


# Machine Learning in a Nutshell

## 15-488 Spring '20

Lecture 25:
Decision Trees 2

Teacher:
Gianni A. Di Caro

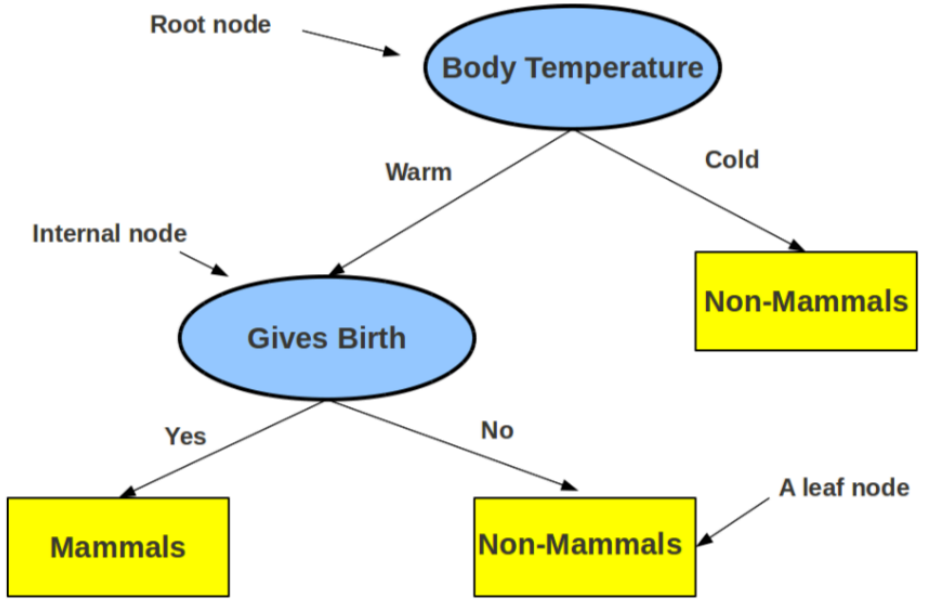
جامعة كارنيجي ميلون في قطر
Carnegie Mellon University Qatar

# Decision Trees: Summary

- A **decision tree** represents a function $f: X \rightarrow Y$, where $X$ = attribute set, $Y$ =decision set
  - Input: a vector of **attribute values**, $X_1, X_2, \cdots, X_n$
  - Output: a **decision** (a classification or a regression), value $y \in Y$

  } Discrete / continuous / categorical / interval

- Structure:
  - ❖ Defines a tree-structured hierarchy of tests / questions (*rules*)
  - ❖ Consists of a root node, internal nodes, and leaf nodes
    - ✓ **Root and internal nodes**: contain tests on attribute values
    - ✓ **Branches:** assign attribute values
    - ✓ **Leaf nodes**: define the predictions




- **Predict** (e.g., classify) input $\boldsymbol{X}$:
  1. traverse the tree from root to leaf nodes:
     *answer questions / perform tests based on input data*
  2. output the labeled $y$

# Divide-and-Conquer with Decision Trees: car fuel example

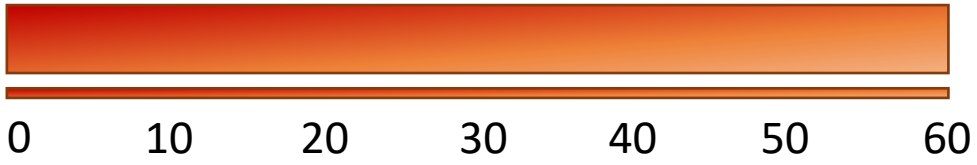
- **Simplest tree**: one node
- Make a decision by a plain *majority* strategy: **the most frequent outcome in the dataset**

➢ bad is the most frequent outcome
  - *bad*: 22
  - good: 18

**Decision Tree**

Predict:
mpg($\boldsymbol{x}$) → bad

Is this a good tree?




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

**Quiz**: Answer all questions

1. Is the classifier consistent?
2. What is its expected error rate?

# Divide-and-Conquer with Decision Trees: car fuel example

- **Simplest tree**: one node
- Make a decision by a plain *majority* strategy: **the most frequent outcome in the dataset**

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

➢ bad is the most frequent outcome
- *bad*: 22
- good: 18

**Decision Tree**

Predict:
mpg($\boldsymbol{x}$) → bad

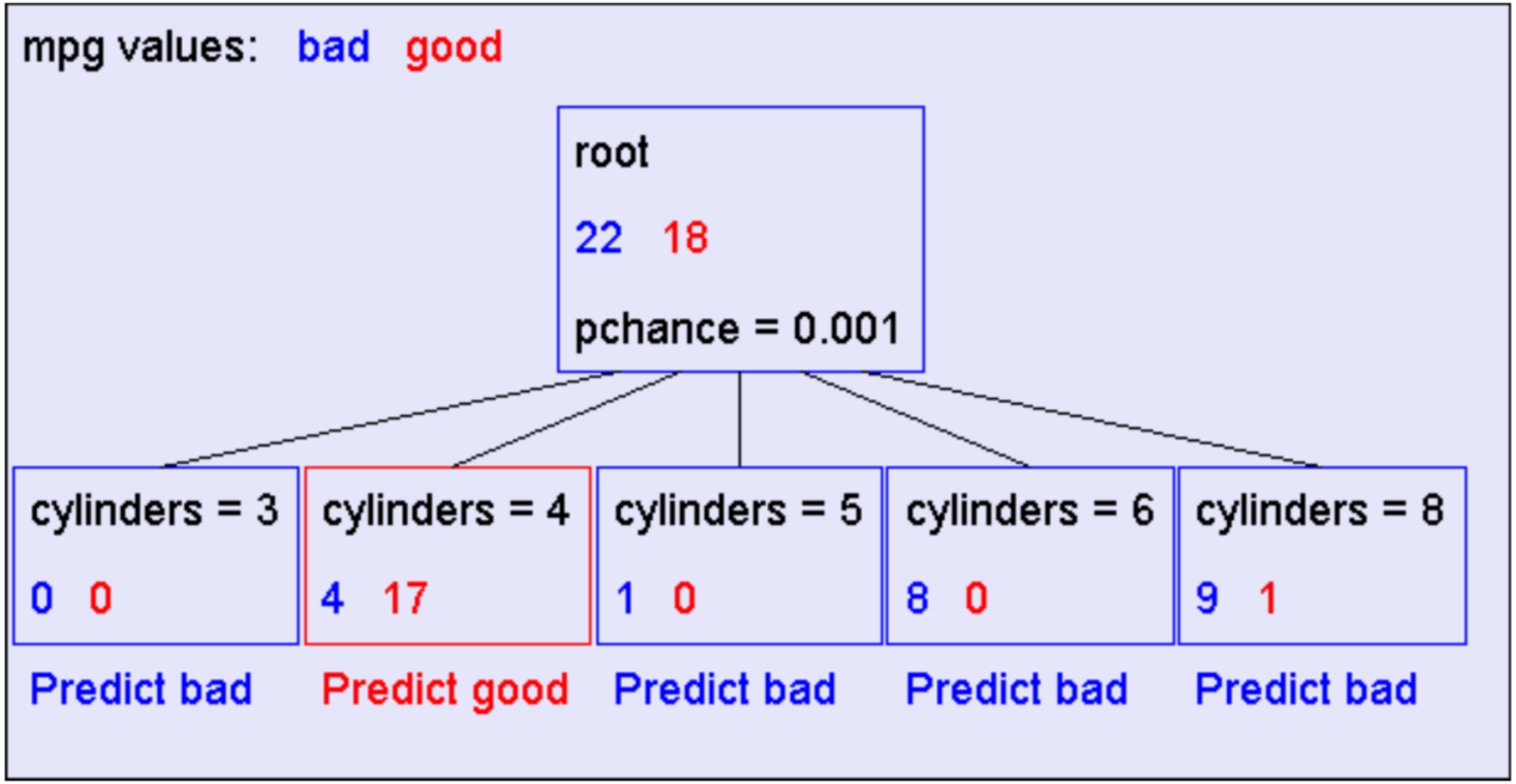
Is this a good tree?
- Correct on 22 examples
- Wrong on 18 examples

Not consistent

Expected error rate: $\frac{18}{40} = 45\%$

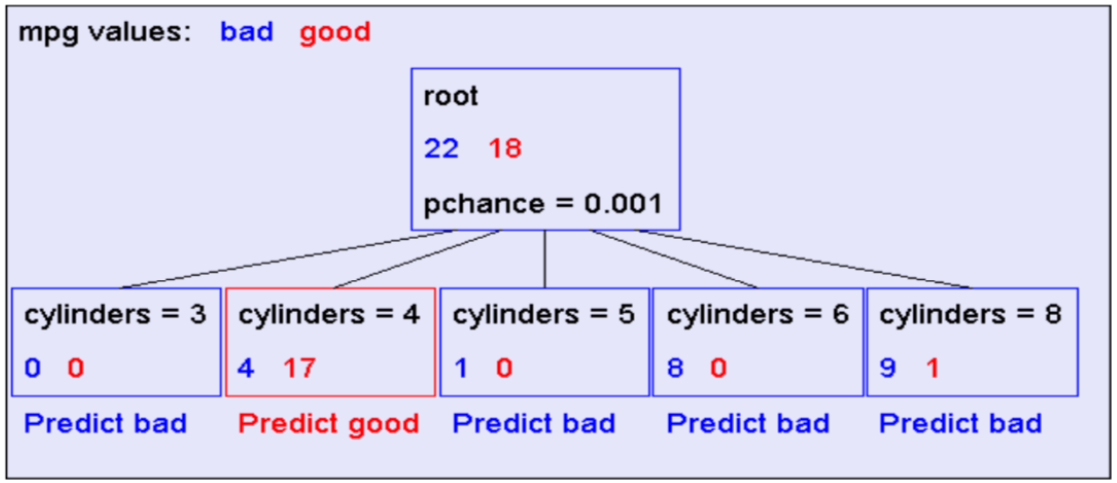
# Divide-and-Conquer: car fuel example

**Decision stump**: a one-level decision tree

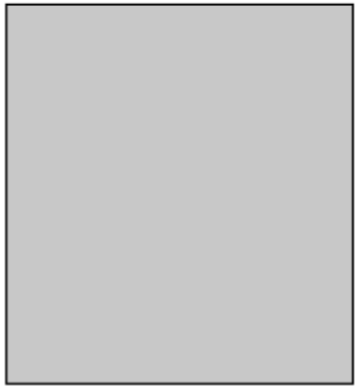



Let's start from the cylinders attribute!

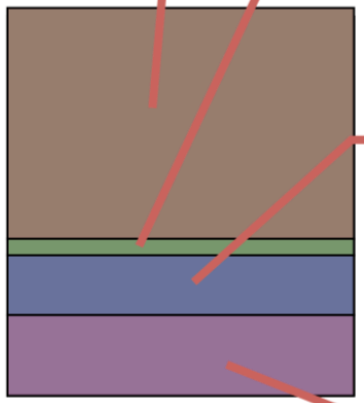
# Divide-and-Conquer: car fuel example




Take the Original Dataset..



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



Records in which cylinders = 4

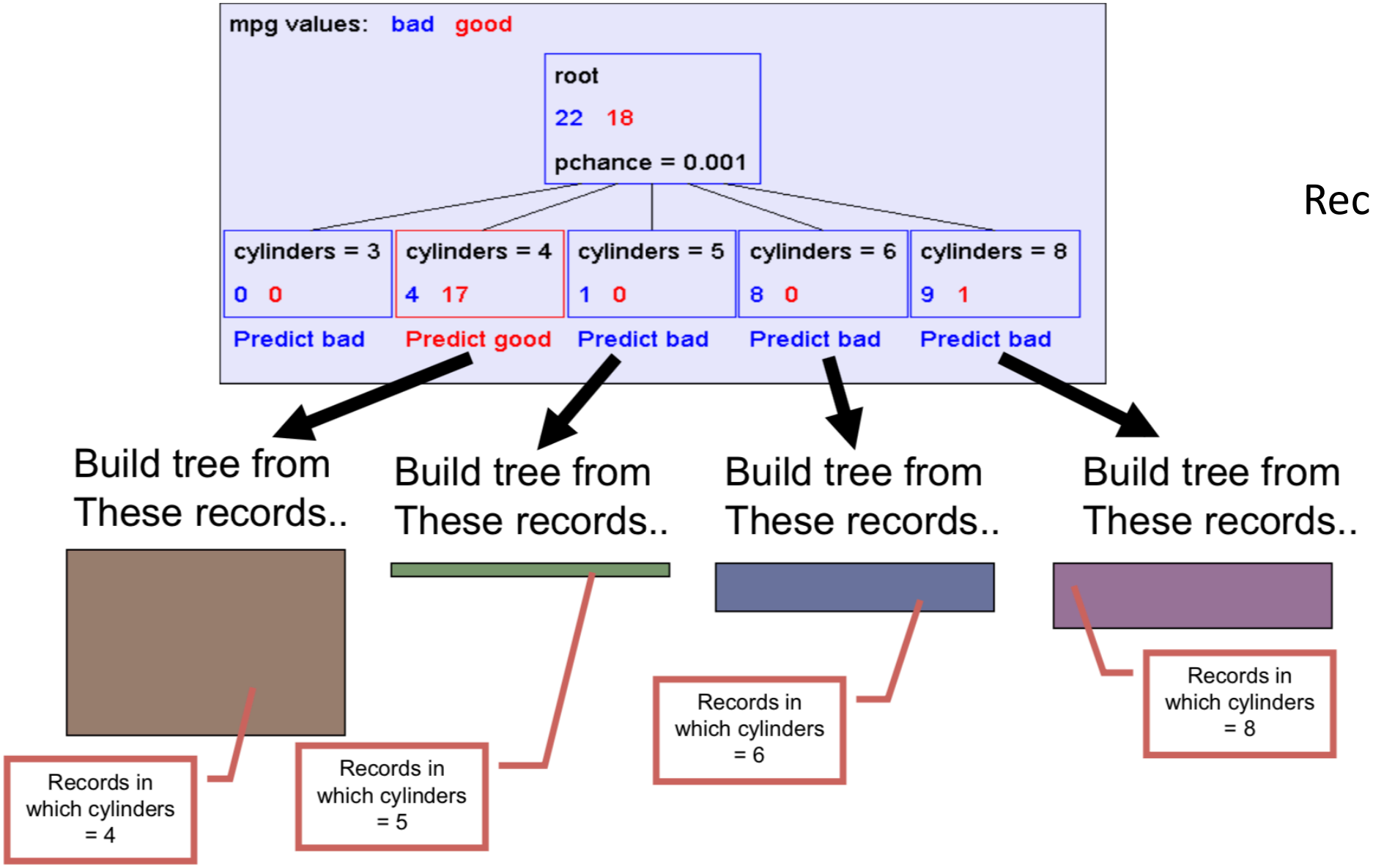
Records in which cylinders = 5

Records in which cylinders = 6

Records in which cylinders = 8

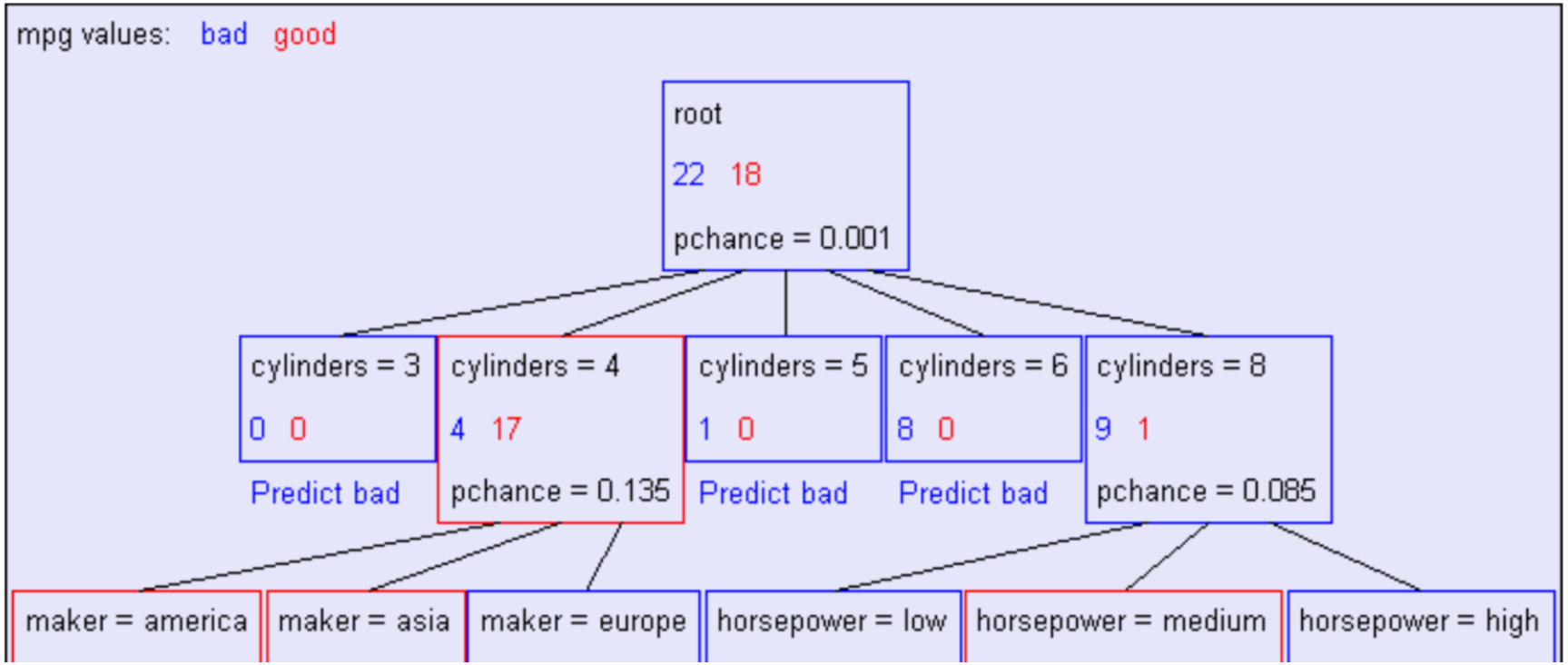
Recursive step

# Divide-and-Conquer: car fuel example




Recursive step

# Divide-and-Conquer: car fuel example

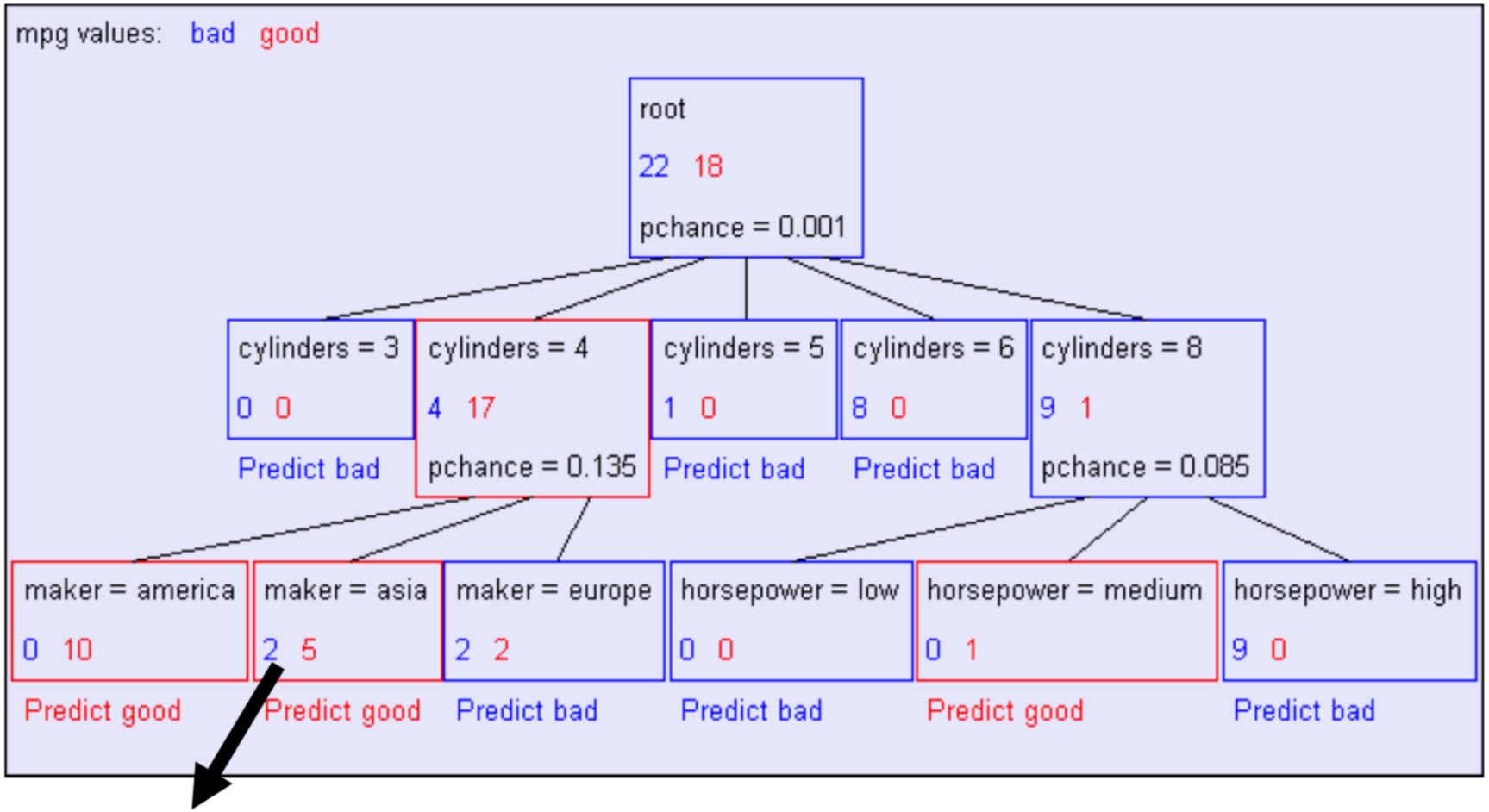



Recursive step

- From node *cylinders* = 4, the selected next node / question is on *maker,* which can take 3 values → three new branches in the tree
- From node *cylinders* = 8, the selected next node / question is on *horsepower*, which can take 3 values → three new branches in the tree

Selected how?

# Divide-and-Conquer: car fuel example



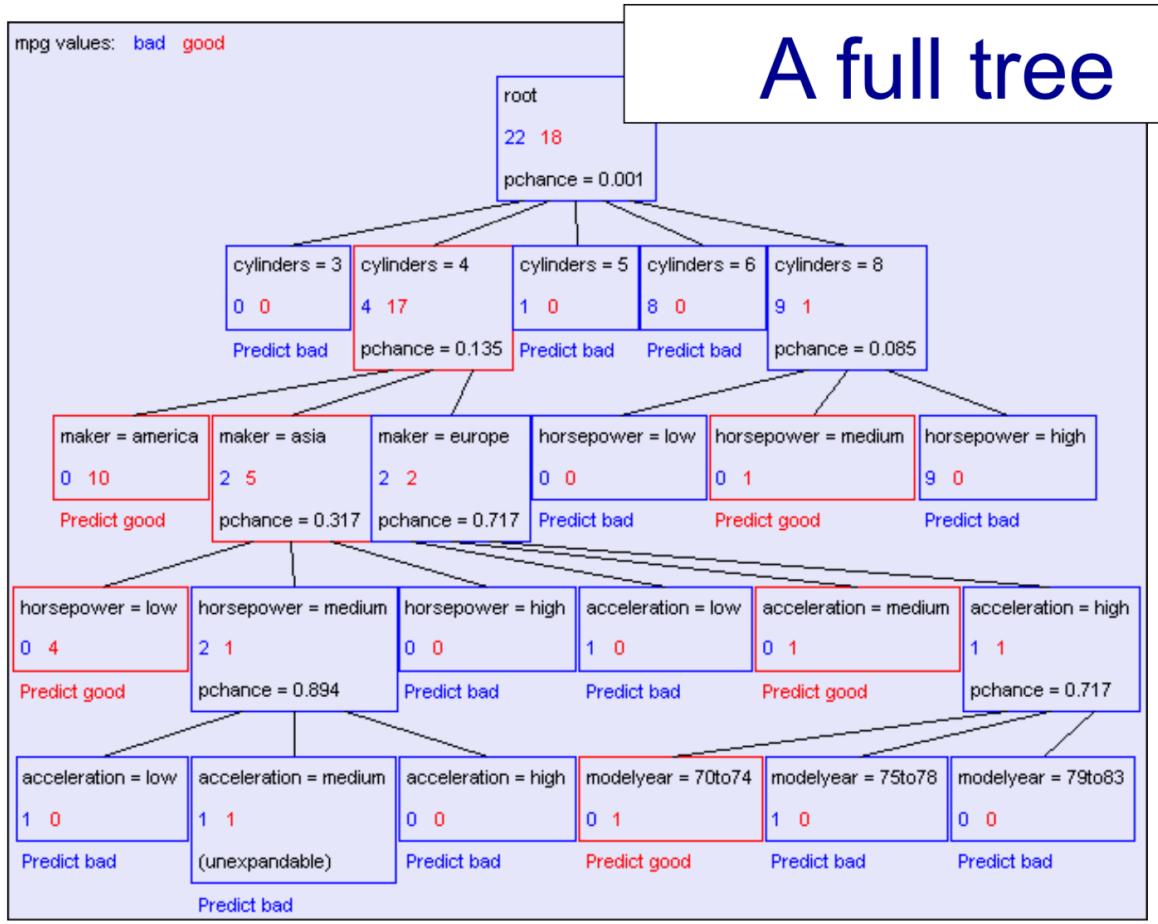

Recursive step

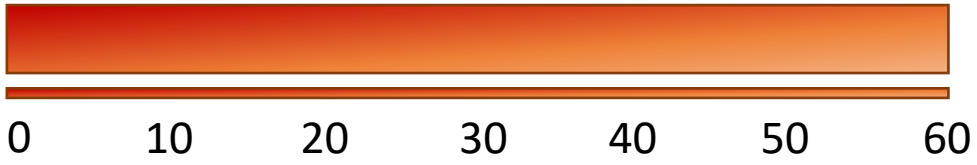
New partitioning of the dataset

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

(Similar recursion in the other cases)

# Divide-and-Conquer: car fuel example




**Quiz**: Is this the *best* tree for the problem? Check the correct answers.

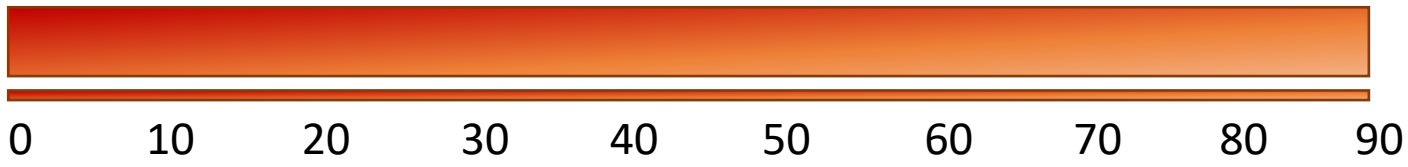
1. Yes, why not?
2. I'm not sure, maybe starting from a different root would I get a better tree?
3. I think that, maybe, checking for different attributes along the levels I might have found a better tree.
4. All trees I could build will be equivalent in generalization

# How was the tree built? Greedy, recursive, top-down heuristic

- One general, widely employed approach is to resort to a greedy, top-down heuristic:
  - Start from an empty decision tree
  - Split data on next best attribute
  - Recurse

**Quiz**: Let's be sure that we understand the terms. Give a short answer to each question:

1. Why is it a top-down approach?
2. Why is it a greedy approach?
3. Why is it a heuristic approach?



# Greedy, recursive, top-down heuristic

- One general, widely employed approach is to resort to a greedy, top-down heuristic:
  - Start from an empty decision tree
  - Split data on next best attribute
  - Recurse

**ID3** (Quinlan, 1986): Natural greedy approach growing a DT from top-down, from top to the leaves by repeatedly replacing an existing leaf with an internal node.

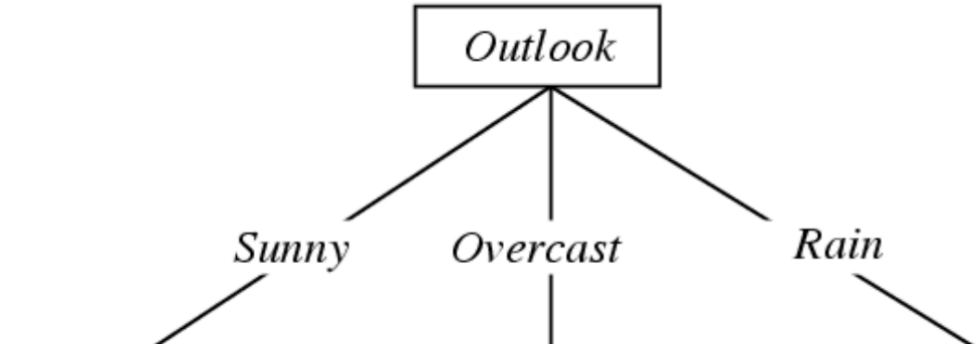
- Pick best attribute to split at the root based on training data
- Recurse on children node that are impure
  - Their data partition does not allow to issue a decision free of uncertainty, not all data entries belong to the same class
- Split data on next best attribute

# Greedy, recursive, top-down heuristic

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

Play tennis dataset

# Greedy, recursive, top-down heuristic




all the examples with
*Outlook = Sunny*

$P_{O_s}$

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |

**Impure set**

$P_{O_o}$

all examples with
*Outlook = Overcast*

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D3 | Overcast | Hot | High | Weak | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |

**Pure set**

$P_{O_r}$

all the examples with
*Outlook = Rain*

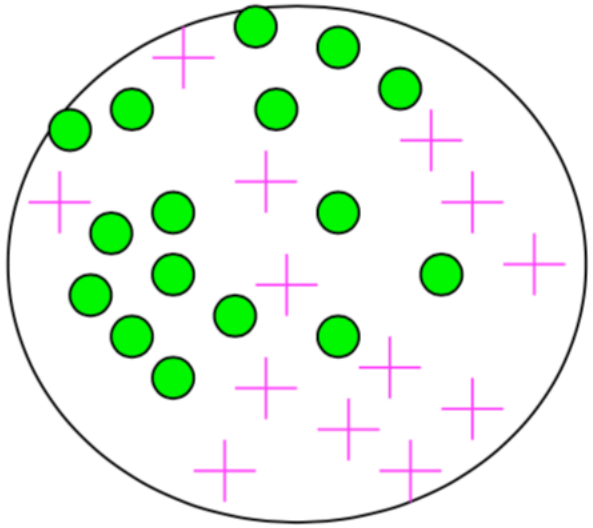
| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

**Impure set**

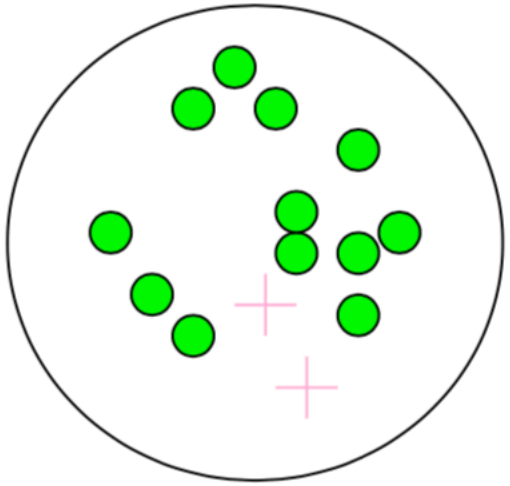
- `Outlook` is selected as *attribute* at the **root node**
- Outlook can take 3 possible values
- → The initial dataset $D$ of 14 examples is **split** in the partitions: $P_{O_s}, P_{O_r}, P_{O_o}$

# Impurity

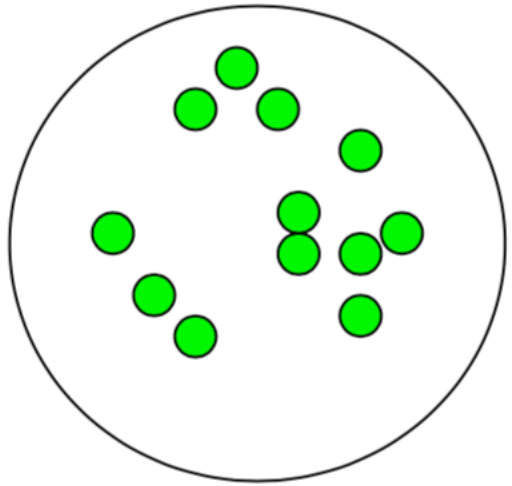
**Very impure group**



**Less impure**

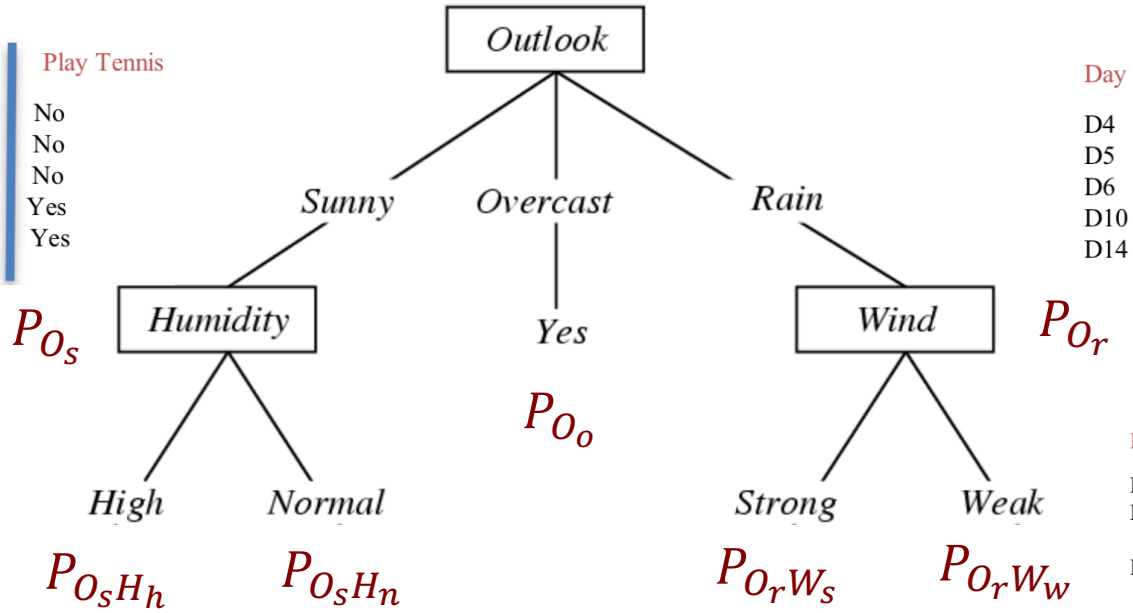


**Minimum impurity**



# Greedy, recursive, top-down heuristic

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |




$P_{O_s}$ $P_{O_o}$ $P_{O_r}$

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D8 | Sunny | Mild | High | Weak | No |

$P_{O_sH_h}$ $P_{O_sH_n}$

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |

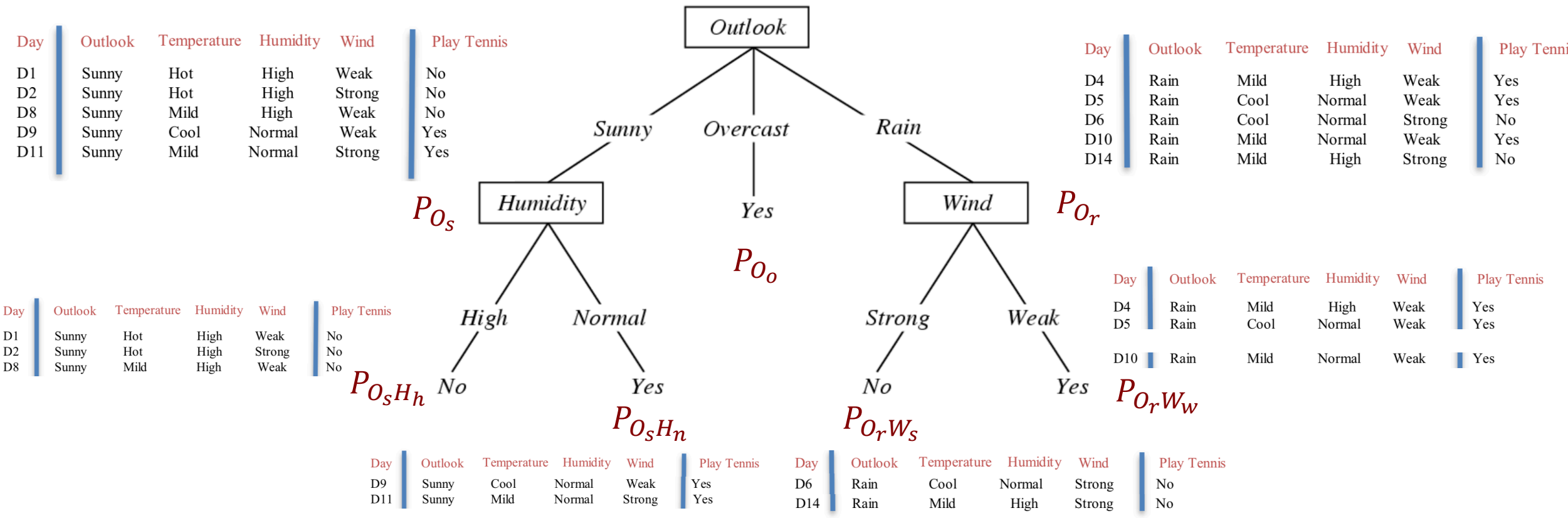
$P_{O_rW_s}$ $P_{O_rW_w}$

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D6 | Rain | Cool | Normal | Strong | No |
| D14 | Rain | Mild | High | Strong | No |

- At each brach / split we need to select the next (best) question / next (best) attribute to further split the data
- `Humidity` is best attribute for $P_{O_s}$, and `Wind` is best attribute for $P_{O_r}$, each attribute creates a 2-partition
- $P_{O_o}$ doesn't need any further split since it is a **pure set** for taking a decision: all entries have the same label
- **At all leafs, the partition sets are pure** → The construction of the tree stops.

# Greedy, recursive, top-down heuristic




- At each brach / split we need to select the next (best) question / next (best) attribute to further split the data
- `Humidity` is best attribute for $P_{O_s}$, and `Wind` is best attribute for $P_{O_r}$, each attribute creates a 2-partition
- $P_{O_o}$ doesn't need any further split since it is a **pure set** for taking a decision: all entries have the same label
- **At all leafs, the partition sets are pure** → The construction of the tree stops: we have all decisions set!

# ID3 (Quinlan, 1986)

*node* = Root

Main loop:

1. $A \leftarrow$ the "best" decision attribute for next *node*
2. Assign $A$ as decision attribute for *node*
3. For each value of $A$, create new descendent of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes.

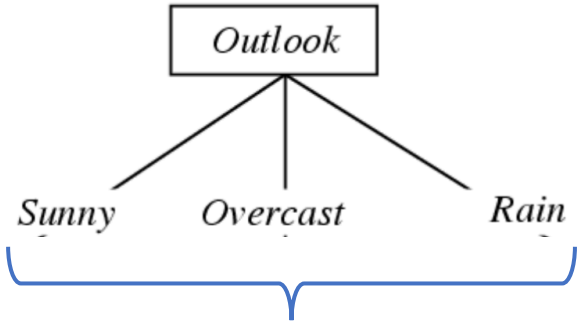
*Best*?

# Key problem: Choosing the next / best attribute

Key problem in the step-by-step building of a decision tree: choosing the **best attribute** to split a given set of examples

1. At any node, given its **partition of the dataset**,
2. which **attribute** shold be selected to **split the partition**,
3. with the aim of reducing the uncertainty towards final decision
4. ↔ in order to quickly generate pure partitions / reach out leafs?

At node *Outlook*




Values for the attribute *Outlook*

Partion of the dataset where *Outlook = Rain*

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

Question: which attribute to select next (which question to ask next) to split further the partition?

- *Temperature*
- *Humidity*
- *Wind*

# Key problem: Choosing the next / best attribute

Some possible strategies to choose the attribute to split a given set of examples:

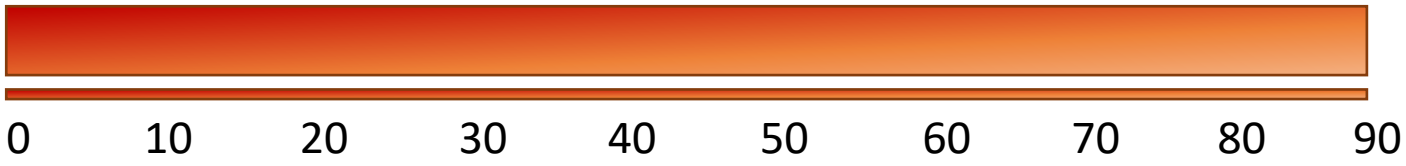
- **Random**: Select any attribute at random
- **Least-Values**: Choose the attribute with the smallest number of possible values
- **Most-Values**: Choose the attribute with the largest number of possible values
- ✓ **Max-Gain**: Choose the attribute that has the largest expected information gain (i.e., reduction of expected uncertainty regarding the decisions)
  - ❖ Attribute that results in smallest expected size of the sub-trees rooted at its children

ID3 uses the Max-Gain criterion, where the max gain can be expressed using the notion of **Entropy**

# Key problem: Choosing the next / best attribute

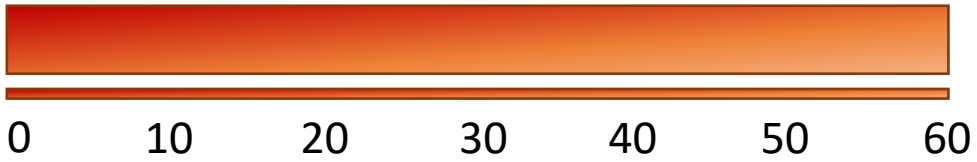
**Quiz**: Check the true statements behind the motivation of the different strategies for choosing the best attribute.

1. **Least-Values**: the attribute with the smallest number of possible values will determine the minimal branching and therefore will minimize the expected growth of the tree
2. **Most-Values**: the attribute with the largest number of possible values will automatically split the the node partition in many different partitions, favoring breaking out uniformity / impurity in the sets
3. **Max-Gain:** the attribute that has the largest expected reduction in the uncertainty also results in smallest expected size of the sub-trees that will be rooted at its children nodes



# Splitting: choosing a good attribute

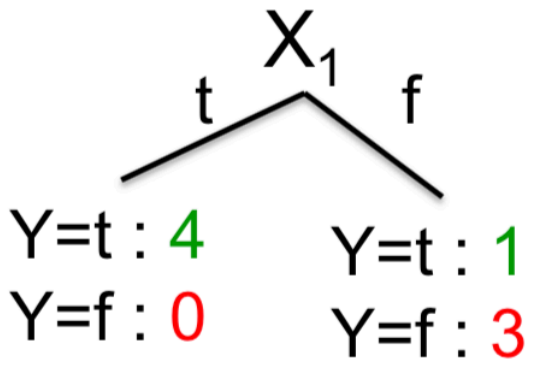
**Quiz:** Would you prefer to split on $X_1$ or $X_2$ to optimize information gain?

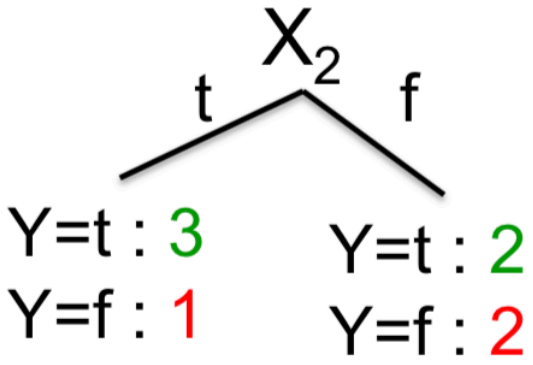



| $X_1$ | $X_2$ | Y |
|---|---|---|
| T | T | T |
| T | F | T |
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |
| F | T | F |
| F | F | F |

# Splitting: choosing a good attribute

Would we prefer to split on $X_1$ or $X_2$?




Splitting on $X_1$ we can estimate:

If $X_1$ = T → Pr($Y$ = T) = 1

If $X_1$ = F → Pr($Y$ = F) = 0.75

Low uncertainty in decisions

Splitting on $X_2$ we can estimate:

If $X_2$ = T → Pr($Y$ = T) = 0.75

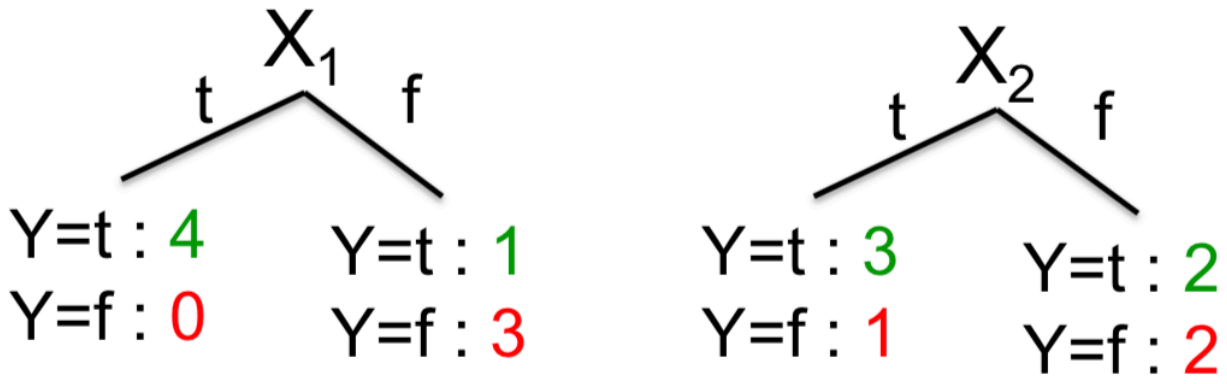
If $X_2$ = F → Pr($Y$ = F) = 0.5

Large uncertainty in decisions

| $X_1$ | $X_2$ | Y |
|---|---|---|
| T | T | T |
| T | F | T |
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |
| F | T | F |
| F | F | F |

# Splitting: choosing a good attribute

Would we prefer to split on $X_1$ or $X_2$?




| $X_1$ | $X_2$ | Y |
|---|---|---|
| T | T | T |
| T | F | T |
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |
| F | T | F |
| F | F | F |

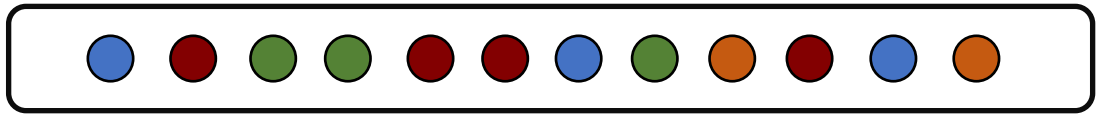
Idea: use counts at leaves to define probability distributions, so we can measure uncertainty!

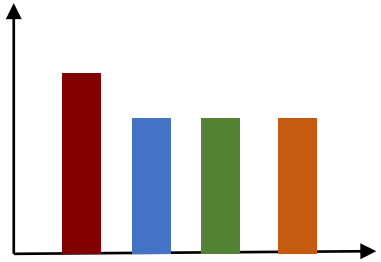
# Measuring uncertainty

➢ A split is good if we are more certain about classification / decision after the split.

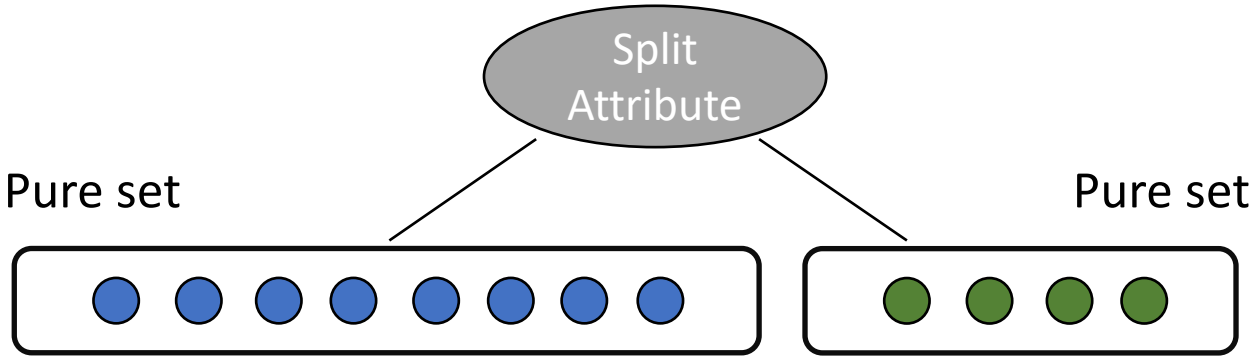
E.g., for the case of four classes/decisions, red, green, blue, orange:

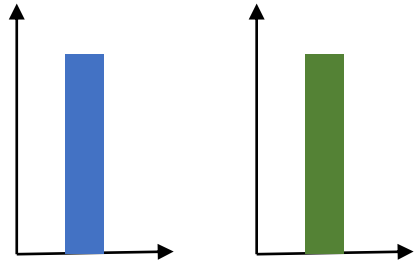
Large uncertainty set
(Very) Impure





✓ **Deterministic, good:** After the split each set contains elements of one and only one class
→ Leafs of the decision tree



$P(blue) = 1$

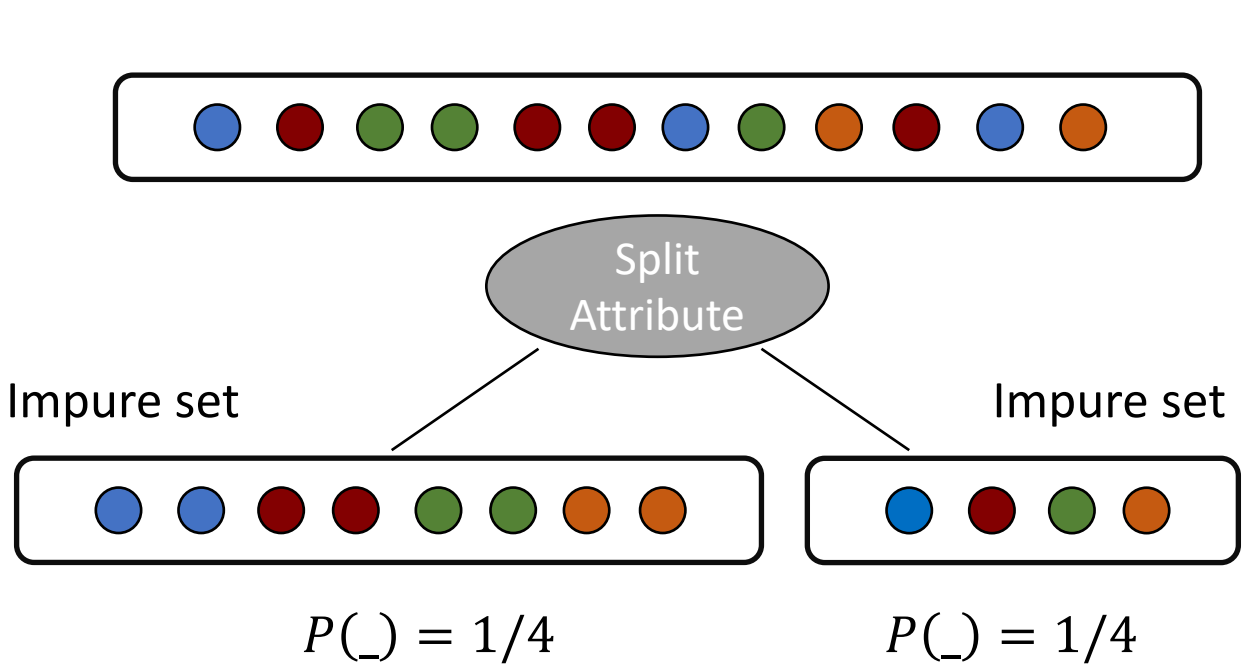
$P(r) = P(g) = P(o) = 0$

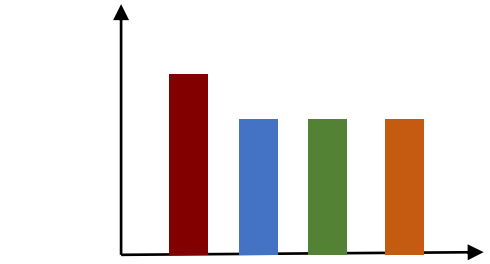
$P(green) = 1$

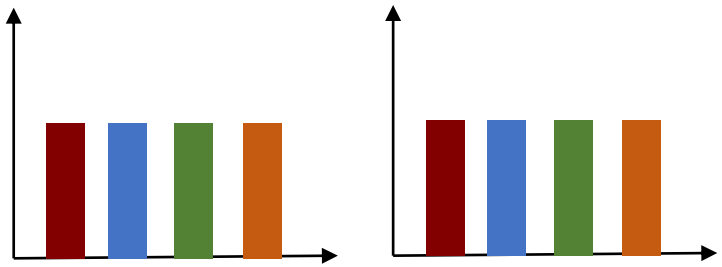
$P(r) = P(b) = P(o) = 0$

# Measuring uncertainty

- **Uniform, bad:** After the split each set contains a uniform distribution of the elements
  → Can't really take any reliable decision, need to expand the tree further

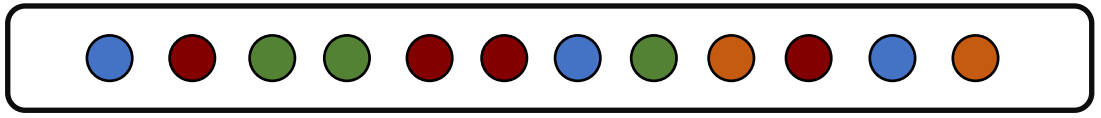


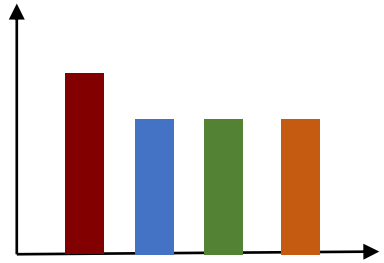



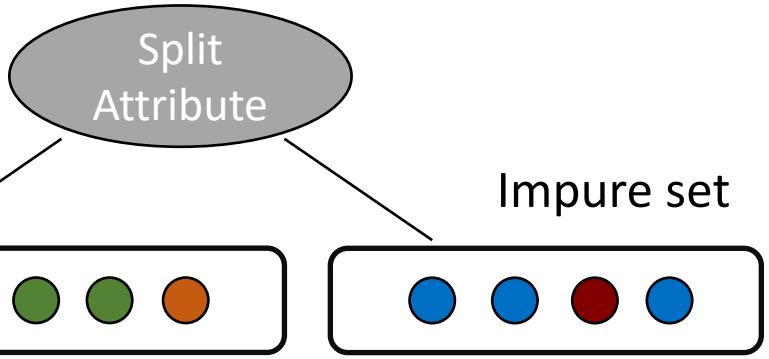


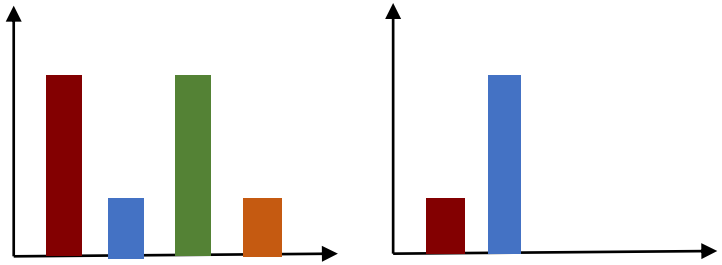
# Measuring uncertainty

- **In-between:** After the split each set contains some degree of impurity / not uniform distribution of the elements







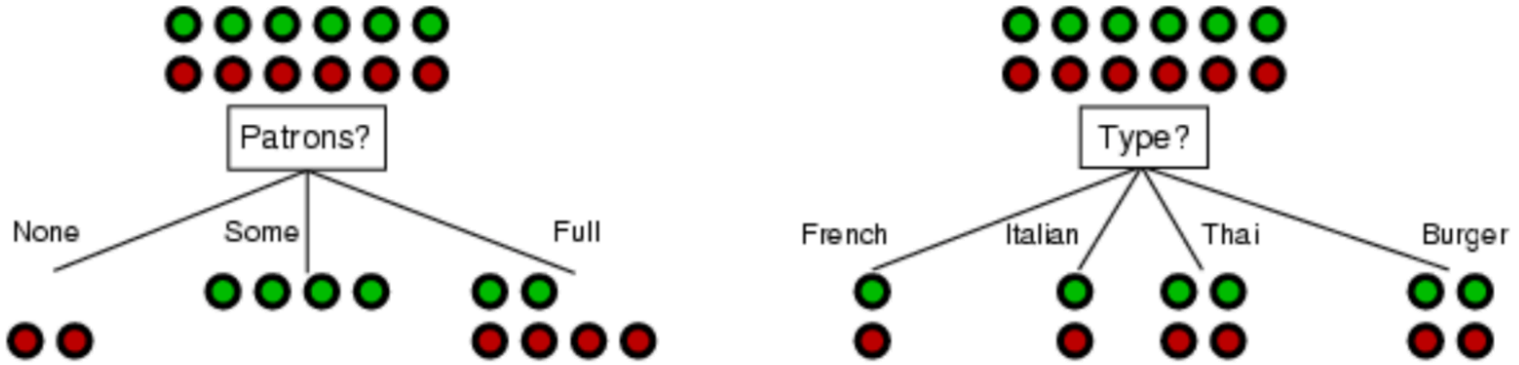



$P(b) = P(o) = 1/8$
$P(r) = P(g) = 3/8$

$P(b) = 3/4$
$P(r) = 1/4$
$P(g) = P(o) = 0$

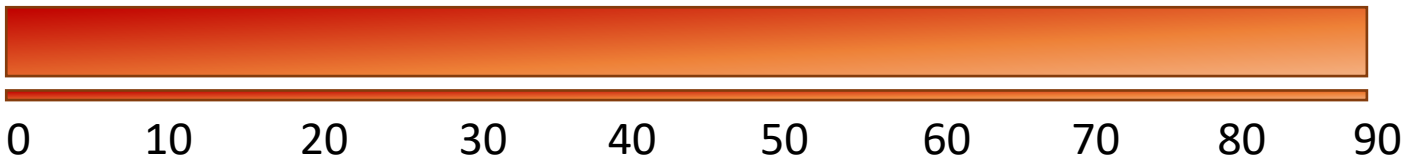
How do we assess whether this split is good or bad? To which degree is this good or bad?

# Splitting: choosing a good attribute

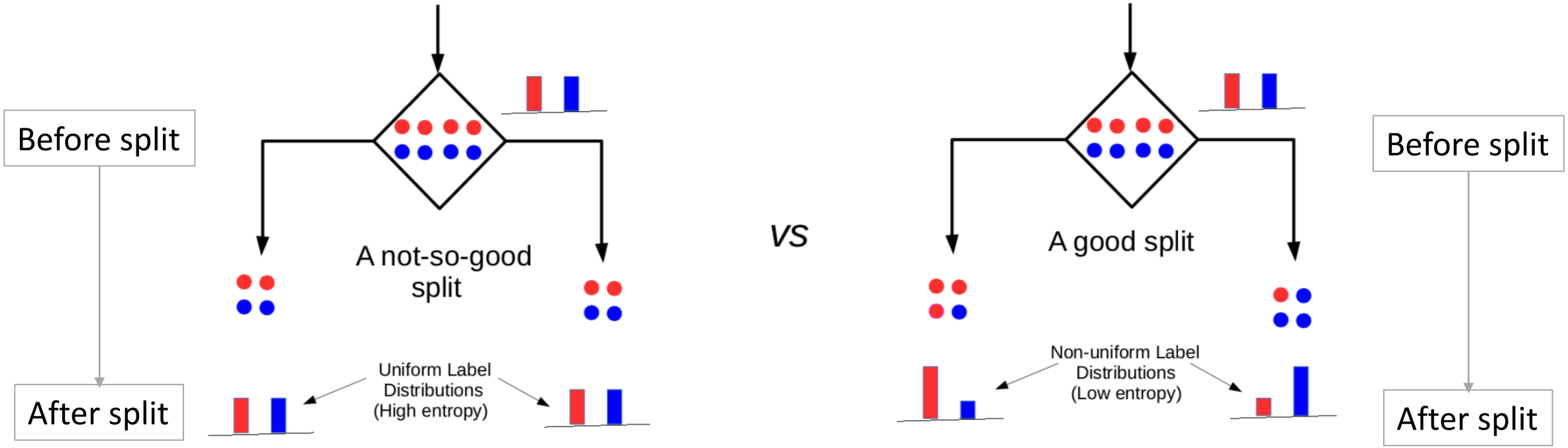



**Quiz**: Given a uniform partition, let's select the best attribute to split on.

1. Which split is better, *Patrons* or *Type?*
2. How many pure partitions are generated by splitting on *Patrons*?
3. How many pure partitions are generated by splitting on *Type*?
4. Given that in the input *Patrons = Some*, what would be the answer of the DT in the case the split is on *Patrons*? And what is the answer in the case the split is on *Type*?
5. Answer the same question 4 for the input *Patrons = Full*



# Splitting: increasing *purity* / decreasing *entropy*



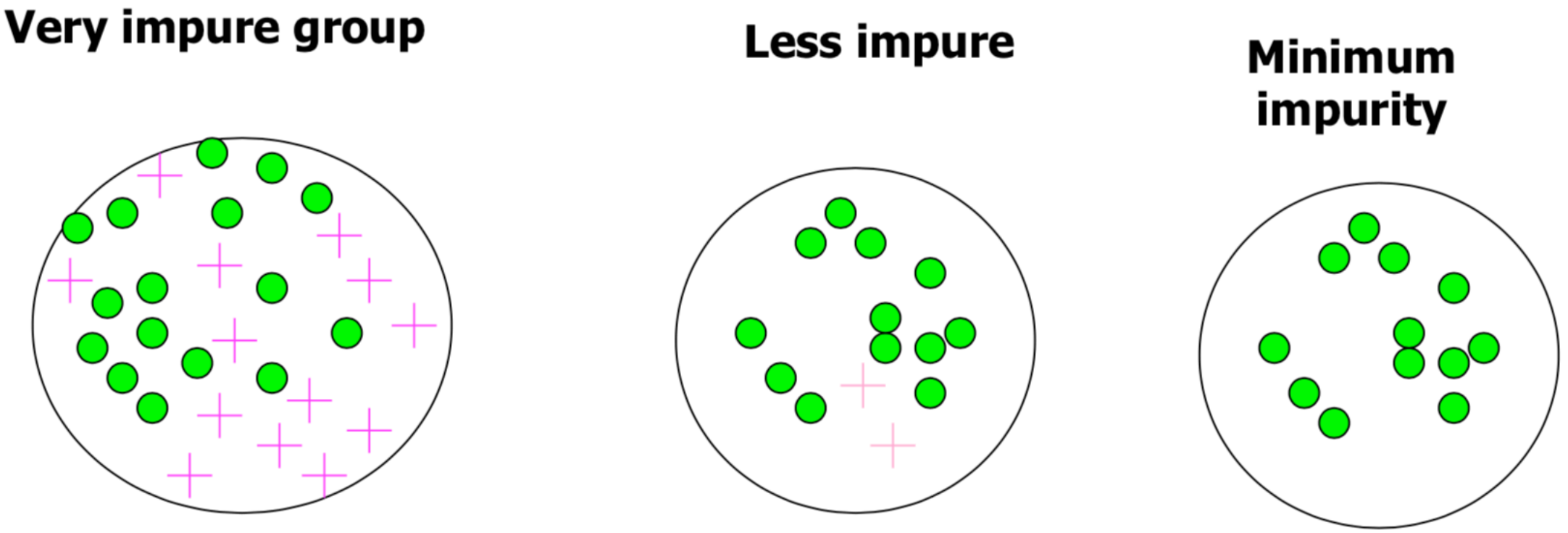

- **Entropy** of the label distribution is a *measure of purity*
  - Low entropy → High purity, **low uncertainty**, not a uniform distribution of the labels
  - ✓ Splits that give the largest reduction in entropy (before split vs. after split) are preferred

- **Information gain** = Entropy(before_split) – Entropy(after_split)

# Impurity / Entropy

**Impurity / Entropy (informal)**

- Measures the level of impurity in a group of examples



# Entropy of a random variable

Entropy $H(X)$ of a random variable $X$

# of possible values for X

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

E.g., the outcome from rolling a 6-face *fair die* can be modeled as a random variable $X$ that can take 6 possible values, $\{1, 2, 3, 4, 5, 6\}$ (the event space of $X$), and each possible outcome is equally probable (it's fair!):

$$P(X = 1) = P(X = 2) = P(X = 3) = P(X = 4) = P(X = 5) = P(X = 6) = 1/6$$

$$H(X) = -\sum_{i=1}^{6} P(X = i) \ln P(X = i) = -\sum_{i=1}^{6} \frac{1}{6} \ln\left(\frac{1}{6}\right) = -6\frac{1}{6} \ln\frac{1}{6} = -\ln\frac{1}{6} = 1.79$$


For a 20-face fair die, $H(X) = -\ln\frac{1}{20} = 4.32$

Uncertainty of outcome is higher, entropy increases!

# Entropy of a random variable


For tossing a fair coin, $H(X) = -\ln\frac{1}{2} = 1$

- If Einstein's face is *more likely* (e.g., there's more weight on the other side of the coin), such that

$$P(X = E) = 0.75 \quad P(X = C) = 0.25 \quad \rightarrow \quad H(X) = -\left(\frac{3}{4}\ln\frac{3}{4} + \frac{1}{4}\ln\frac{1}{4}\right) = 0.81$$

There's less uncertainty on outcome
→ Entropy decreases!

- If Einstein's face is *sure*, such that:

$$P(X = E) = 1 \quad P(X = C) = 0 \quad \rightarrow \quad H(X) = -(1\ln 1 + 0) = 0$$

There's certainty on outcome
→ Entropy is zero!

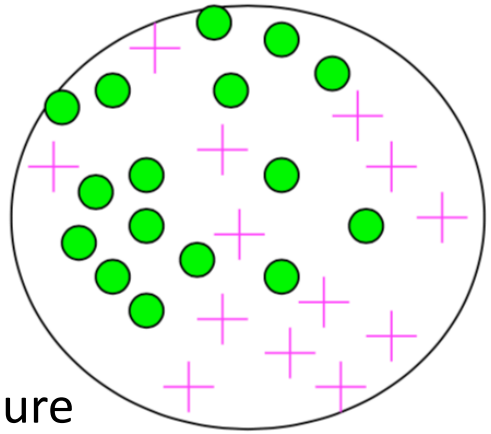
✓ No uncertainty in the outcome → **Entropy is 0**

✓ Maximal uncertainty in the outcome (all $n$ outcomes are equally possible)
→ **Entropy is maximal**, $H(X) = -\ln\frac{1}{n}$

# Entropy as a measure of impurity of a (labeled) set

$$Y = \{circle, cross\}$$

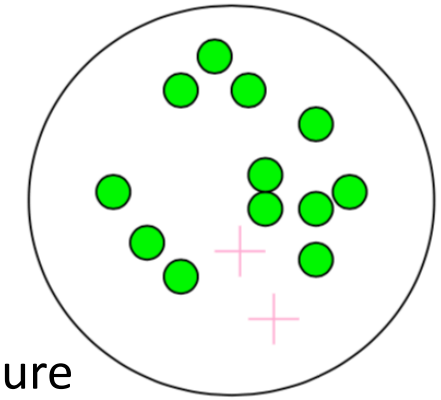
$S = \{16\ circle, 13\ cross\}$



Very impure

$$H_S(Y) = 0.99$$

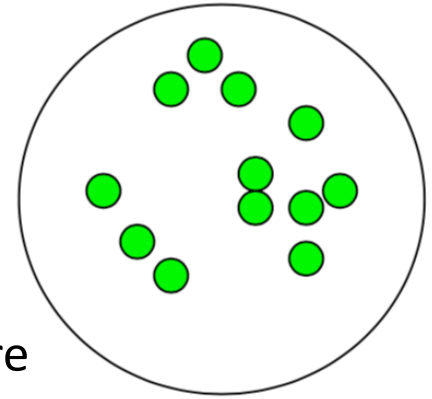
$S = \{12\ circle, 2\ cross\}$



Impure

$$H_S(Y) = 0.59$$

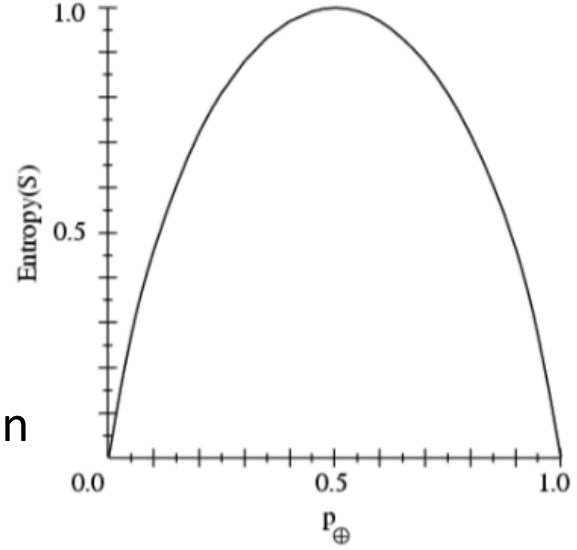
$S = \{12\ circle, 0\ cross\}$



Pure

$$H_S(Y) = 0$$

Entropy of the target variable $Y$ for each data sample $S$



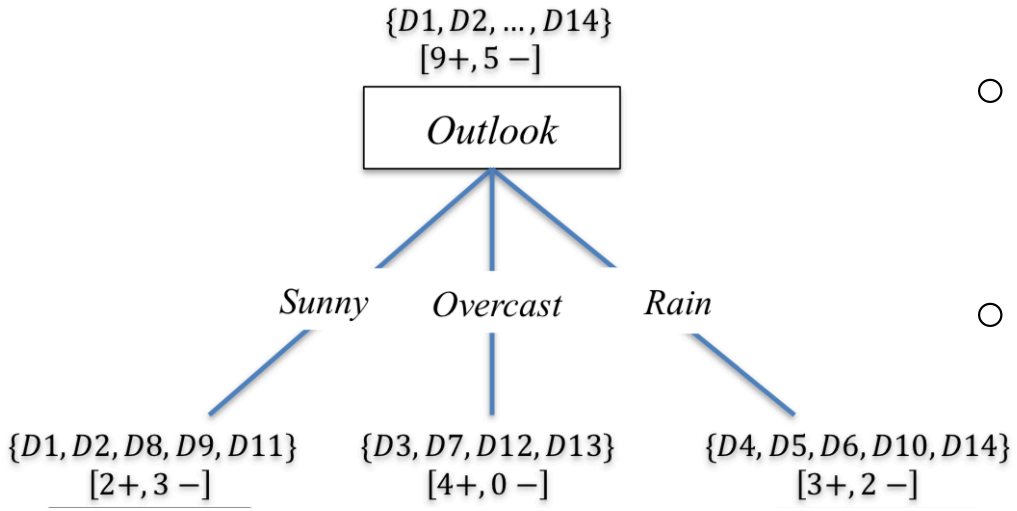

Entropy as a function of the fraction of +ve examples in a 2-class set

# ID3 Splitting: the attribute with the highest information gain

**Information Gain:** Expected reduction in entropy of the target variable $Y$ for a data sample $S$ as the result of splitting $S$ on the attribute $A$

$$Gain(S, A) = H_S(Y) - H_S(Y \mid A)$$

- **ID3**: Select the attribute $A$ with the highest information gain




- Since splitting on an attribute produces $m$ data partitions, one per each value of the attribute $A$, the term $H_S(Y|A)$ is computed as the **weighted average of the entropies of each partition set**
- Weights are the number of elements in the partition out of the total in $S$ (a more crowded partition should weight more!)

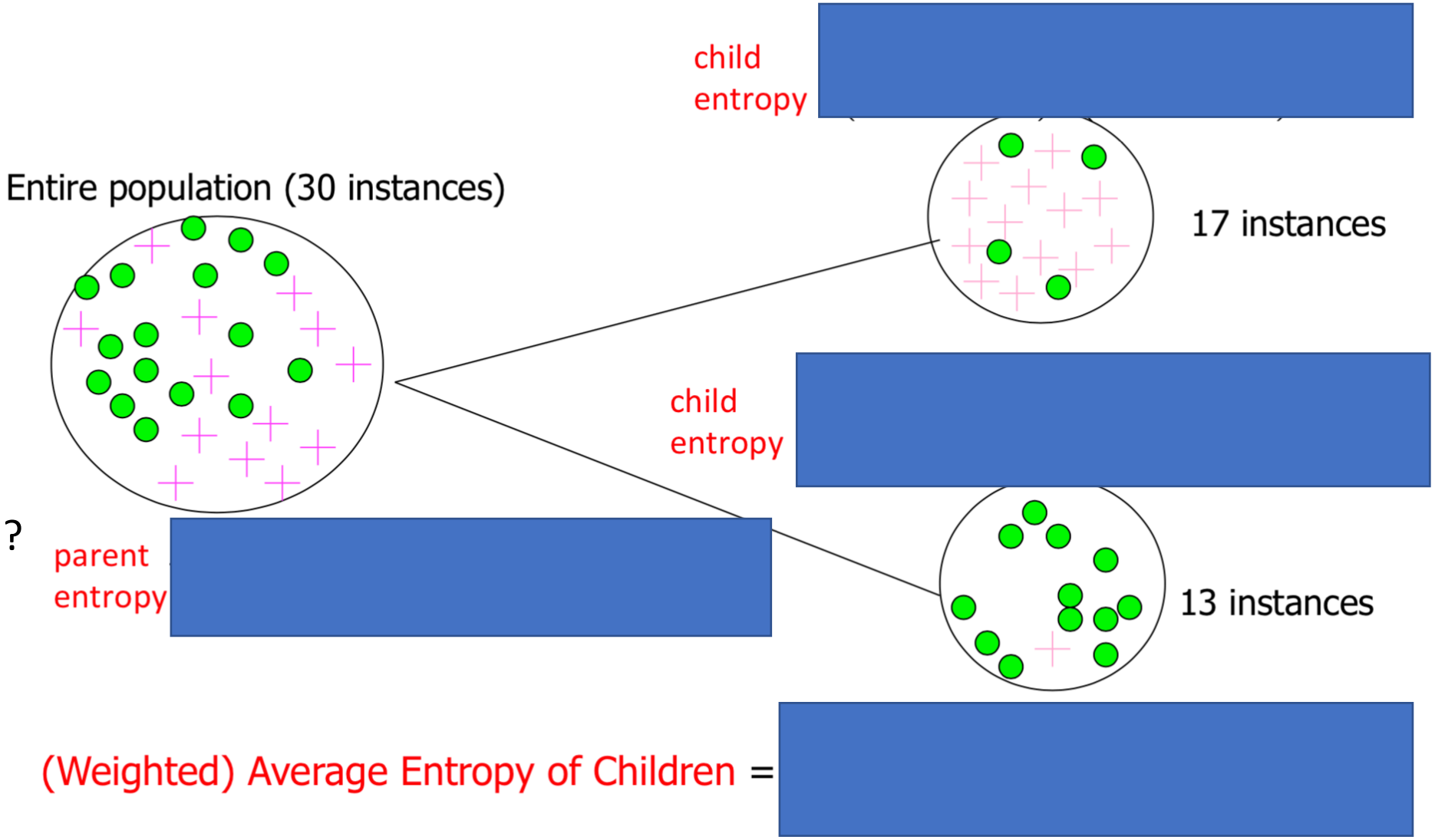
$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

# From entropy to information gain

**Information Gain** = entropy(parent) – [average entropy(children)]

**Quiz**: Compute Gains.

1. Parent $H$?
2. Child (up) $H$?
3. Child (down) $H$?
4. Information gain?

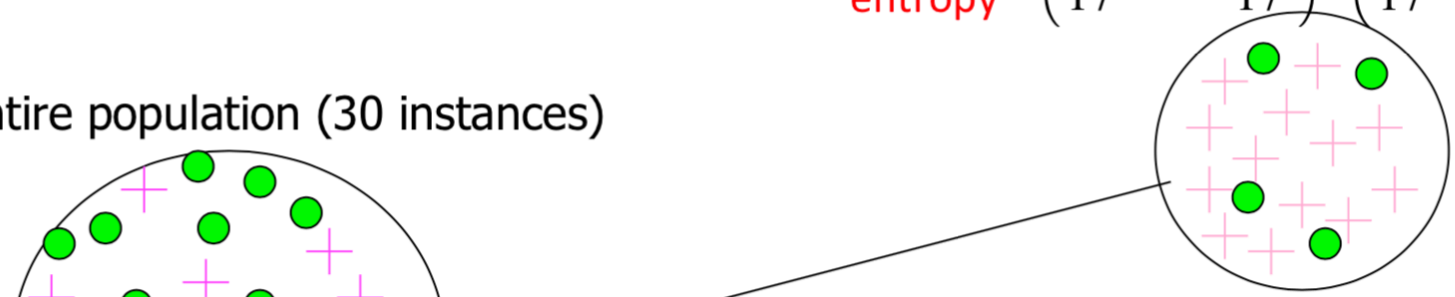



**Information Gain= 0.996 - 0.615 = 0.38**

# From entropy to information gain

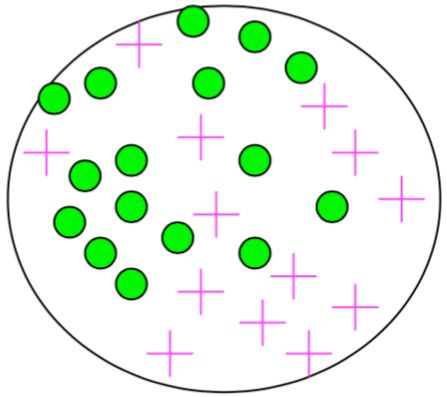
**Information Gain** = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$

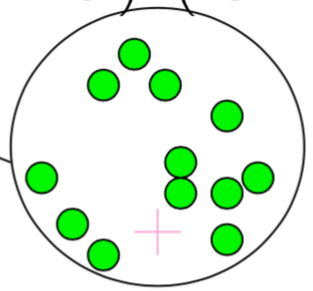


17 instances

Entire population (30 instances)



child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$



13 instances

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

**Information Gain= 0.996 - 0.615 = 0.38**