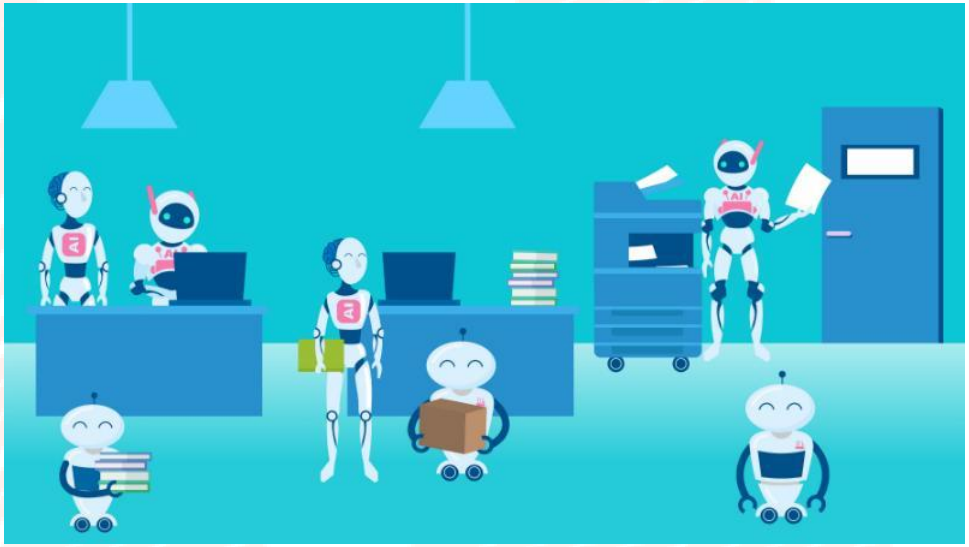

FACULTY OF INFORMATION TECHNOLOGY

# Artificial Intelligence Fundamentals (NM TTNT)

**Semester 1, 2021/2022**

# Chapter 8. Learning



# Content

- Learning agents
- Inductive learning
- Decision tree learning
- Measuring learning performance

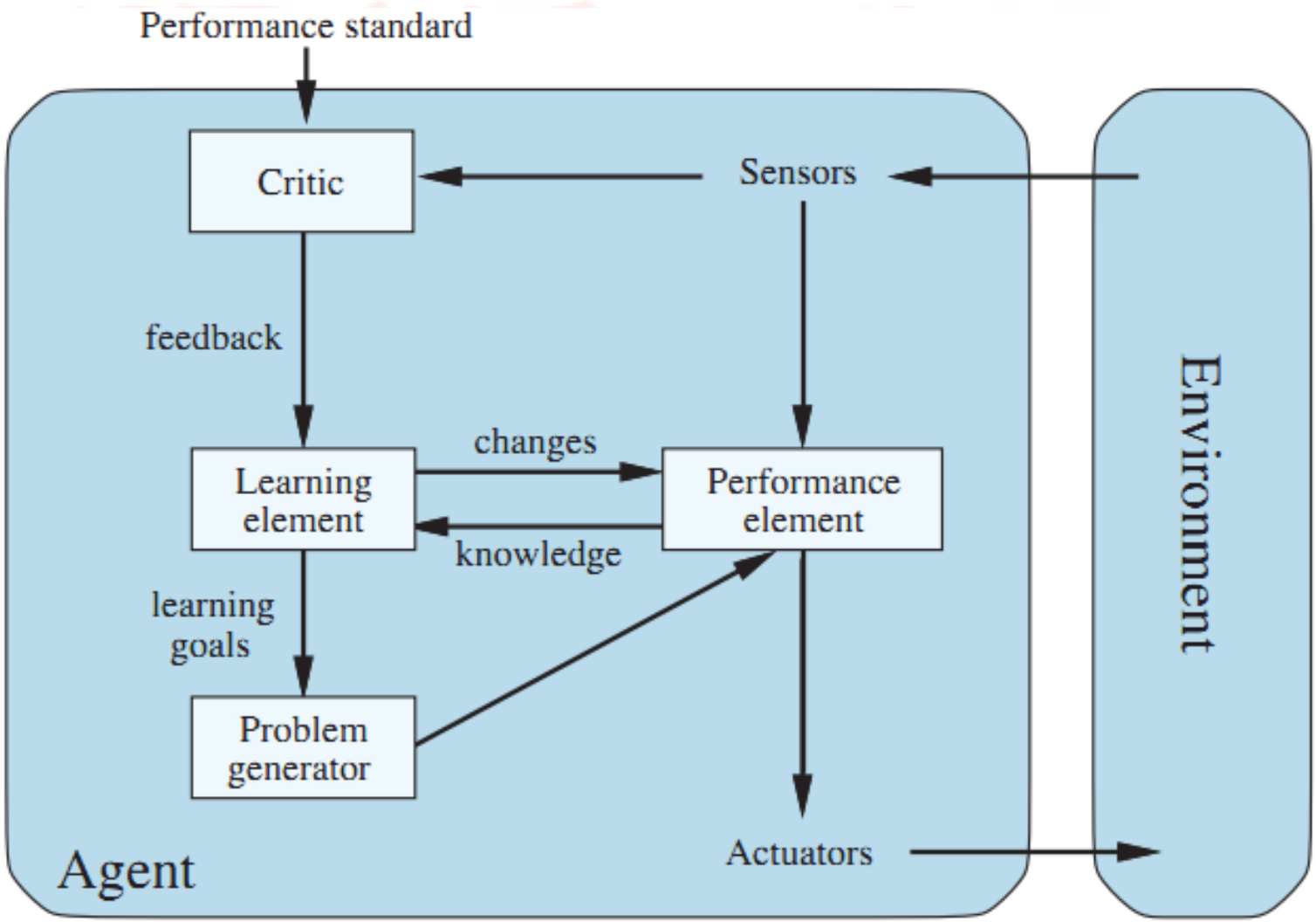
# Learning

- Learning is essential for unknown environments,
  - i.e., when designer lacks omniscience
- Learning is useful as a system construction method,
  - i.e., expose the agent to reality rather than trying to write it down
- Learning modifies the agent's decision mechanisms to improve performance

# Why do Machine Learning?

- Understand and improve efficiency of human learning
  - Use to improve methods for teaching and tutoring people, as done in CAI -- Computer-Aided Instruction
- Discover new things or structure that is unknown to humans
  - Data mining
- Fill in skeletal or incomplete specifications about a domain
  - Large, complex AI systems cannot be completely derived by hand and require dynamic updating to incorporate new information.
  - Learning new characteristics expands the domain or expertise and lessens the "brittleness" of the system

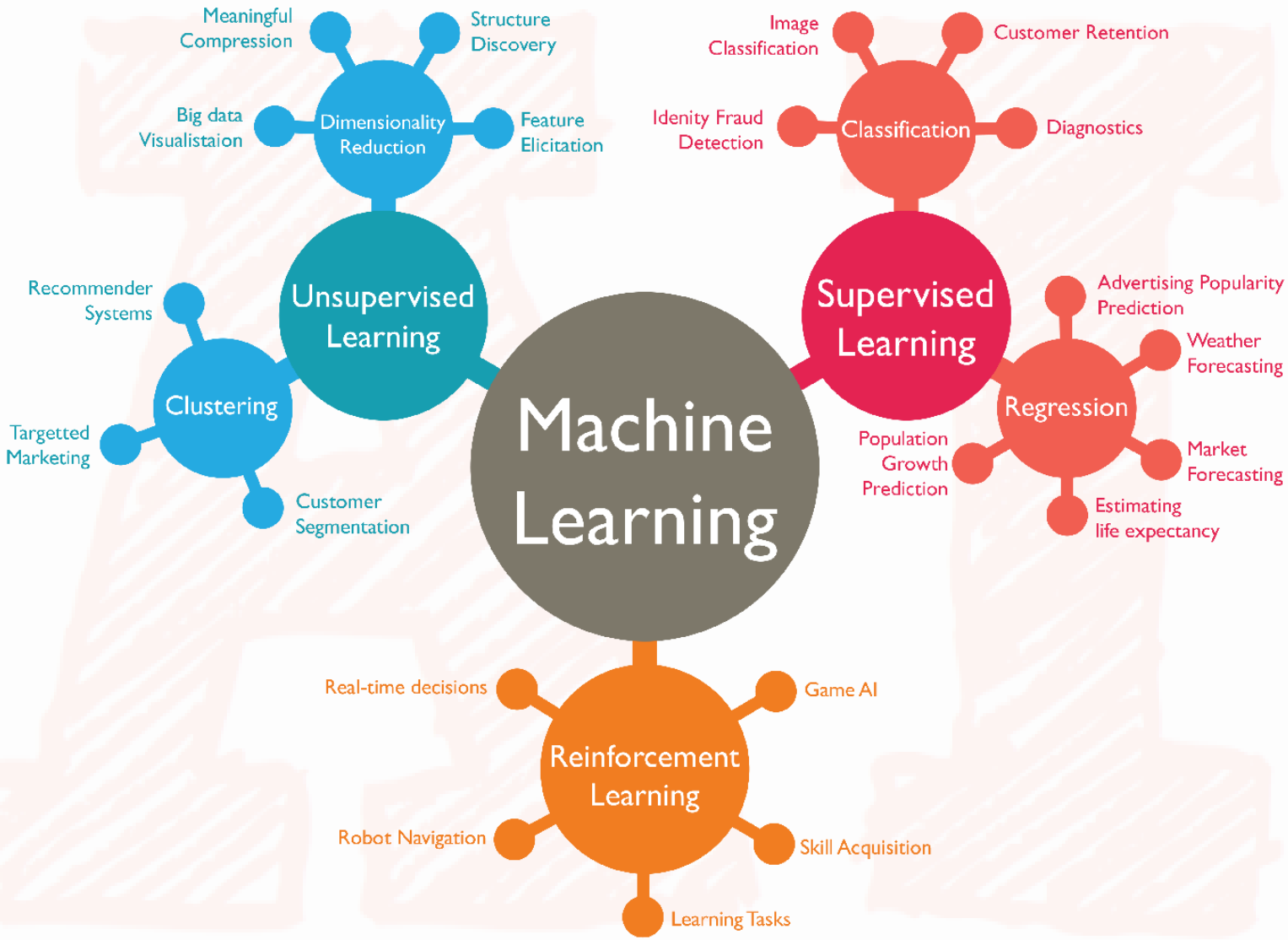
# Learning agents



# Learning element

- Types of feedback:
  - **Supervised learning**: correct answers for each example
    - Identity fraud detection, Spam E-mail detection, Image Classification etc.
  - **Unsupervised learning**: correct answers not given
    - targeted marketing and customer segmentation
  - **Reinforcement learning**: occasional rewards
    - Self driving cars, Game AI (bots), Robot navigation etc.

# Types of Machine Learning

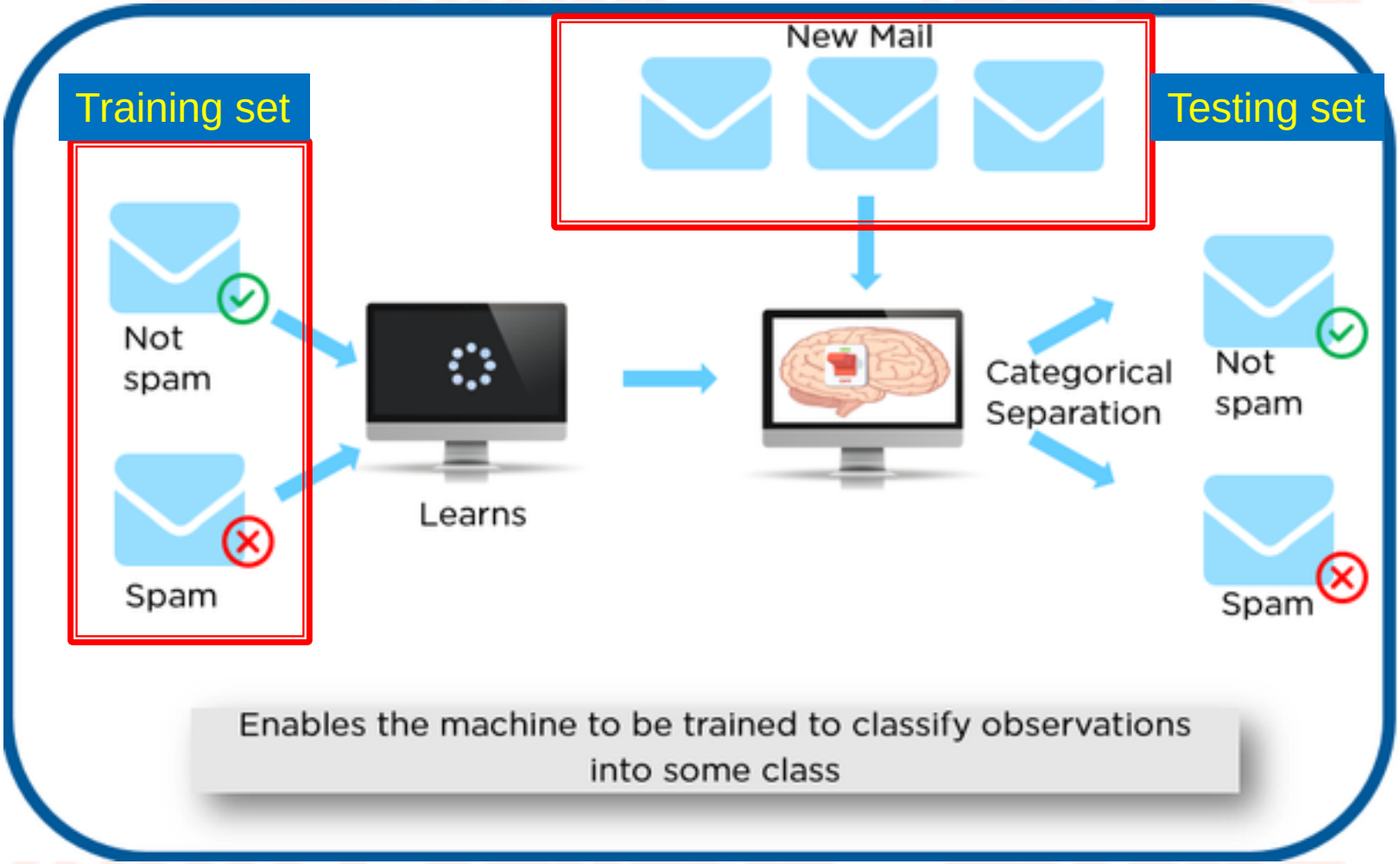



https://medium.com/swlh/types-of-machine-learning-algorithms-62608e83d709

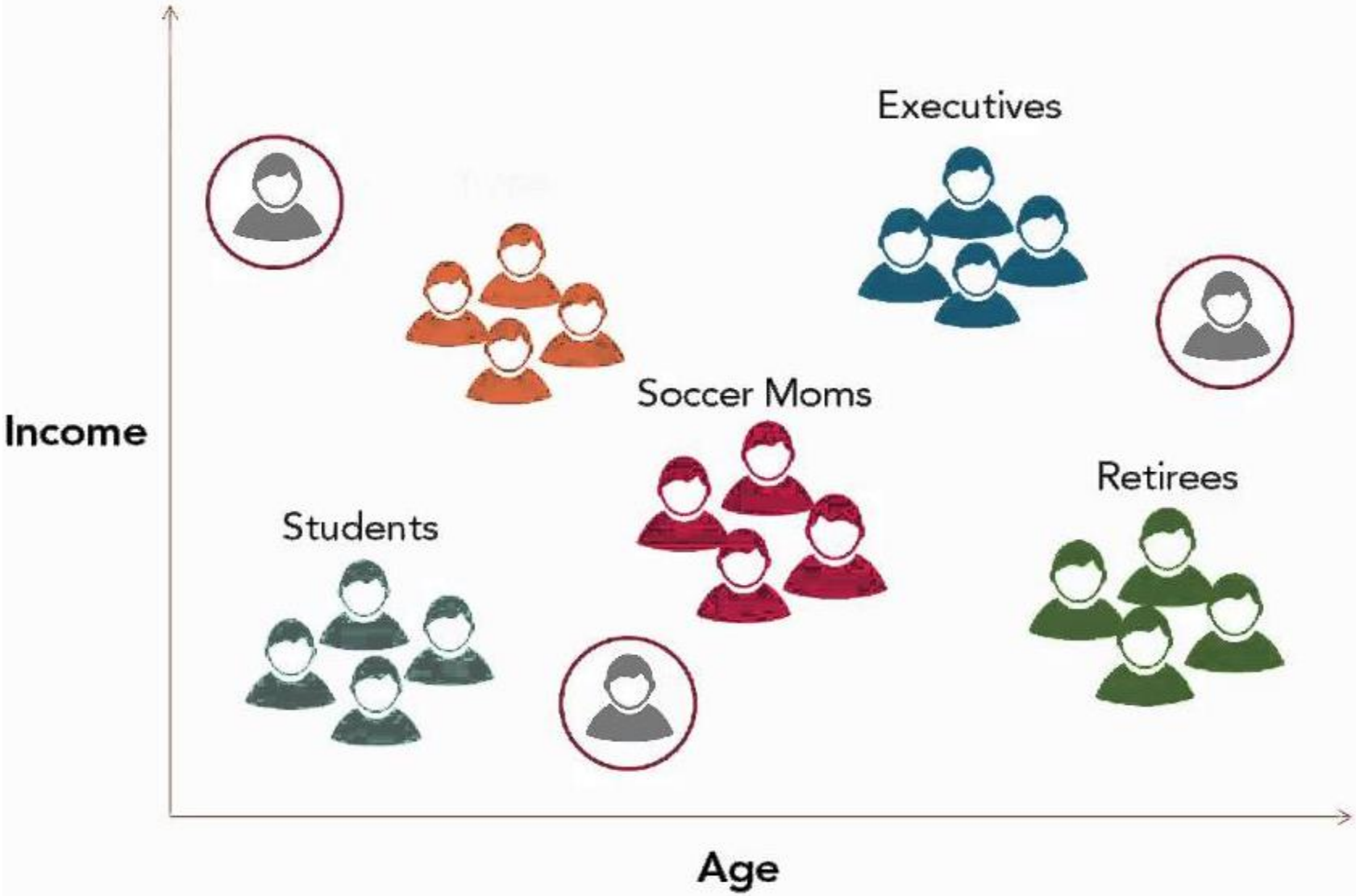
# Supervised vs. Unsupervised learning

- Supervised:
  - "teacher" gives a set of both the input examples and desired outputs, i.e. (x, f(x)) pairs
- Unsupervised:
  - only given the input examples, i.e. the x
- In either case, the goal is to determine an hypothesis h that estimates f.

# Spam mail detection



# Customer segmentation



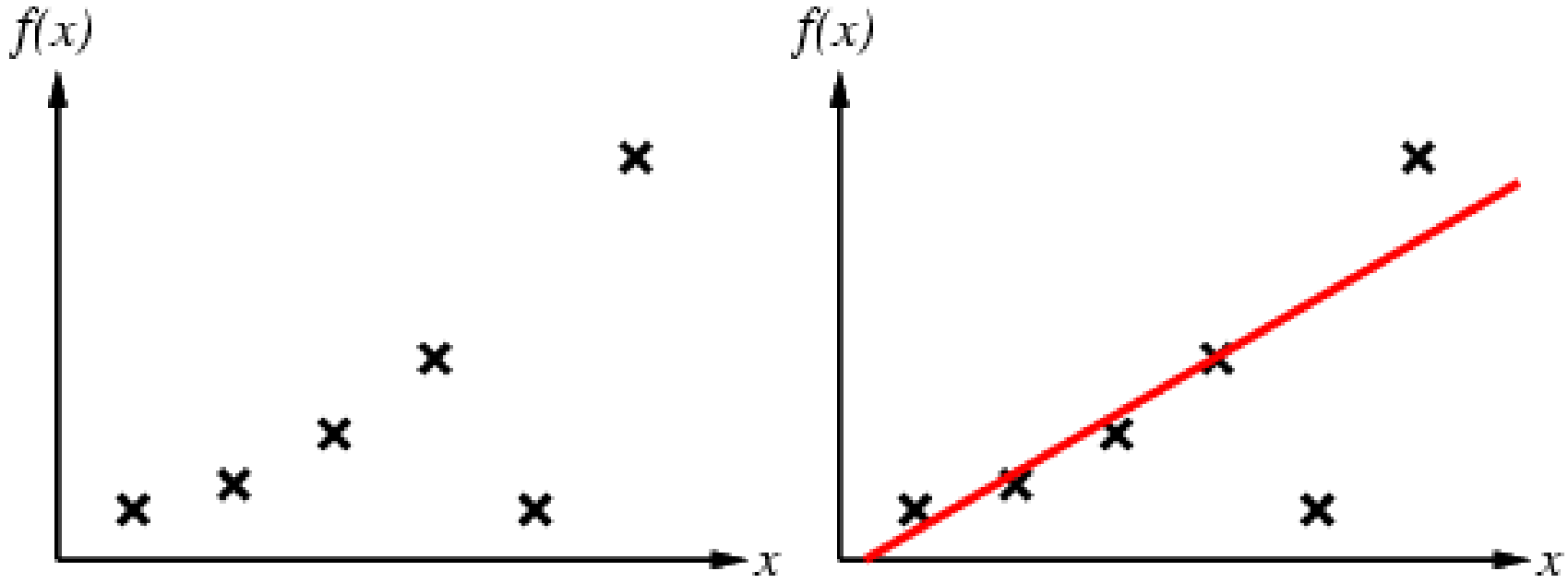
# Inductive Learning

# Inductive learning

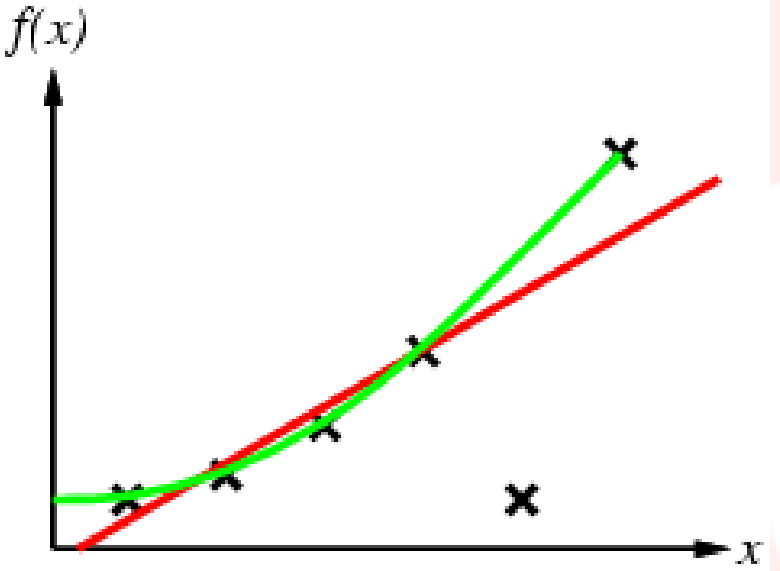
- Simplest form: Learn a function from examples
- Extrapolates from a given set of examples so that accurate predictions can be made about future examples.
- f is the target function
  - An example is a pair (x, f(x))
  - Problem: find a hypothesis h
    - such that **h ≈ f**
    - given a **training set of examples**

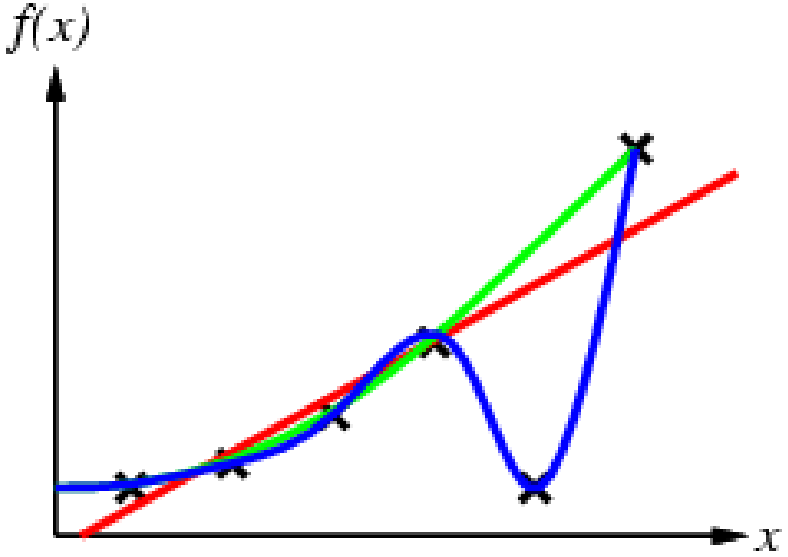
# Inductive learning method

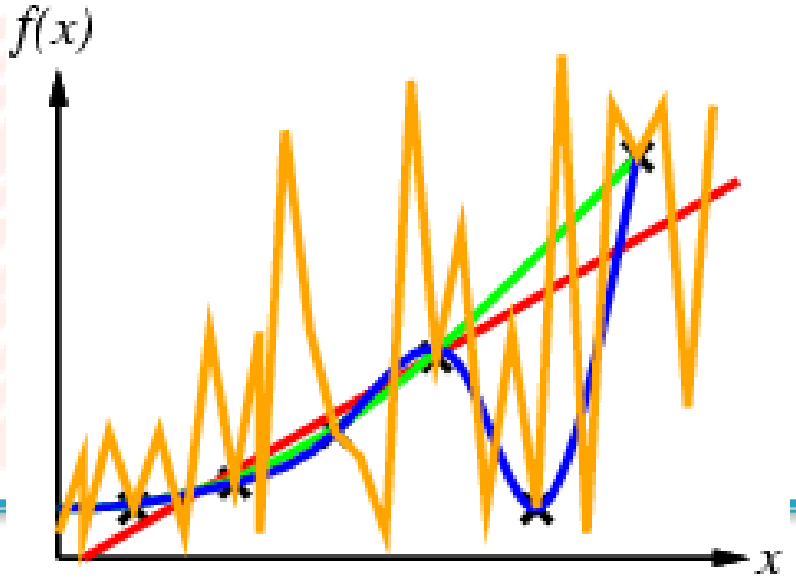
- Construct/adjust h to agree with f on training set (h is consistent if it agrees with f on all examples)
- E.g., curve fitting:



# Inductive learning method: Curve fitting

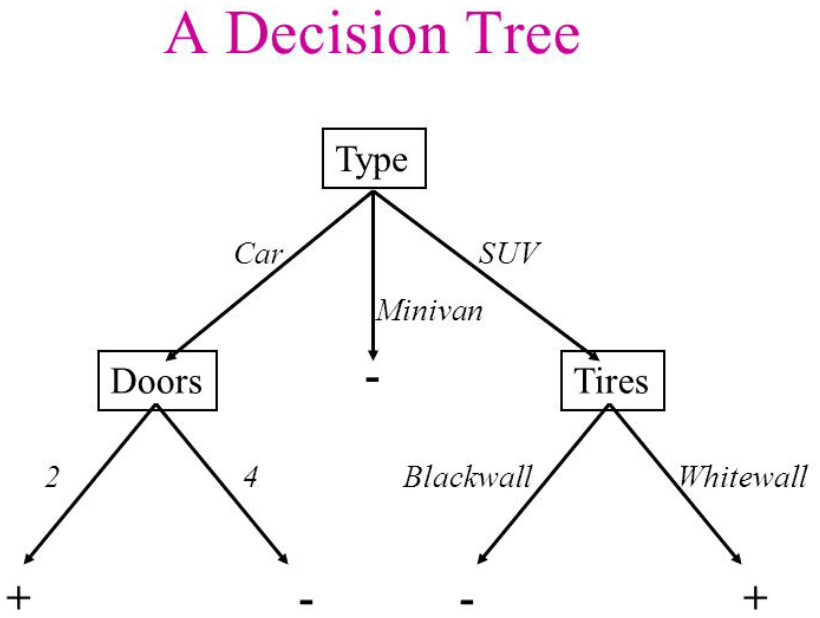



Ockham's razor:
maximize a combination of consistency and simplicity

# Learning Decision Trees



# Learning decision trees

- **Problem**: decide whether to wait for a table at a restaurant, based on the following attributes:
  - **Alternate**: is there an alternative restaurant nearby?
  - **Bar**: is there a comfortable bar area to wait in?
  - **Fri/Sat**: is today Friday or Saturday?
  - **Hungry**: are we hungry?
  - **Patrons**: number of people in the restaurant (**None**, **Some**, **Full**)
  - **Price**: price range (**$**, **$$**, **$$$**)
  - **Raining**: is it raining outside?
  - **Reservation**: have we made a reservation?
  - **Type**: kind of restaurant (**French**, **Italian**, **Thai**, **Burger**)
  - **WaitEstimate**: estimated waiting time (**0–10**, **10–30**, **30–60**, **>60**)

# Attribute-based representations

- Examples described by attribute values (Boolean, discrete, continuous)
- 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* | *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 |

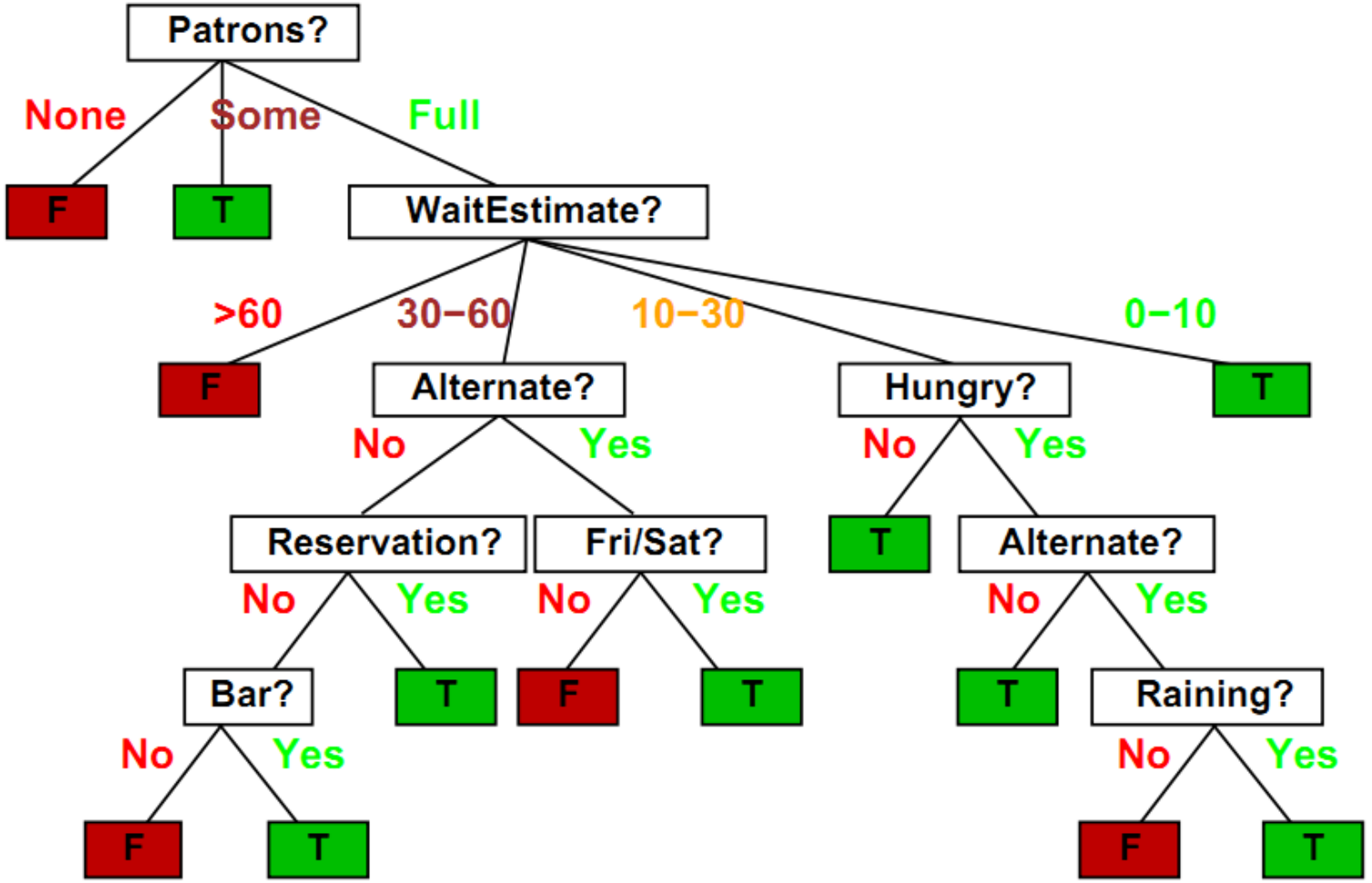
Classification of examples is positive (T) or negative (F)

# How to summarize data

- Idea: Try to capture the logical structure of the data.
  - Create a node for some features, with a descendent for each value,
  - repeat at each node for a different feature,
  - until we can reach a decision.
- Such an object is called a decision tree.
- Seems almost ridiculously simple, but turns out to be extremely useful way to summarize data.

# Decision trees

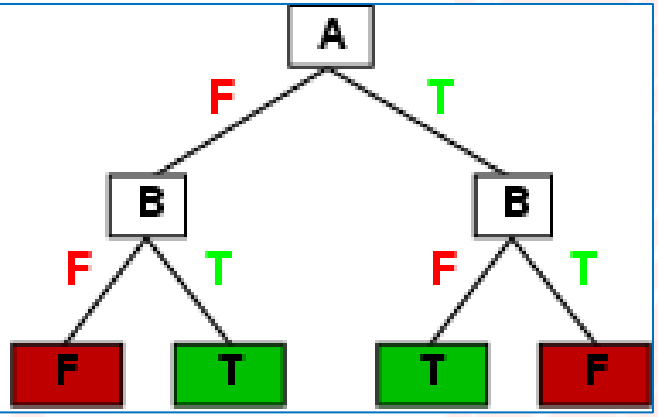
- One possible representation for hypotheses
- E.g., here is the "true" tree for deciding whether to wait:



# Expressiveness

- Decision trees can express any function of the input attributes.
- E.g., 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 |




- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless ***f*** nondeterministic in ***x***) but it probably won't generalize to new examples
- Prefer to find more compact decision trees

# Decision tree

- We can always come up with some decision tree for a data set:
  - Pick any feature not used yet, branch on its values, continue.
  - However, starting with a random feature may lead to a large, unmotivated tree.

- In general, we prefer short trees over larger ones.
  - Why?!
  - Intuitively, a simple (consistent) hypothesis is more likely to be true.

# Hypothesis spaces

- How many distinct decision trees with n Boolean attributes?

  = number of Boolean functions

  = number of distinct truth tables with $2^n$ rows = $2^{2^n}$

- E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

# Decision tree learning

- **Aim**: find a small tree consistent with the training examples
- **Idea**: (recursively) choose "most significant" attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

# Generating decision tree

-Suppose "**age**" is the best attribute
- Split D into D1, D2, D3 by "age" values

| RID | age | income | student | credit_rating | Class: Buys_computer |
|---|---|---|---|---|---|
| 1 | <=30 | high | no | fair | no |
| 2 | <=30 | high | no | excellent | no |
| 3 | 31...40 | high | no | fair | yes |
| 4 | >40 | medium | no | fair | yes |
| 5 | >40 | low | yes | fair | yes |
| 6 | >40 | low | yes | excellent | no |
| 7 | 31…40 | low | yes | excellent | yes |
| 8 | <=30 | medium | no | fair | no |
| 9 | <=30 | low | yes | fair | yes |
| 10 | >40 | medium | yes | fair | yes |
| 11 | <=30 | medium | yes | excellent | yes |
| 12 | 31…40 | medium | no | excellent | yes |
| 13 | 31…40 | high | yes | fair | yes |
| 14 | >40 | medium | no | excellent | no |


<=30

31…40

>40

$D_1$

| RID | income | student | credit_rating | Class: Buys_computer |
|---|---|---|---|---|
| 1 | high | no | fair | no |
| 2 | high | no | excellent | no |
| 8 | medium | no | fair | no |
| 9 | low | yes | fair | yes |
| 11 | medium | yes | excellent | yes |

$D_2$

| RID | income | student | credit_rating | Class: Buys_computer |
|---|---|---|---|---|
| 3 | high | no | fair | yes |
| 7 | low | yes | excellent | yes |
| 12 | medium | no | excellent | yes |
| 13 | high | yes | fair | yes |

$D_3$

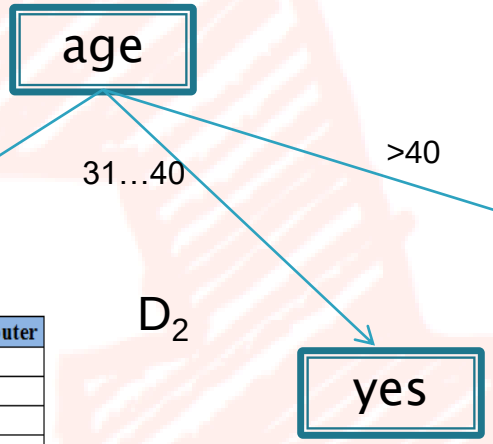
| RID | income | student | credit_rating | Class: Buys_computer |
|---|---|---|---|---|
| 4 | medium | no | fair | yes |
| 5 | low | yes | fair | yes |
| 6 | low | yes | excellent | no |
| 10 | medium | yes | fair | yes |
| 14 | medium | no | excellent | no |

# Generating decision tree

| RID | age | income | student | credit_rating | Class: Buys_computer |
|---|---|---|---|---|---|
| 1 | <=30 | high | no | fair | no |
| 2 | <=30 | high | no | excellent | no |
| 3 | 31...40 | high | no | fair | yes |
| 4 | >40 | medium | no | fair | yes |
| 5 | >40 | low | yes | fair | yes |
| 6 | >40 | low | yes | excellent | no |
| 7 | 31…40 | low | yes | excellent | yes |
| 8 | <=30 | medium | no | fair | no |
| 9 | <=30 | low | yes | fair | yes |
| 10 | >40 | medium | yes | fair | yes |
| 11 | <=30 | medium | yes | excellent | yes |
| 12 | 31…40 | medium | no | excellent | yes |
| 13 | 31…40 | high | yes | fair | yes |
| 14 | >40 | medium | no | excellent | no |

-All class values in D2 belong to "yes".

-Replace leaf node by "yes".




$D_1$

| RID | income | student | credit_rating | Class: Buys_computer |
|---|---|---|---|---|
| 1 | high | no | fair | no |
| 2 | high | no | excellent | no |
| 8 | medium | no | fair | no |
| 9 | low | yes | fair | yes |
| 11 | medium | yes | excellent | yes |

$D_2$

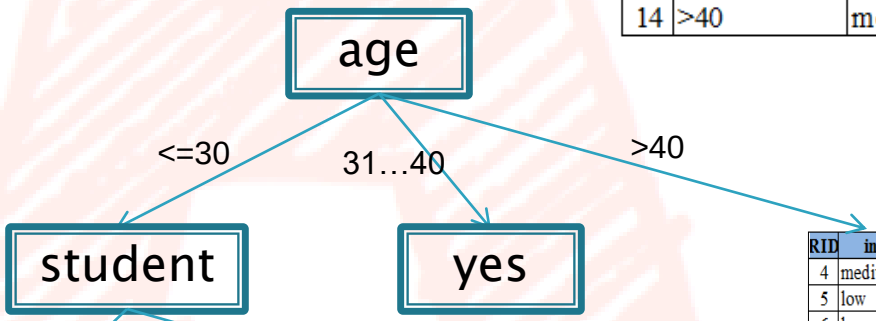
$D_3$

| RID | income | student | credit_rating | Class: Buys_computer |
|---|---|---|---|---|
| 4 | medium | no | fair | yes |
| 5 | low | yes | fair | yes |
| 6 | low | yes | excellent | no |
| 10 | medium | yes | fair | yes |
| 14 | medium | no | excellent | no |

# Generating decision tree

-Suppose "**student**" is selected
-Split D1 into D11, D12

| RID | age | income | student | credit_rating | Class: Buys_computer |
|---|---|---|---|---|---|
| 1 | <=30 | high | no | fair | no |
| 2 | <=30 | high | no | excellent | no |
| 3 | 31…40 | high | no | fair | yes |
| 4 | >40 | medium | no | fair | yes |
| 5 | >40 | low | yes | fair | yes |
| 6 | >40 | low | yes | excellent | no |
| 7 | 31…40 | low | yes | excellent | yes |
| 8 | <=30 | medium | no | fair | no |
| 9 | <=30 | low | yes | fair | yes |
| 10 | >40 | medium | yes | fair | yes |
| 11 | <=30 | medium | yes | excellent | yes |
| 12 | 31…40 | medium | no | excellent | yes |
| 13 | 31…40 | high | yes | fair | yes |
| 14 | >40 | medium | no | excellent | no |




$D_3$

| RID | income | student | credit_rating | Class: Buys_computer |
|---|---|---|---|---|
| 4 | medium | no | fair | yes |
| 5 | low | yes | fair | yes |
| 6 | low | yes | excellent | no |
| 10 | medium | yes | fair | yes |
| 14 | medium | no | excellent | no |

$D_{11}$

| RID | income | credit_rating | Class: Buys_computer |
|---|---|---|---|
| 1 | high | fair | no |
| 2 | high | excellent | no |
| 8 | medium | fair | no |

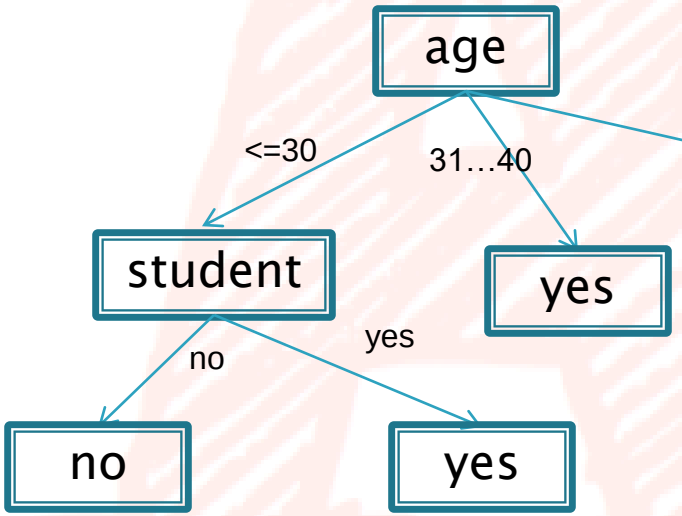
$D_{12}$

| RID | income | credit_rating | Class: Buys_computer |
|---|---|---|---|
| 9 | low | fair | yes |
| 11 | medium | excellent | yes |

# Generating decision tree (cont.)

- $D_{11}$: all tuples belong to class "no" ⇒ leaf node "no"
- $D_{12}$ : all tuples belong to class "yes" ⇒ leaf node "yes"

| RID | age | income | student | credit_rating | Class: Buys_computer |
|---|---|---|---|---|---|
| 1 | <=30 | high | no | fair | no |
| 2 | <=30 | high | no | excellent | no |
| 3 | 31…40 | high | no | fair | yes |
| 4 | >40 | medium | no | fair | yes |
| 5 | >40 | low | yes | fair | yes |
| 6 | >40 | low | yes | excellent | no |
| 7 | 31…40 | low | yes | excellent | yes |
| 8 | <=30 | medium | no | fair | no |
| 9 | <=30 | low | yes | fair | yes |
| 10 | >40 | medium | yes | fair | yes |
| 11 | <=30 | medium | yes | excellent | yes |
| 12 | 31…40 | medium | no | excellent | yes |
| 13 | 31…40 | high | yes | fair | yes |
| 14 | >40 | medium | no | excellent | no |




$D_3$

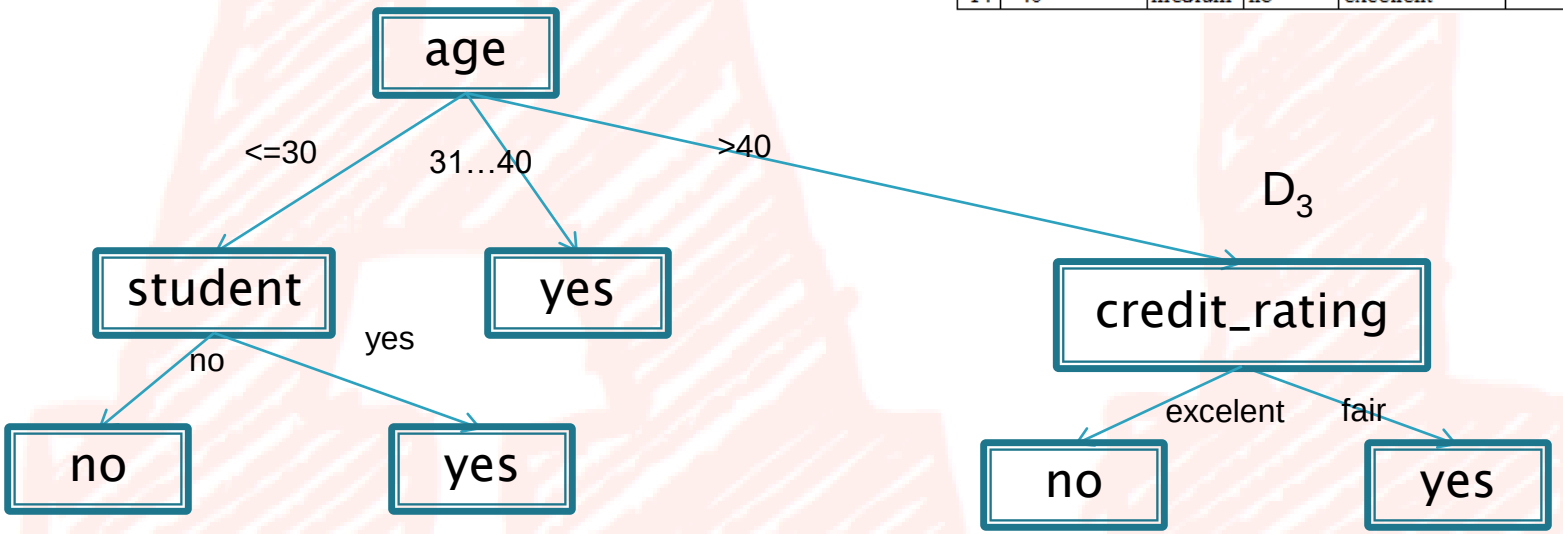
| RID | income | student | credit_rating | Class: Buys_computer |
|---|---|---|---|---|
| 4 | medium | no | fair | yes |
| 5 | low | yes | fair | yes |
| 6 | low | yes | excellent | no |
| 10 | medium | yes | fair | yes |
| 14 | medium | no | excellent | no |

• For D3, suppose credit_rating is selected for splitting. Generate a tree for D3?

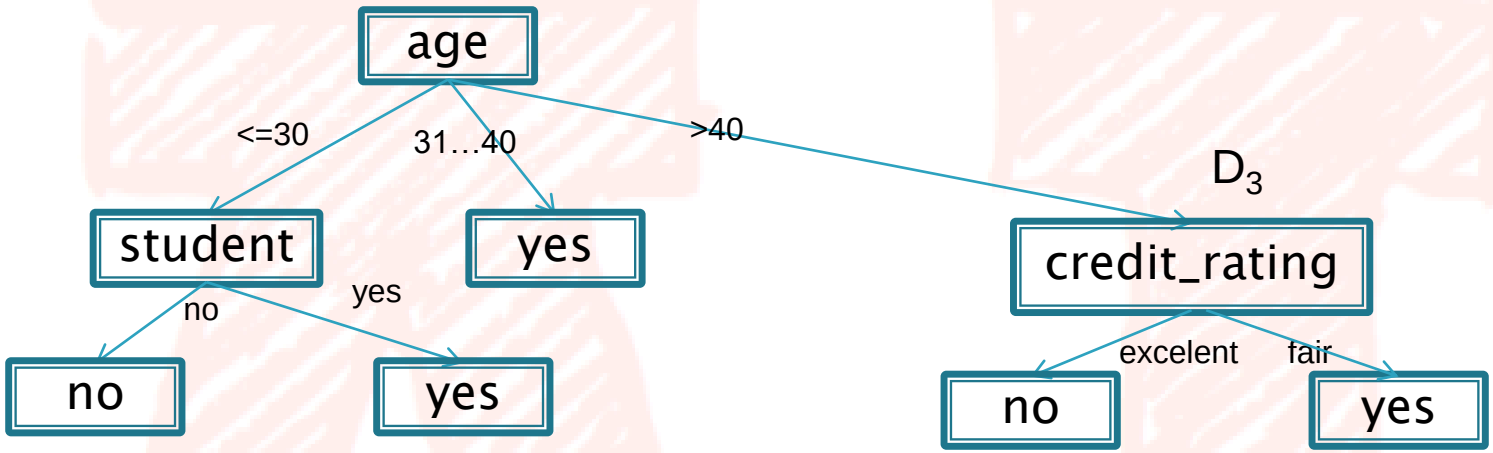
# Generating decision tree (cont.)

- $D_{31}$: all values of credit_rating are excelent belonging to “no” ⇒ leaf node “no”
- $D_{32}$: all values of credit_rating are excelent belonging to “yes” ⇒ leaf node “yes”

| RID | age | income | student | credit_rating | Class: Buys_computer |
|---|---|---|---|---|---|
| 1 | <=30 | high | no | fair | no |
| 2 | <=30 | high | no | excellent | no |
| 3 | 31…40 | high | no | fair | yes |
| 4 | >40 | medium | no | fair | yes |
| 5 | >40 | low | yes | fair | yes |
| 6 | >40 | low | yes | excellent | no |
| 7 | 31…40 | low | yes | excellent | yes |
| 8 | <=30 | medium | no | fair | no |
| 9 | <=30 | low | yes | fair | yes |
| 10 | >40 | medium | yes | fair | yes |
| 11 | <=30 | medium | yes | excellent | yes |
| 12 | 31…40 | medium | no | excellent | yes |
| 13 | 31…40 | high | yes | fair | yes |
| 14 | >40 | medium | no | excellent | no |

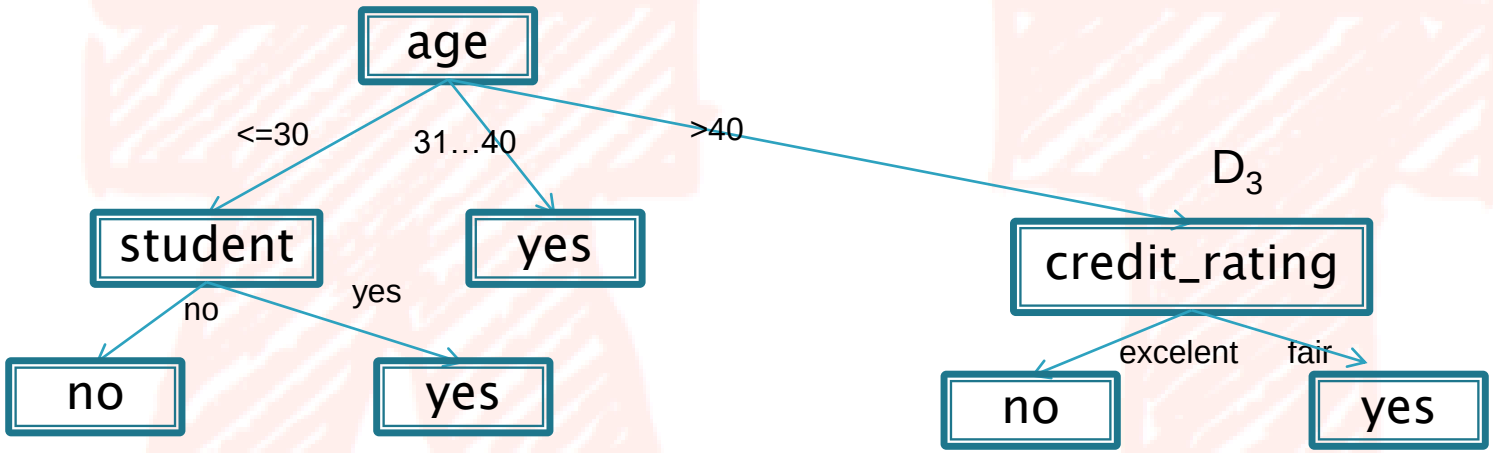


# Extract Rules from Decision Tree




- If age<=30 and student=no, then buy_computer=no
- If age<=30 and student=yes, then buy_computer=yes
- If age=31…40, then buy_computer = Yes
- If age>40 and credit-rating=excelent, then buy_computer=no
- If age>40 and credit-rating=fair, then buy_computer=no

# Decision Tree: model usage

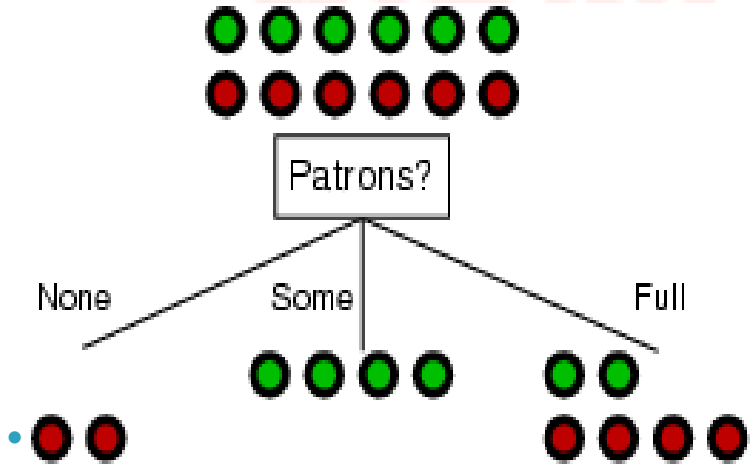


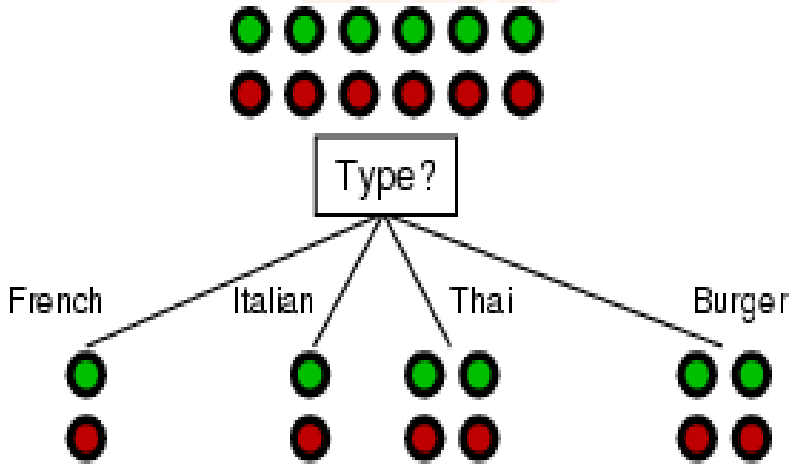

- X=(age<=30, student=yes, credit_rating=fair)
- ➔ yes
- Y=(age>40, student=no, credit_rating=excelent)
- ➔ no

# Attribute Selection

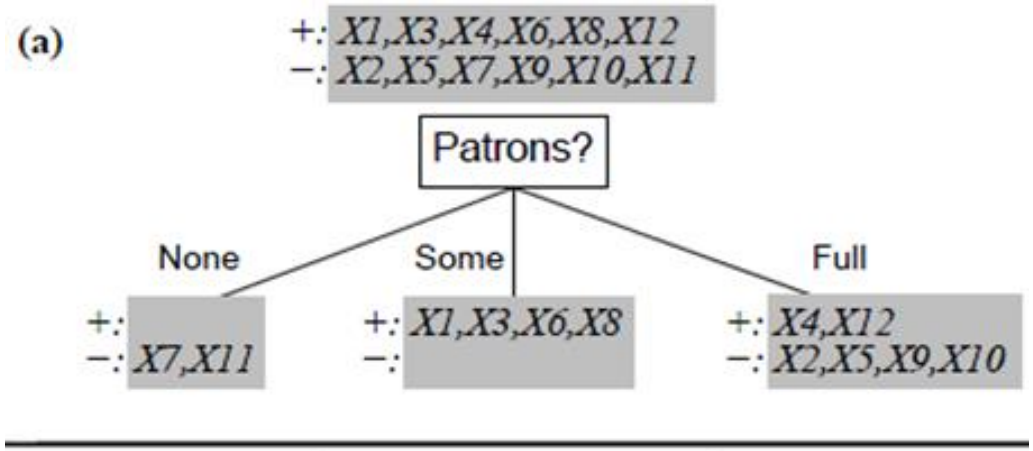
# Choosing an attribute

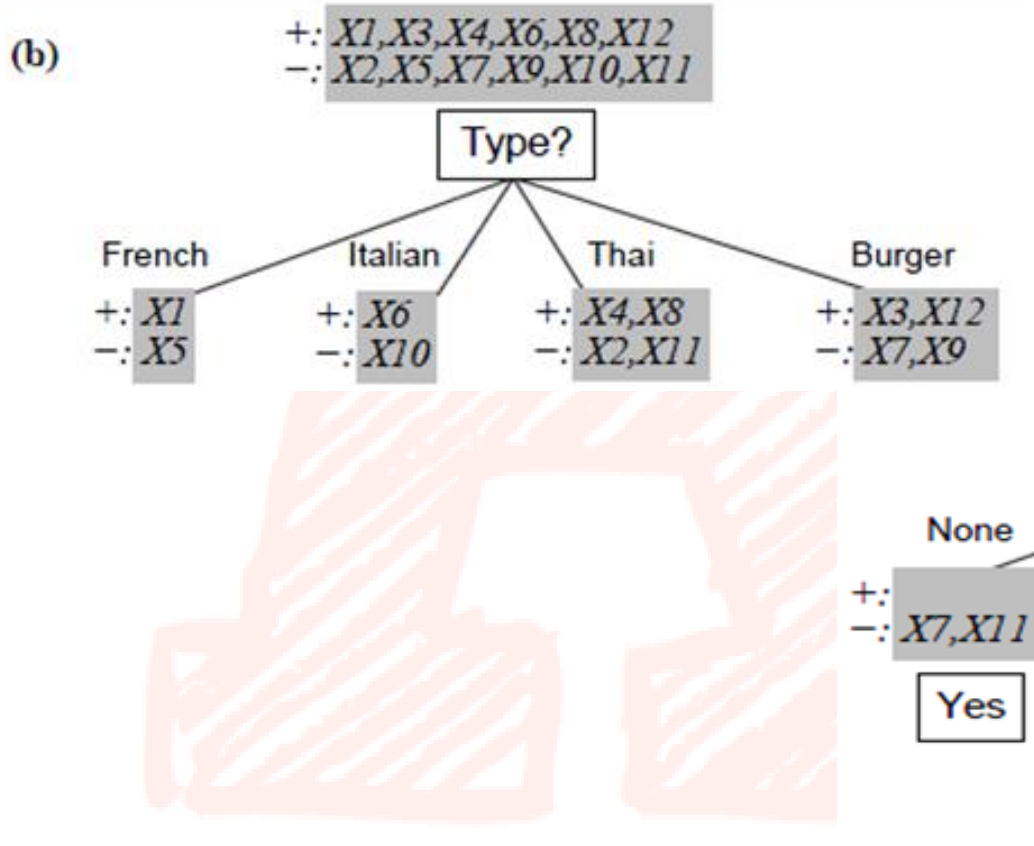
- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

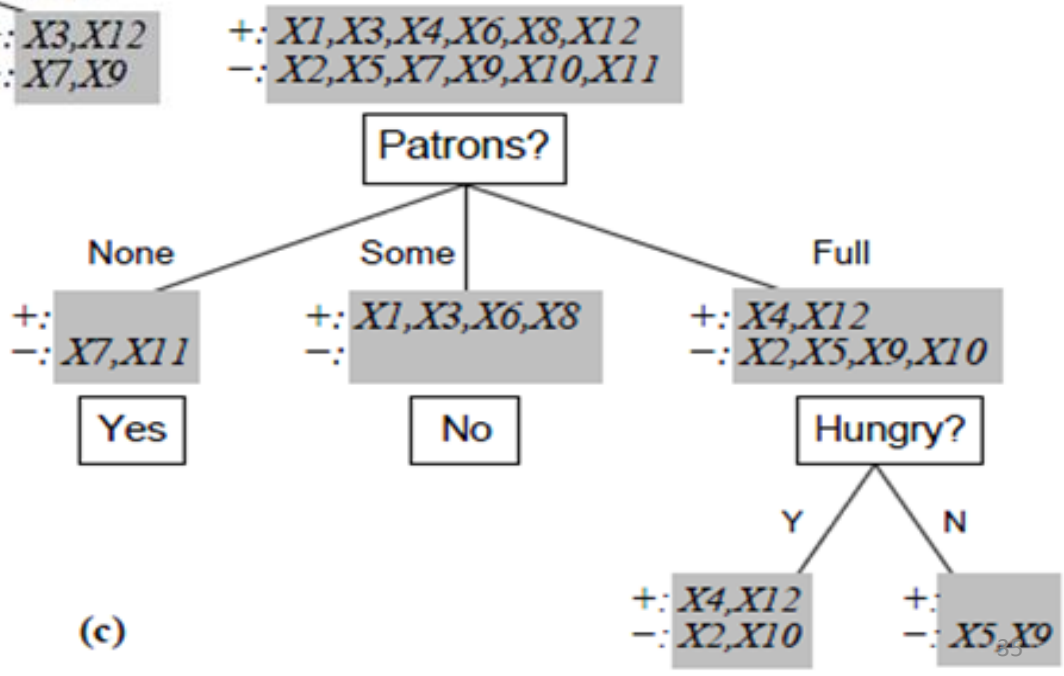



Patrons? is a better choice - gives information about the classification


(a)
+: X1,X3,X4,X6,X8,X12
−: X2,X5,X7,X9,X10,X11
Patrons?
None
Some
Full
+:
−: X7,X11
+: X1,X3,X6,X8
−:
+: X4,X12
−: X2,X5,X9,X10


(b)
+: X1,X3,X4,X6,X8,X12
−: X2,X5,X7,X9,X10,X11
Type?
French
Italian
Thai
Burger
+: X1
−: X5
+: X6
−: X10
+: X4,X8
−: X2,X11
+: X3,X12
−: X7,X9


+: X1,X3,X4,X6,X8,X12
−: X2,X5,X7,X9,X10,X11
Patrons?
None
Some
Full
+:
−: X7,X11
+: X1,X3,X6,X8
−:
+: X4,X12
−: X2,X5,X9,X10
Yes
No
Hungry?
Y
N
+: X4,X12
−: X2,X10
+:
−: X5,X9
(c)

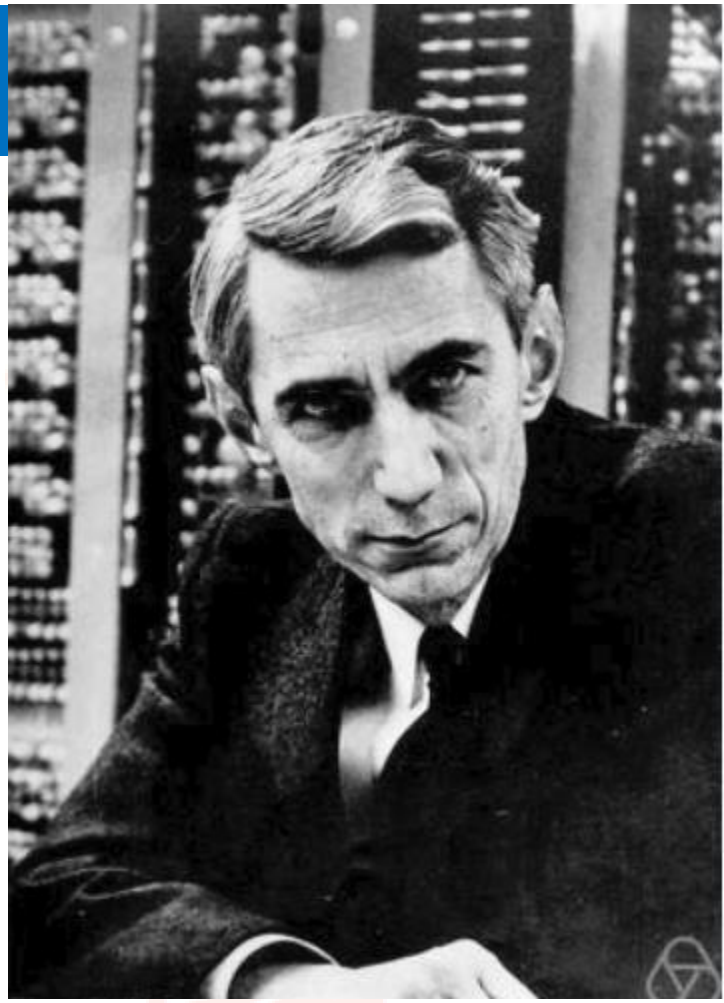
# Choosing an attribute

- Finding the smallest decision tree turns out to be intractable.
- However, there are simple heuristics that do a good job of finding small trees.
- Basic question is: Which attribute do we split on next?
- Idea: Using information theory
  - Define a statistical property, called information gain, to measure how good a feature is at separating the data according to the target.

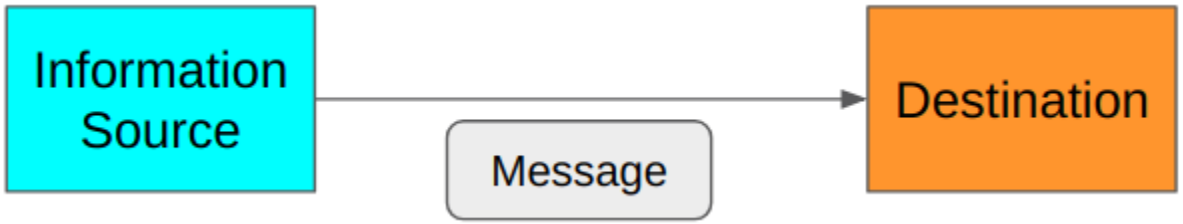
# Entropy

# Who Invented Entropy?

- **Claude Shannon** introduced information entropy in paper "A Mathematical Theory of Communication", 1948.



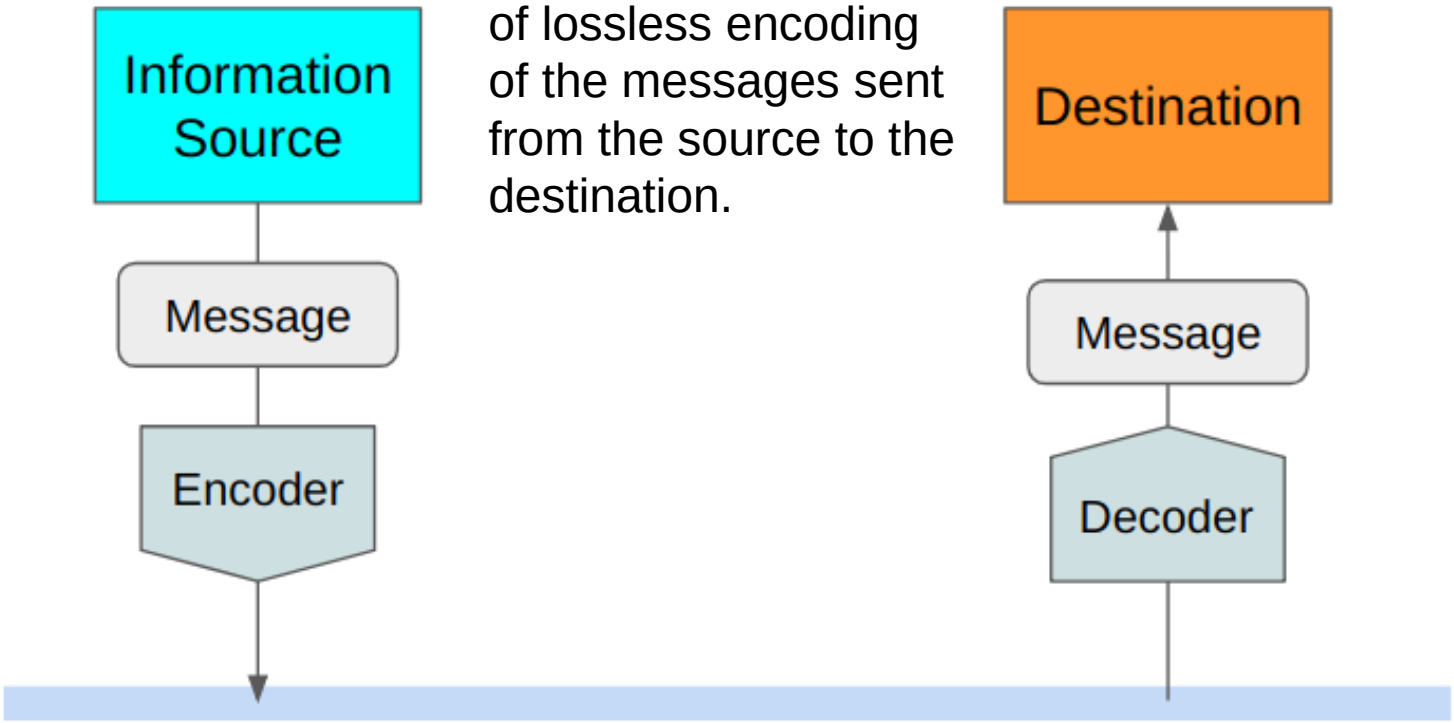
How to efficiently send messages without losing any information



# Why Entropy?

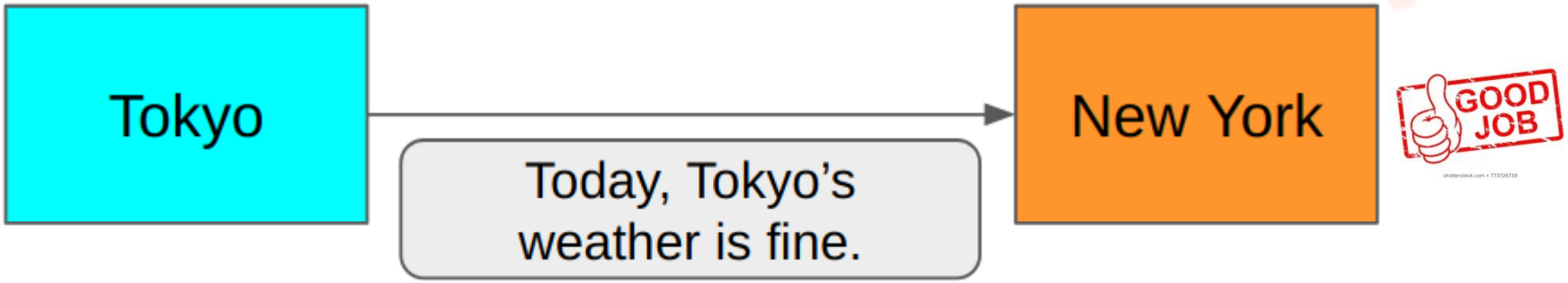
- How to *encode the original message into the smallest possible data structure* by taking into account average message length.

**Entropy**: the smallest possible average size of lossless encoding of the messages sent from the source to the destination.

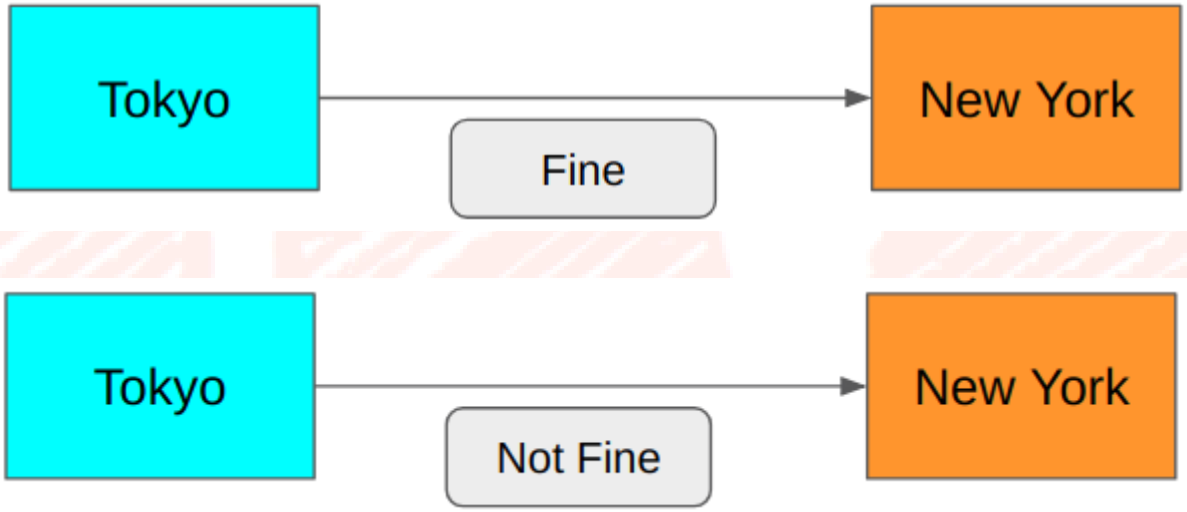


# How to Make Efficient and Lossless Encoding?

- Suppose a message sent from Tokyo to New York regarding Tokyo's weather today.

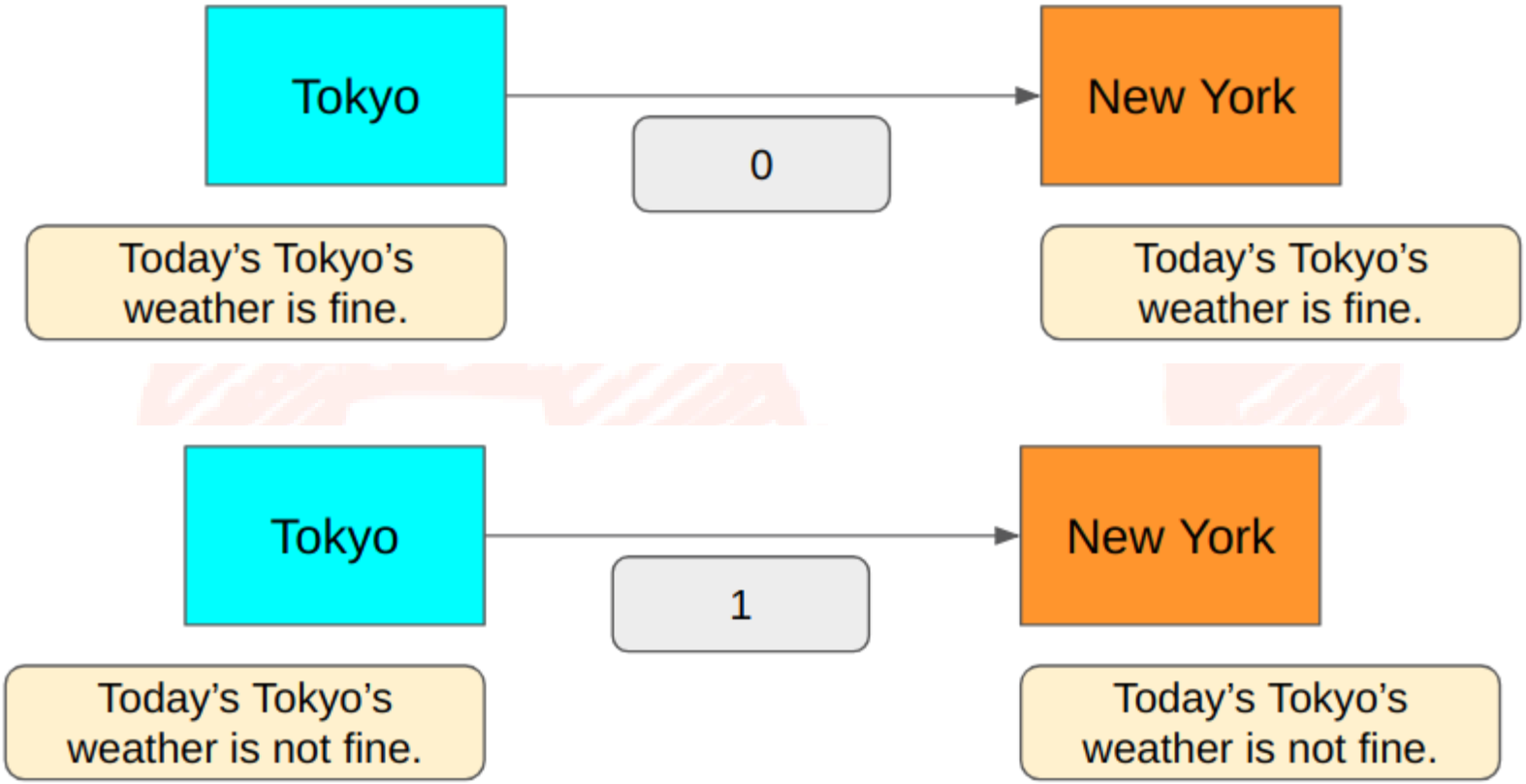



- Other approaches:



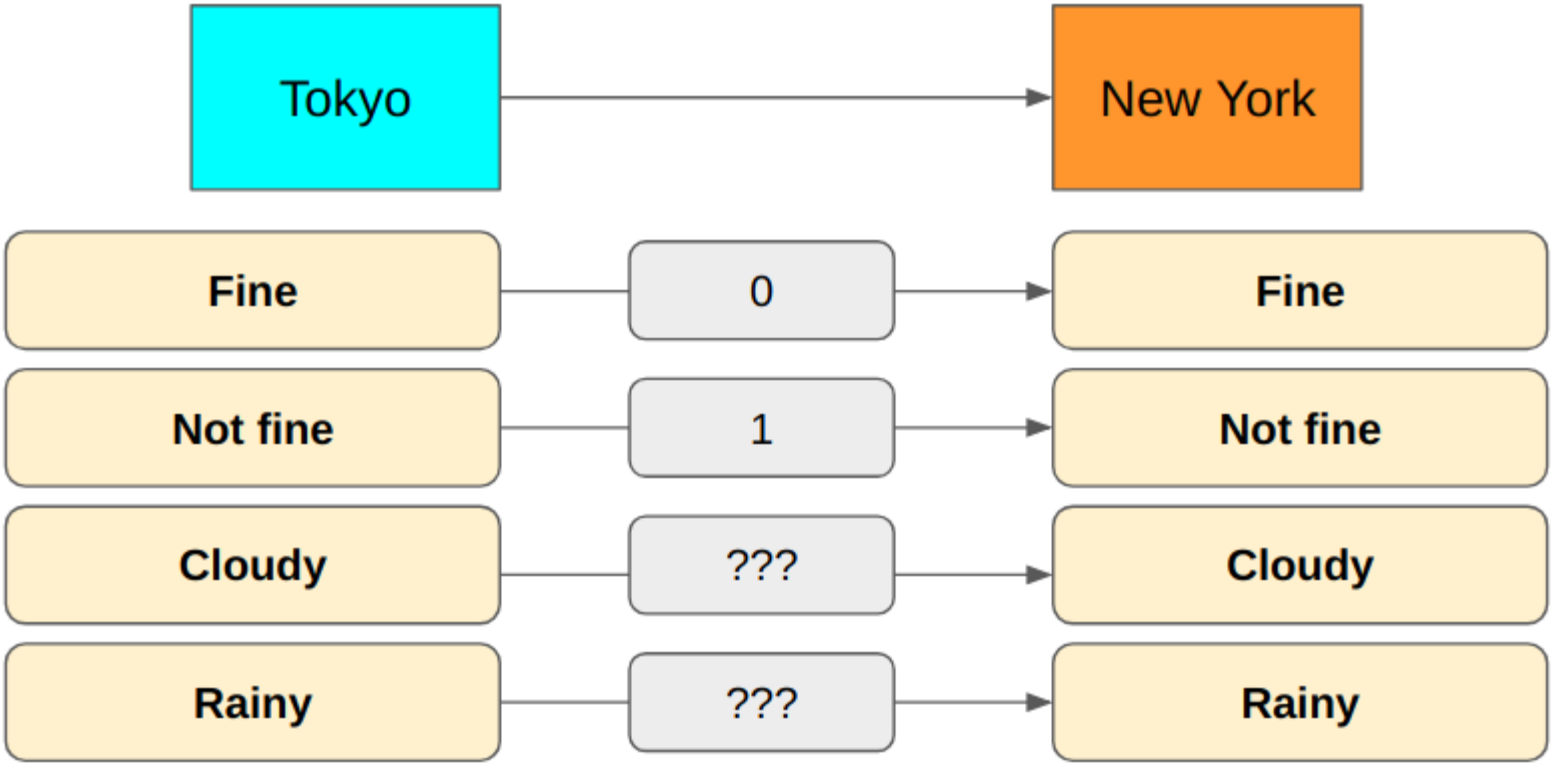
# How to Make Efficient and Lossless Encoding?

- It is much shorter. Can we do better?
- We need precisely 1 bit to encode the above messages.



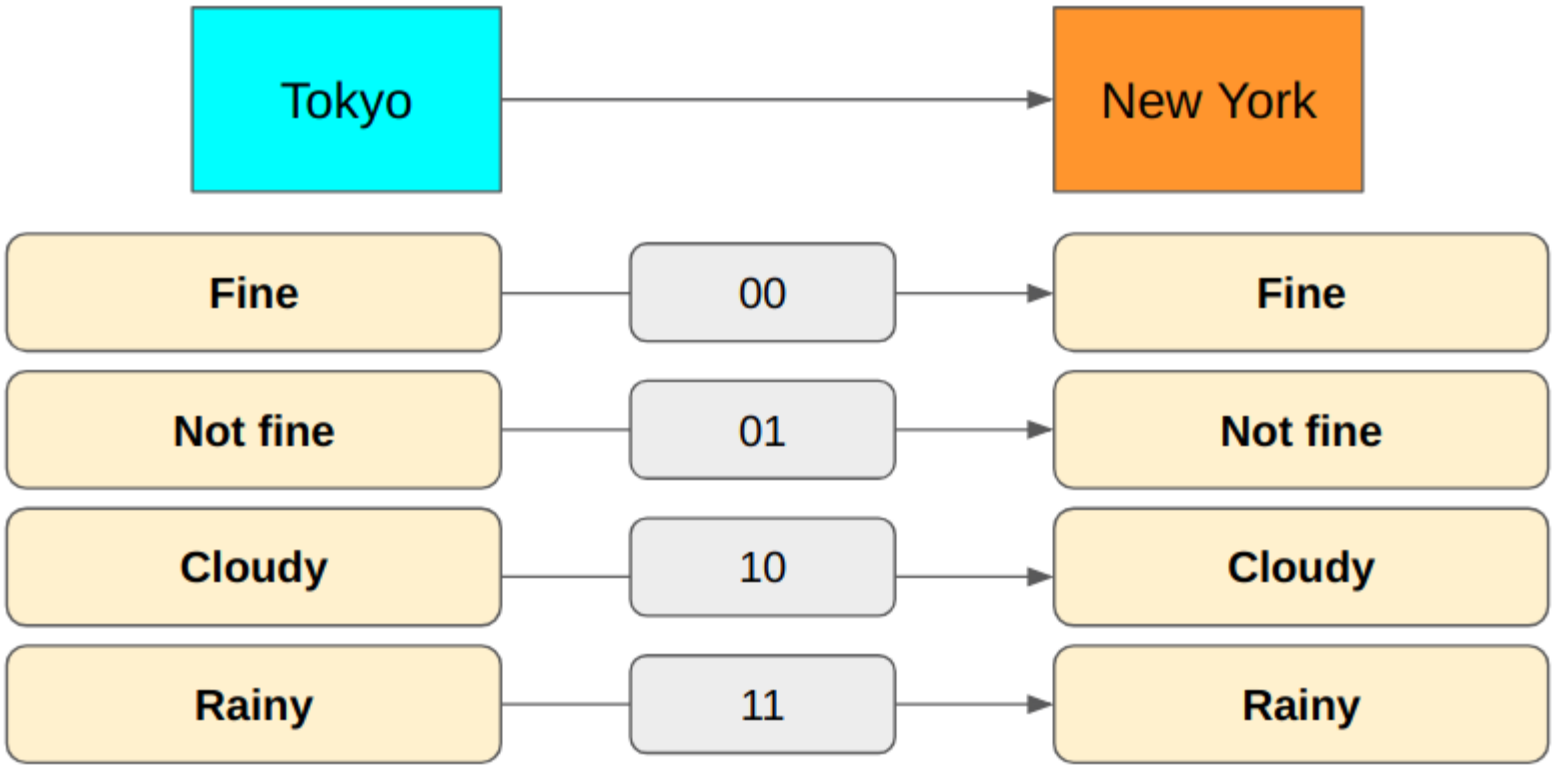
# How to Make Efficient and Lossless Encoding?

- How about “Cloudy” or “Rainy” information? 1-bit encoding won’t be able to cover all cases.



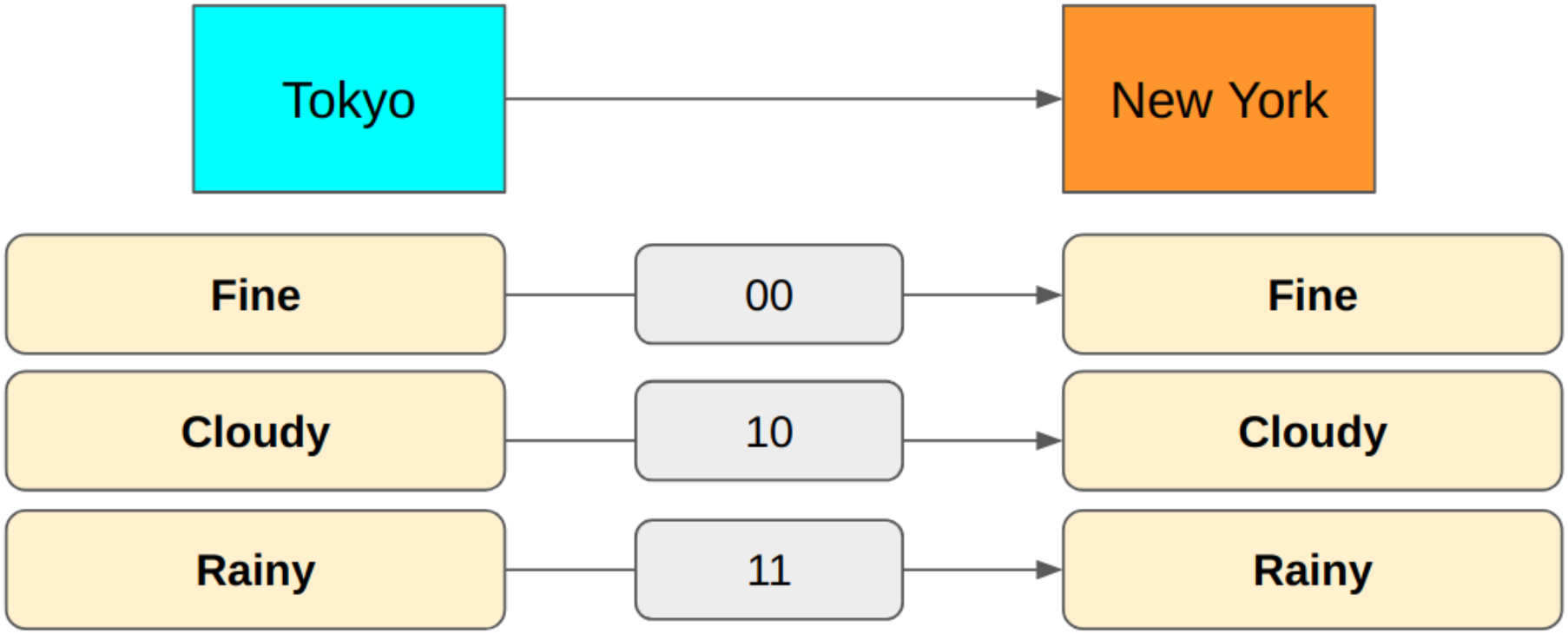
# How to Make Efficient and Lossless Encoding?

- How about using 2 bits instead?



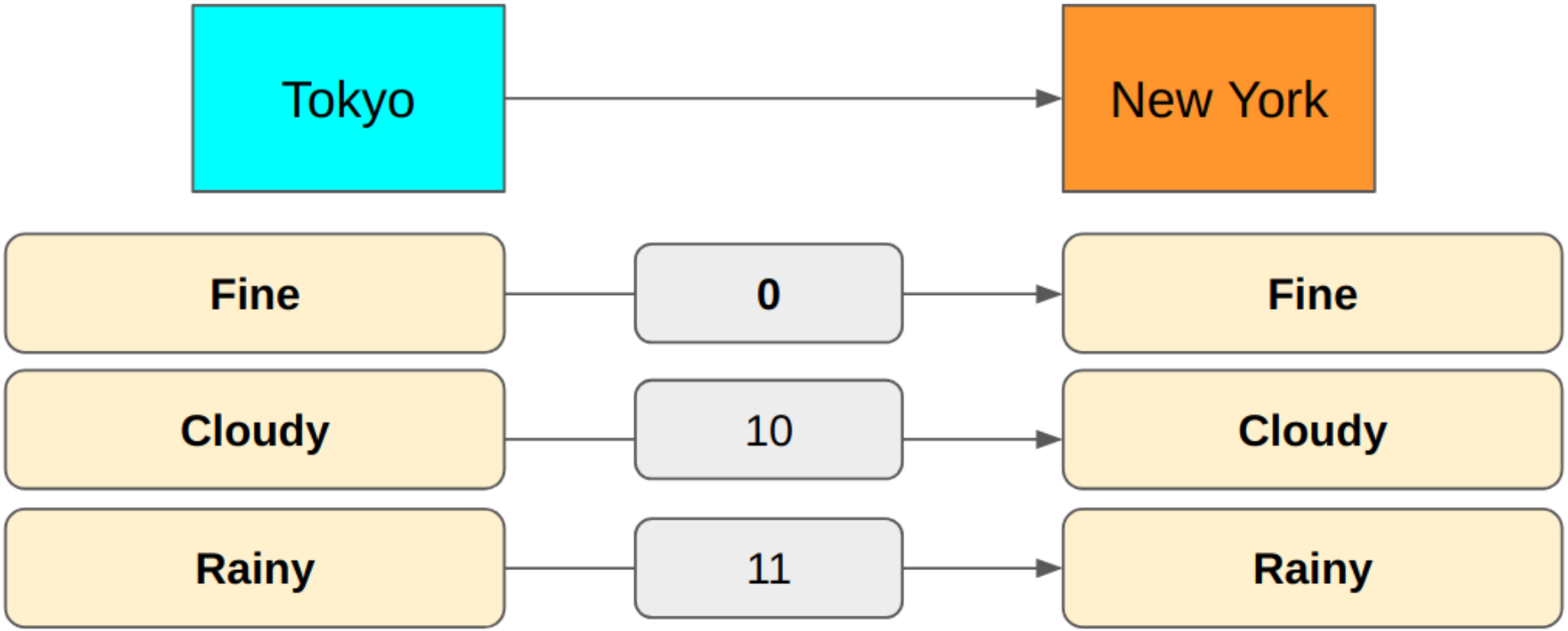
# How to Make Efficient and Lossless Encoding?

- Semantically, "Cloudy" and "Rainy" are both "Not fine" as well. Remove it!



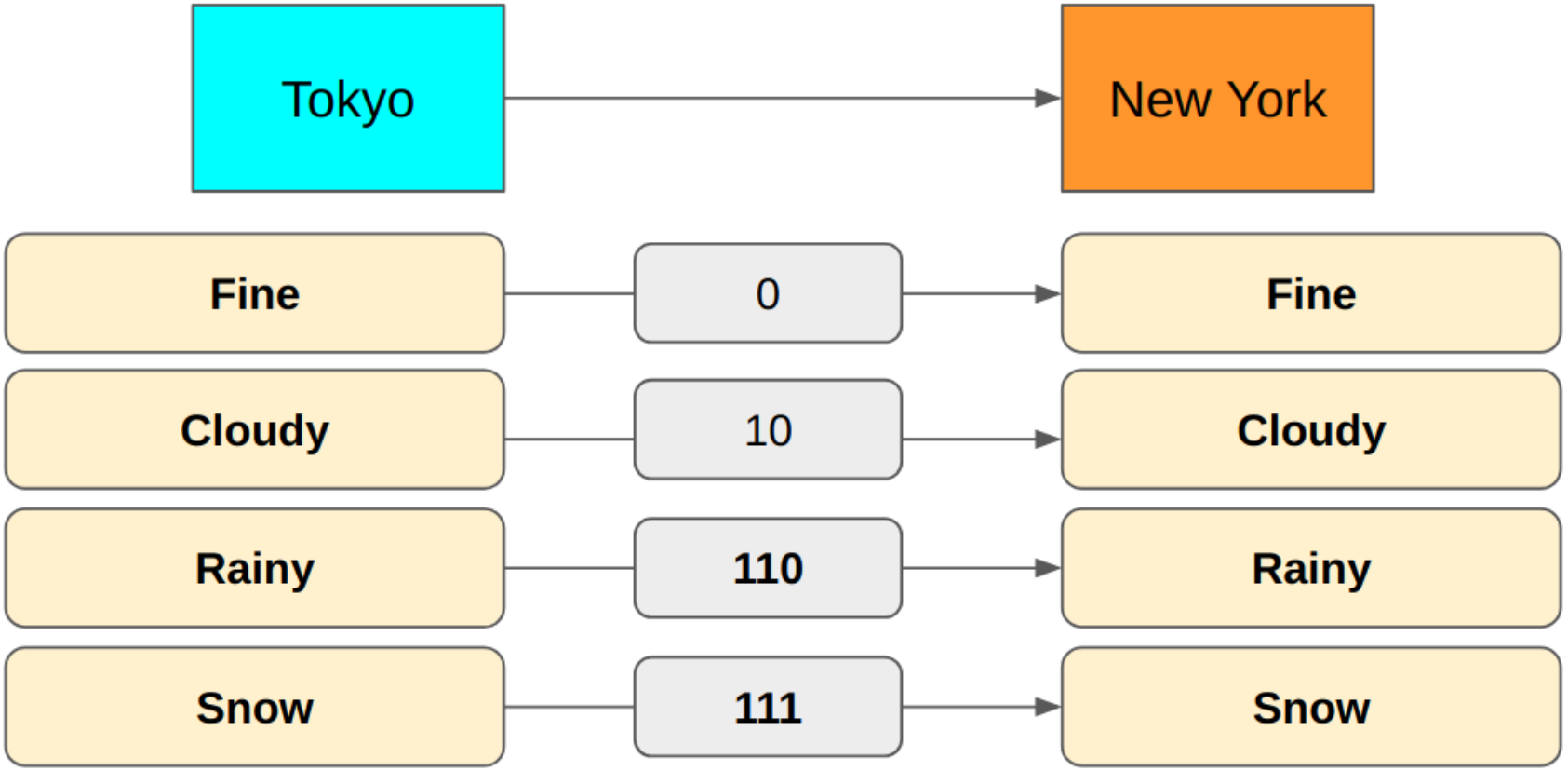
# How to Make Efficient and Lossless Encoding?

- "Fine" is "00" but we don't need the second zero



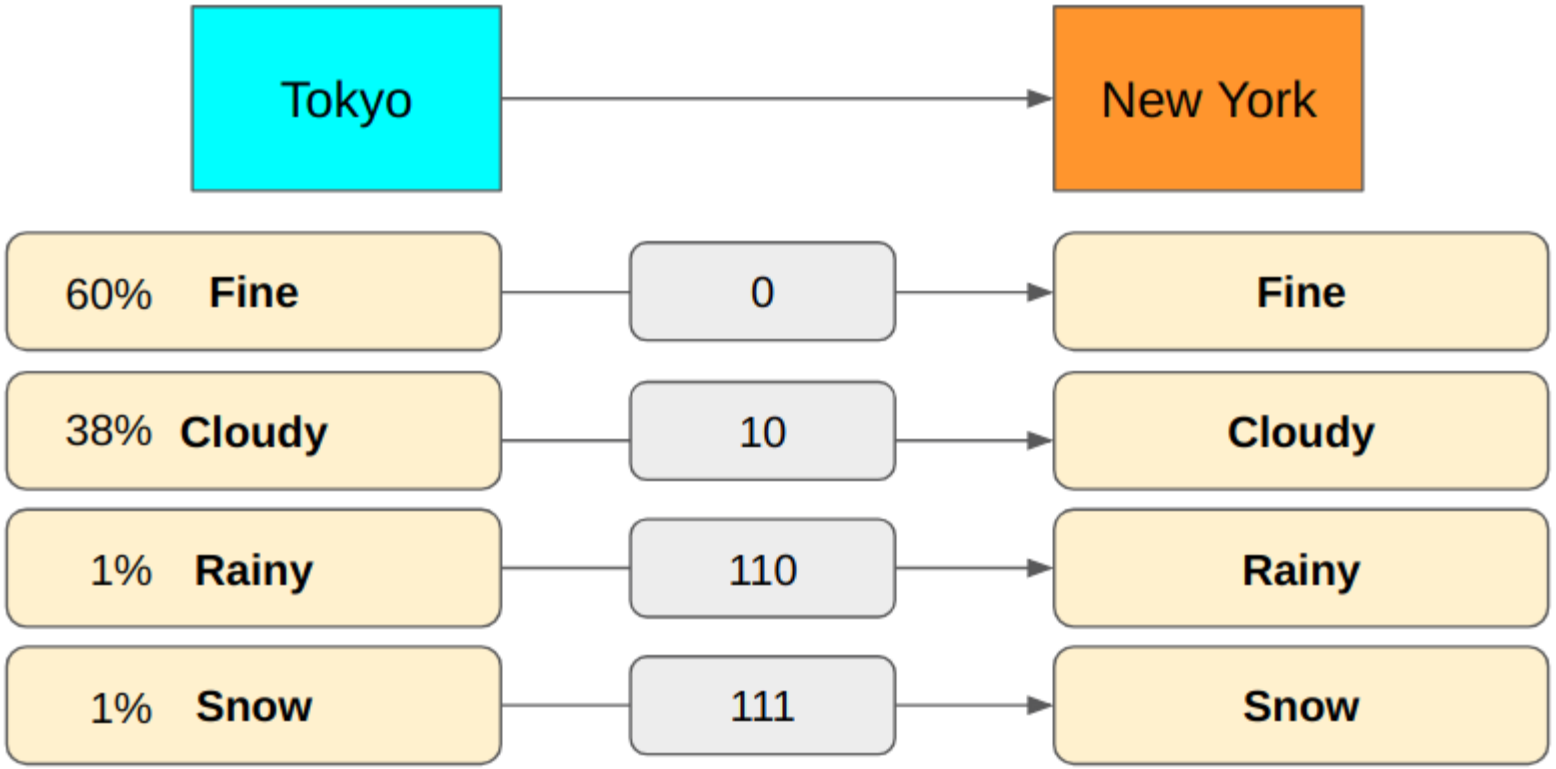
# How to Make Efficient and Lossless Encoding?

- Similarly, "Snow" information can be encoded as follows:



# How to Calculate Average Encoding Size?

- Suppose Tokyo is sending the weather messages to New York many times (every hour) using the above lossless encoding.
- The probability distribution of message types:



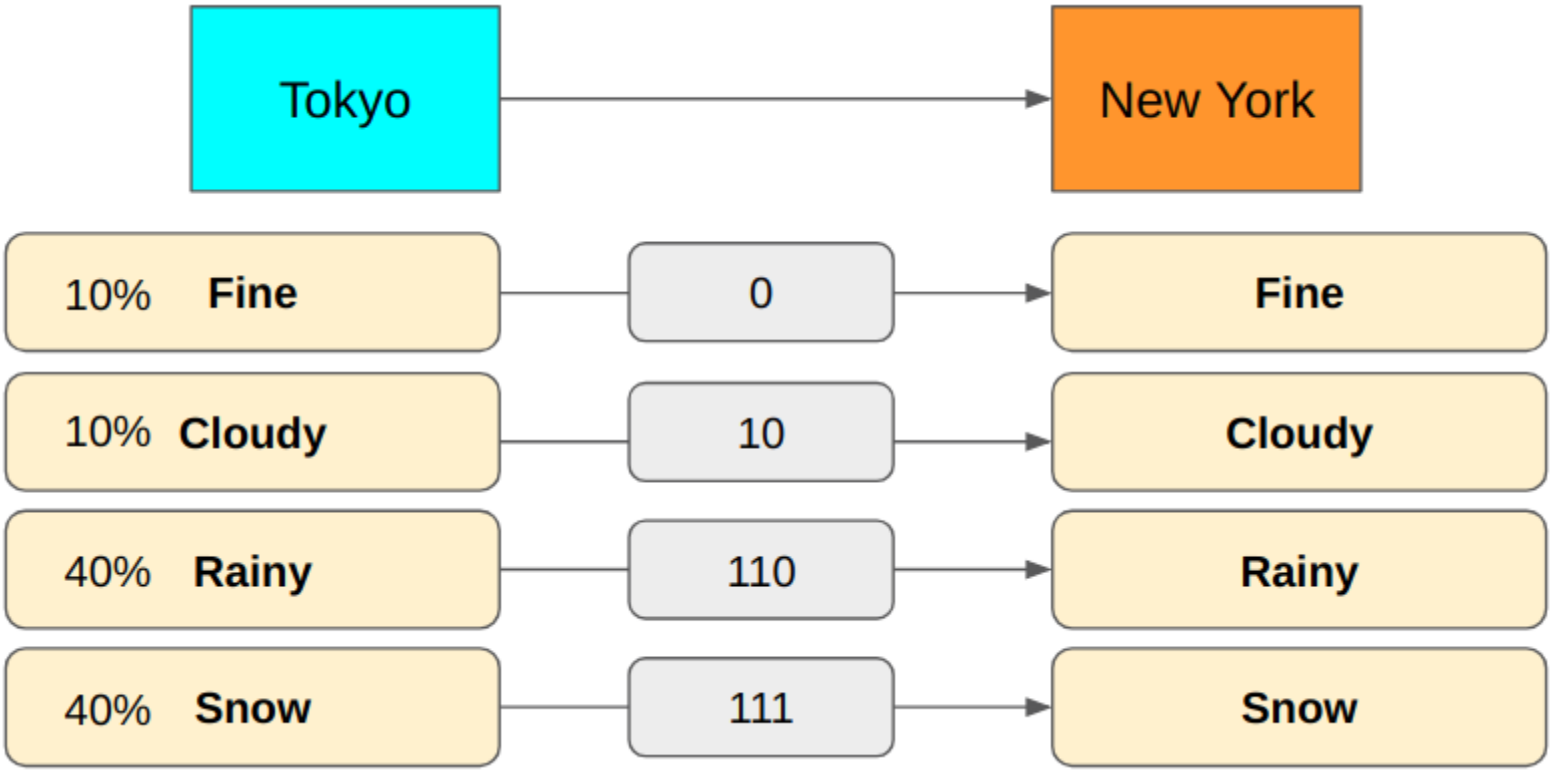

(0.6 x 1 bit)+(0.38 x 2 bits)+(0.01 x 3 bits)+(0.01 x 3 bits)=1.42 bits

# How to Calculate Average Encoding Size?

- In conclusion: The average number of bits depends on
  - the probability distribution and
  - the message encoding (how many bits we use for each message type).

# How to Calculate Average Encoding Size?

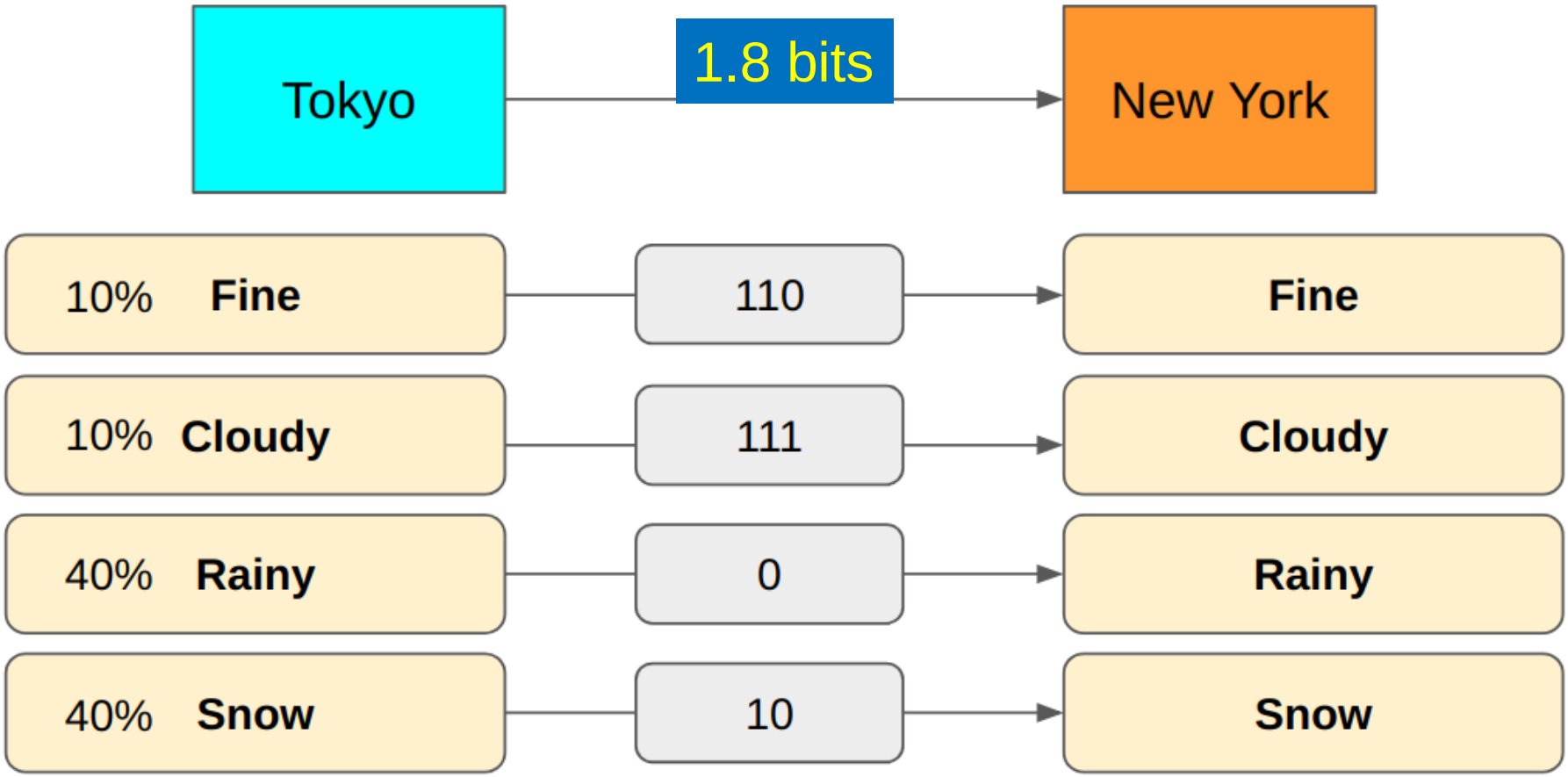
- Consider:




(0.1 x 1 bit)+(0.1 x 2 bits)+(0.4 x 3 bits)+(0.4 x 3 bits)=**2.7 bits**

# How to Calculate Average Encoding Size?

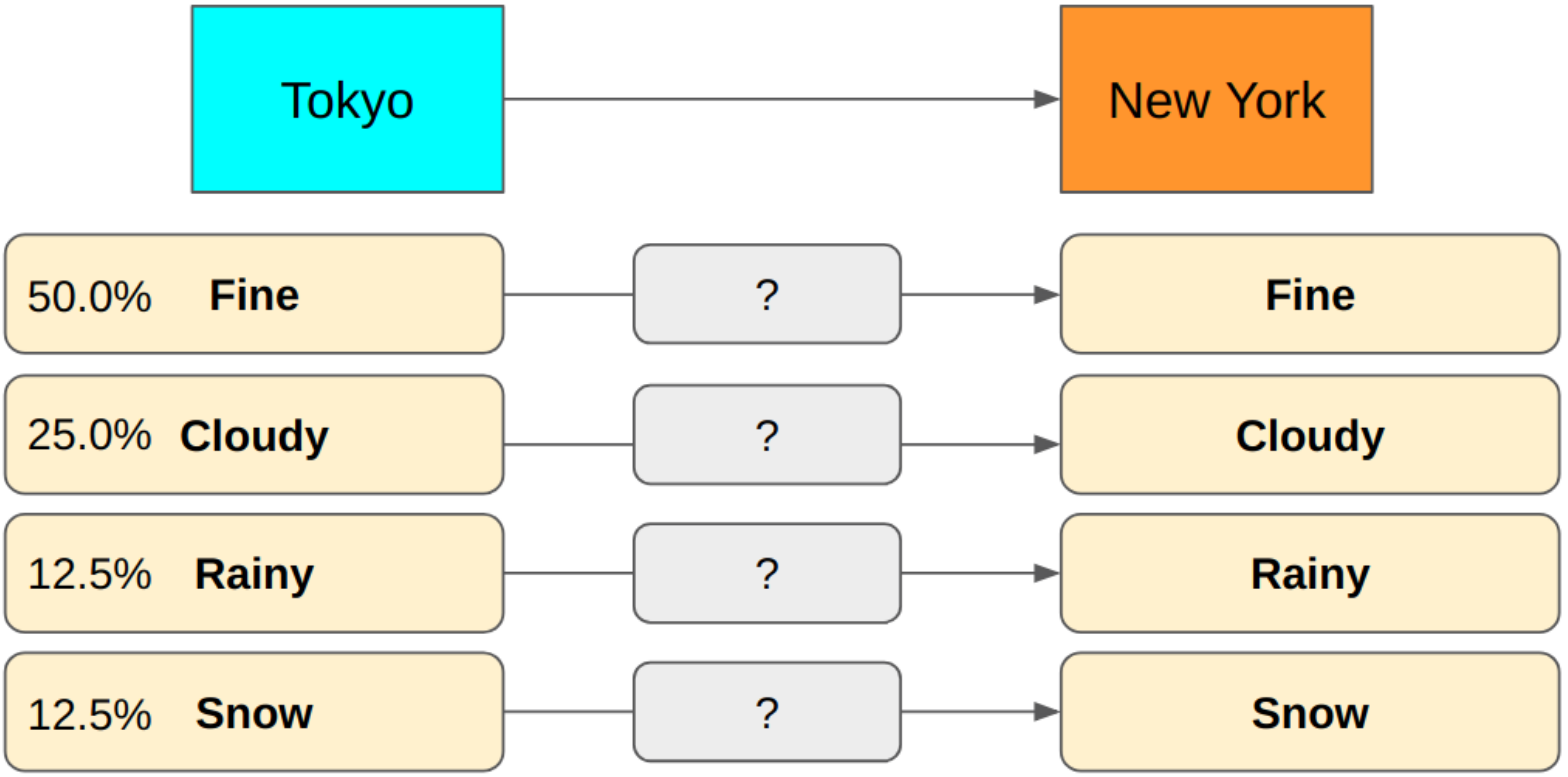
- Reduce the average encoding size by changing the encoding to use fewer bits for "Rainy" and "Snow"



# Finding the Smallest Average Encoding Size?

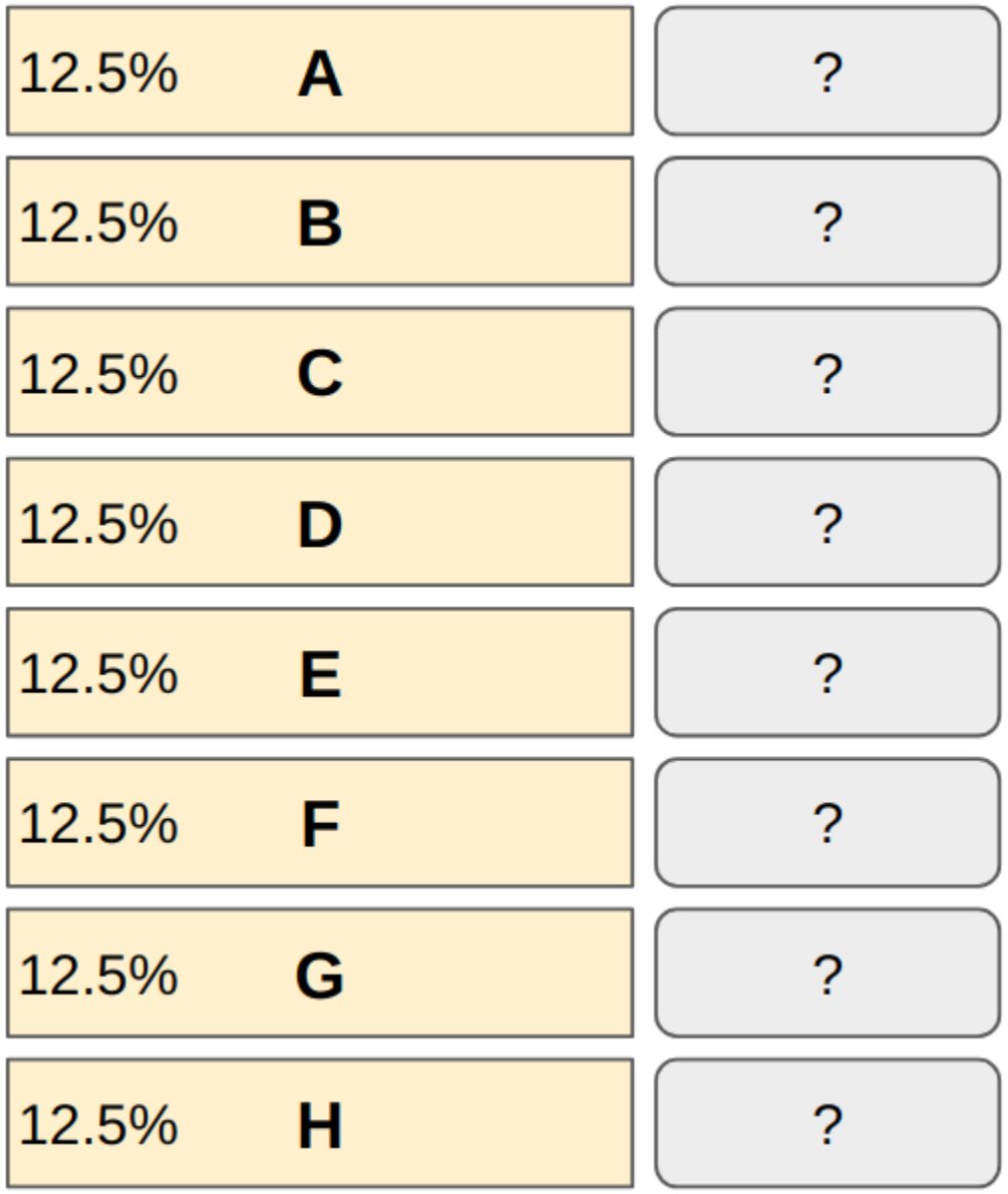
- Try to encode and check?
- Calculate entropy?

**Entropy**: the smallest possible average size of lossless encoding of the messages sent from the source to the destination.



# How to Calculate Entropy?

- Suppose we have 8 message types, each of which happens with equal probability (⅛ = 12.5%).



# How to Calculate Entropy?

- 3 bits to encode 8 message types

1 bit: 2 values (0, 1)

2 bits: 2x2 = 4 values (00, 01, 10, and 11)

3 bits: 2x2x2 = 8 values (000, 001, 010, 011, 100, 101, 110 and 111).

| | | |
|---|---|---|
| 12.5% | A | 000 |
| 12.5% | B | 001 |
| 12.5% | C | 010 |
| 12.5% | D | 011 |
| 12.5% | E | 100 |
| 12.5% | F | 101 |
| 12.5% | G | 110 |
| 12.5% | H | 111 |

# How to Calculate Entropy?

- In general, when we need N different values expressed in bits, we need

$$\log_2 N$$

- If a message type happens 1 out of N times, the above formula gives the minimum size required.
- The probability of the message, P=1/N.

$$\log_2 N = -\log_2 1/N = -\log_2 P$$

# How to Calculate Entropy?

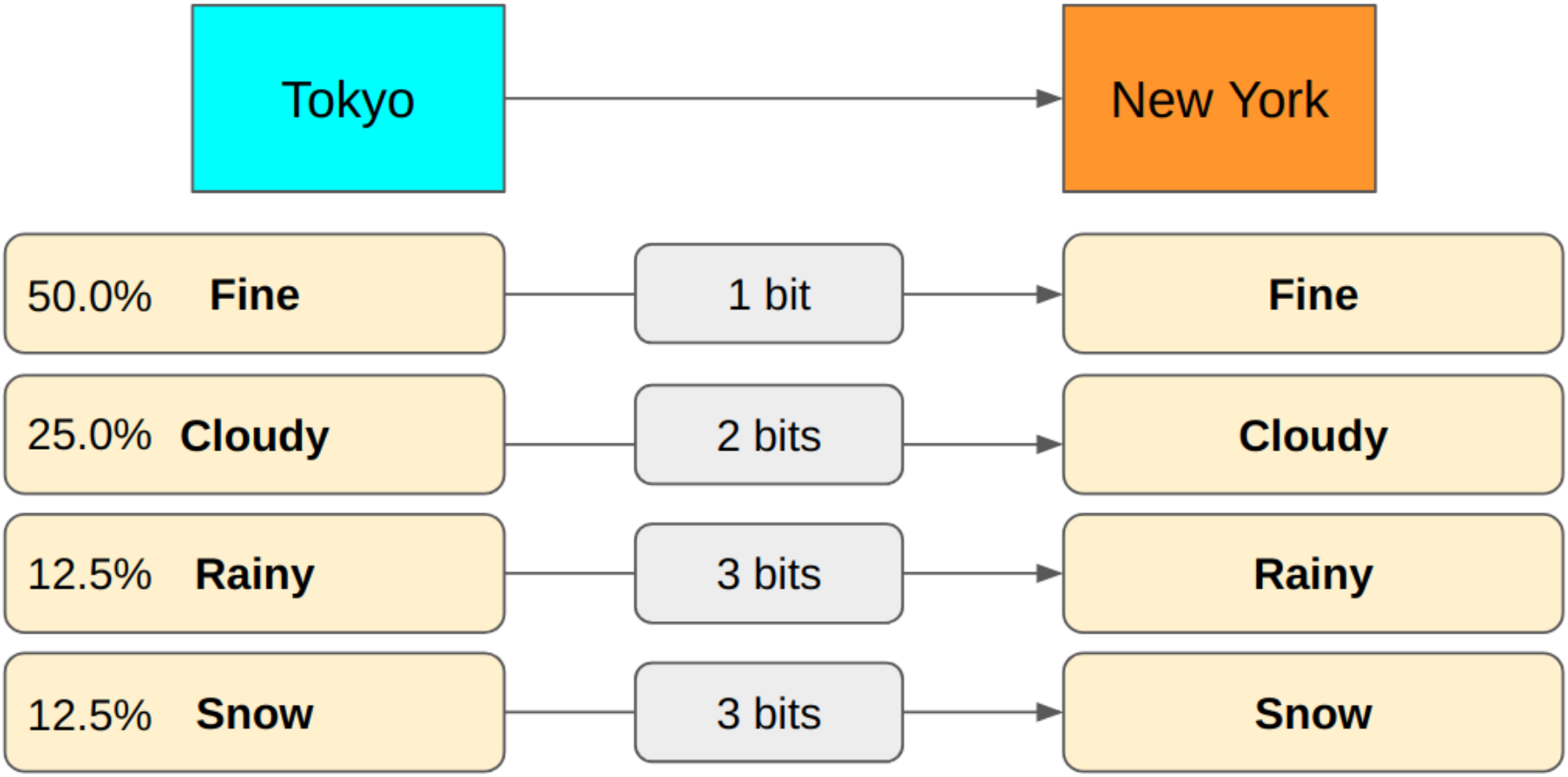
- Therefore, the minimum average encoding size (in bits) which is entropy:

$$Entropy = -\sum_{i} P(i)\, log_2 P(i)$$

- Where P(i) is the probability of i-th message type.

# How to Calculate Entropy?

- Consider:



# How to Calculate Entropy?

P(Fine) = 0.5

$-\log_2 P(\text{Fine}) = -\log_2 0.5 = -\log_2 1/2 = \log_2 2 = 1$ bit

P(Cloudy) = 0.25

$-\log_2 P(\text{Cloudy}) = -\log_2 0.25 = -\log_2 1/4 = \log_2 4 = 2$ bits

P(Rainy) = 0.125

$-\log_2 P(\text{Rainy}) = -\log_2 0.125 = -\log_2 1/8 = \log_2 8 = 3$ bits

P(Snow) = 0.125

$-\log_2 P(\text{Snow}) = -\log_2 0.125 = -\log_2 1/8 = \log_2 8 = 3$ bits

(0.5 x 1 bit)+(0.25 x 2 bits)+(0.125 x 3 bits)+(0.125 x 3 bits)=1.75 bits

# Properties of Entropy

- Entropy: disorder, uncertainty, surprise, unpredictability, amount of information, etc.
- If entropy is high (encoding size is big on average), then we have many message types with small probabilities.
- Low entropy means that most of the times we are receiving the more predictable information

# Information Gain

# Information theory – Entropy

- For a training set D containing *p* positive and *n* negative examples
  - For 12 restaurant examples, wait p = 6, nowait n = 6

$$Entropy = -\sum_i P(i)\log_2 P(i)$$

$$Entropy(D) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

$$= -\frac{6}{12}\log_2\frac{6}{12} - \frac{6}{12}\log_2\frac{6}{12} = -\log_2\frac{1}{2} = 1$$

# Information gain

- A chosen attribute A divides the training set D into subsets $D_1, \ldots, D_v$ according to their values for A, where A has v distinct values.

- After testing attribute A, we need a remainder of bits to classify the samples.

$$Remainder(A) = \sum_{i=1}^{v} \frac{|D_i|}{|D|} Entropy(D_i)$$

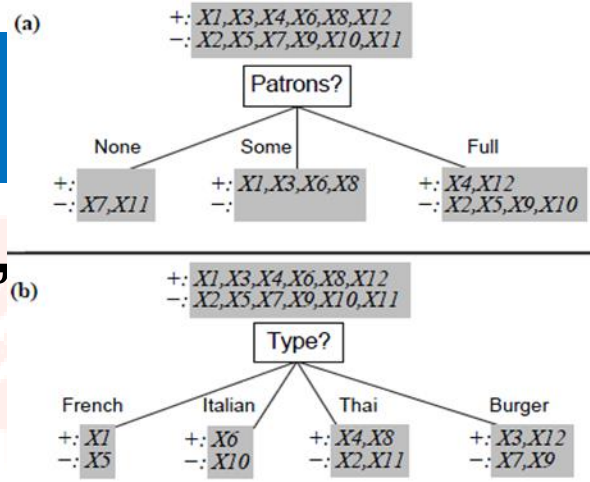
# Information gain

- Information Gain (IG) or reduction in entropy from the attribute test:

$$IG(A) = Entropy(D) - Remainder(A)$$

- Choose the attribute with the **largest IG**

# Information gain




- For the training set D, p = n = 6, *Entropy(D) = 1 bit*
- Consider the attributes **Patrons** and **Type**:

$$Remainder(Patrons) = 0.459$$

$$IG(Patrons) = Entropy(D) - Remainer(Patrons) = 0.541 \text{ bits}$$

$$IG(Type) = 1 - 1 = 0 \text{ bits}$$

- Patrons has the highest IG of all attributes and so is chosen by the DTL algorithm as the root

# Attribute-based representations

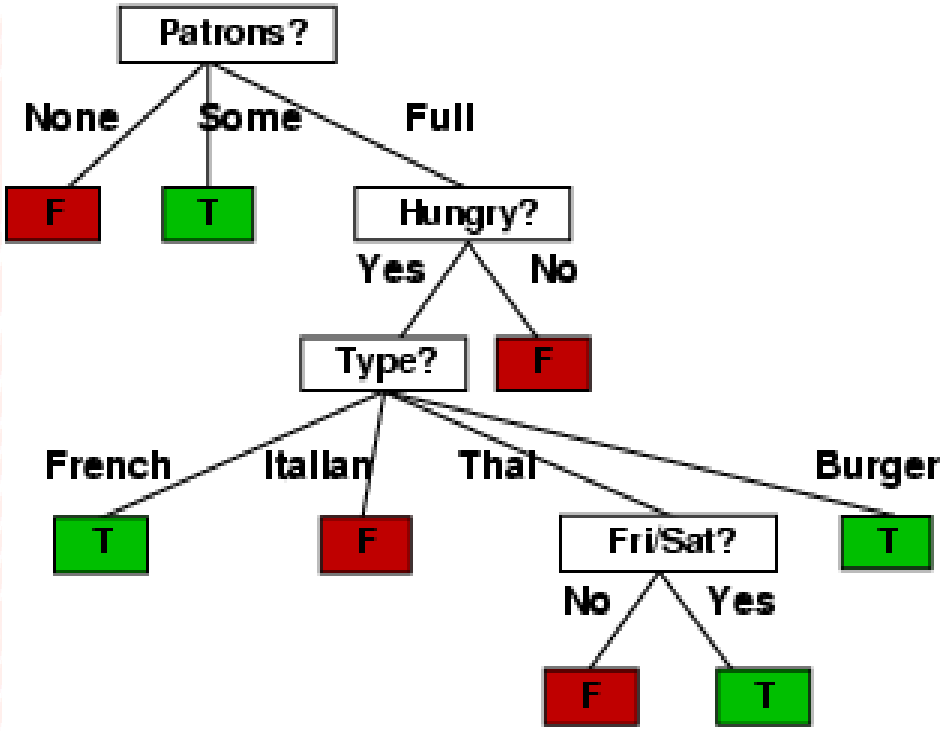
- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

| Example | Attributes | | | | | | | | | | Target |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Alt* | *Bar* | *Fri* | *Hun* | *Pat* | *Price* | *Rain* | *Res* | *Type* | *Est* | *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 |

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

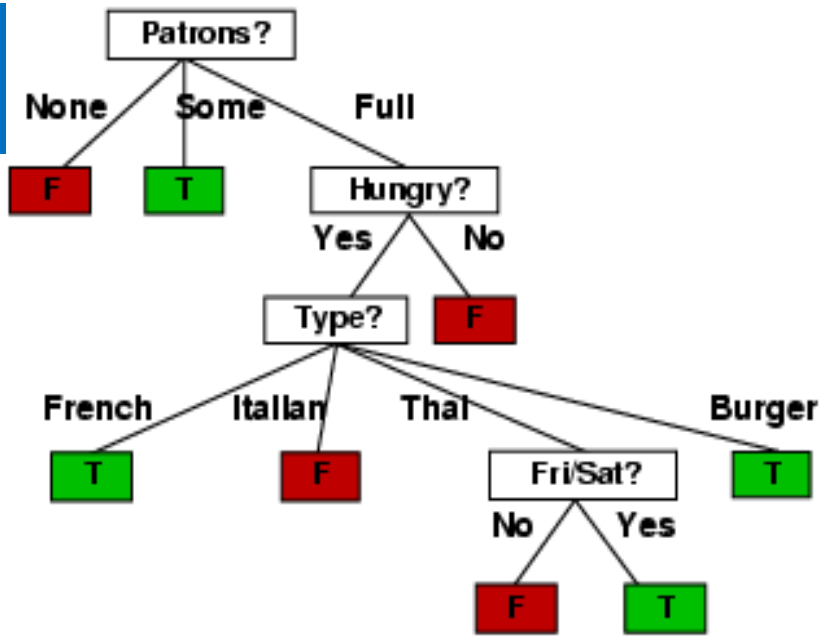
# Example cont.

- Decision tree learned from the 12 examples:




- Substantially simpler than "true" tree---a more complex hypothesis isn't justified by small amount of data

# Extract Rules from DT




If Patrons = None then NoWait

If Patrons = Some then Wait

If (Patrons = Full and Hungry = No) then NoWait

If (Patrons = Full and Hungry = Yes and Type = French) then Wait

If (Patrons = Full and Hungry = Yes and Type = Italian) then NoWait

If (Patrons = Full and Hungry = Yes and Type = Burger) then Wait

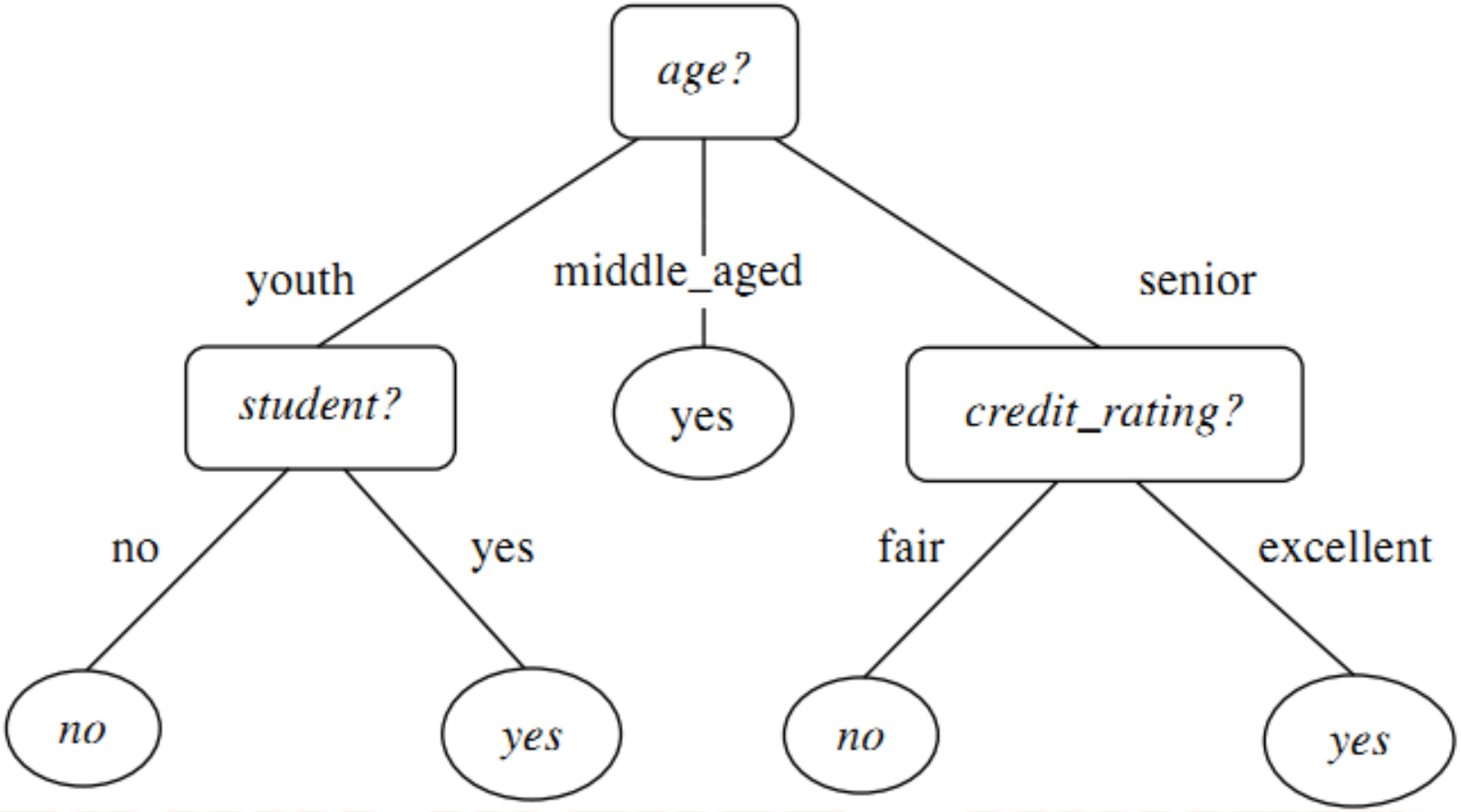
If (Patrons = Full and Hungry = Yes and Type = Thai and Fri = Yes) then Wait

If (Patrons = Full and Hungry = Yes and Type = Thai and Fri = No) then NoWait

# Attribute Selection: Information Gain

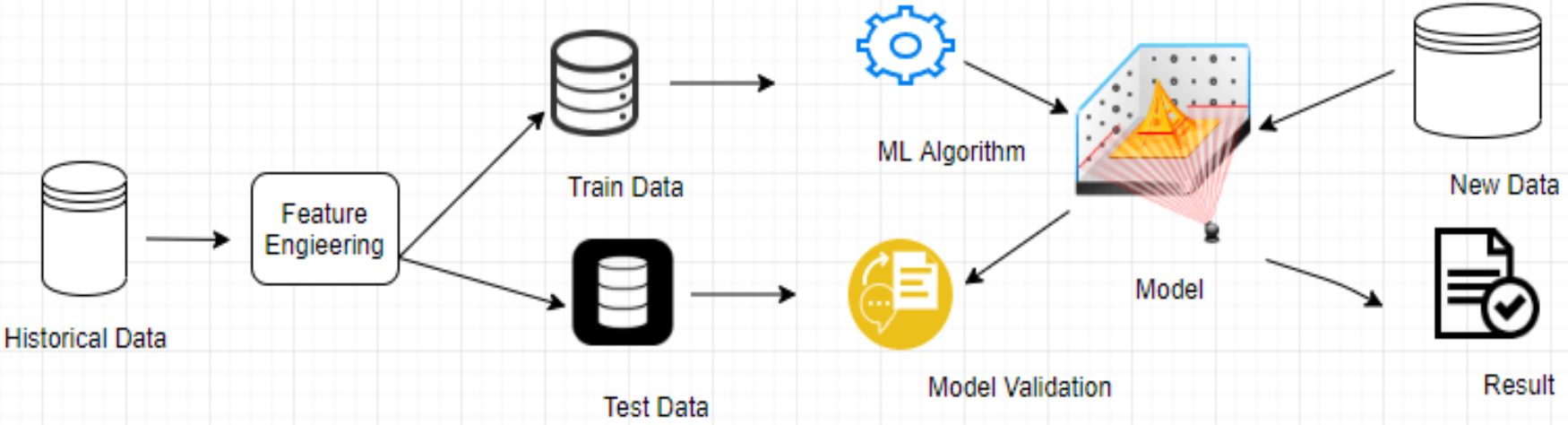
| RID | age | income | student | credit_rating | Class: buys_computer |
|---|---|---|---|---|---|
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle_aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle_aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle_aged | medium | no | excellent | yes |
| 13 | middle_aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

# Attribute Selection: Information Gain



# Performance measurement

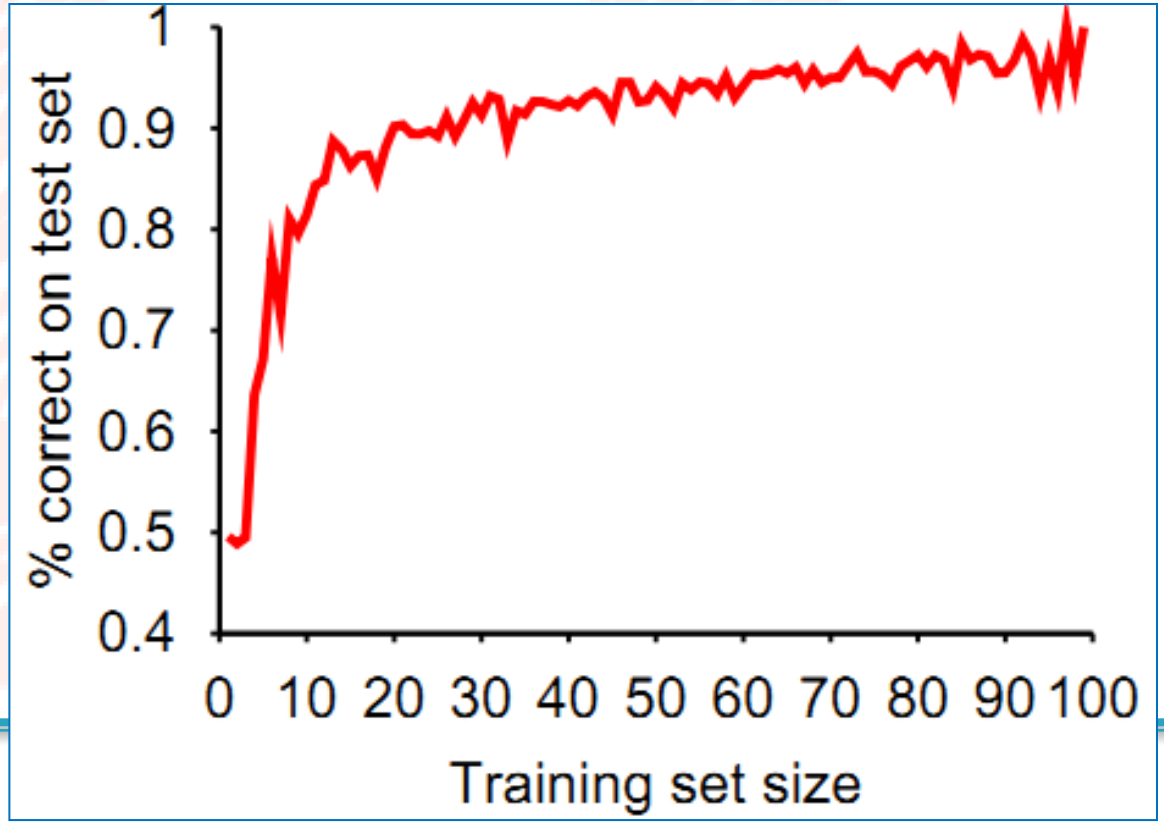
# Model Evaluation in ML



# Performance measurement

- How do we know that $h \approx f$?
  - Use theorems of computational/statistical learning theory
  - Try h on a new test set of examples
    - (use same distribution over example space as training set)

Learning curve = % correct on test set as a function of training set size



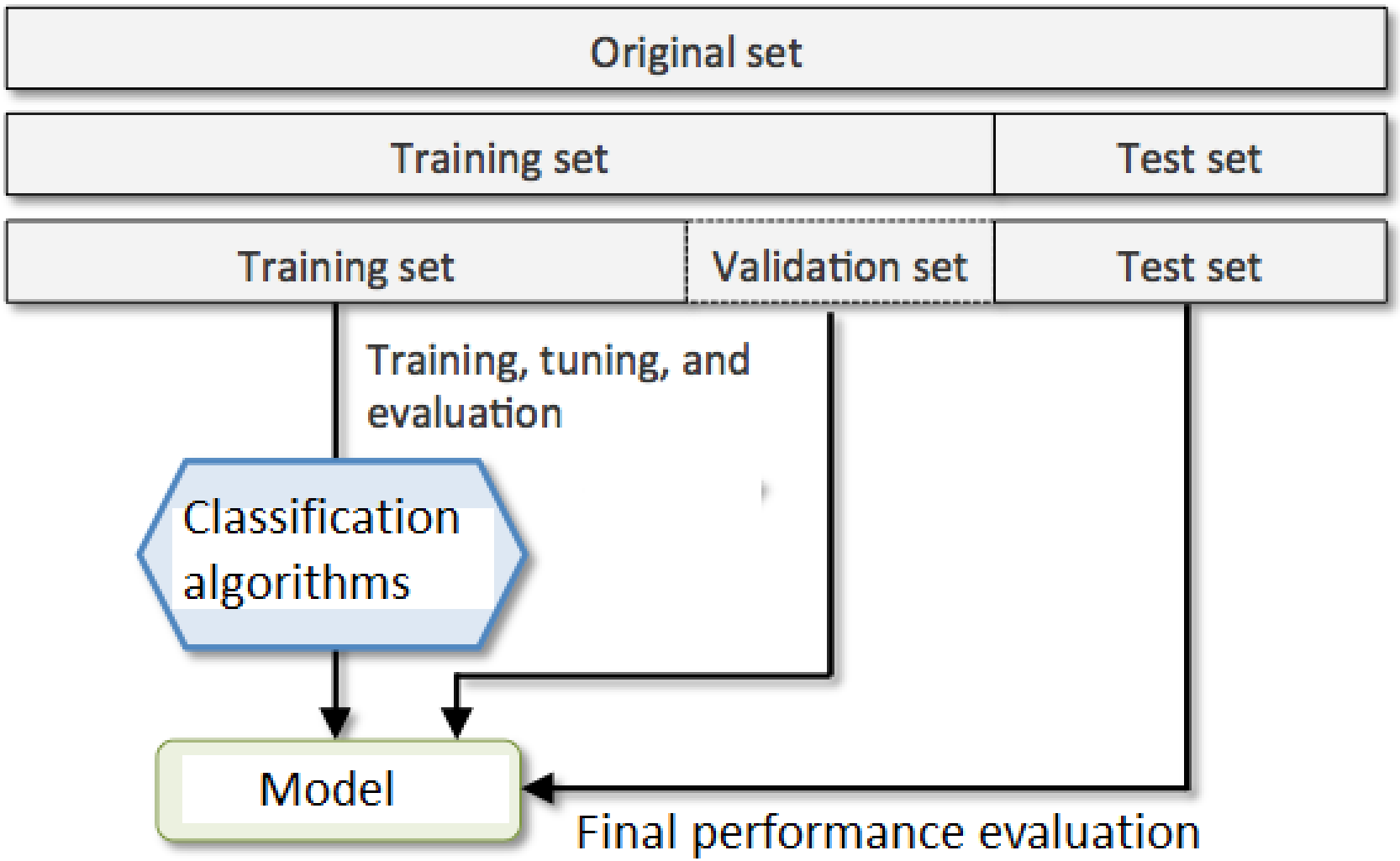
# Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use **validation test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap
- Comparing classifiers:
  - Confidence intervals
  - Cost-benefit analysis and ROC Curves

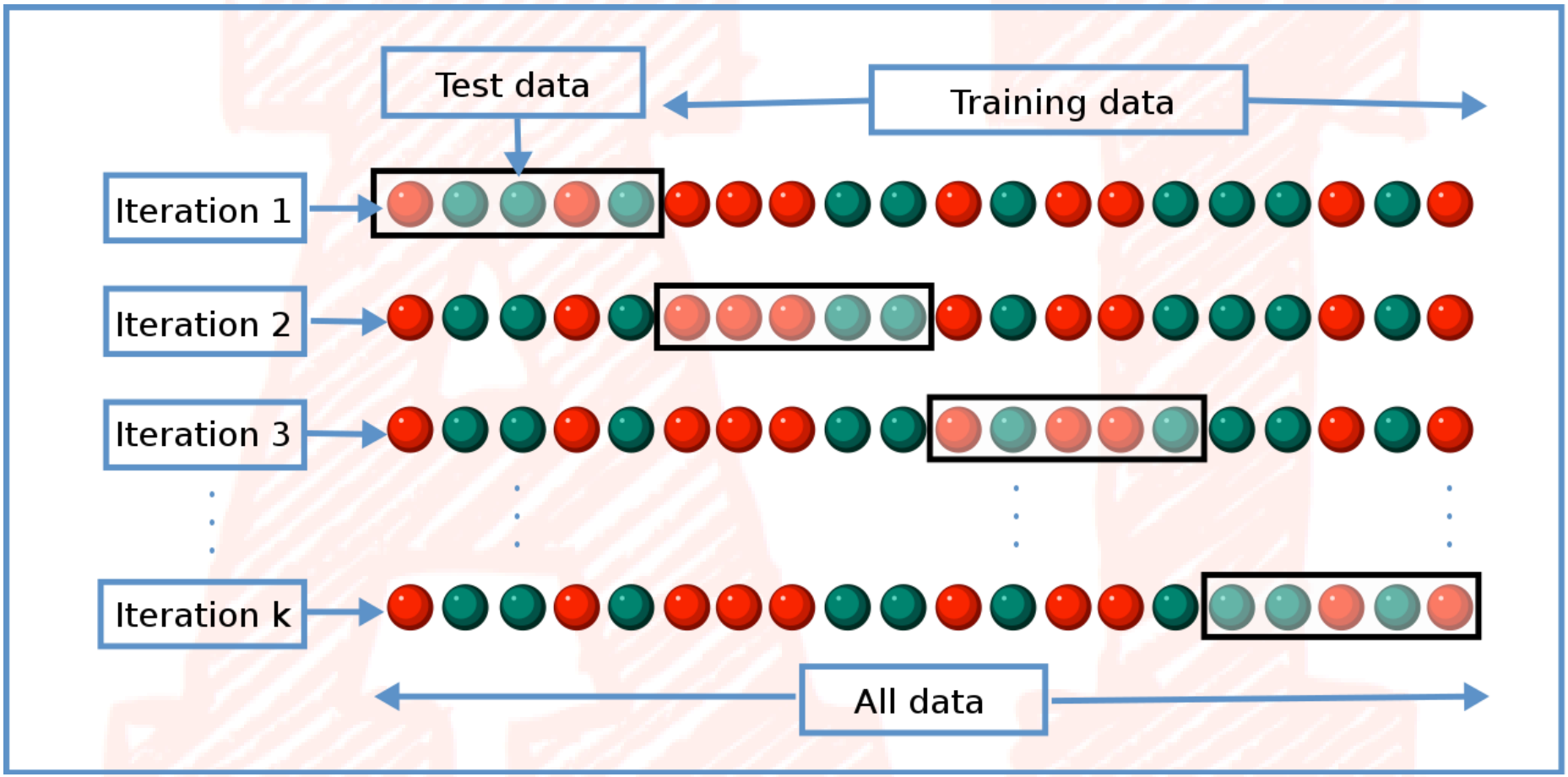
# Model Evaluation

- Purpose:
  - To estimate performance of classifier on previously unseen data (test set)
- Holdout
  - Reserve k% for training and (100-k)% for testing
  - Random subsampling: repeated holdout
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n

# Model Evaluation: Holdout



# Model Evaluation: Cross validation




https://en.wikipedia.org/wiki/Cross-validation_(statistics)

# Classifier Evaluation Metrics: Confusion Matrix

| | | Predicted class | |
|---|---|---|---|
| | | $C_1$ | $\neg C_1$ |
| Actual class | $C_1$ | **True Positives (TP)** | **False Negatives (FN)** |
| | $\neg C_1$ | **False Positives (FP)** | **True Negatives (TN)** |

# Classifier Evaluation Metrics: Confusion Matrix

**Example of Confusion Matrix:**

| | | Predicted class | | Total |
|---|---|---|---|---|
| | | buy_computer = yes | buy_computer = no | |
| **Actual class** | **buy_computer = yes** | **6954** | **46** | 7000 |
| | **buy_computer = no** | **412** | **2588** | 3000 |
| **Total** | | 7366 | 2634 | 10000 |

- Given *m* classes, an entry, $CM_{i,j}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals


# FACULTY OF INFORMATION TECHNOLOGY

**Thank you for your attention!**