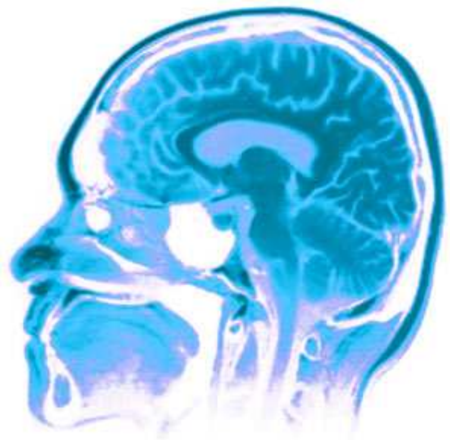
DS4400



Decision trees

* This presentation is highly inspired by Nando de Freitas lecture notes the blog post by Luis Serrano on Udacity.

.

Outline of the lecture

This lecture provides an introduction to decision trees. It discusses:

- ❑ Decision trees
- ❑ Using reduction in entropy as a criterion for constructing decision trees.
- ❑ The application of decision trees to classification

- Trees can be used for **regression**, **classification**, **clustering** and **density estimation**

Outline of the lecture

This lecture provides an introduction to decision trees. It discusses:

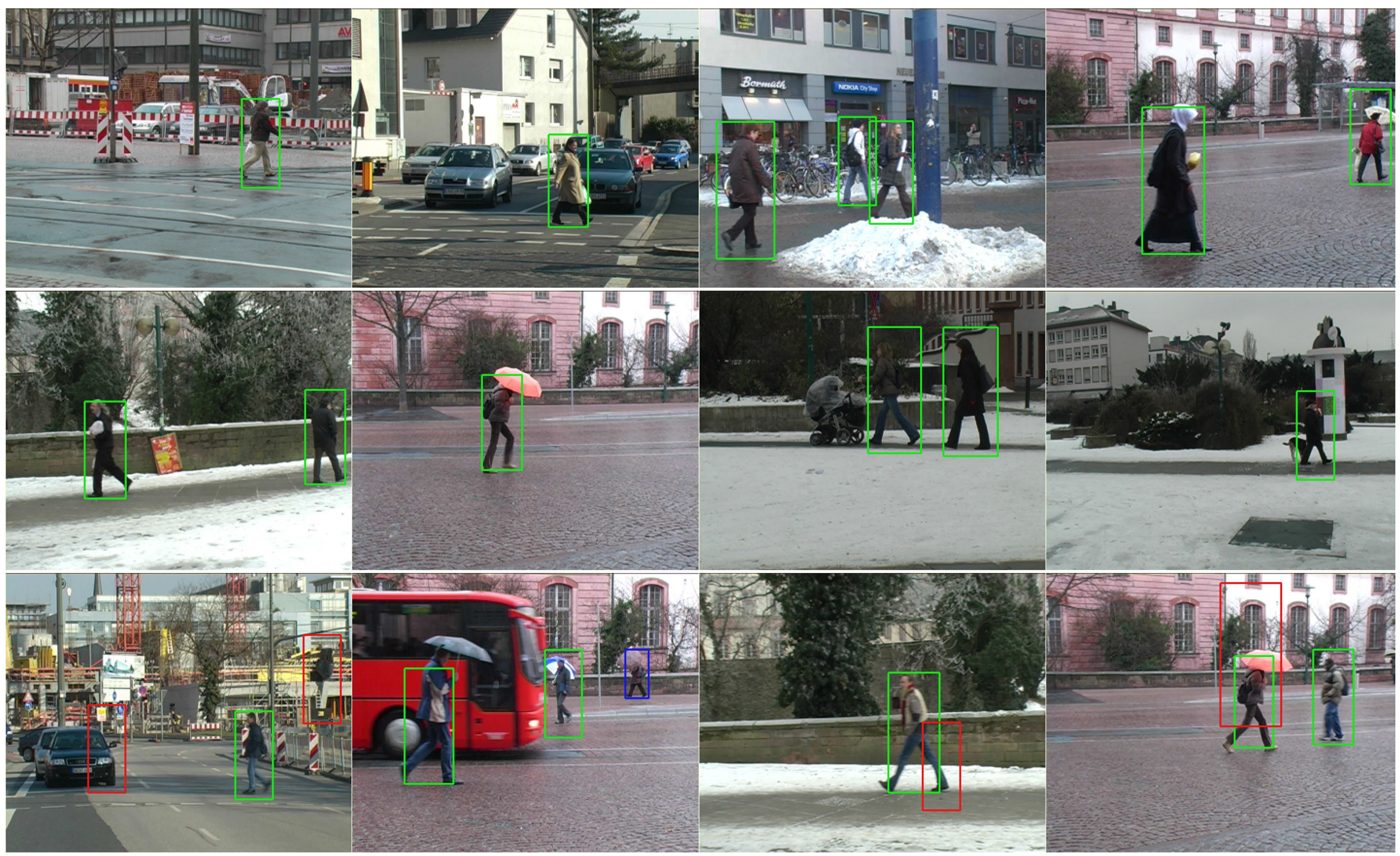
- ❑ Decision trees
- ❑ Using reduction in entropy as a criterion for constructing decision trees.
- ❑ The application of decision trees to classification

- Trees can be used for **regression**, **classification**, **clustering** and **density estimation**

Applications:

- Face detection, tagging, Kinect
- Text classification, email spam detection

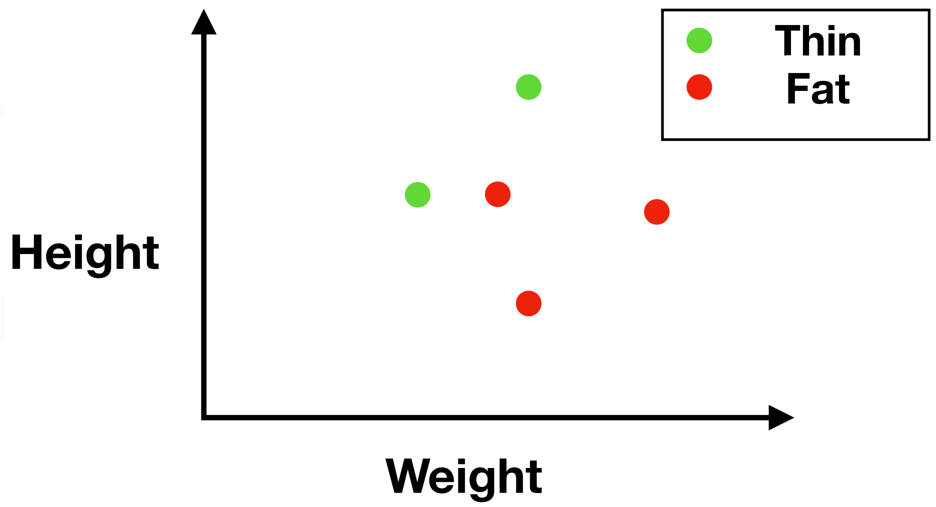
Motivation example 1: object detection



Motivation example 2: Kinect

- A sensor projects infrared grid on the subject
- Get a depth image
- Detects which point on the image is a hand or a shoulder or etc, with a **Random Forest**

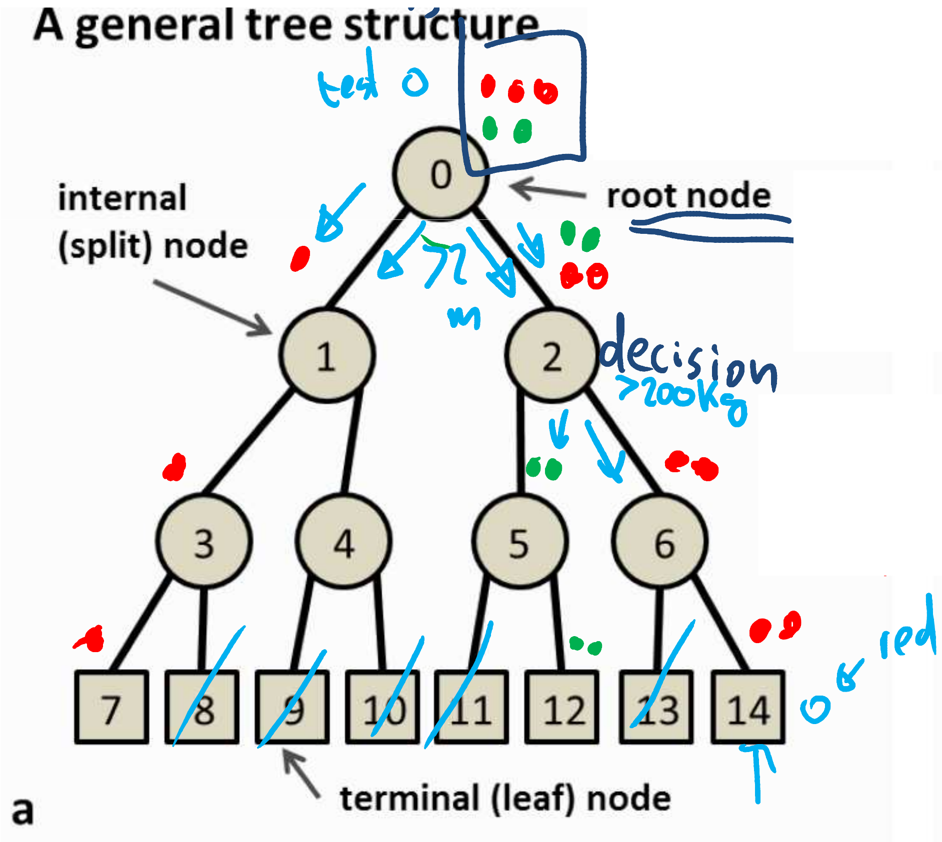
Image classification example

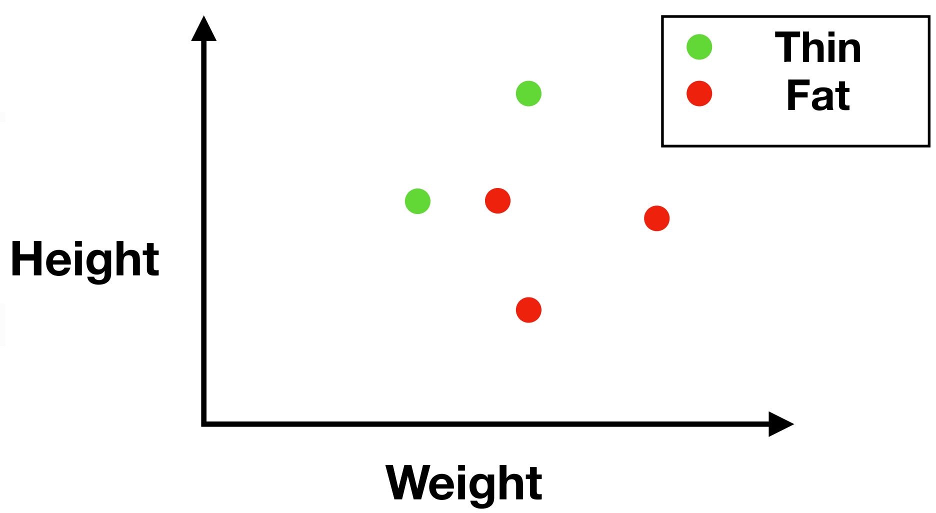


[MSR Tutorial on decision forests by Criminisi et al, 2011]

Image classification example

A **decision tree** is a structure that
Split the data into bins

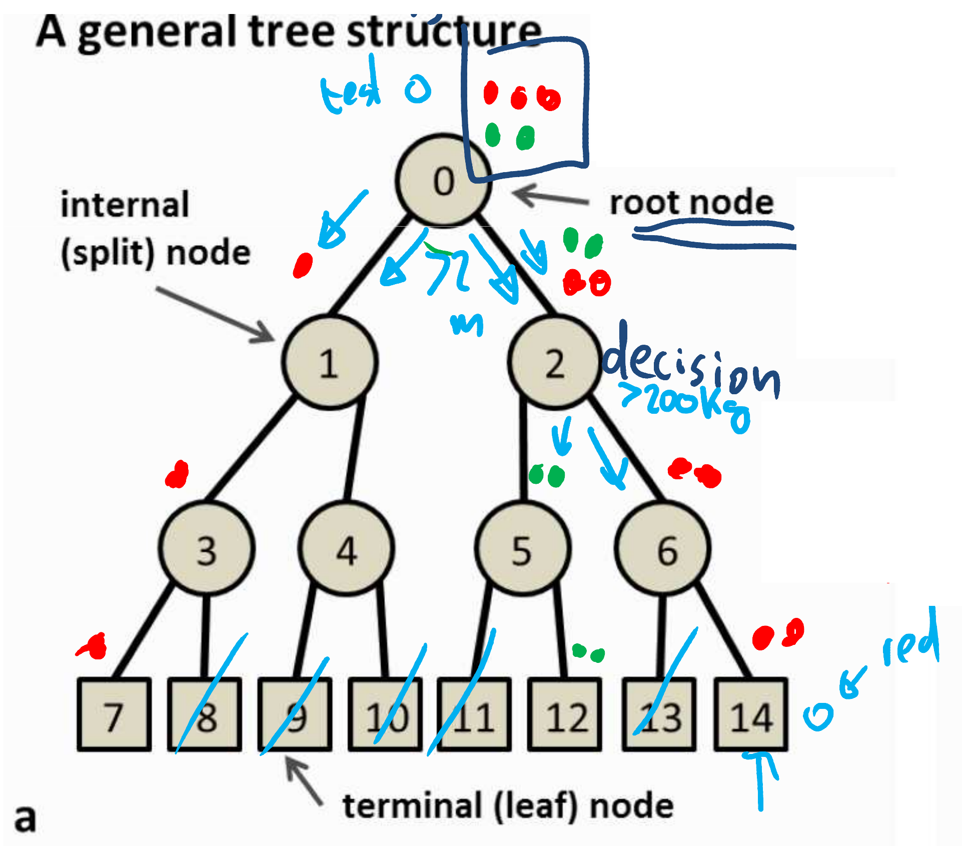




[MSR Tutorial on decision forests by Criminisi et al, 2011]

Image classification example

A **decision tree** is a structure that
Split the data into bins

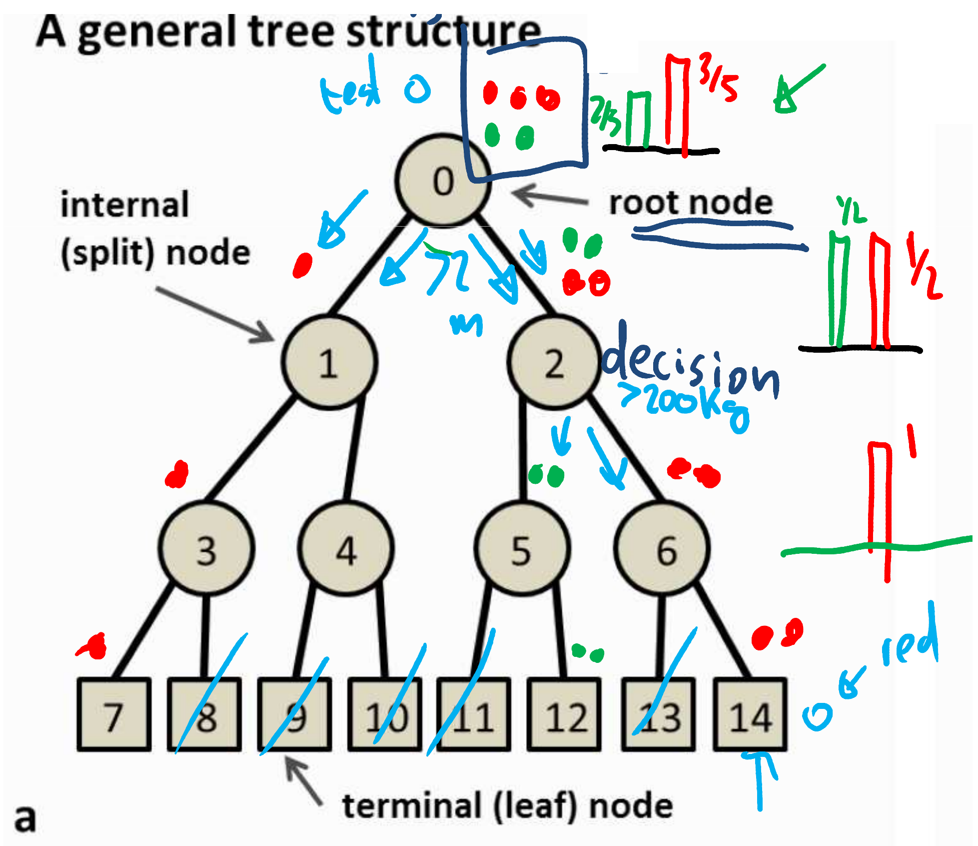


- Start from a **root node**
- Apply a **decision**
- **Split** the data

[MSR Tutorial on decision forests by Criminisi et al, 2011]

Image classification example

A **decision tree** is a structure that
Split the data into bins

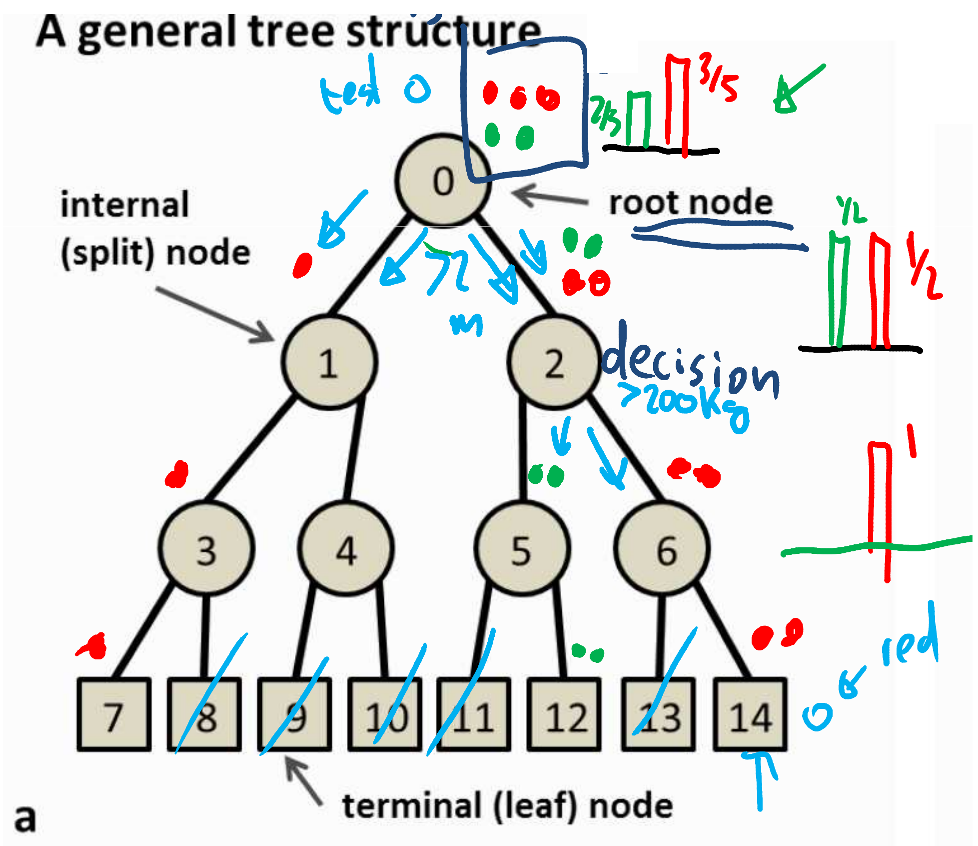


- Start from a **root node**
- Apply a **decision**
- **Split** the data

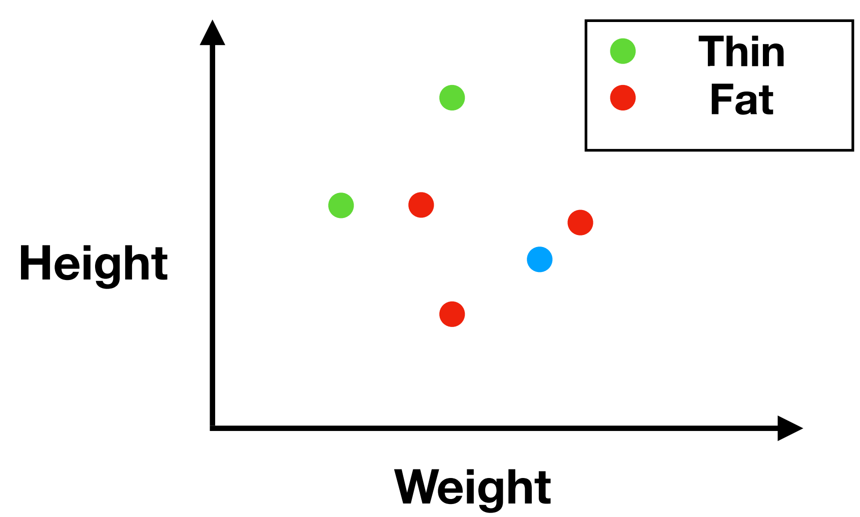
- A **histogram** at each node denotes **Probability** of each class

[MSR Tutorial on decision forests by Criminisi et al, 2011]

Image classification example

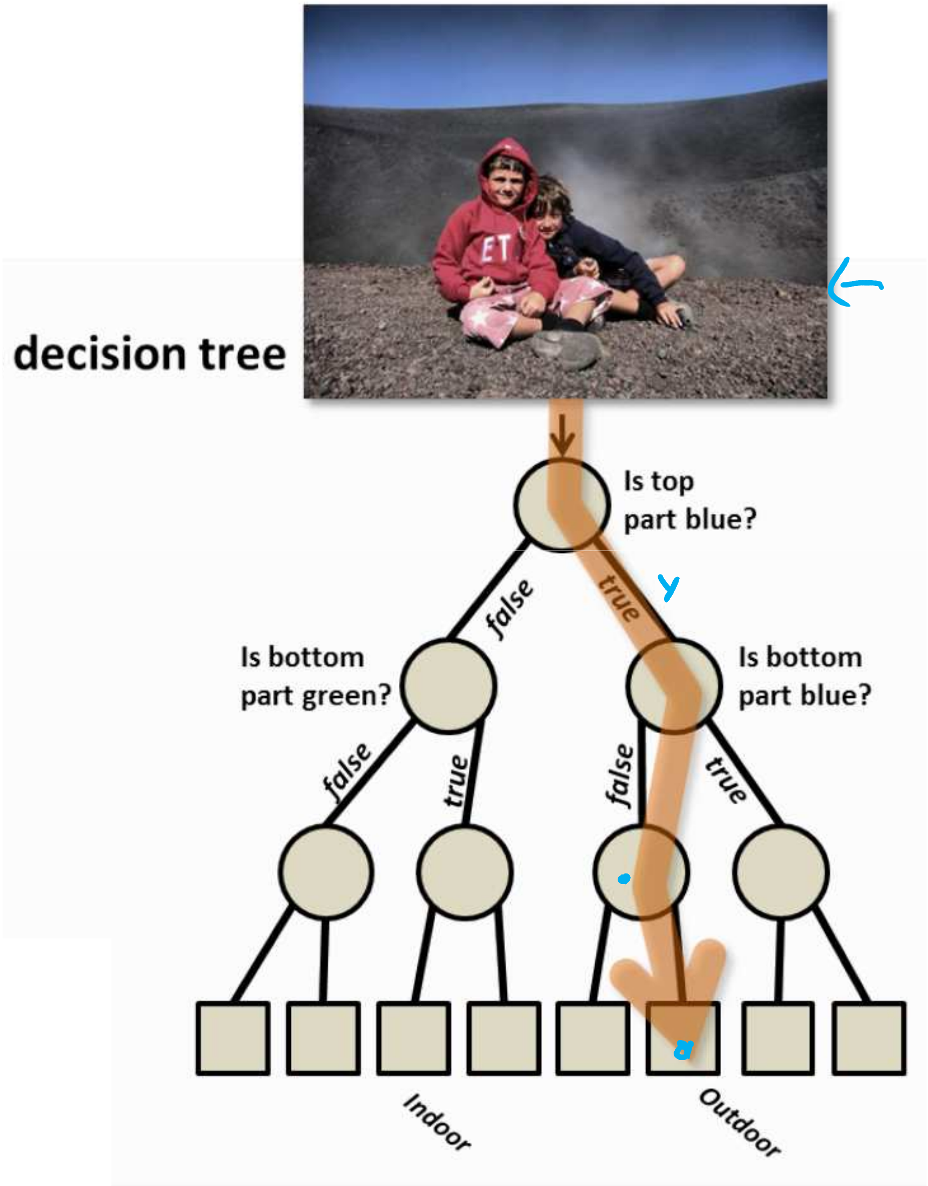


Goal is to **build the rules/questions** in the tree.
Once we have the tree, passing a new point is straight-forward.



[MSR Tutorial on decision forests by Criminisi et al, 2011]

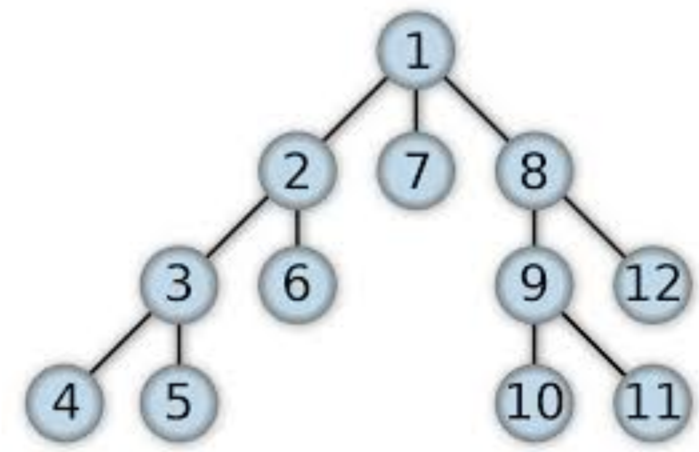
Image classification example



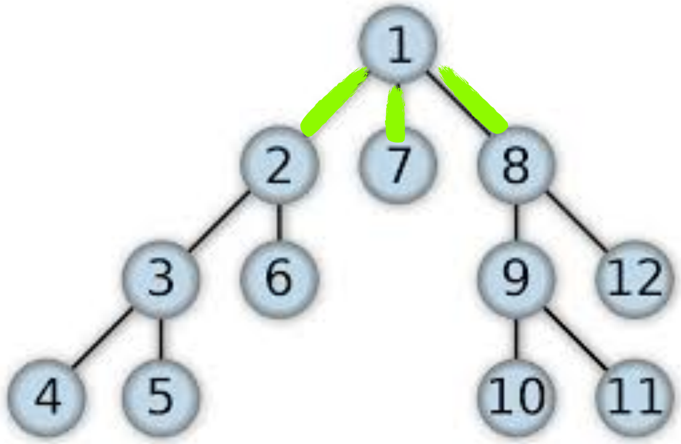
[MSR Tutorial on decision forests by Criminisi et al, 2011]

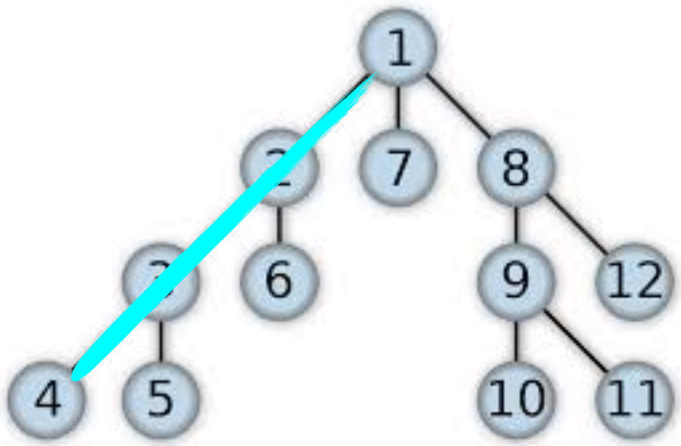
How to build the tree?

- We can learn the trees in a **greedy** fashion



- We can learn the trees is a **greedy** fashion
- Two ways:
    - Breadth First
    - Depth First





<u>First question</u>: How to build the ***root***?

- Assume a **marketing** agency wants to know if a customer **will wait** or **not** in a line.
- It is helpful to ***design strategies*** to increase the revenue of a restaurant.
- The following data is collected:

Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won't wait for a table:

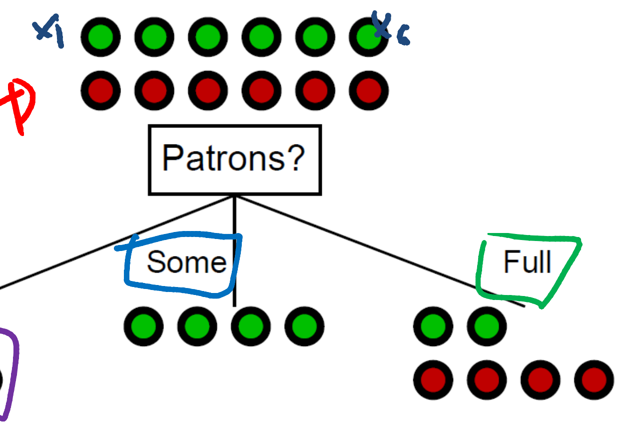
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	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

Classification of examples is positive (T) or negative (F)

[AI book of Stuart Russell and Peter Norvig]

Data Matrix

- We can use **any feature** to create a **Node** in the tree.



Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won't wait for a table:

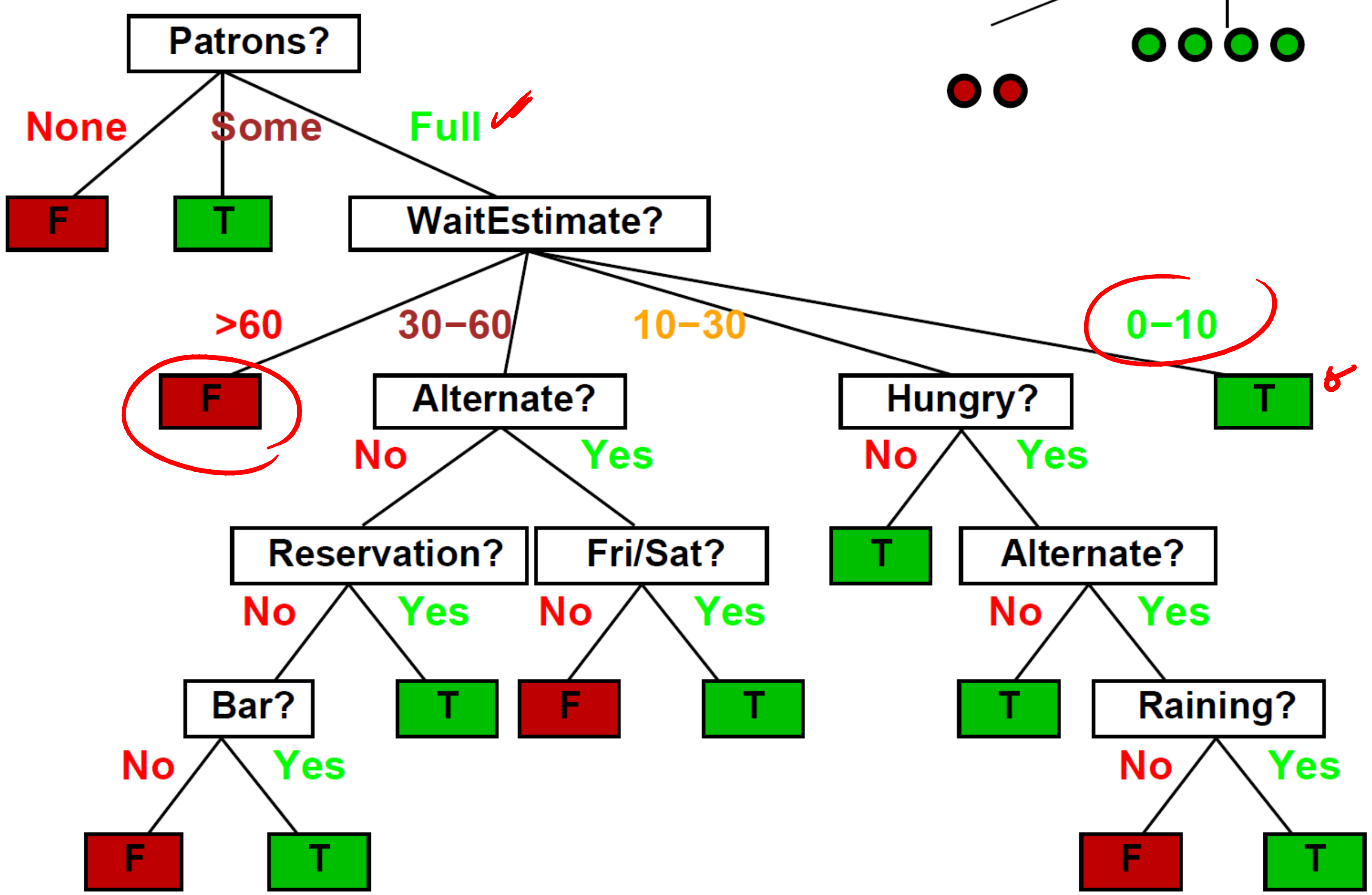
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	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

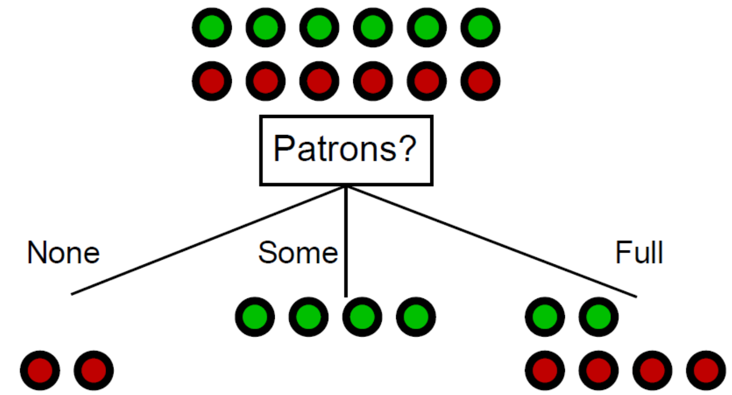
Classification of examples is positive (T) or negative (F)

[AI book of Stuart Russell and Peter Norvig]

Data Matrix

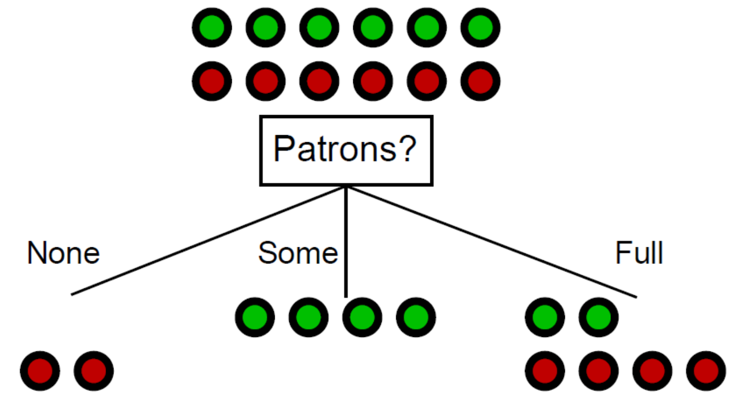
A learned decision tree

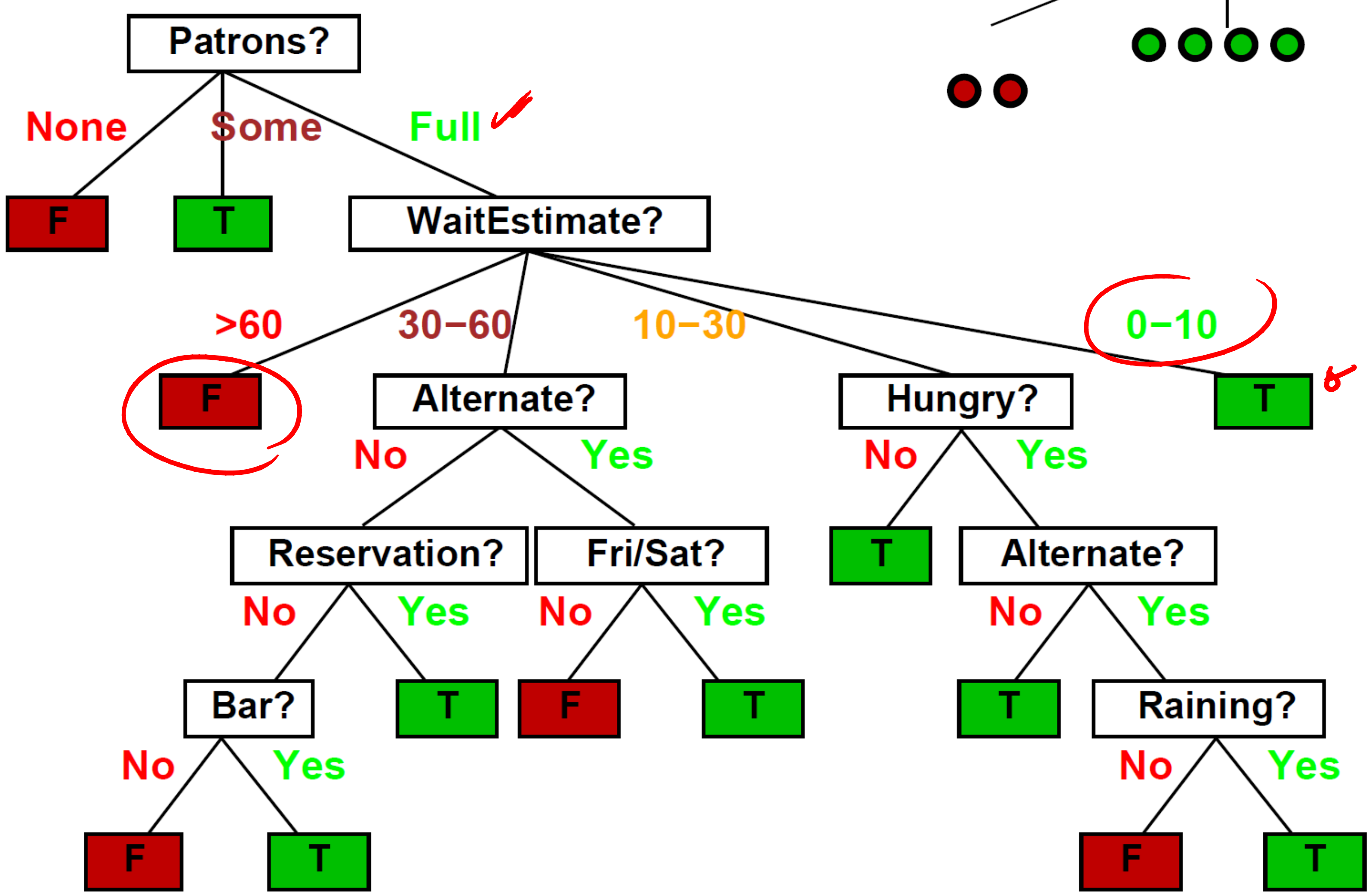




[AI book of Stuart Russell and Peter Norvig]

A learned decision tree



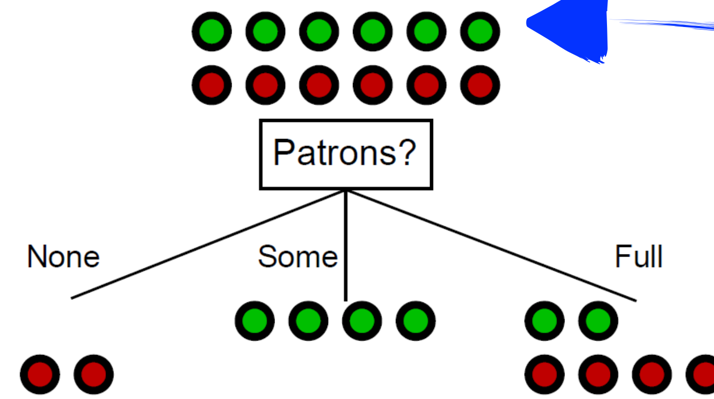


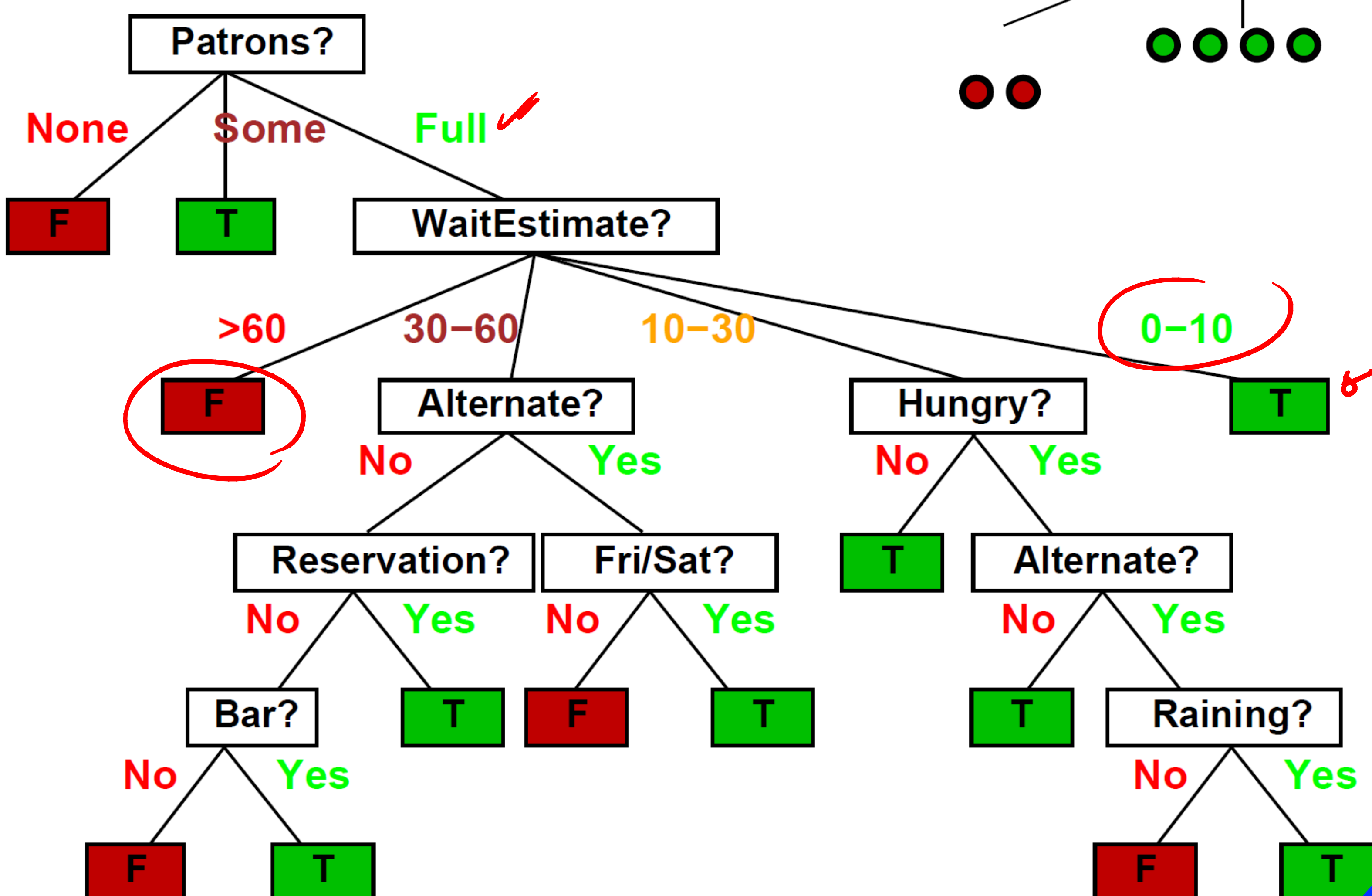
[AI book of Stuart Russell and Peter Norvig]

Dealing with mixed Features

interpretability

A learned decision tree

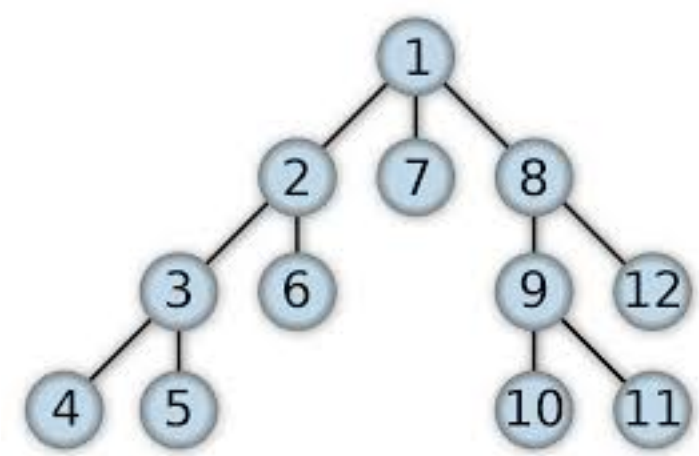




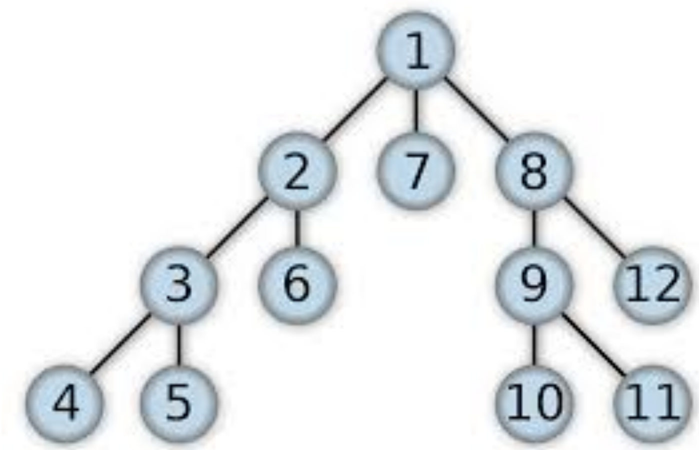
tuart Russell and Peter Norvig]

Marketing Strategy:

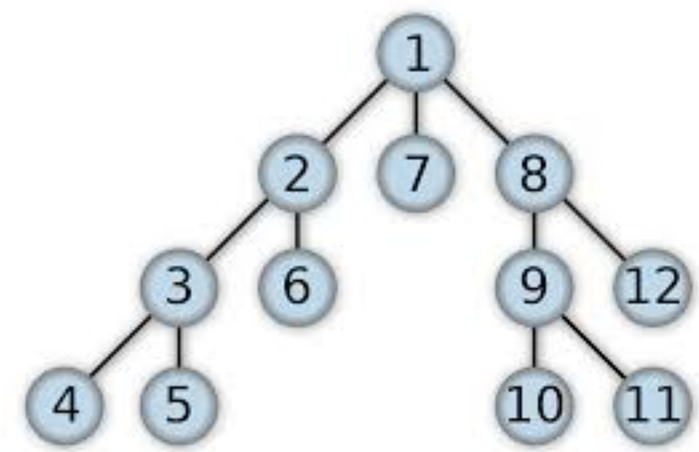
- Why learn the trees in a **greedy** fashion?



- Why learn the trees in a **greedy** fashion?
- Are the trees **unique**?

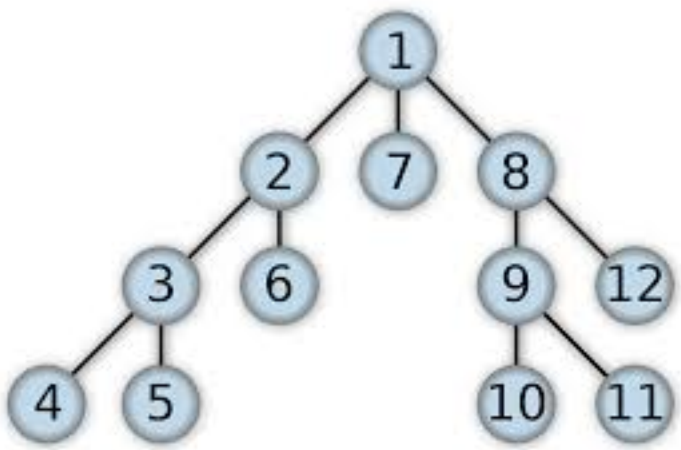


- Why learn the trees in a **greedy** fashion?
- Are the trees **unique**?
- Which **attribute** to use **first**?



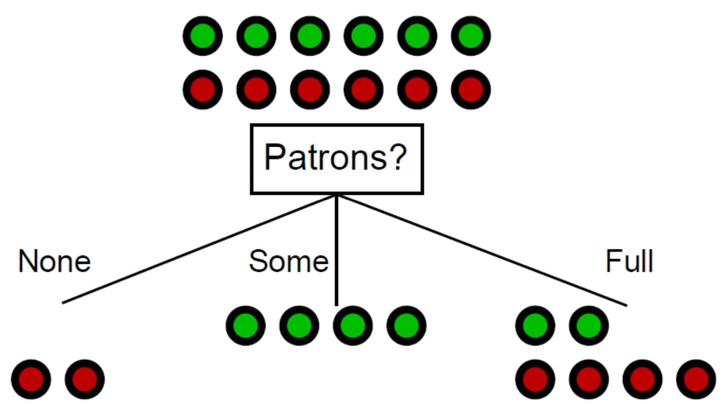
- Why learn the trees in a **greedy** fashion?
- Are the trees **unique**?
- Which **attribute** to use **first**?

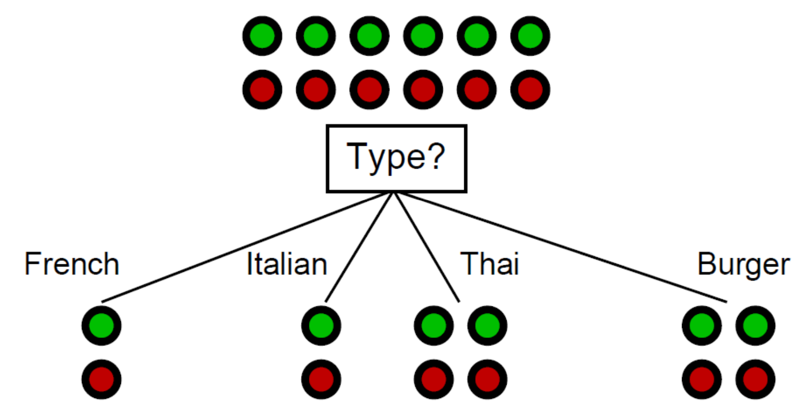
  The one with **maximum information!**



How do we construct the tree ?
i.e., how to pick attribute (nodes)?

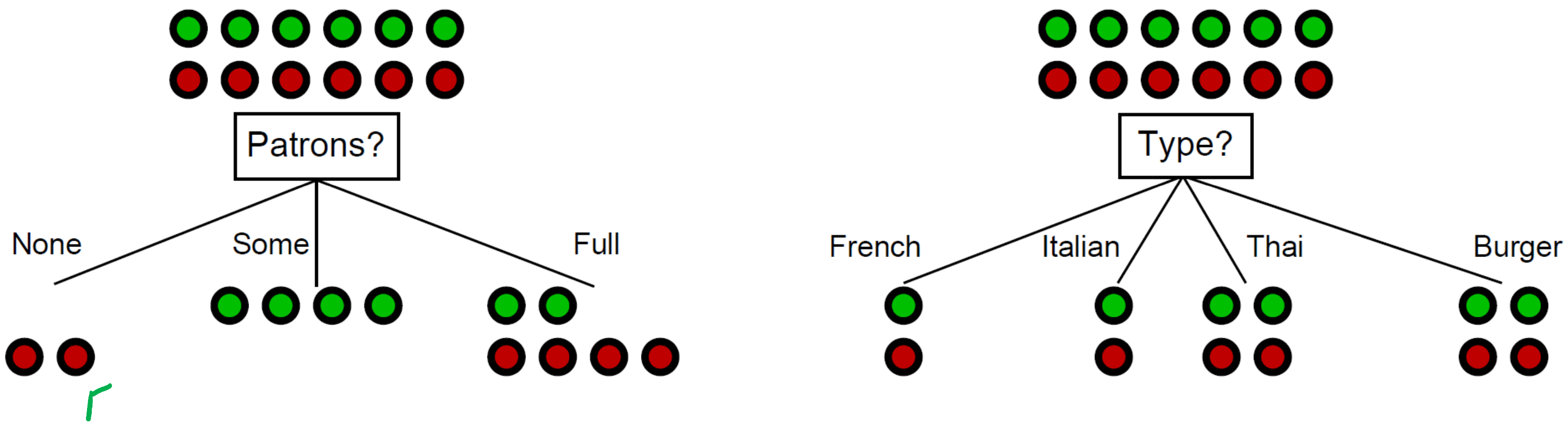
Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"





How do we construct the tree ?
i.e., how to pick attribute (nodes)?

Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

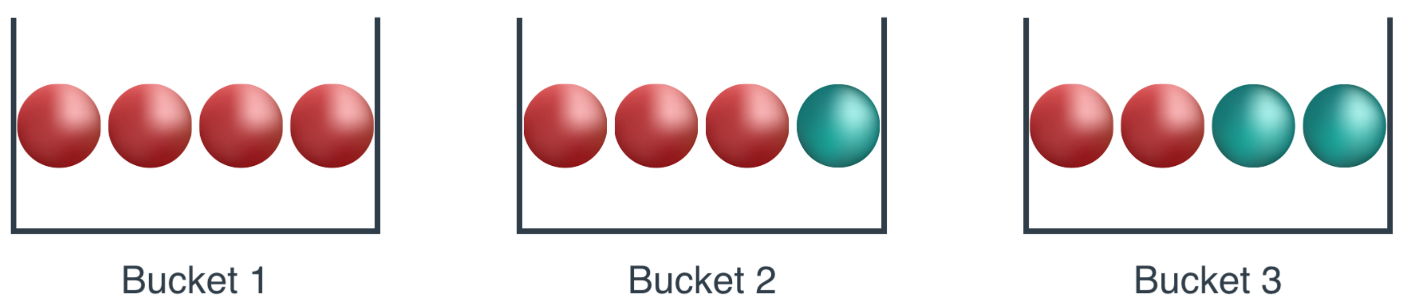


Patrons? is a better choice—gives **information** about the classification

Entropy & Information Gain

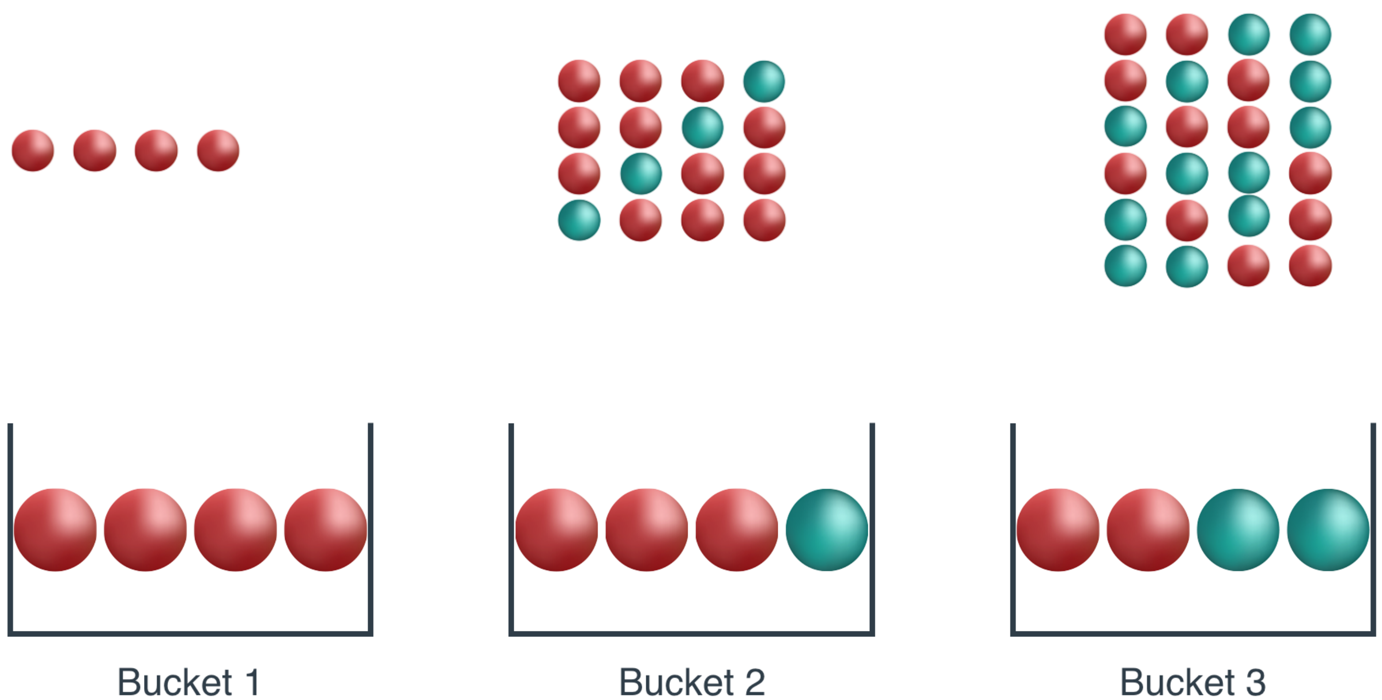
- **Entropy concept**: A measure of how quickly the molecules are moving

- In the probability sense, assume we have three buckets with *identical same color* balls:



Entropy & Information Gain

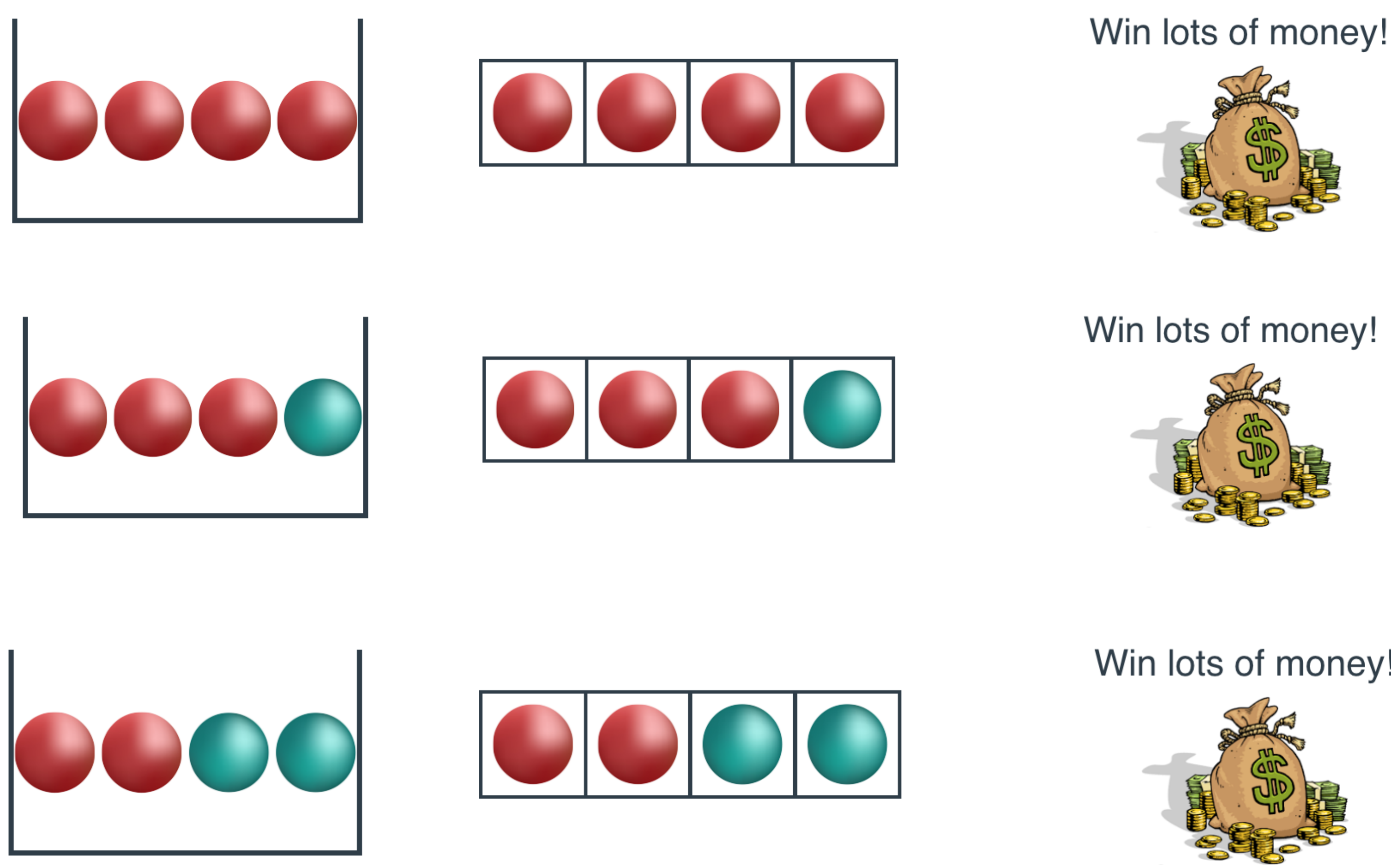
- **Entropy concept**: How much can we **rearrange** each bucket to get **unique sequences**?



- **Information concept**: How much am I certain a ball I'm choosing is red?
  - Important: *same color* balls are *identical.*

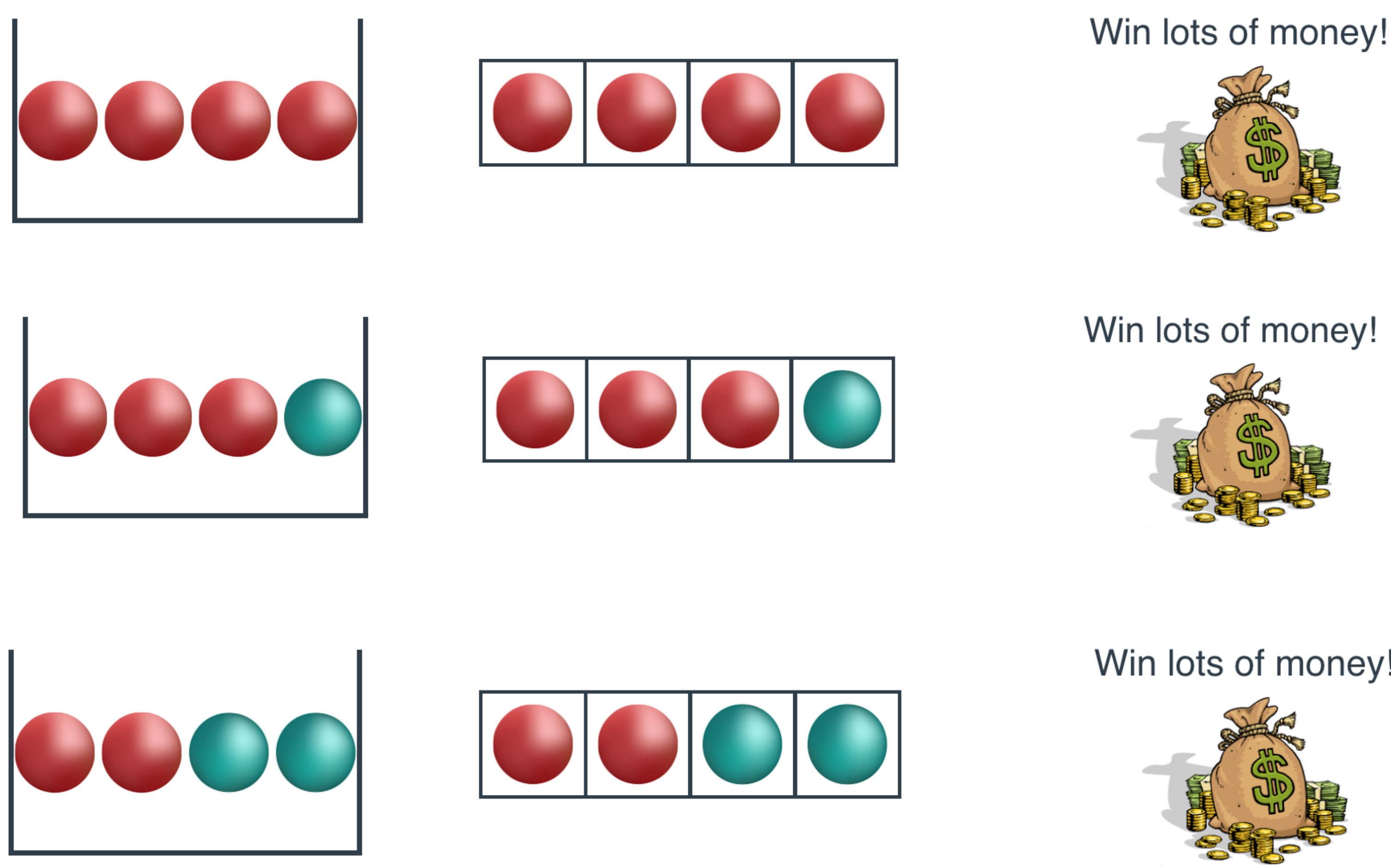
Entropy & Information Gain

- *Let's play a game:* Can we guess the correct sequence for sampling with **replacement**?
- What's the winning probability?



Entropy & Information Gain

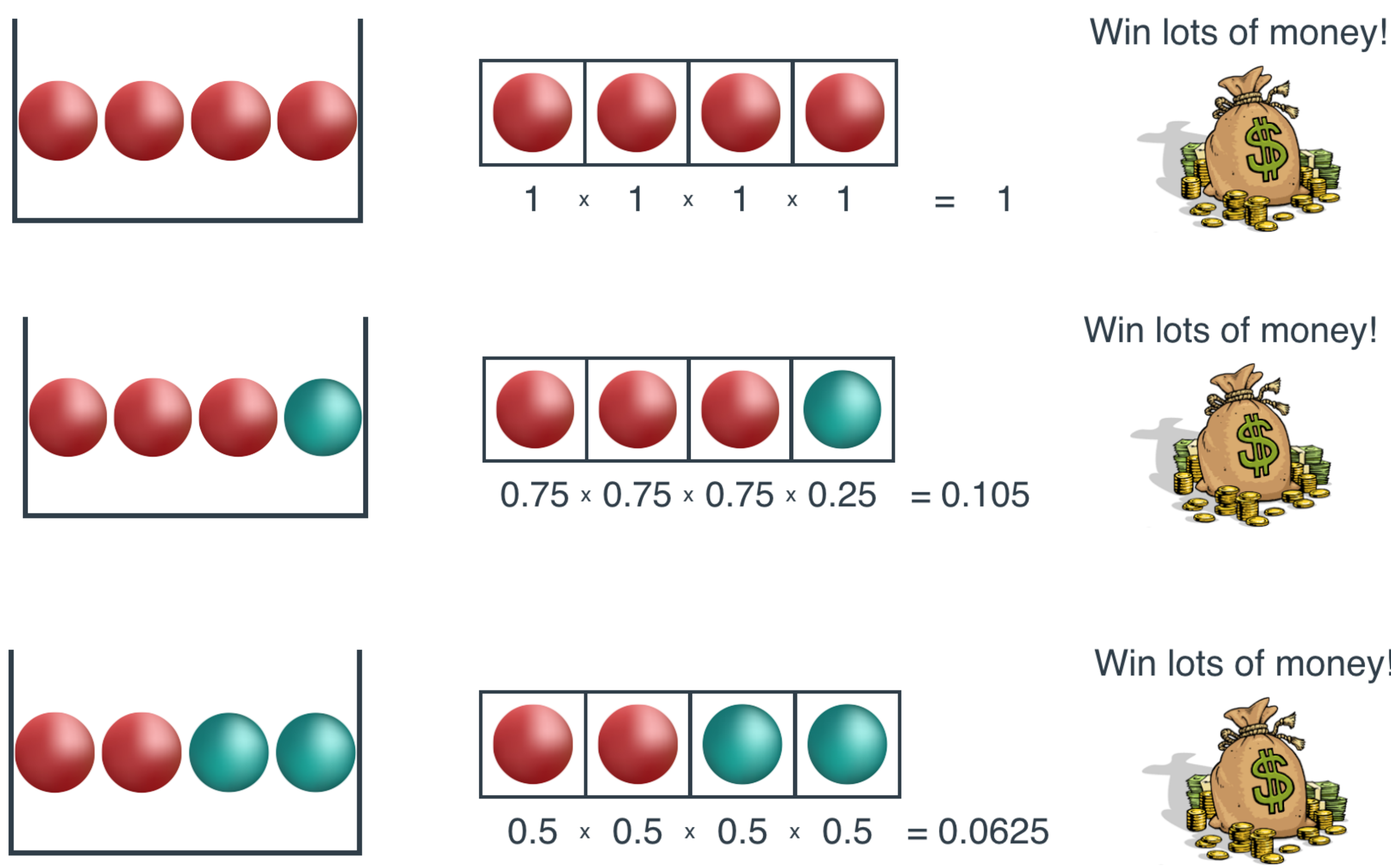
- *Let's play a game:* Can we guess the correct sequence for sampling with **replacement**?
- What's the winning probability?



Assuming **independent** events: $p(a, b) = p(a) \times p(b)$

Entropy & Information Gain

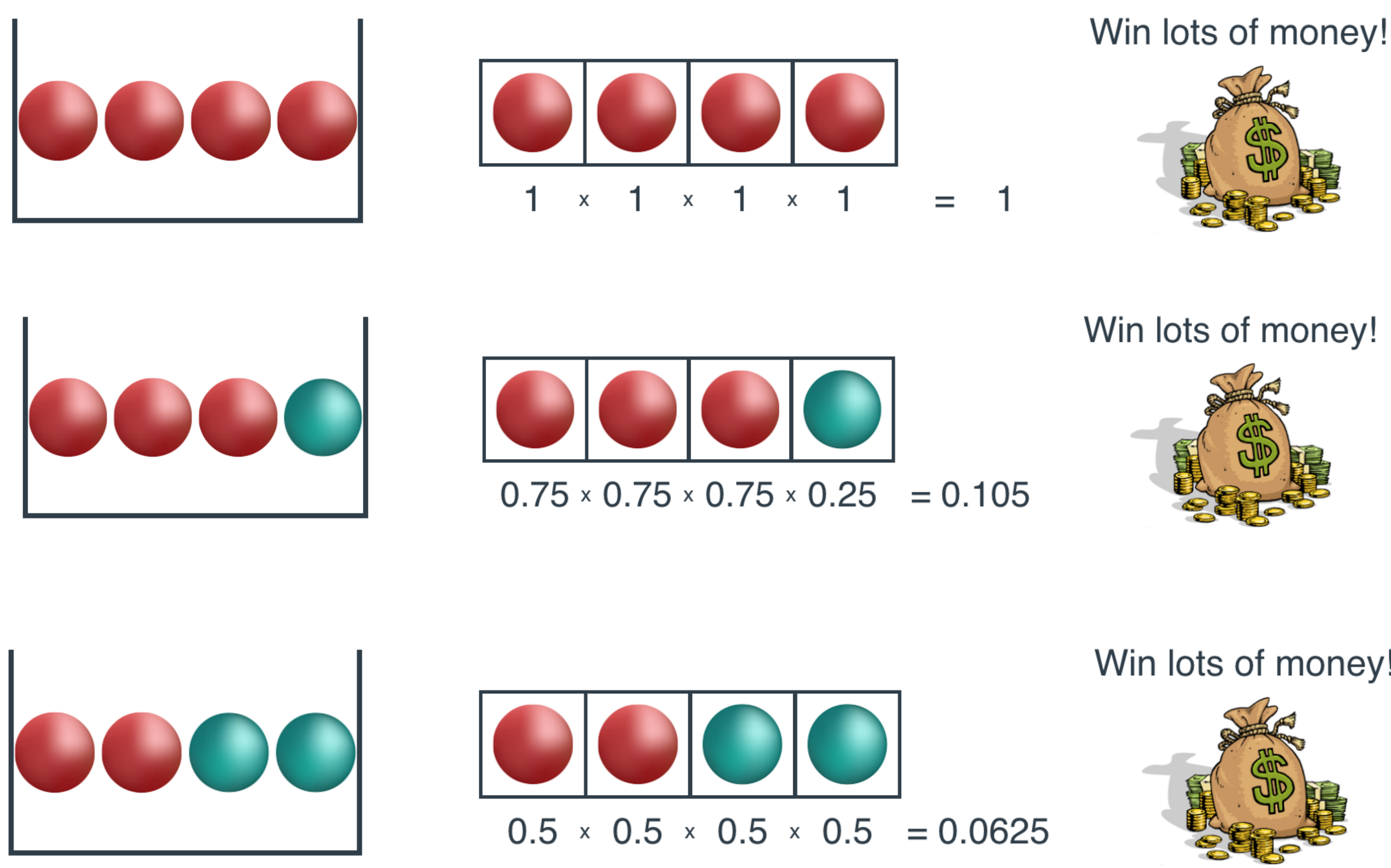
- *Let's play a game:* Can we guess the correct sequence for sampling with **replacement**?
- What's the winning probability?



Assuming **independent** events: $p(a, b) = p(a) \times p(b)$

Entropy & Information Gain

- *Let's play a game:* Can we guess the correct sequence?
- What's the winning probability?



Issue: increasing number of events shrinks the probability.
Solution: use **logarithm of probability** instead and take **the average**.

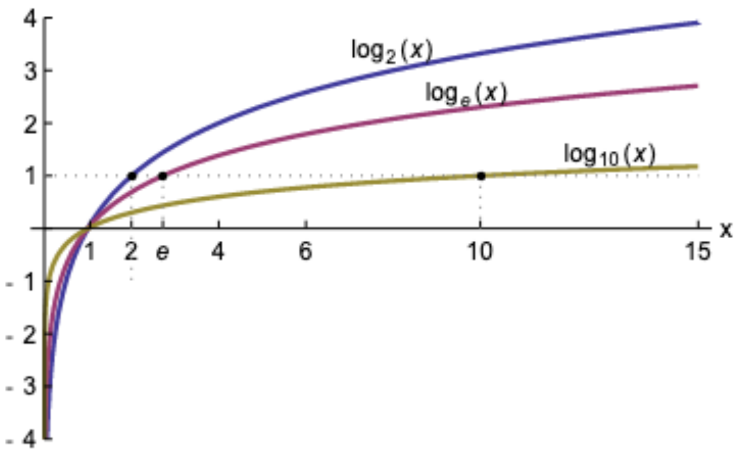
Entropy & Information Gain

- Why a logarithm function?

$$log(p_1 \times p_2) = log(p_1) + log(p_2)$$

- **Shannon Entropy:**

$$H(p_1, \ldots, p_N) = -\sum_{i=1}^{N} p_i \,.\, log\left(p_i\right)$$

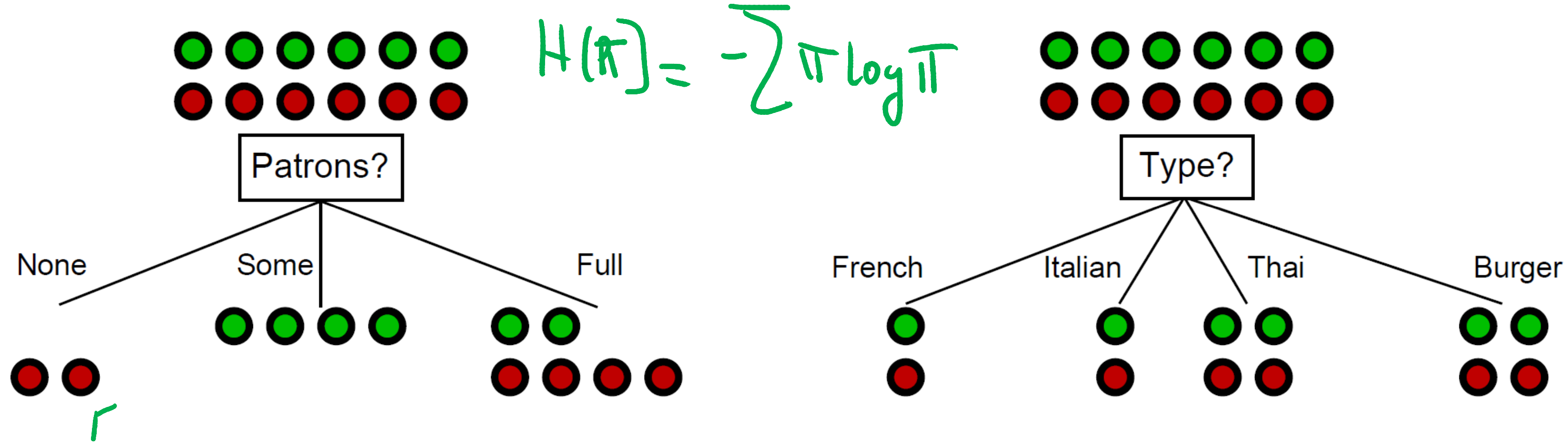


Issue: increasing number of events shrinks the probability.
Solution: use **logarithm of probability** instead and take **the average.**

How do we construct the tree ?
i.e., how to pick attribute (nodes)?

Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

$H(\pi) = -\sum \pi \log \pi$



Patrons? is a better choice—gives **information** about the classification

For a training set containing p positive examples and n negative examples, we have:

$$H\left(\frac{p}{\boxed{p+n}}, \frac{n}{p+n}\right) = -\underbrace{\frac{\overset{\text{prob}(p)}{p}}{p+n}\log_2 \frac{\overset{\text{prob}(p)}{p}}{p+n}} - \underbrace{\frac{\overset{\text{prob}(n)}{n}}{p+n}\log_2 \frac{\overset{\text{prob}(n)}{n}}{p+n}}$$

How to pick nodes?

- A chosen attribute A, with K distinct values, divides the training set E into subsets $E_1, \ldots, E_K$.
- The **Expected Entropy (EH) remaining** after trying attribute A (with branches $i=1,2,\ldots,K$) is

points in child i

$$EH(A) = \sum_{i=1}^{K} \frac{p_i + n_i}{p+n} H\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

- **Information gain (I)** or **reduction in entropy** for this attribute is:

$$I(A) = H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - EH(A)$$

= **Entropy** in the **parent node -** remaining **Expected Entropy** in the **child nodes**

[Hwee Tou Ng & Stuart Russell]

How to pick nodes?

- A chosen attribute A, with K distinct values, divides the training set E into subsets $E_1, \ldots, E_K$.
- The **Expected Entropy (EH) remaining** after trying attribute A (with branches $i=1,2,\ldots,K$) is

points in child i

$$EH(A) = \sum_{i=1}^{K} \frac{p_i + n_i}{\boxed{p+n}} H\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

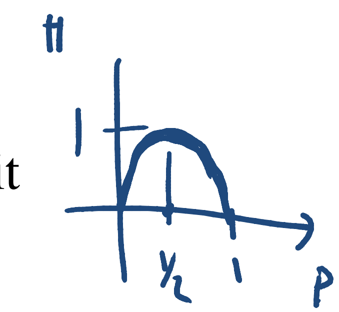
- **Information gain (I)** or **reduction in entropy** for this attribute is:

$$I(A) = H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - EH(A)$$

- Choose the attribute with the largest I

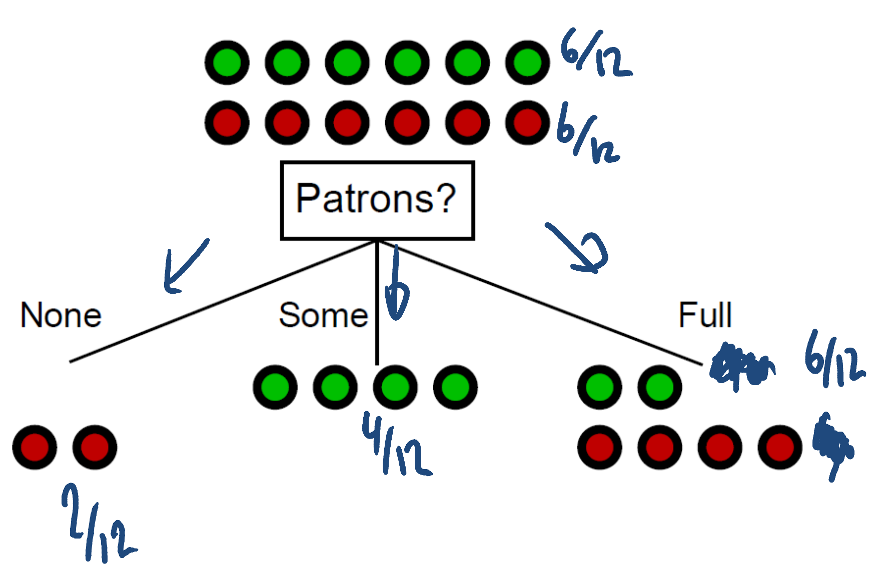
[Hwee Tou Ng & Stuart Russell]

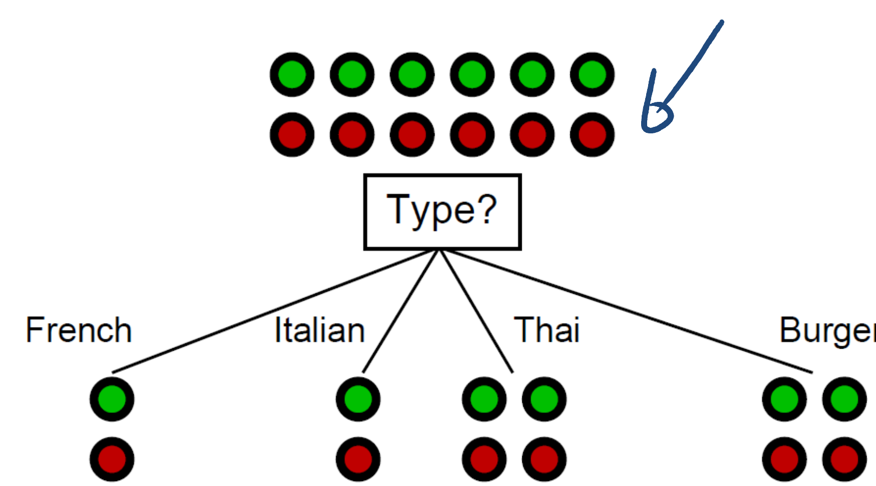
Example



- **Convention**: For the training set, $p = n = 6$, $\underbrace{H(6/12, 6/12)} = \underline{1}$ bit

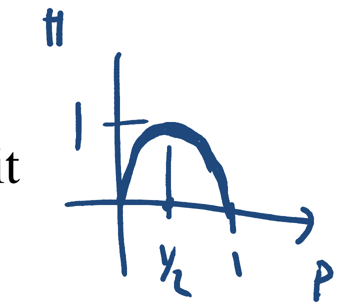
$$H(\frac{6}{12},\frac{6}{12}) = -\frac{6}{12} \times log(\frac{6}{12}) - \frac{6}{12} \times log(\frac{6}{12}) = -2 \times \frac{6}{12} \times -1 = 1$$





[Hwee Tou Ng & Stuart Russell]

Example

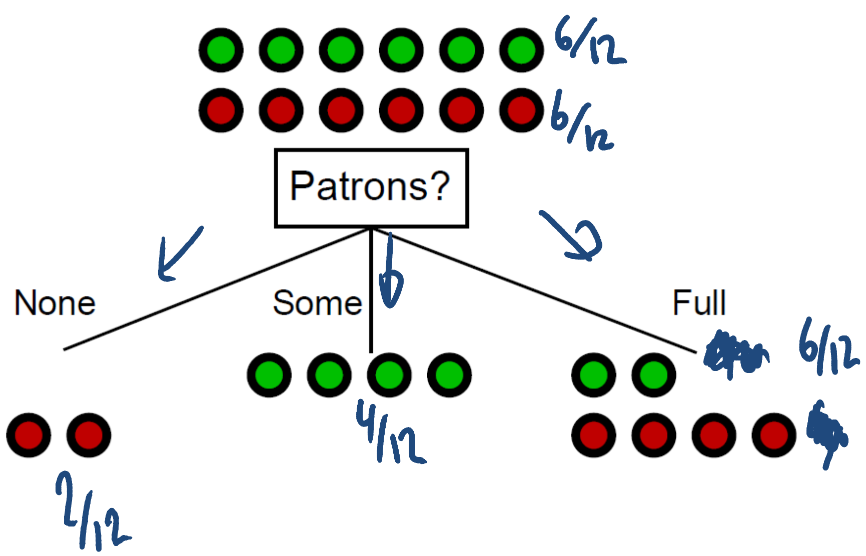


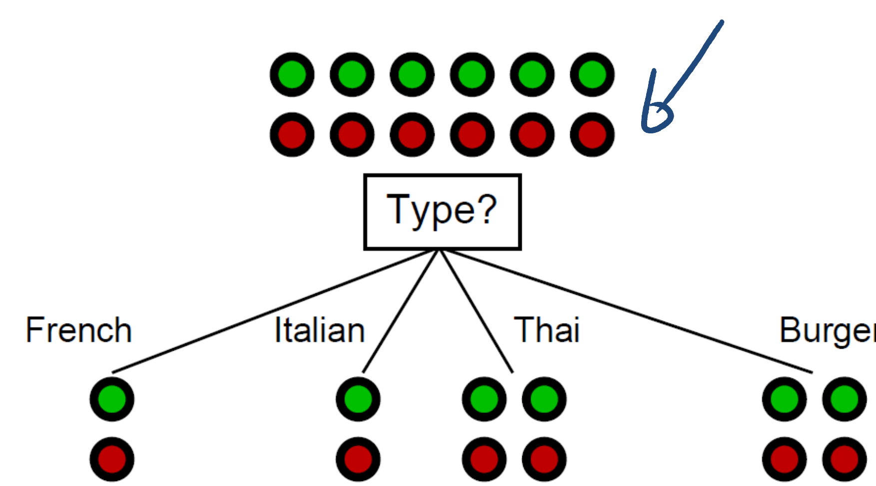
- **Convention**: For the training set, $p = n = 6$, $\underbrace{H(6/12, 6/12)} = 1$ bit

- Consider the attributes *Patrons* and *Type* (and others too):

$$I(Patrons) = 1 - \left[\frac{2}{12}H(0,1) + \frac{4}{12}H(1,0) + \frac{6}{12}H\left(\frac{2}{6},\frac{4}{6}\right)\right] = .0541 \text{ bits}$$

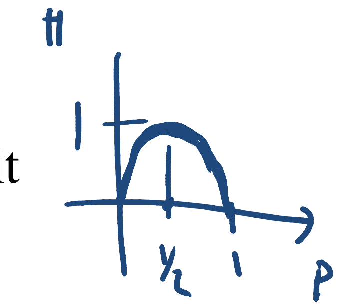
$$I(Type) = \text{?}$$





[Hwee Tou Ng & Stuart Russell]

Example

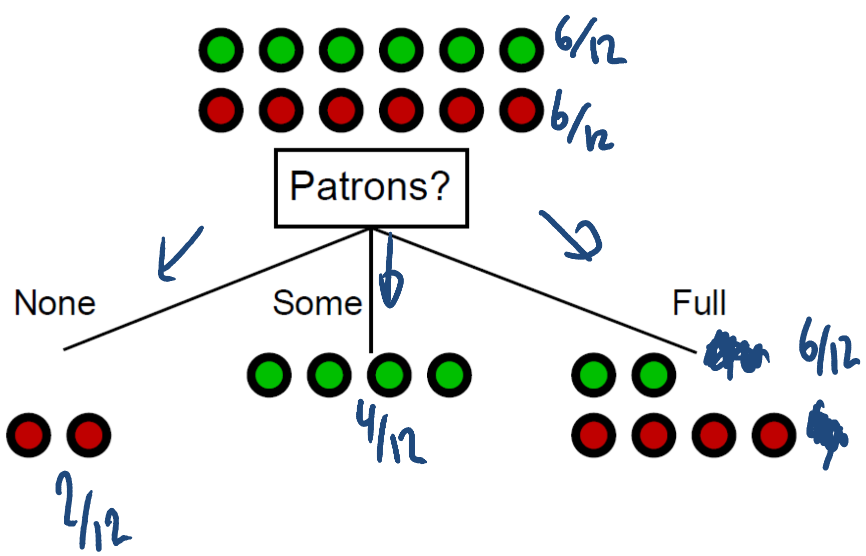


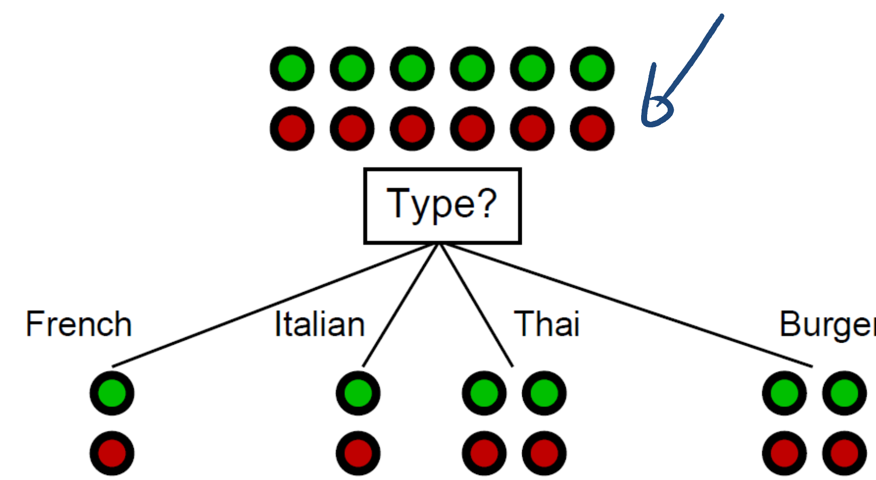
- **Convention**: For the training set, $p = n = 6$, $\underbrace{H(6/12, 6/12)} = \underline{1}$ bit

- Consider the attributes *Patrons* and *Type* (and others too):

$$I(Patrons) = 1 - [\frac{2}{12}H(0,1) + \frac{4}{12}H(1,0) + \frac{6}{12}H(\frac{2}{6},\frac{4}{6})] = .0541 \text{ bits}$$

$$I(Type) = 1 - [\frac{2}{12}H(\frac{1}{2},\frac{1}{2}) + \frac{2}{12}H(\frac{1}{2},\frac{1}{2}) + \frac{4}{12}H(\frac{2}{4},\frac{2}{4}) + \frac{4}{12}H(\frac{2}{4},\frac{2}{4})] = 0 \text{ bits}$$

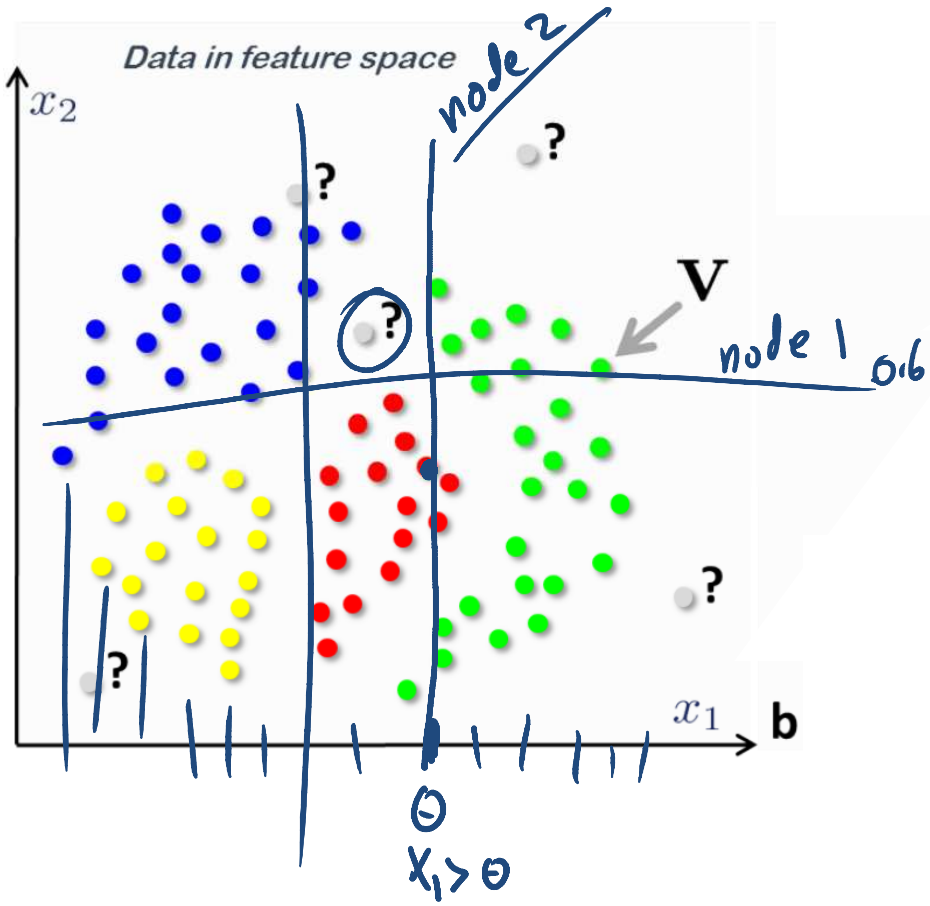




[Hwee Tou Ng & Stuart Russell]

- Entropy of a fair coin at the point of highest uncertainty ($p = \frac{1}{2}$) equals 1 bit.

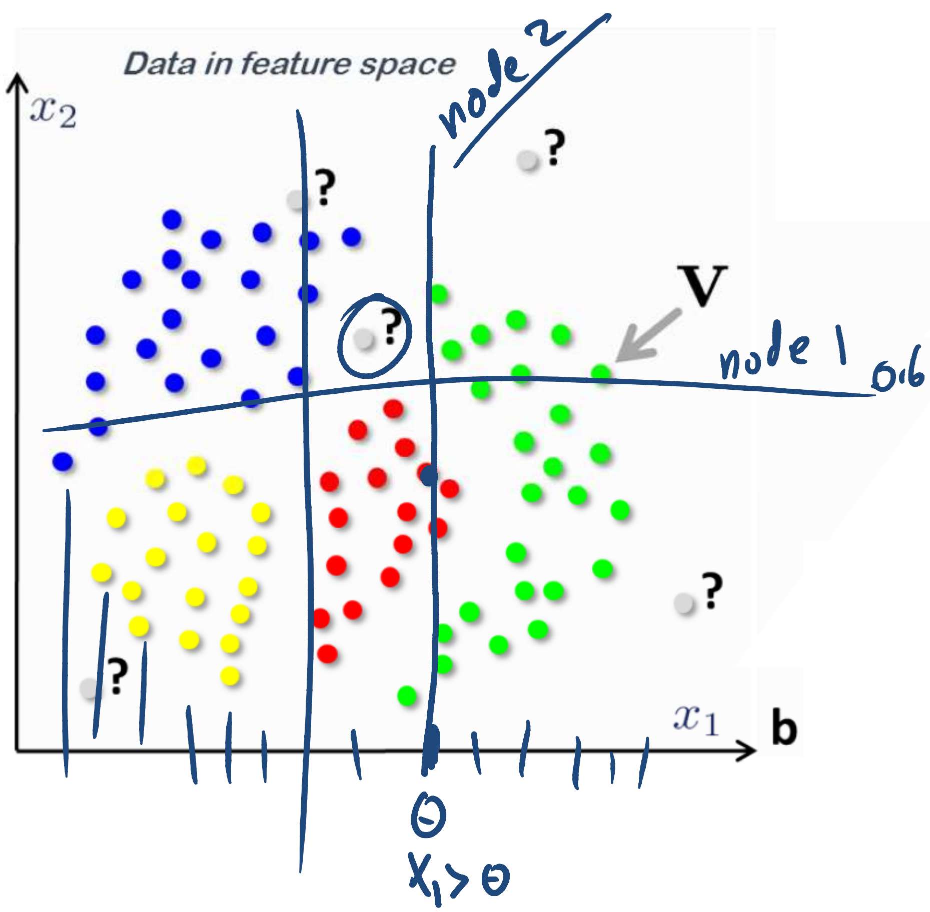
Classification tree



- How to deal with ***continuous features***?

A generic data point is denoted by a vector $\mathbf{v} = (x_1, x_2, \cdots, x_d)$

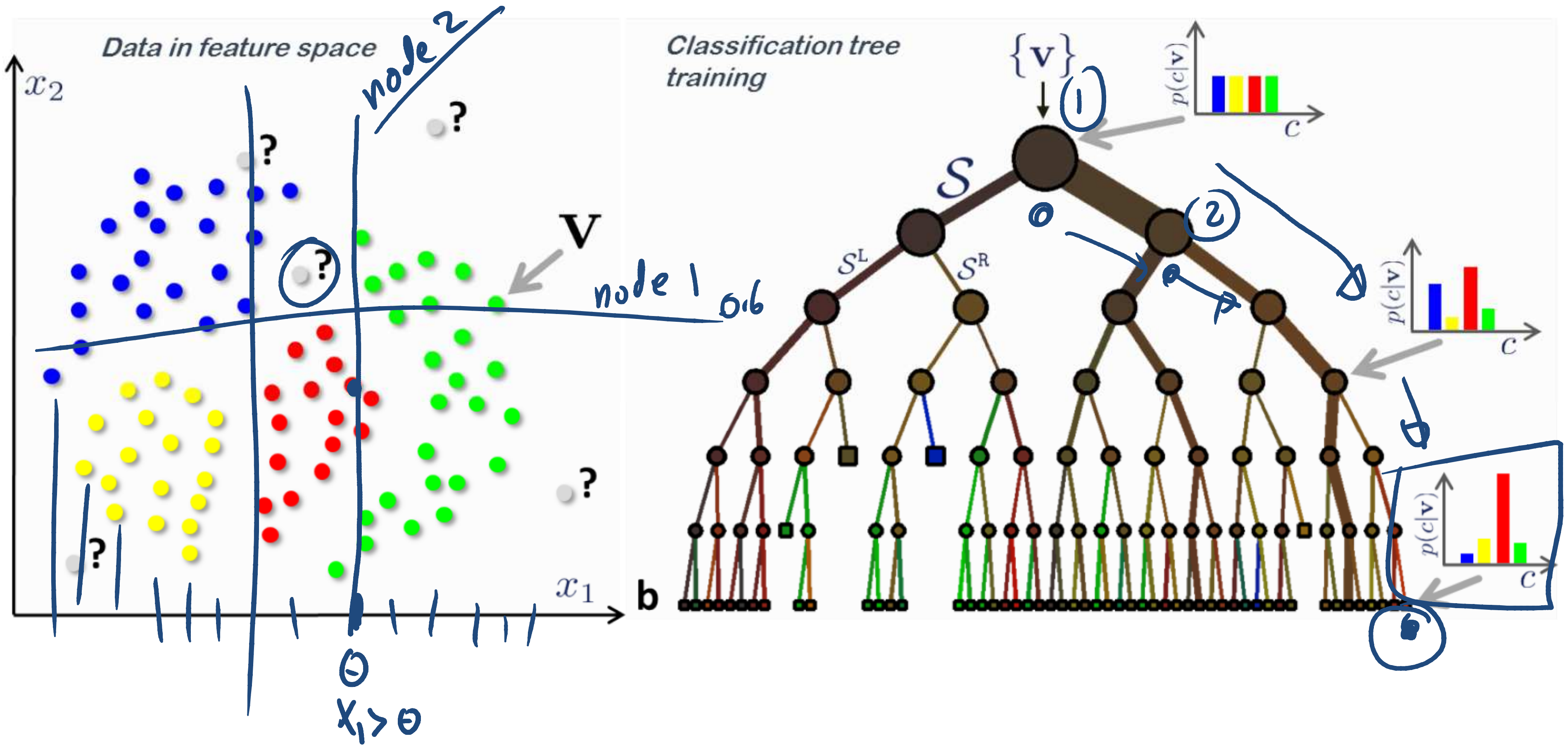
Classification tree



- How to deal with ***continuous features***?
  - Create the splits **randomly**
  - Compute **information gain** for each split
  - Choose the one with **maximum gain**

A generic data point is denoted by a vector $\mathbf{v} = (x_1, x_2, \cdots, x_d)$

Classification tree



A generic data point is denoted by a vector $\mathbf{v} = (x_1, x_2, \cdots, x_d)$

$$\mathcal{S}_j = \mathcal{S}_j^{\mathrm{L}} \cup \mathcal{S}_j^{\mathrm{R}}$$

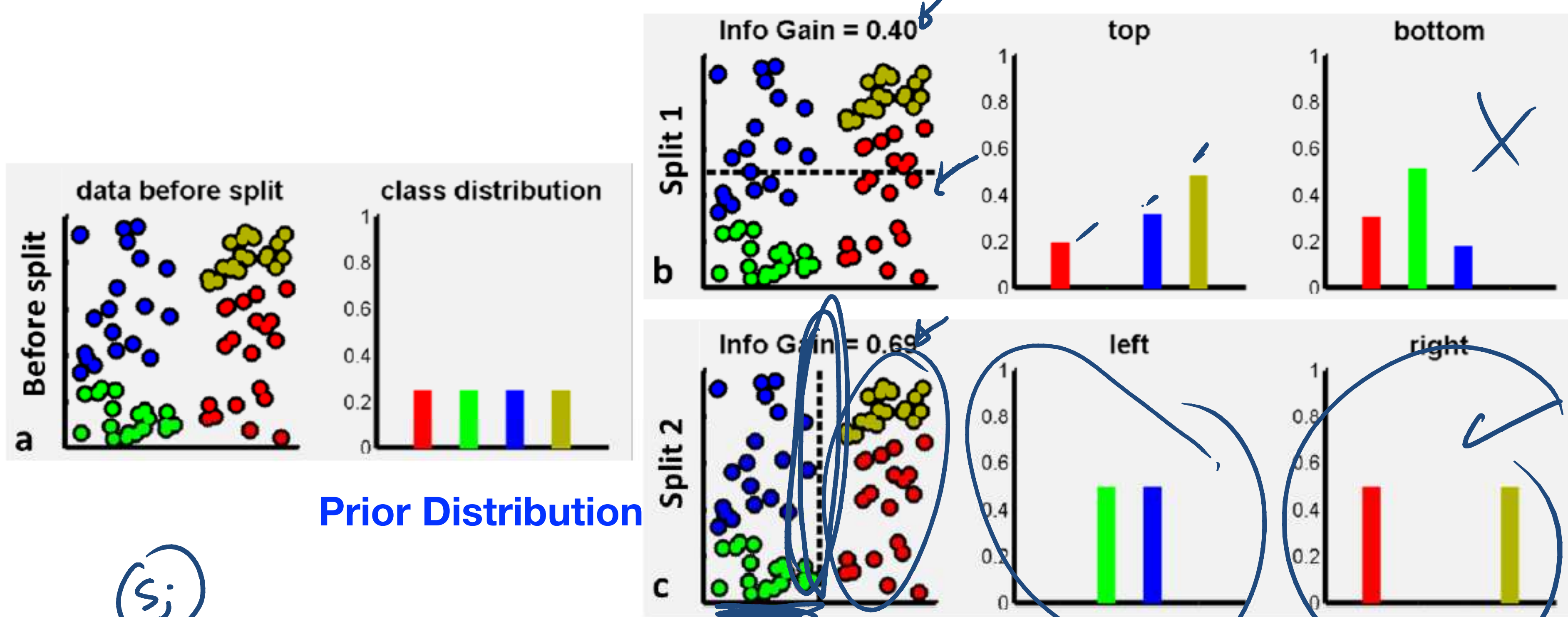
[Criminisi et al, 2011]

- Note that the **histogram** shows the **posterior distribution** for each class:

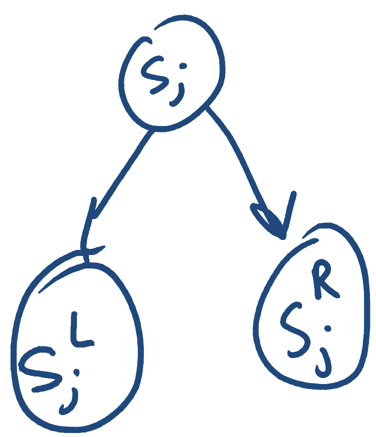
$$p(Class\,|\,Data)$$

Use information gain to decide splits

Posterior Distribution



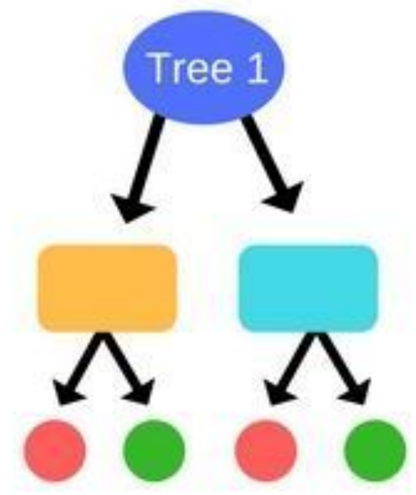
Prior Distribution



$$I_j = H(\mathcal{S}_j) - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}_j^i|}{|\mathcal{S}_j|} H(\mathcal{S}_j^i)$$

[Criminisi et al, 2011]

- Decision Trees are **interesting** because:
    - **Interpretable** → easy to understand for ML practitioners
    - **Scales** well → can deal with data with many features
    - *But* does NOT **generalize** well → **High variance**



- Decision Trees are **interesting** because:
    - **Interpretable** → easy to understand for ML practitioners
    - **Scales** well → can deal with data with many features
    - *But* does NOT **generalize** well → **High variance**
    - **Solution: Random Forest**
        - **Remove** variance by **averaging** over trees