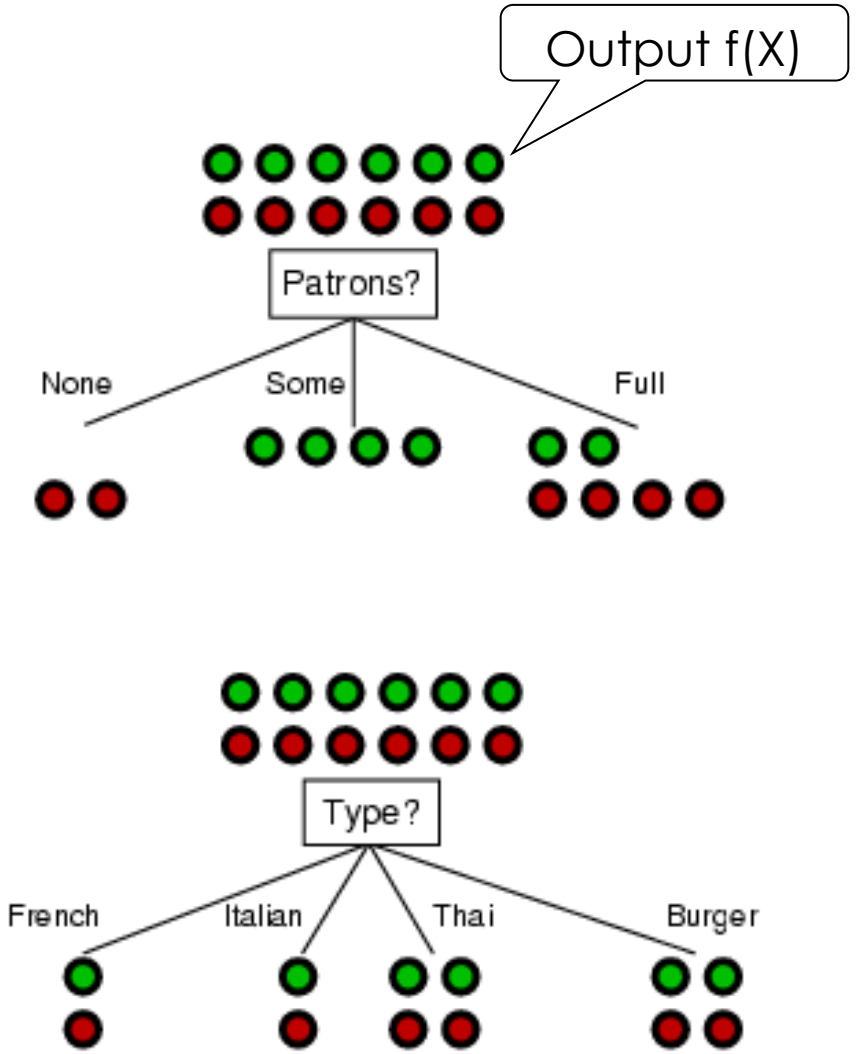


Choosing the Next Principle

- The key problem is choosing which feature to split a given set of examples
- ID3 uses **Maximum Information-Gain**:
  - Choose the attribute that has the largest information gain
    - i.e., the attribute that will result in the smallest expected size of the subtrees rooted at its children
    - information theory

Intuitively

- ***Patron*:**
  - If value is *Some*… all outputs=*Yes*
  - If value is *None*… all outputs=*No*
  - If value is *Full*… we need more tests
- ***Type*:**
  - If value is *French*… we need more tests
  - If value is *Italian*… we need more tests
  - If value is *Thai*… we need more tests
  - If value is *Burger*… we need more tests
- …
- So ***patron*** may lead to shorter tree…

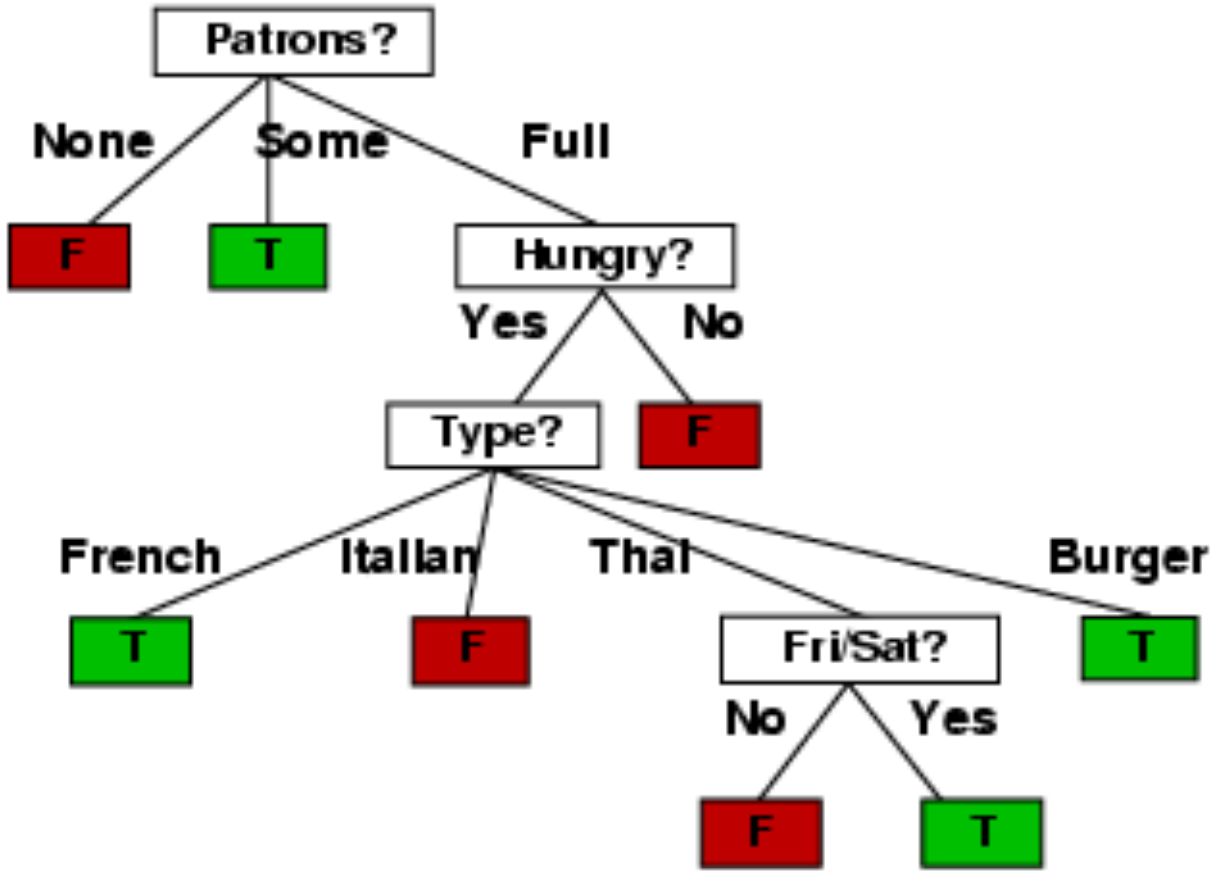


Next Feature

- For only data where *patron = Full*
- ***hungry***
  - If value is *Yes*… we need more tests
  - If value is *No*… all output= *No*
- ***type*:**
  - If value is *French*… all output= *No*
  - If value is *Italian*… all output= *No*
  - If value is *Thai*… we need more tests
  - If value is *Burger*… we need more tests
- …
- So ***hungry*** is more discriminating (only 1 new branch)…

Next Feature

- 4 tests instead of 9
- 11 branches instead of 21



Choosing the Next Attribute

- The key problem is choosing which feature to split a given set of examples
- Most used strategy: information theory

$$H(X) = -\sum_{x_i \in X} p(x_i) \log_2 p(x_i)$$ Entropy (or information content)

$$H(\text{fair coin toss}) = -\sum_{x_i \in X} p(x_i) \log_2 p(x_i) = H\left(\frac{1}{2}, \frac{1}{2}\right)$$

$$= \left(\frac{1}{2}\log_2 \frac{1}{2} + \frac{1}{2}\log_2 \frac{1}{2}\right) = 1 \text{ bit}$$

entropy of a fair coin toss with 2 possible outcomes, each with a probability of 1/2

Essential Information Theory

- Developed by Shannon in the 40s
- Notion of entropy (information content):
  - How informative is a piece of information?
    - If you already have a good idea about the answer
    - low entropy
    - then a "hint" about the right answer is not very informative…
    - If you have no idea about the answer (ex. 50-50 split)
    - high entropy
    - then a "hint" about the right answer is very informative…

Entropy

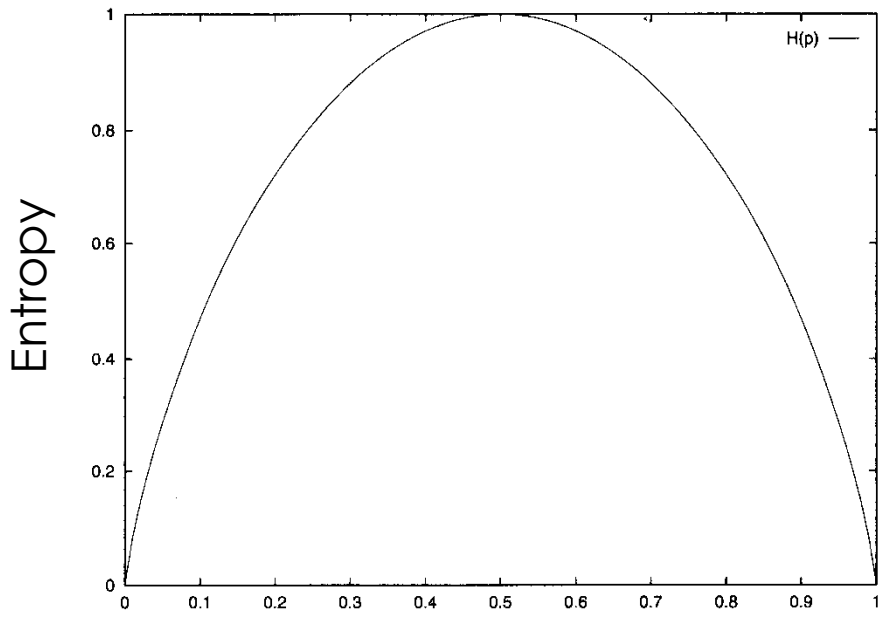
- Let X be a discrete random variable (RV) with *i* possible outcomes x_i
- Entropy (or information content)

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

- measures the *amount of information* in a RV
  - *average uncertainty* of a RV
  - *the average length of the message* needed to transmit an outcome x_i of that variable
- measured in bits
- for only 2 outcomes x_1 and x_2, then $1 \geq H(X) \geq 0$

Example: The Coin Flip

- Fair coin: $H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1\,\text{bit}$
- Rigged coin: $H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i) = -\left(\frac{99}{100}\log_2\frac{99}{100} + \frac{1}{100}\log_2\frac{1}{100}\right) = 0.08\,\text{bits}$



fair coin -> high entropy

rigged coin -> low entropy

Choosing the Best Feature

- The "discriminating power" of an attribute A given a data set S
- Let Values(A) = the set of values that attribute A can take
- Let S_v = the set of examples in the data set which have value v for attribute A (for each value v from Values(A))

information gain (or entropy reduction)

$$gain(S, A) = H(S) - H(S \mid A)$$

$$= H(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} x\, H(S_v)$$

Some Intuition

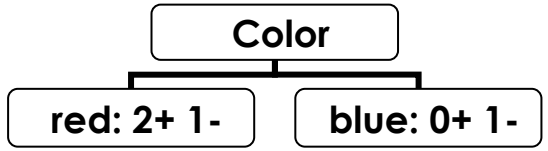
Size	Color	Shape	Output
Big	Red	Circle	+
Small	Red	Circle	+
Small	Red	Square	-
Big	Blue	Circle	-

- *Size* is the least discriminating attribute (i.e. smallest information gain)
- *Shape* and *color* are the most discriminating attributes (i.e. highest information gain)

A Small Example 1

Size	Color	Shape	Output
Big	Red	Circle	+
Small	Red	Circle	+
Small	Red	Square	-
Big	Blue	Circle	-

Values(Color) = {red,blue}



$$gain(S, Color) = H(S) - \sum_{v \in values(Color)} \frac{|S_v|}{|S|} x\, H(S_v)$$

$$H(S) = -\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}\right) = 1$$

for each v of Values(Color)

$$H(S \mid Color = red) = H\left(\frac{2}{3},\frac{1}{3}\right) = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right) = 0.918$$

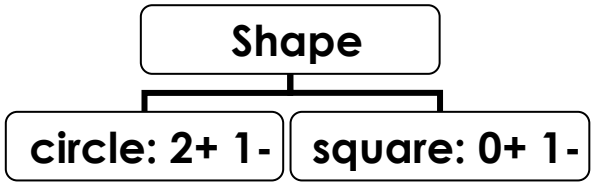
$$H(S \mid Color = blue) = H(1,0) = -\left(\frac{1}{1}\log_2\frac{1}{1}\right) = 0$$

$$H(S \mid Color) = \frac{3}{4}(0.918) + \frac{1}{4}(0) = 0.6885$$

$$gain(Color) = H(S) - H(S \mid Color) = 1 - 0.6885 = 0.3115$$

A Small Example 2

Size	Color	Shape	Output
Big	Red	Circle	+
Small	Red	Circle	+
Small	Red	Square	-
Big	Blue	Circle	-



$$H(S) = -\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}\right) = 1$$

Note: by definition,

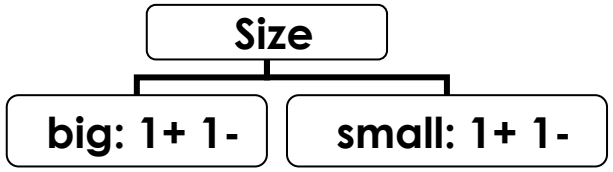
- **Log 0 = -∞**
- **0log0 is 0**

$$H(S|Shape) = \frac{3}{4}(0.918) + \frac{1}{4}(0) = 0.6885$$

$$gain(Shape) = H(S) - H(S|Shape) = 1 - 0.6885 = 0.3115$$

A Small Example 3

Size	Color	Shape	Output
Big	Red	Circle	+
Small	Red	Circle	+
Small	Red	Square	-
Big	Blue	Circle	-



$$H(S) = -\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}\right) = 1$$

$$H(S|Size) = \frac{1}{2}(1) + \frac{1}{2}(1) = 1$$

$$gain(Size) = H(S) - H(S|Size) = 1 - 1 = 0$$

A Small Example 4

Size	Color	Shape	Output
Big	Red	Circle	+
Small	Red	Circle	+
Small	Red	Square	-
Big	Blue	Circle	-

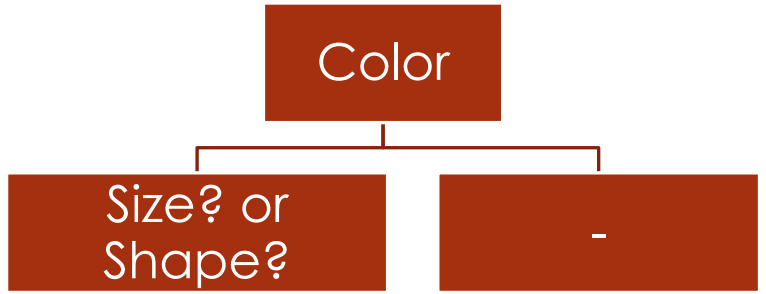
$$gain(Shape) = 0.3115$$
$$gain(Color) = 0.3115$$
$$gain(Size) = 0$$

- So first separate according to either *color* or *shape* (root of the tree)

A Small Example 4

Size	Color	Shape	Output
Big	Red	Circle	+
Small	Red	Circle	+
Small	Red	Square	-
Big	Blue	Circle	-

S_2



$$H(S_2) = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right)$$

for each v of Values(Size)

$$H(S_2 \mid Size = big) = H\left(\frac{1}{1}, \frac{0}{1}\right) = 0$$

$$H(S_2 \mid Size = small) = H\left(\frac{1}{2}, \frac{1}{2}\right) = 1$$

$$H(S_2 \mid Size) = \frac{1}{3}(0) + \frac{2}{3}(1)$$

$$gain(Size) = H(S_2) - H(S_2 \mid Color)$$

for each v of Values(Shape)

$$H(S_2 \mid Shape = circle) = H\left(\frac{2}{2}, \frac{0}{2}\right) = 0$$

$$H(S_2 \mid Shape = square) = H\left(\frac{0}{1}, \frac{1}{1}\right) = 0$$

$$H(S_2 \mid Shape)$$

$$gain(Shape) = H(S_2) - H(S_2 \mid Shape)$$

Back to the Restaurant

- **Training data:**

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

The Restaurant Example

$gain(alt) = \ldots$ $gain(bar) = \ldots$ $gain(fri) = \ldots$ $gain(hun) = \ldots$

$$gain(pat) = 1 - \left(\frac{2}{12} \times H\left(\frac{0}{2}, \frac{2}{2}\right) + \frac{4}{12} \times H\left(\frac{0}{4}, \frac{4}{4}\right) + \frac{6}{12} \times H\left(\frac{2}{6}, \frac{4}{6}\right) \right)$$

$$= 1 - \left(\frac{2}{12} \times - \left(\frac{0}{2} \log_2 \frac{0}{2} + \frac{2}{2} \log_2 \frac{2}{2} \right) + \frac{4}{12} \times - \left(\frac{0}{4} \log_2 \frac{0}{4} + \frac{4}{4} \log_2 \frac{4}{4} \right) + \ldots \right) \approx 0.541 \text{bits}$$

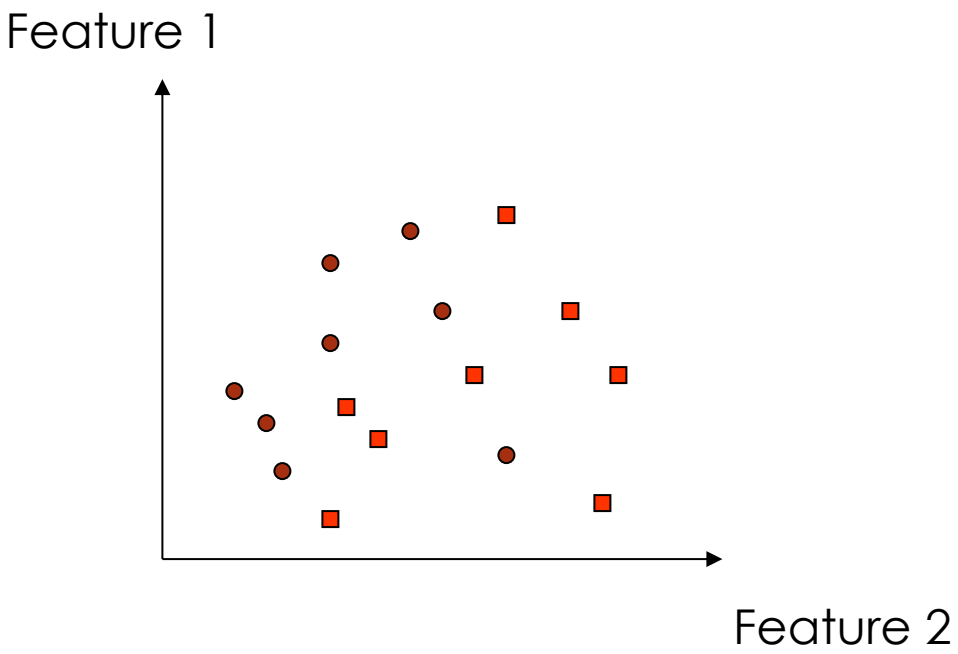
$gain(price) = \ldots$ $gain(rain) = \ldots$ $gain(res) = \ldots$

$$gain(type) = 1 - \left(\frac{2}{12} \times H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} \times H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} \times H\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} \times H\left(\frac{2}{4}, \frac{2}{4}\right) \right) = 0 \text{ bits}$$

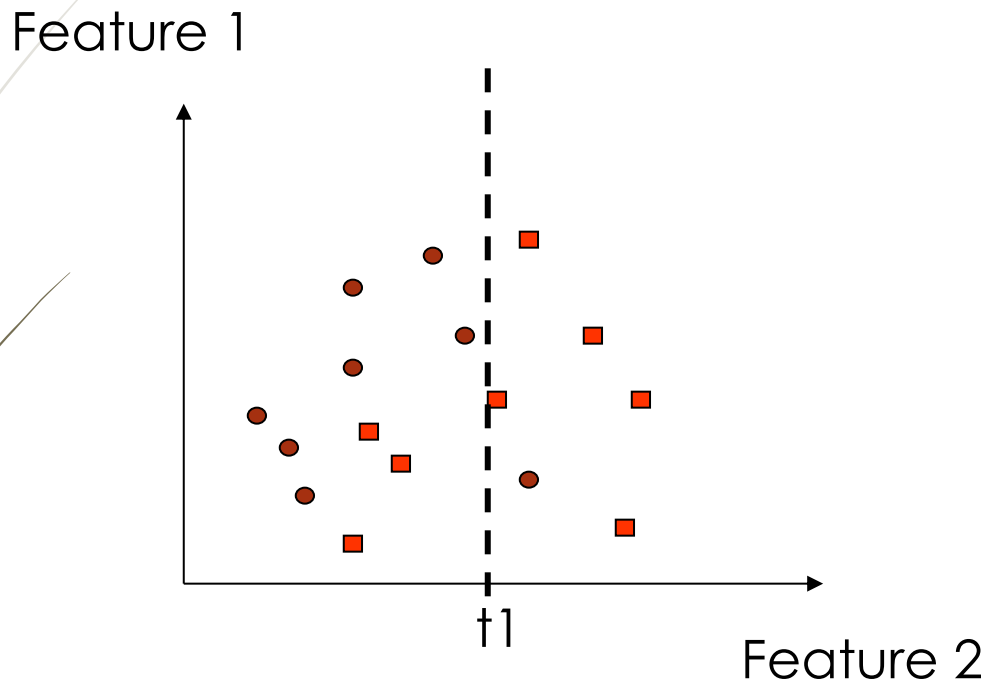
$gain(est) = \ldots$

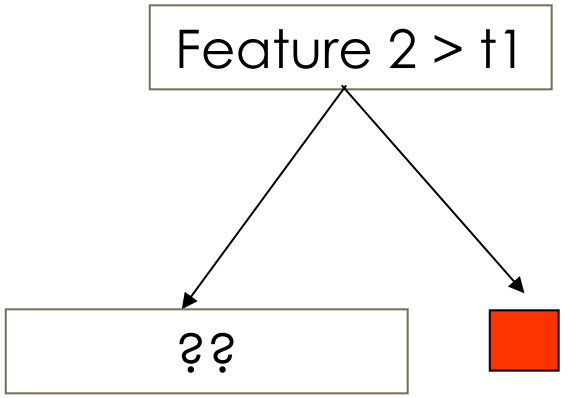
- Attribute pat (Patron) has the highest gain, so root of the tree should be attribute *Patrons*
- do recursively for subtrees

Decision Boundaries

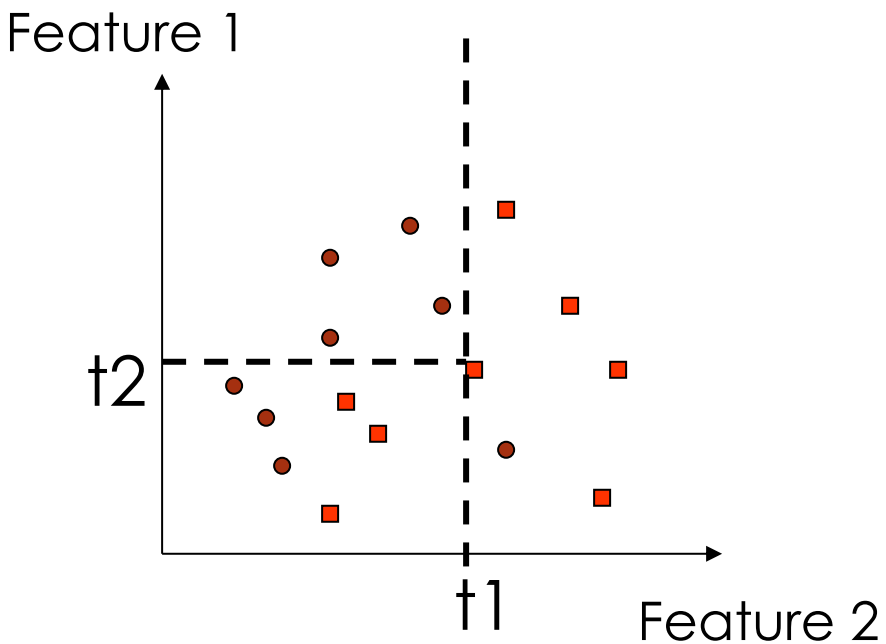


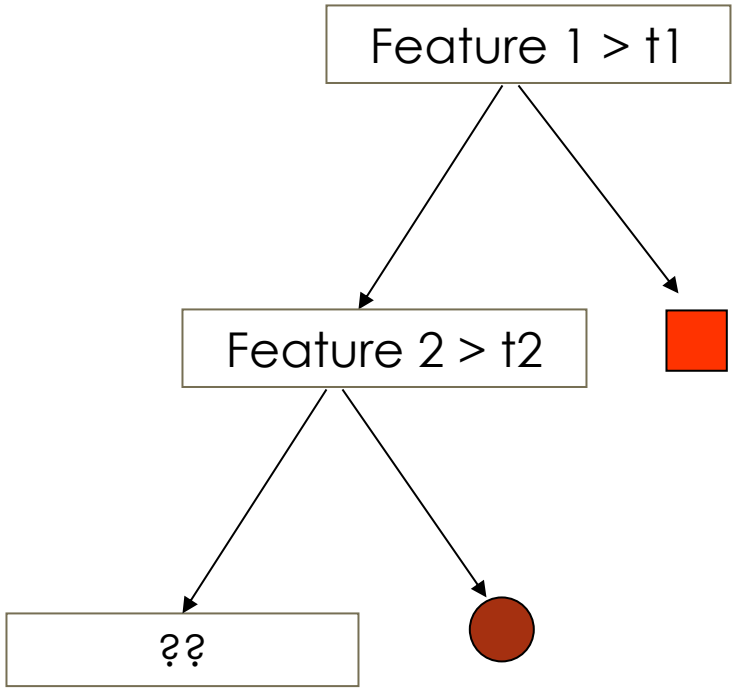
Decision Boundaries



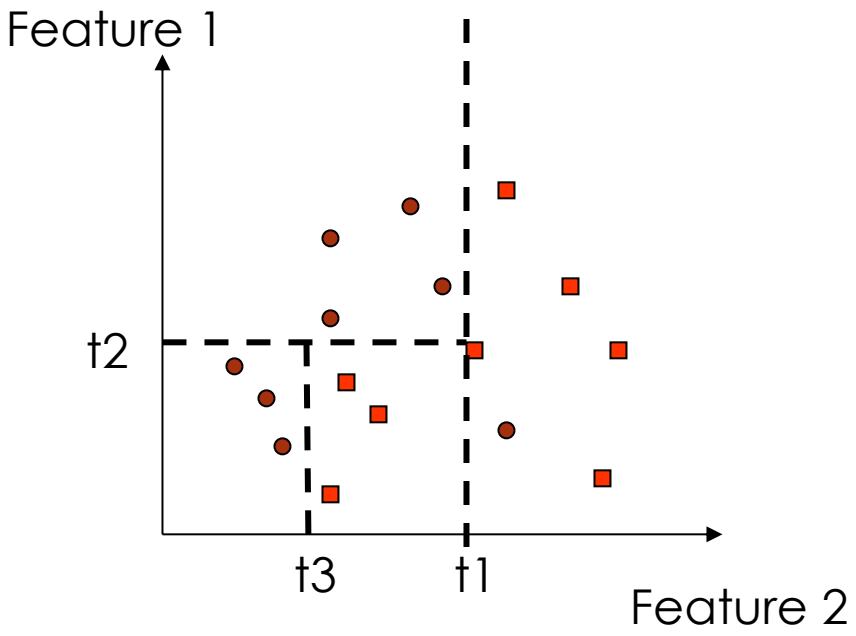


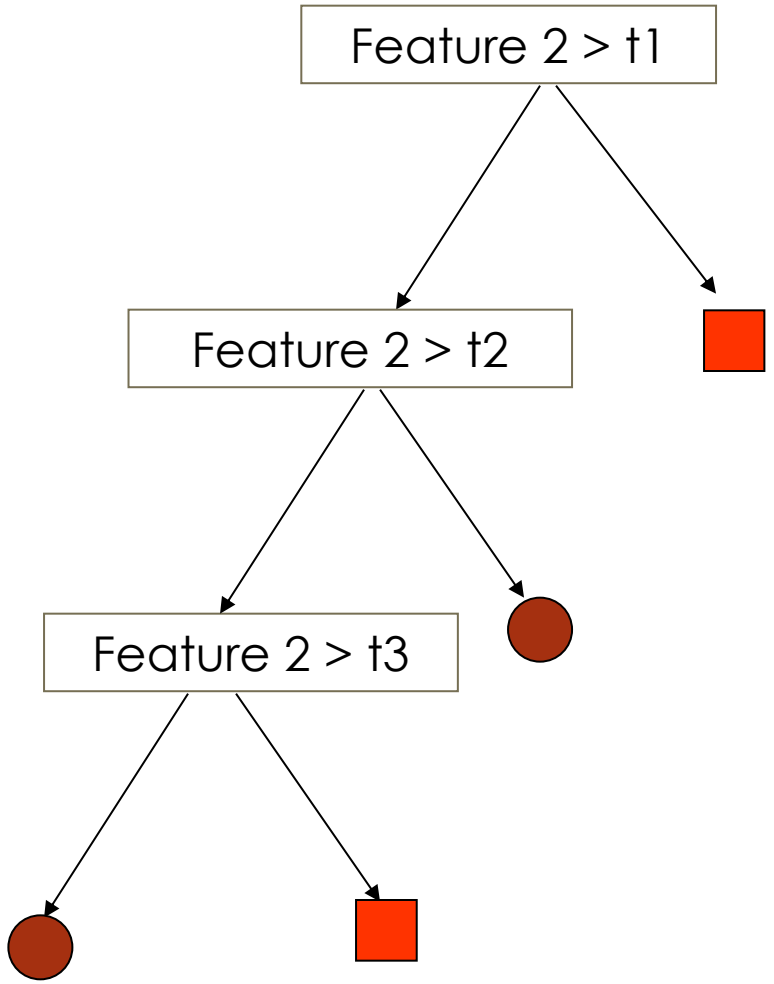
Decision Boundaries





Decision Boundaries





Applications

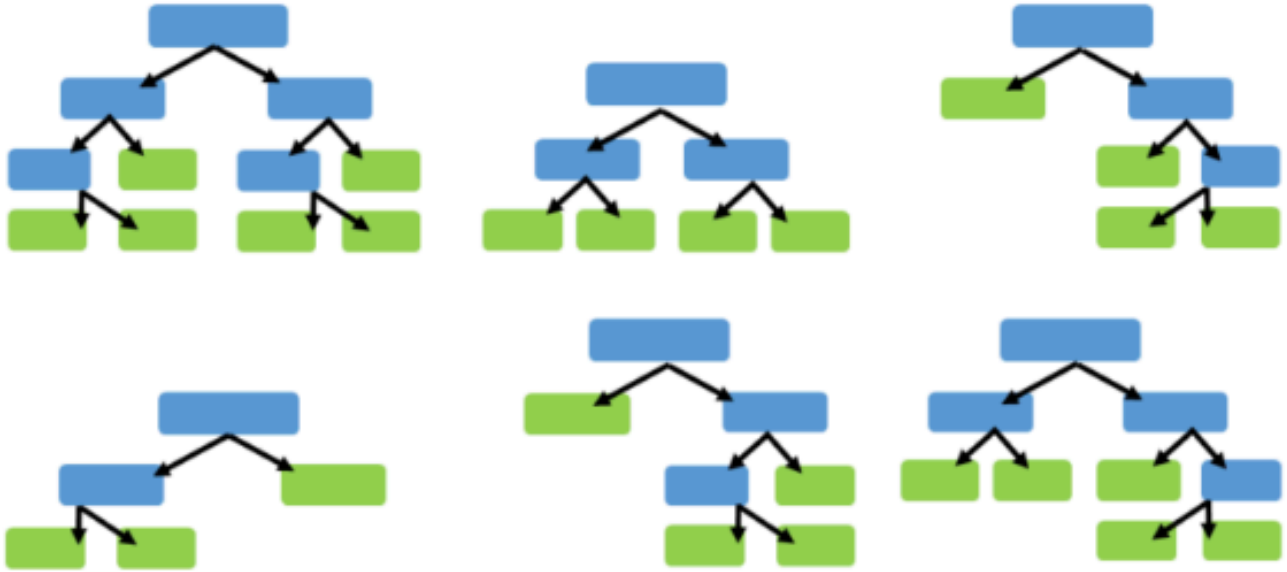
- One of the most widely used learning methods in practice
  - Fast, simple, and traceable
- Can out-perform human experts in many problems

Supervised Learning Algorithms

- ***Linear Regression***
- ***Logistic Regression***
- ***Decision Tree***
- Random Forest
- Naïve Bayes Classifier

Random Forest

- **Random forest** builds multiple decision trees (called the forest) and glues them together to get a more accurate and stable prediction.



Why Random Forest

- **More Accuracy**
- **Avoid Overfitting**
- **Bagging** to reduce the variations and the predictions by combining the result of multiple decision trees on different samples of the data set.
- Example: use the following data set to create a Random Forest that predicts of a person has heart disease or not.

Creating a Random Forest

- Example: use the following data set to create a Random Forest that predicts of a person has heart disease or not.

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Normal	Yes	Yes	195	Yes
Abnormal	No	No	130	No
Normal	No	Yes	218	No
Abnormal	Yes	Yes	180	Yes

Creating a Random Forest

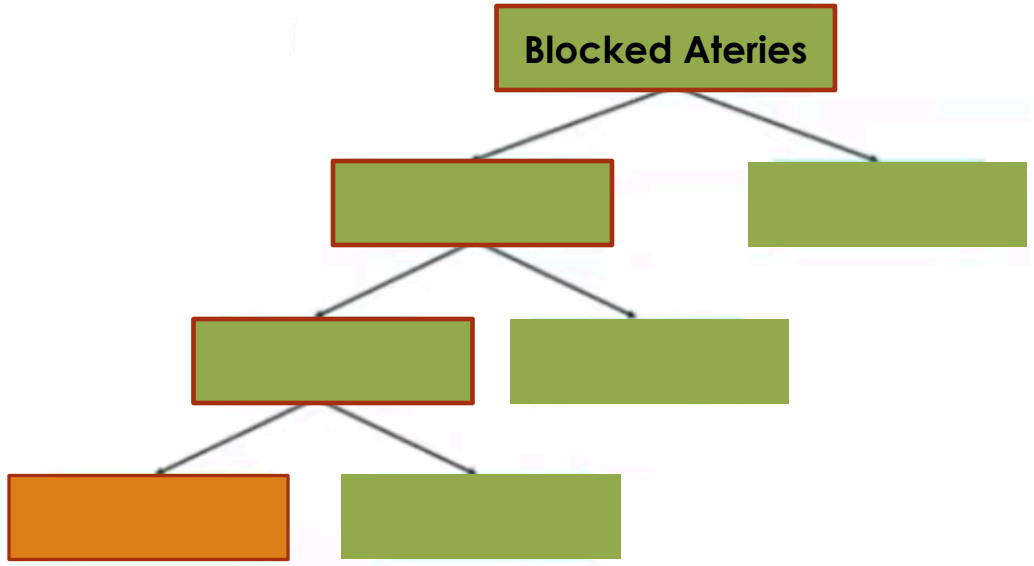
- Step 1: Create a Bootstrapped Data Set

Bootstrapping is an estimation methods used to make predictions on a data set by re-sampling it.

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Normal	Yes	Yes	195	Yes
Abnormal	No	No	130	No
Abnormal	Yes	Yes	180	Yes
Abnormal	Yes	Yes	180	Yes

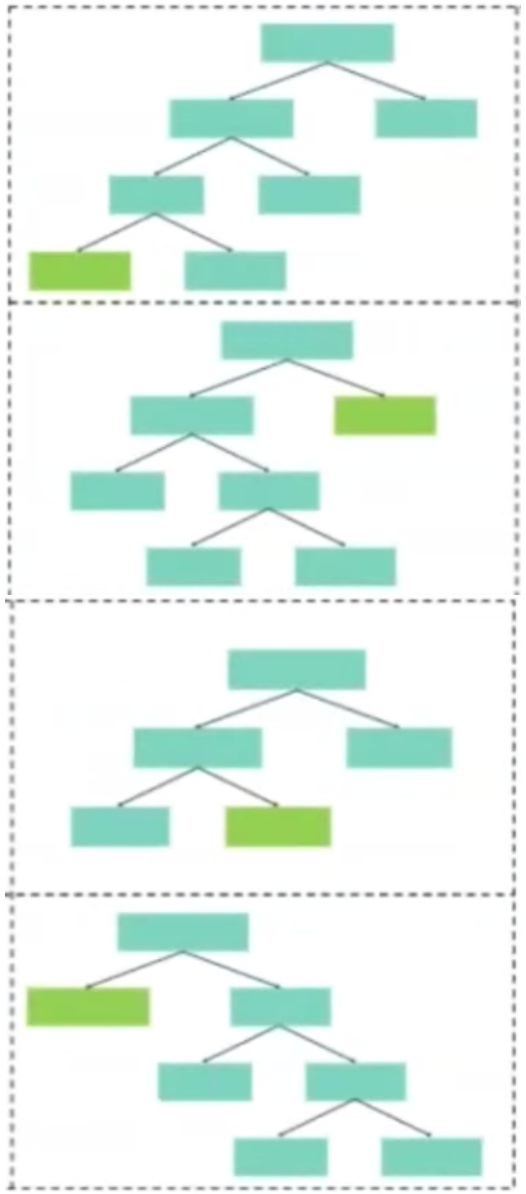
Creating a Random Forest

- Step 2: Creating Decision Tree
- Build a Decision Tree by using the bootstrapped data set
- Begin at the root nod & choose the best attribute to split the data set
- Repeat the same process for each of the upcoming branch nodes.



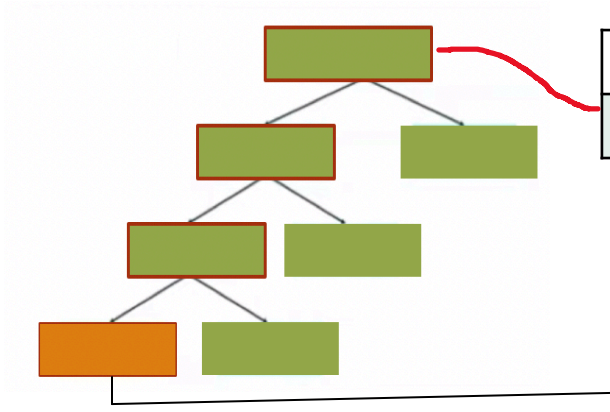
Creating a Random Forest

- Step 3: Go back to Step1 and Repeat
- Each Decision Tree predicts the output class based on the respective predictor variables used in the tree.
- Go back to step1, create a new bootstrapped data set and then build a Decision Tree by considering only a subset of variables at each step.
- This iteration is performed 100's of times, creating multiple decision trees.



Creating a Random Forest

- Step 4: Predicting the outcome of a new data point
- To predict whether a new patient has heart disease, run the new data down the decision trees.
- After running the data down all the trees in the Random Forest, we check which class got the majority votes.



A new patient's record

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Abnormal	No	Yes	185	

Heart Disease	
Yes	No
1	0

Output

Creating a Random Forest

- Step 5: Evaluate the Model
- In a real-world problem, about 1/3rd of the original data set is not included in the bootstrapped data set.
- This sample data set that does not include in the bootstrapped data set is known as the Out-Of-Bag(OOB) data set.
- We can measure the accuracy of a Random Forest by the proportion of OOB samples that are correctly classified.

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Normal	Yes	Yes	195	Yes
Abnormal	No	No	130	No
Abnormal	Yes	Yes	180	Yes
Abnormal	Yes	Yes	180	Yes

Bootstrapped data set

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Normal	No	Yes	218	No

OOB data set (testing data set)

Supervised Learning Algorithms

- ***Linear Regression***
- ***Logistic Regression***
- ***Decision Tree***
- ***Random Forest***
- Naïve Bayes Classifier

The End