Chapter 18

Learning from Examples

The Importance of Learning

- An agent is learning if it improves its performance upon observing the world.
- Why is it important to learn?
  - The designer can not anticipate the all the situations in which the agent will be, e.g., a robot navigating a maze.
  - The designer can not anticipate all changes over time, e.g., stock market.
  - Sometimes designers have no idea how to program the solution themselves, e.g., facial recognition.

Types of Learning

- In order to learn, the agent needs to observe the world and maybe have feedback.
- The different types of feedback determine the different types of learning:
  - Supervised learning
  - Unsupervised learning
  - Semi-supervised learning
  - Reinforcement learning

Types of Learning

- Supervised learning: The agent observes a set of input-output examples (labeled examples) and learns a map from inputs to outputs.
  - Classification: output is discrete (e.g., spam email)
  - Regression: output is real-valued (e.g., stock market)
- Unsupervised learning: No explicit feedback is given, only the inputs (unlabeled examples). The agent learns patterns in the input.
  - Clustering: grouping data into K groups. (e.g. clustering images of fish into different species)

Supervised Learning

- Given a **training set** of m example input-output pairs:

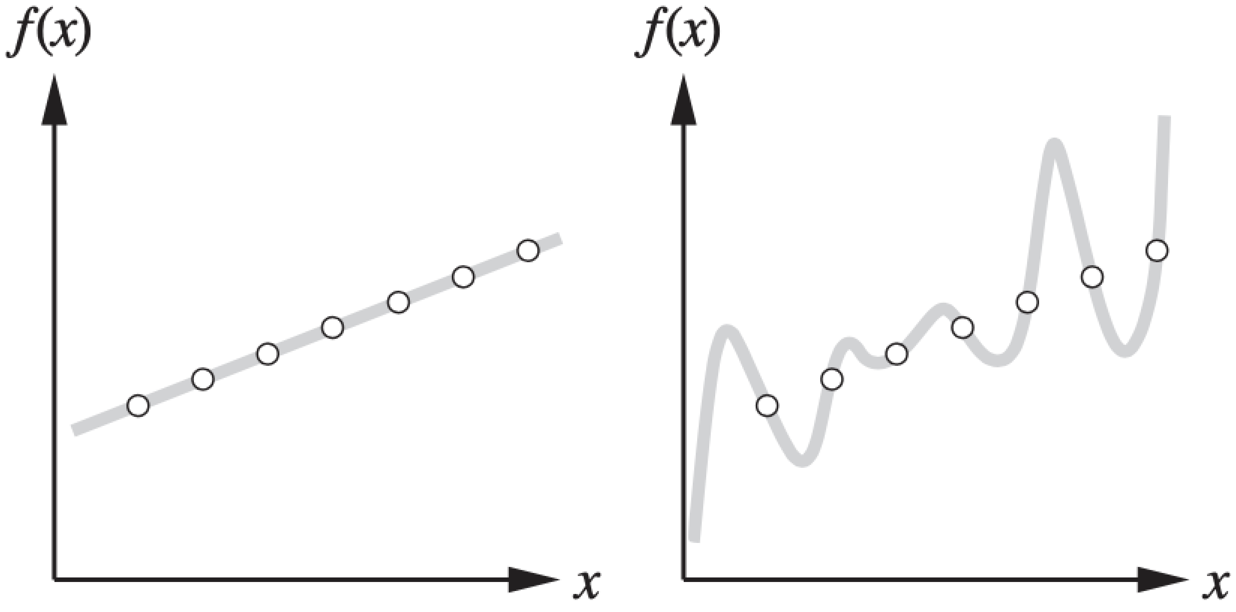
  $(x_1, y_1),\ (x_2, y_2), \dots\ (x_m, y_m),$

  where, $y_j\ = \ f(x_j)$, where f is unknown function, the goal is to find a function h that approximates f.
- The function h is called a **hypothesis**.
- How to measure the accuracy of h?
  - We give a **test set** of examples, which is different from the training set.
  - The hypothesis **generalizes** well if it correctly predicts the output for the test set.

How to Choose the Hypothesis ?

- First, select the hypothesis space: in this case, the set of polynomials.

- Ockham's razor: Choose the simplest hypothesis which is consistent with the data.



(a)

The line is consistent with the data

(b)

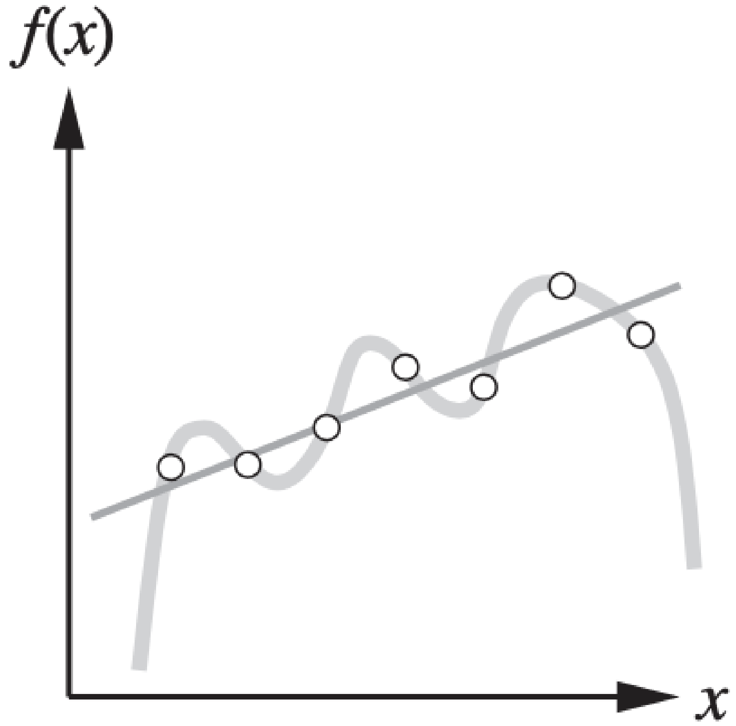
The high-degree polynomial is also consistent with the data

How to Choose the Hypothesis ?

Do we choose the line or the 6-degree polynomial?

- The line detects a pattern and will **generalize** well.
- The 6-degree polynomial does not detect any pattern.

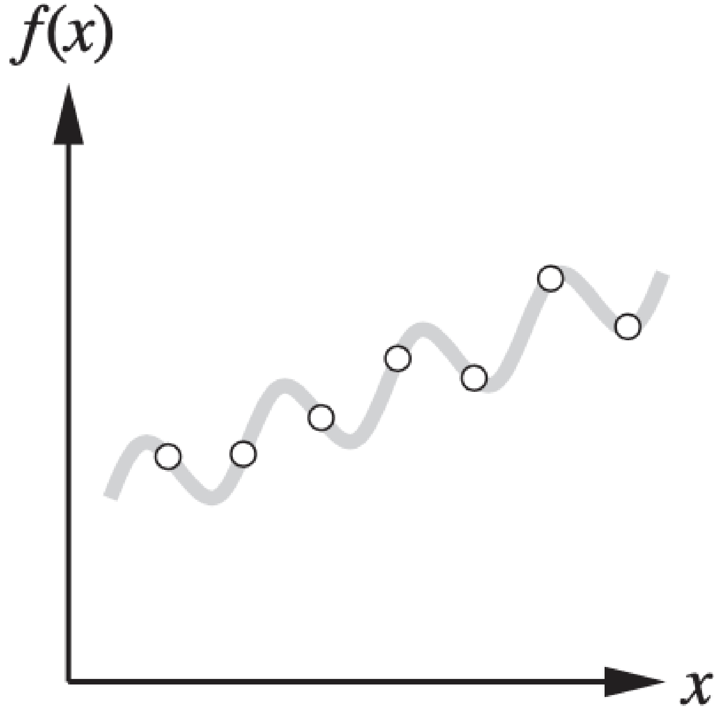
➢Choose the hypothesis that generalizes well, even if it is not consistent with the data.



(c)

How to Choose the Hypothesis ?

- $ax + b + c\sin(x)$ is consistent with the data → The choice of the hypothesis space is important.
- The learning problem is **realizable** if the hypothesis space contains the true function (we can not know this of course, because f is unknown).
- Complex hypothesis space → better hypothesis but complex search.
- Simple hypothesis space → simple search, but less good hypothesis.



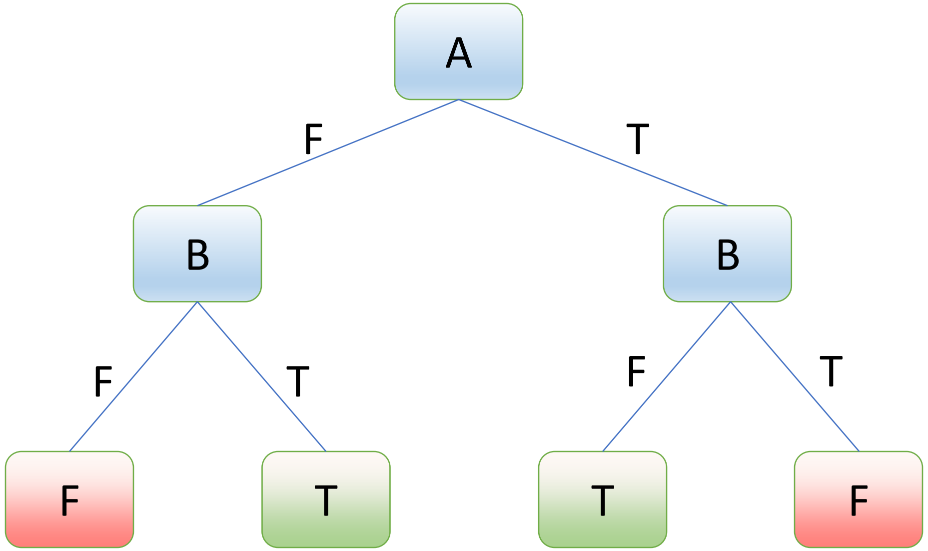
(d)

Decision Trees

- A **decision tree** represents a function f that has multiple inputs but a single output.
  - We focus on discrete input and Boolean output (**Boolean classification**)
- A decision tree reaches the decision by a set of tests on the **attributes** (the inputs).
- The internal nodes are test nodes
- The leaf nodes are the decision nodes

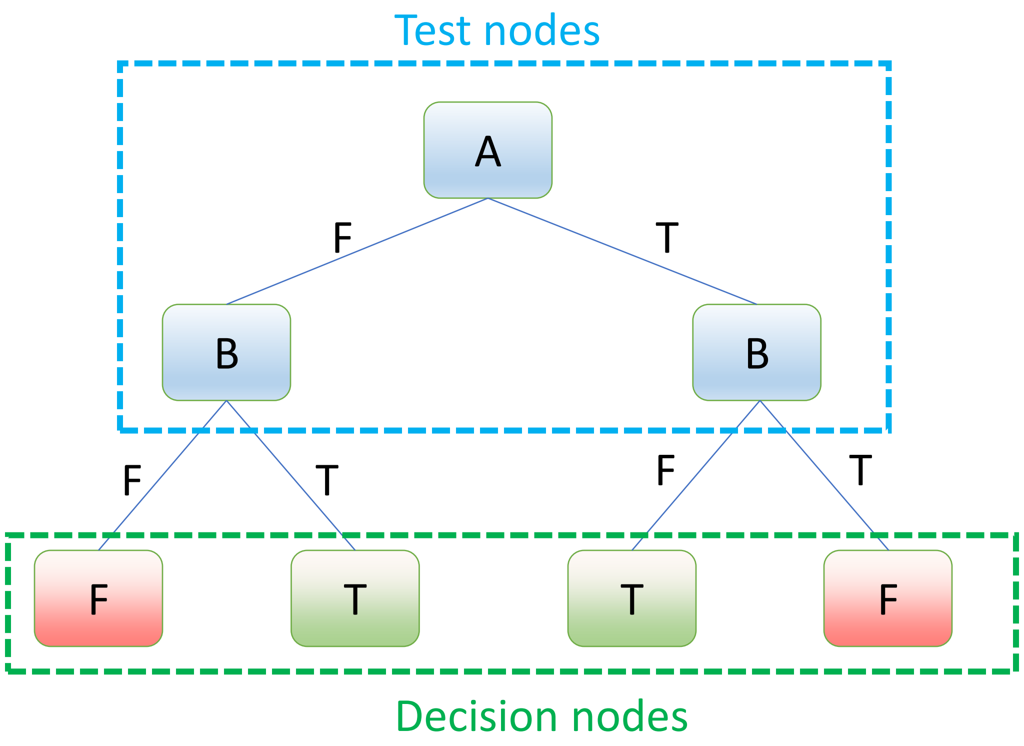
Simple Example

A	B	A xor B
True	True	False
True	False	True
False	True	True
False	False	False



Simple Example

A	B	A xor B
True	True	False
True	False	True
False	True	True
False	False	False



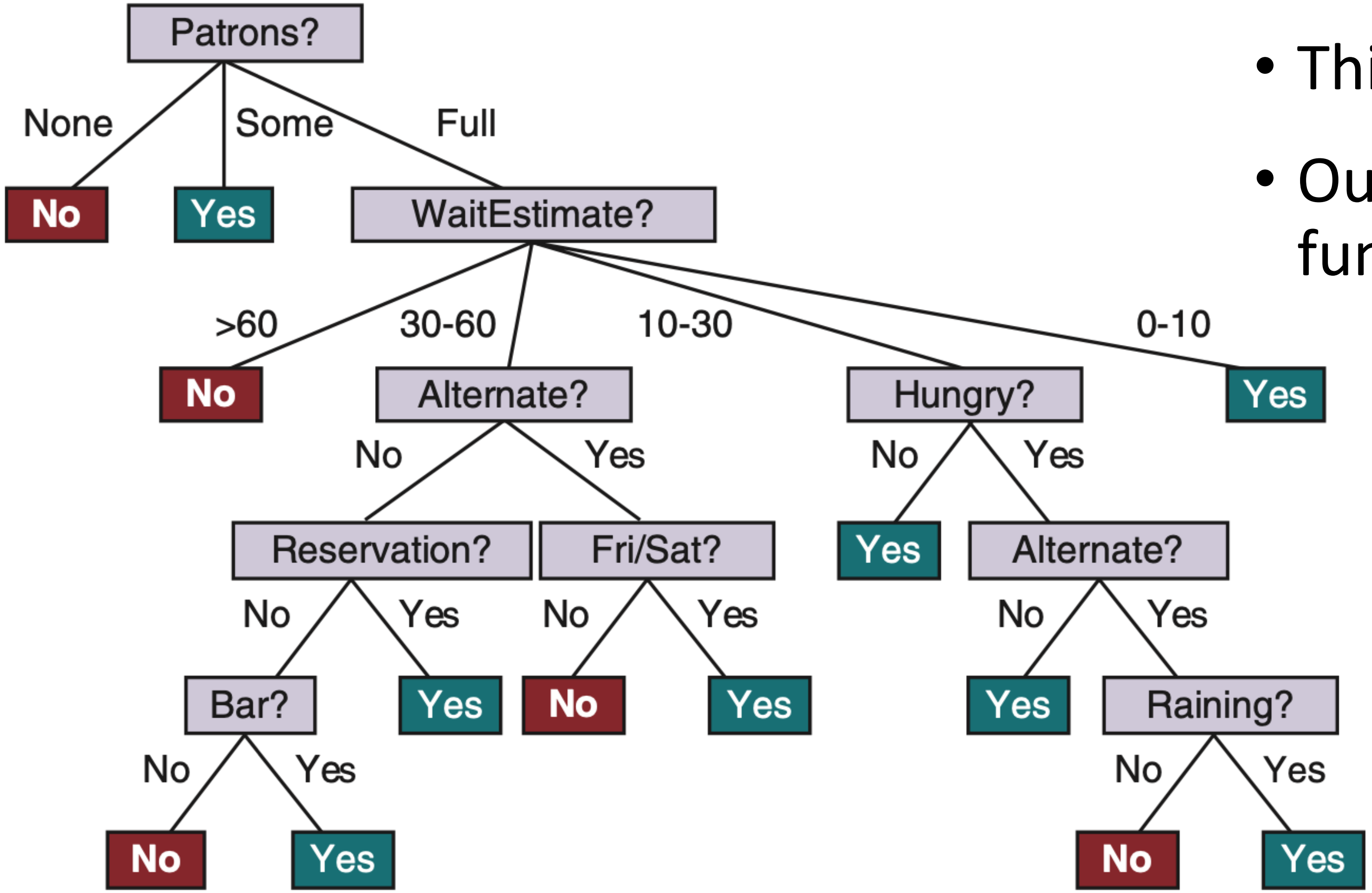
A more complex example: deciding to wait at a restaurant

- The attributes :
  1. **Alternate**: whether there is a suitable alternative restaurant nearby.
  2. **Bar**: whether the restaurant has a comfortable bar area to wait in.
  3. **Fri I Sat**: true on Fridays and Saturdays.
  4. **Hungry**: whether we are hungry.
  5. **Patrons**: how many people are in the restaurant (values are None, Some, and Full).
  6. **Price**: the restaurant's price range ($, $$, $$$).
  7. **Raining**: whether it is raining outside.
  8. **Reservation**: whether we made a reservation.
  9. **Type**: the kind of restaurant (French, Italian, Thai, or burger).
  10. **WaitEstimate**: the wait estimated by the host (0-10 minutes, 10-30, 30-60, or >60).

Decision Trees

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 =$ *Yes*
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 =$ *Yes*
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 =$ *No*
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 =$ *Yes*
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 =$ *No*
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 =$ *Yes*
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 =$ *No*
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} =$ *No*
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} =$ *No*
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} =$ *Yes*

Decision Trees

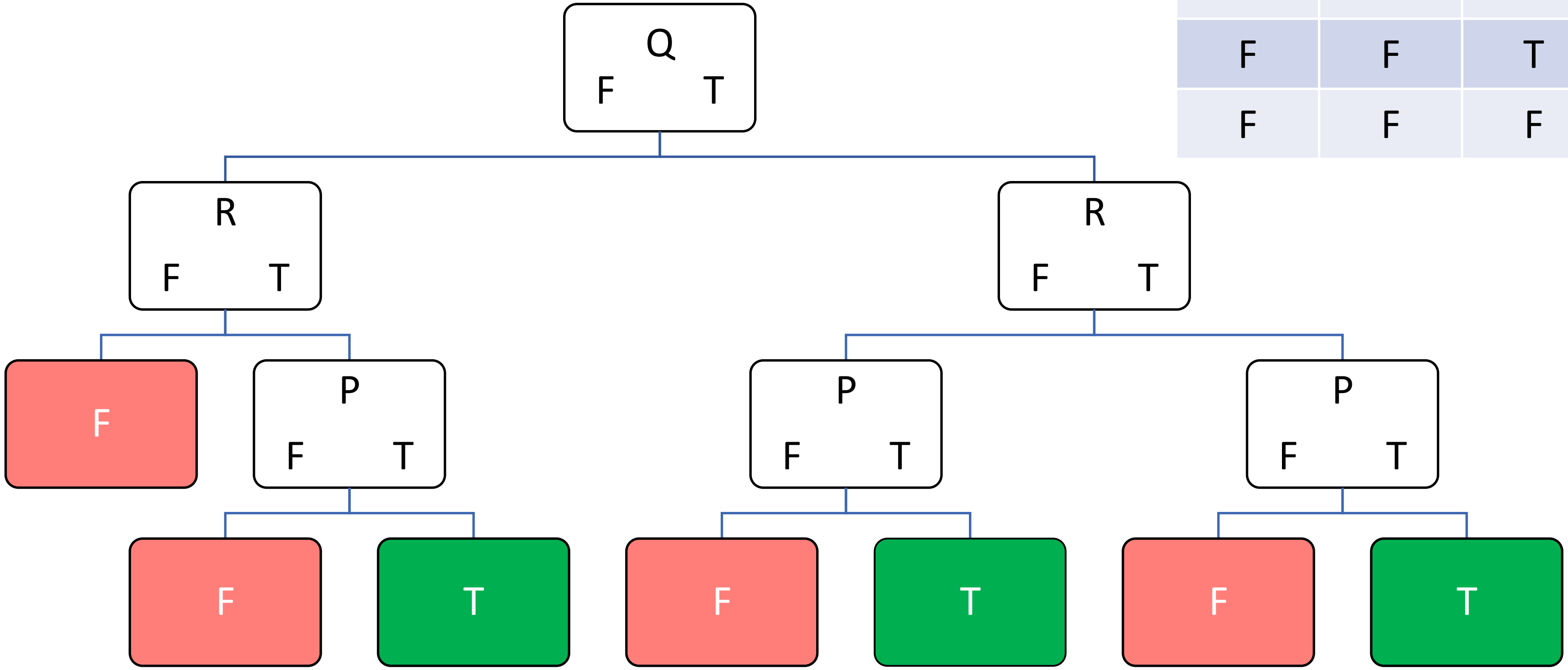


- This is the real function.
- Our goal is to learn this function from examples.

Decision Trees

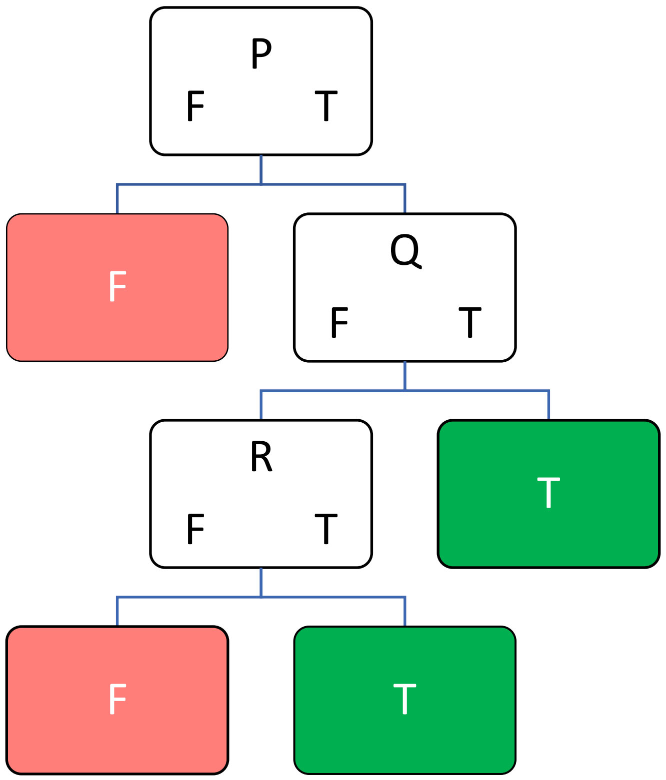
p	q	r	$q \vee r$	$p \wedge (q \vee r)$
T	T	T	T	T
T	T	F	T	T
T	F	T	T	T
T	F	F	F	F
F	T	T	T	F
F	T	F	T	F
F	F	T	T	F
F	F	F	F	F

A decision tree for the function: $P \wedge (Q \vee R)$.
The order of the attributes: Q, R, P



Decision Trees

A decision tree for the function: $P \wedge (Q \vee R)$.
The order of the attributes: P, Q, R



p	q	r	$\boldsymbol{q \vee r}$	$\boldsymbol{p \wedge (q \vee r)}$
T	T	T	T	T
T	T	F	T	T
T	F	T	T	T
T	F	F	F	F
F	T	T	T	F
F	T	F	T	F
F	F	T	T	F
F	F	F	F	F

Smaller number of nodes → The order is important

Learning Decision Trees: Example 1

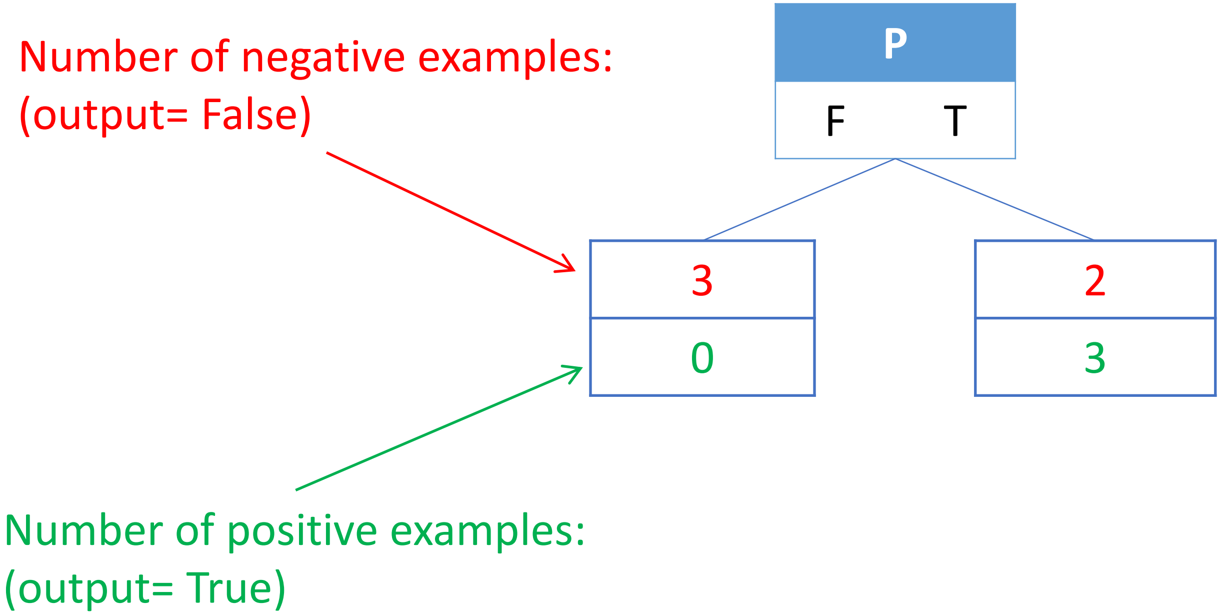
- Training set for $P \land (Q \lor R)$
- Notice that some combinations of inputs do not appear

Example	P	Q	R	Output
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	0	1
8	1	1	0	0

Noise

Learning Decision Trees: Example 1

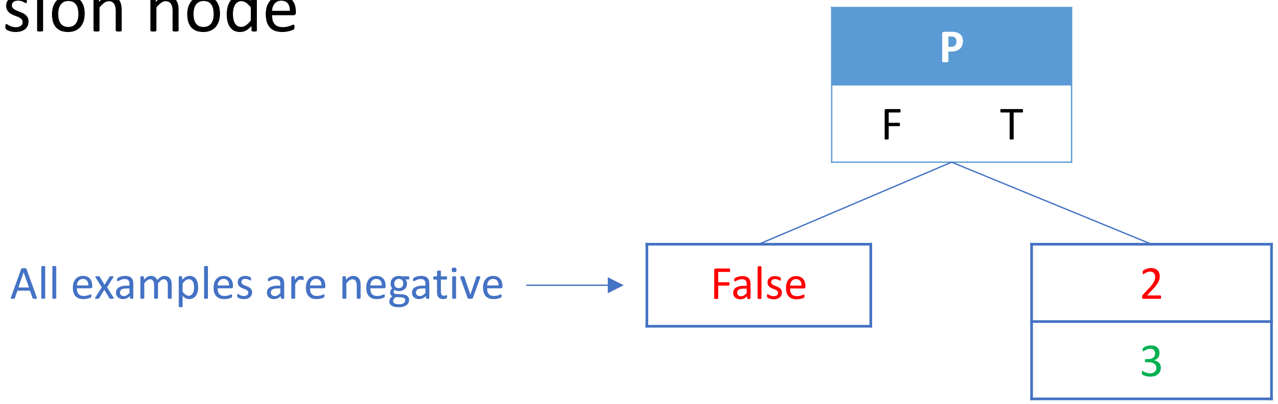
- Choose the most important attribute (how?): in this case it is P, and split the examples.



Example	P	Q	R	Output
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	0	1
8	1	1	0	0

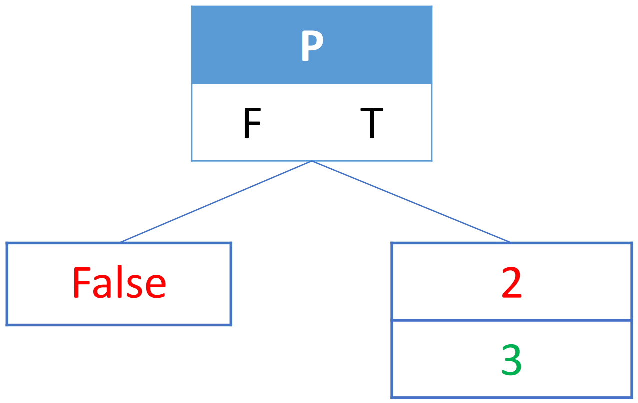
Learning Decision Trees: Example 1

- When $P = False$ all the examples have the same classification (all false) → We stop and make a decision node with the value False.



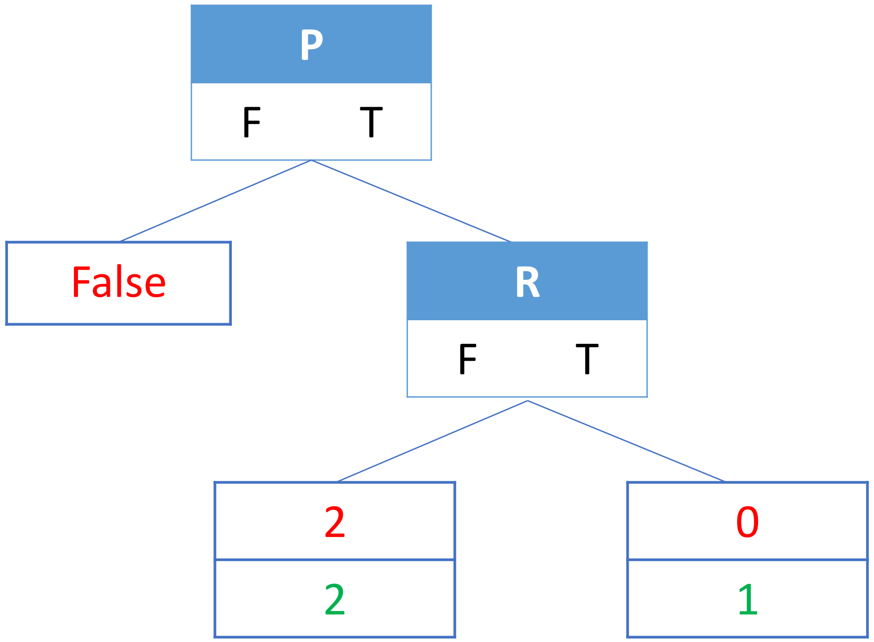
Learning Decision Trees: Example 1

- For the node $P = True$, we have both positive and negative examples, so we choose an attribute (the most important: how?)
- In this case it is R

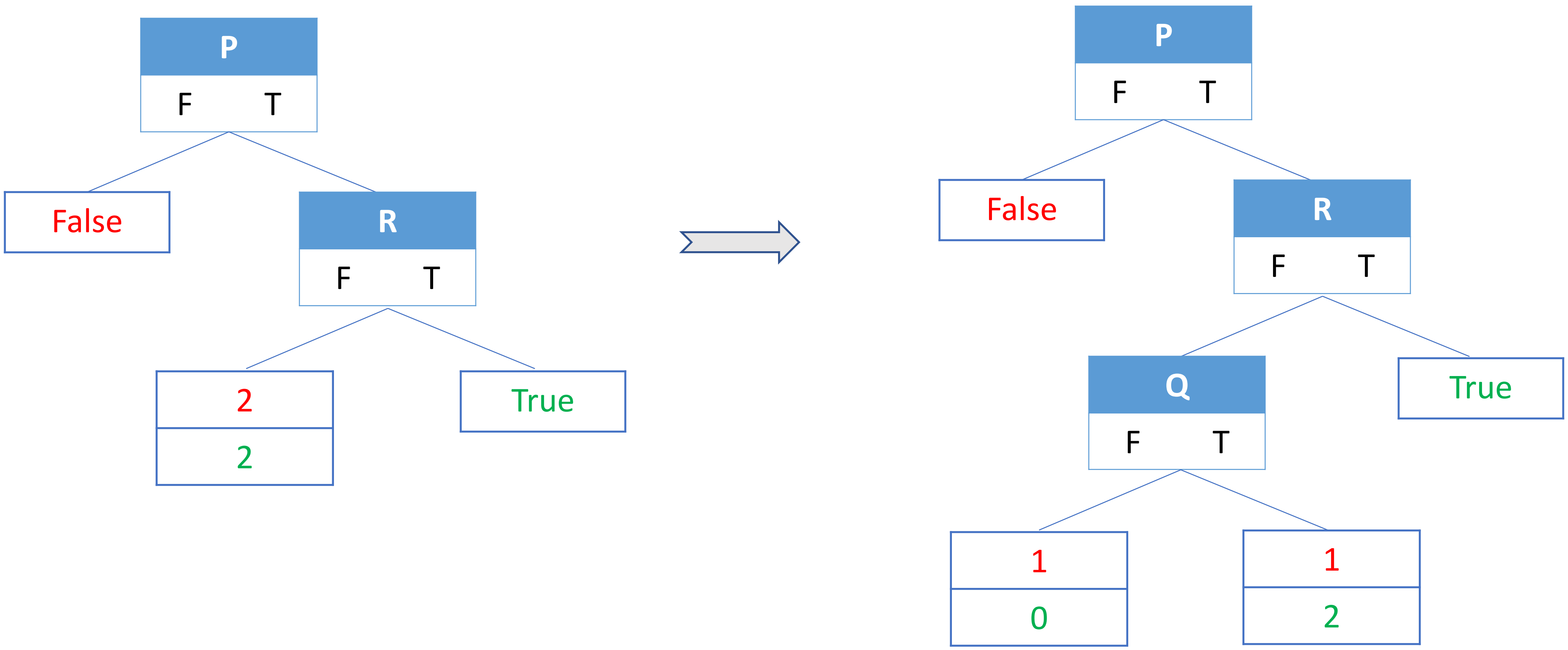


Learning Decision Trees: Example 1

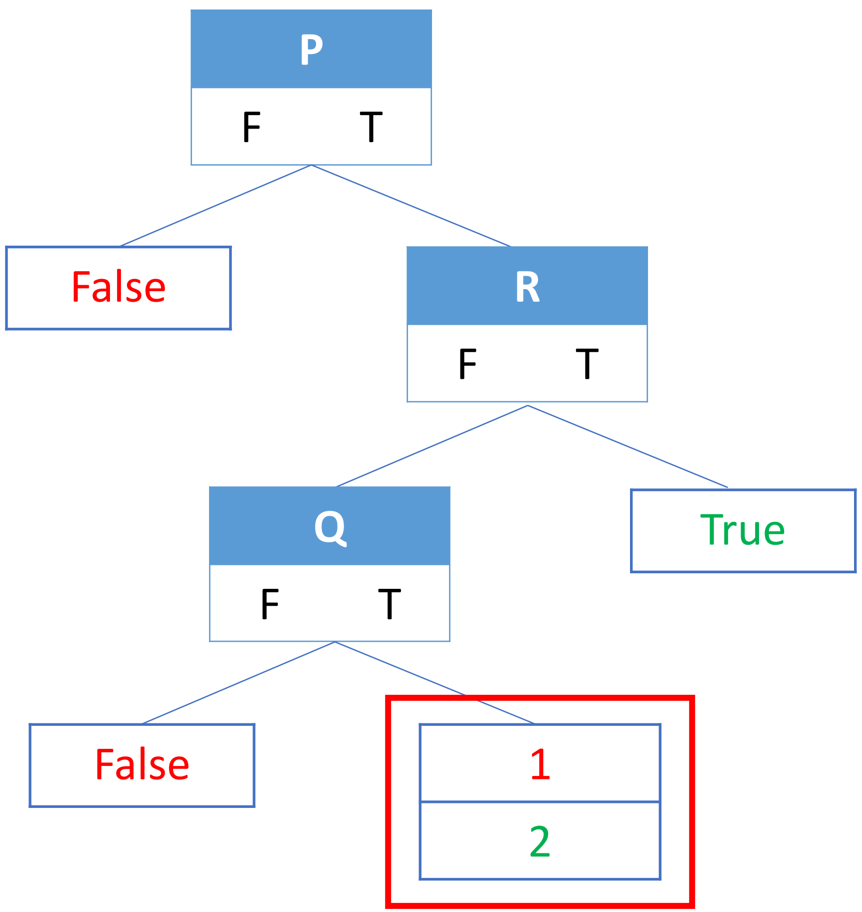
- For the node $R = True$, all the examples have the same classification (all true) → We stop and make a decision node with the value True.
- For the node $R = False$, we have both positive and negative examples, so we choose an attribute: Q



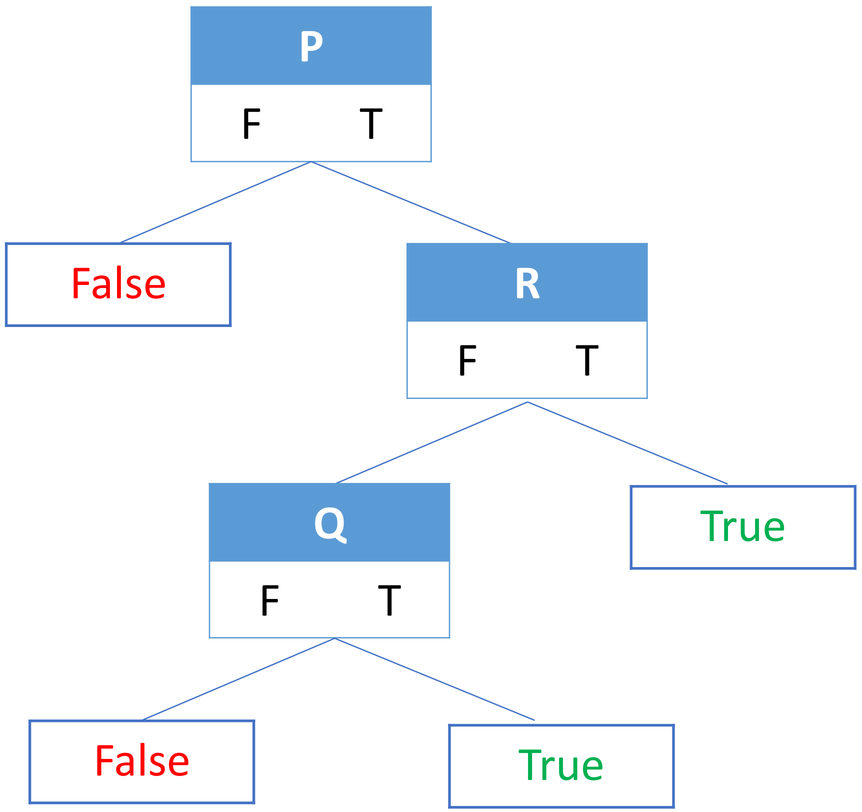
Learning Decision Trees: Example 1



Learning Decision Trees: Example 1



No attributes remaining
→ Take the majority



We obtained the true function in this case , but this will not always happen

Learning Decision Trees: Example 2

Training set for $P \wedge (Q \vee R)$

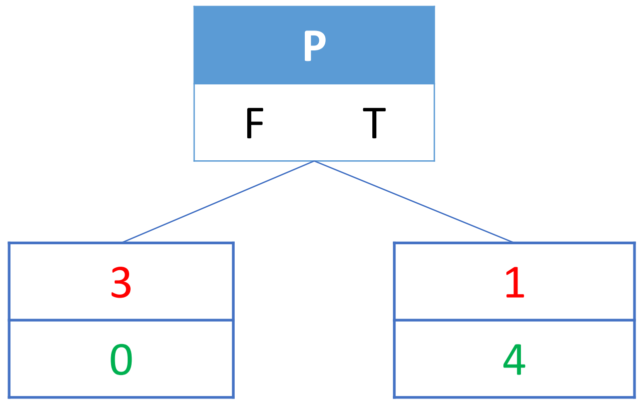
Example	P	Q	R	P ∧ (Q ∨ R)
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	1	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	0	1
8	1	1	0	0

Noise

Notice that some combinations of inputs do not appear

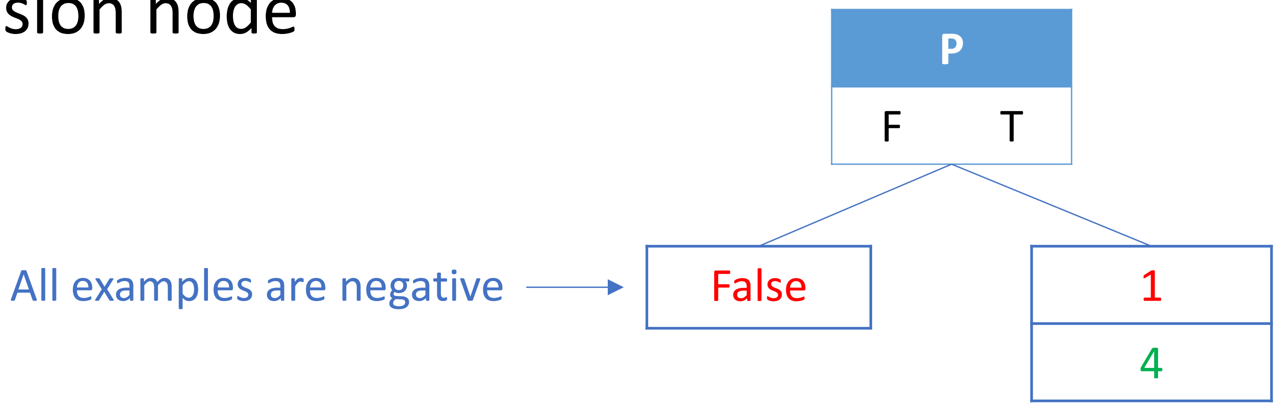
Learning Decision Trees: Example 2

- Choose the most important attribute (how?): in this case it is P, and split the examples.



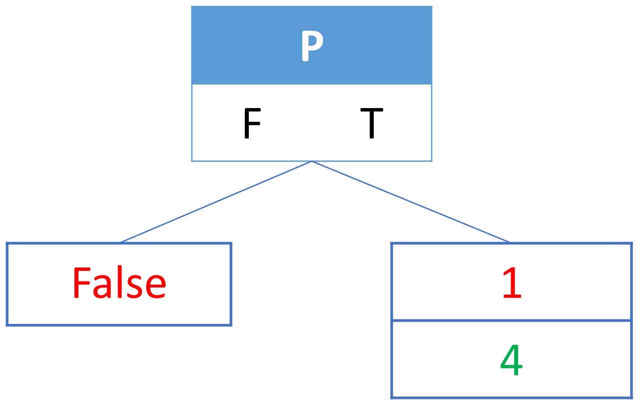
Learning Decision Trees: Example 2

- When $P = False$ all the examples have the same classification (all false) → We stop and make a decision node with the value False.



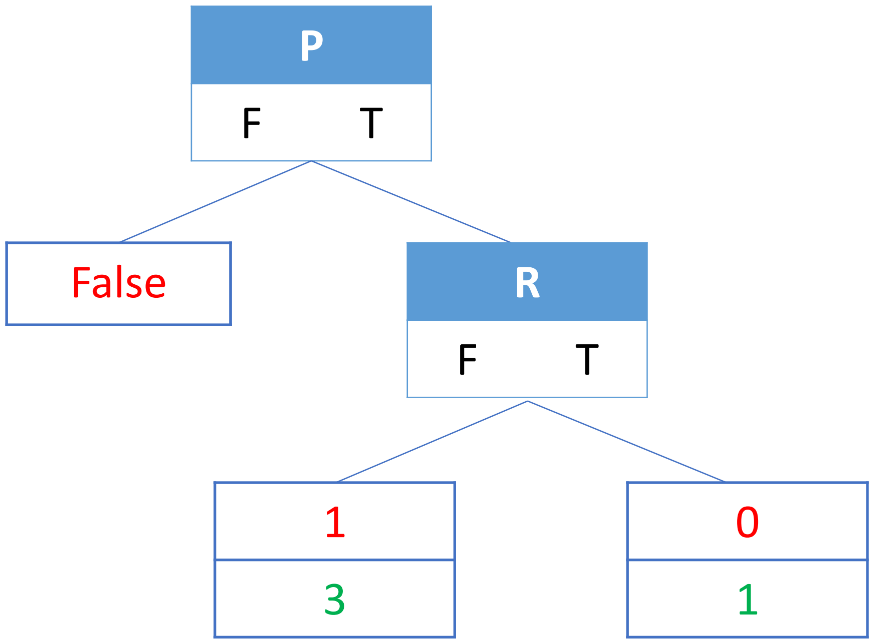
Learning Decision Trees: Example 2

- For the node $P = True$, we have both positive and negative examples, so we choose an attribute (the most important: how?)
- In this case it is R

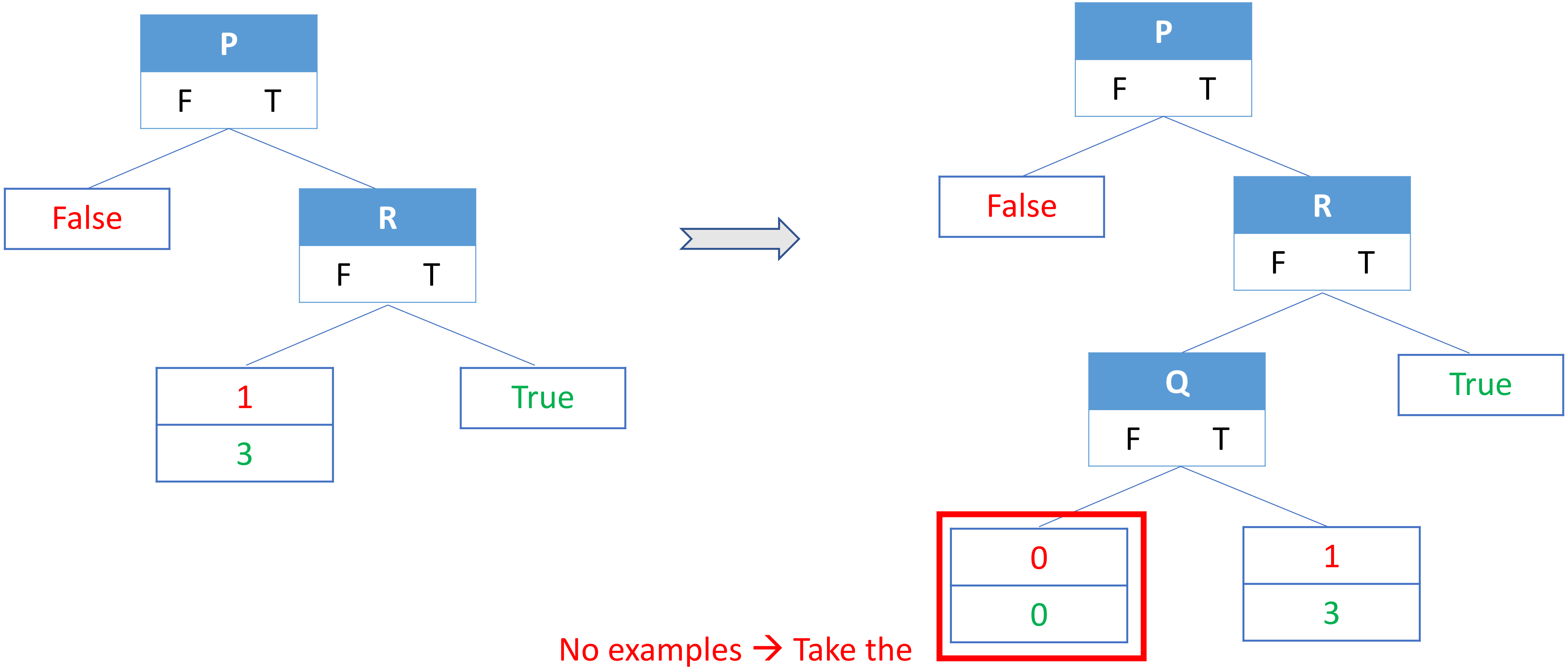


Learning Decision Trees: Example 2

- For the node $R = True$, all the examples have the same classification (all true) → We stop and make a decision node with the value True.
- For the node $R = False$, we have both positive and negative examples, so we choose an attribute: Q

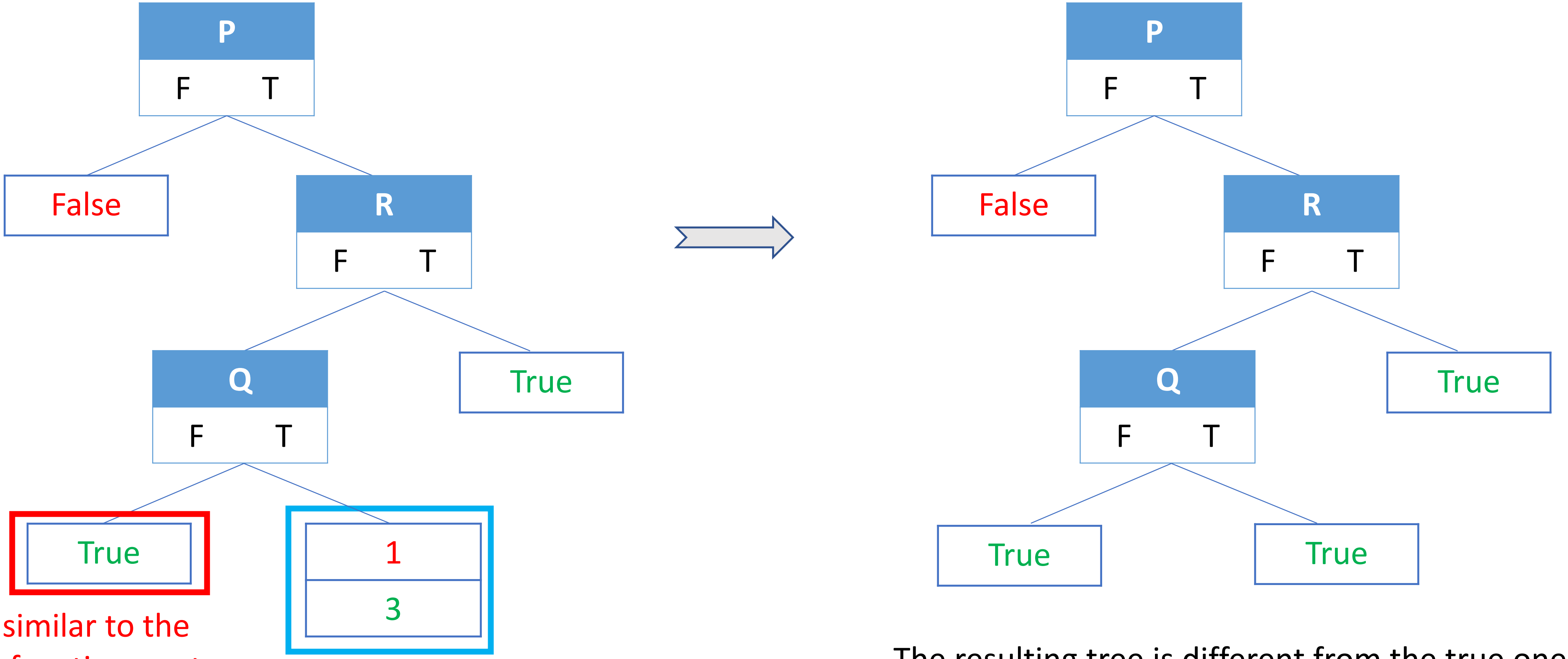


Learning Decision Trees: Example 2



No examples → Take the majority at the parent

Learning Decision Trees: Example 2



Not similar to the true function : not enough examples

No attributes (noise): Take Majority

The resulting tree is different from the true one

Choosing an attribute

- At the beginning, we have some positive (p) and negative (n) examples
- We use the notion of information gain, which is defined in terms of **entropy:**
  - Entropy is a measure of the uncertainty of a random variable
  - More information $\Longrightarrow$ less entropy
- In general, the entropy of a random variable V with values v_k, each with probability $P(v_k)$, is defined as:

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k)$$

What does the function look like?

- A variable with two options: True and False

- If I am not sure if it is true or false:

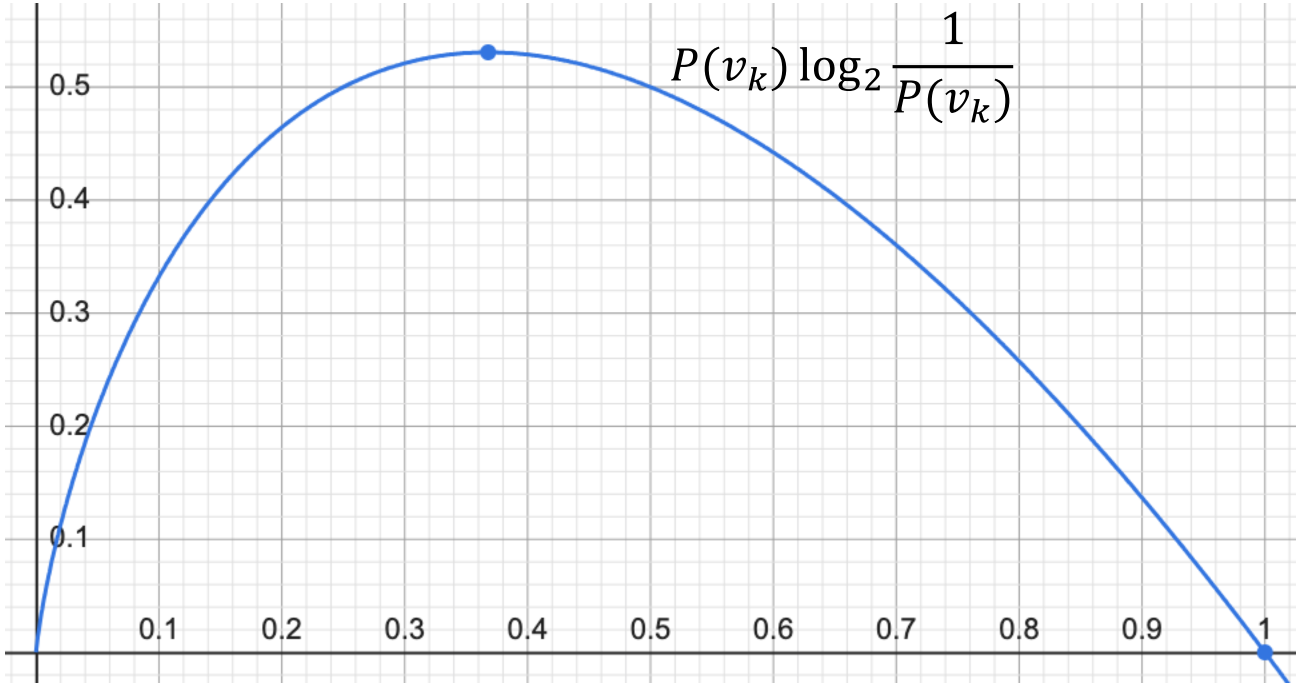
$$-(0.5 \log_2 0.5 \; + \; 0.5 \log_2 0.5) = 1$$

➢ High Entropy, Low information

- I am 99% sure it is true:

$$-(0.99 \log_2 0.99 \; + \; 0.01 \log_2 0.01) \; \approx \; 0.08$$

➢ Low Entropy, High information



Students

Entropy

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k)$$

- Entropy of a fair coin flip is 1 bit:

$$H(Fair) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1\ bit$$

- If the coin is loaded to give 99% heads:

$$H(Loaded) = -(0.99 \log_2 0.99 + 0.01 \log_2 0.01) \approx 0.08\ bits$$

- We define $B(q)$ as the entropy of a Boolean random variable that is true with probability q:

$$B(q) = -(q \log_2 q + (1-q) \log_2 (1-q))$$

Choosing an attribute

- If a training set contains p positive examples and n negative examples, then the entropy of the goal attribute on the whole set is

$$H(Goal) = B\left(\frac{p}{p+n}\right)$$

- Choose the attribute that gives the largest information possible about the function
  - We choose the attribute which if tested gives the maximum reduction in entropy (maximum gain in information).

Students

Choosing an attribute

- Each attribute has k possible values
  - For an RGB attribute, $k = 3$, Red or Green or Blue
  - For the function $P \wedge (Q \vee R)$, $k = 2$
- For each value k, we have a set of positive (p_k) and negative (n_k) examples
  - For attribute P, $k = 2$, 0 and 1
  - $P = 0$: $p_0 = 0, n_0 = 3$
  - $P = 1$: $p_1 = 3, n_1 = 2$

Example	P	Q	R	Output
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	0	1
8	1	1	0	0

Students

Choosing an attribute

- The entropy for each branch is:

$$B\left(\frac{p_k}{p_k + n_k}\right)$$

- 0: $p_0 = 0, n_0 = 3$
- $B\left(\frac{p_0}{p_0+n_0}\right) = B\left(\frac{0}{3}\right)$
- 1: $p_1 = 3, n_1 = 2$
- $B\left(\frac{p_1}{p_1+n_1}\right) = B\left(\frac{3}{5}\right)$

Example	P	Q	R	Output
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	0	1
8	1	1	0	0

Choosing an attribute

- But the expected entropy depends on the branch
- So, we also need the probability of going down either branch:

$$prob = \frac{p_k + n_k}{p + n}$$

- Branch 0: $\frac{p_0+n_0}{p+n} = \frac{3}{8}$
- Branch 1: $\frac{p_1+n_1}{p+n} = \frac{5}{8}$

Example	P	Q	R	Output
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	0	1
8	1	1	0	0

Choosing an attribute

- The expected entropy remaining after testing an attribute A is:

$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right)$$

- The **information gain** from the attribute test on A is the expected reduction in entropy:

$$Gain(A) = B\left(\frac{p}{p + n}\right) - Remainder(A)$$

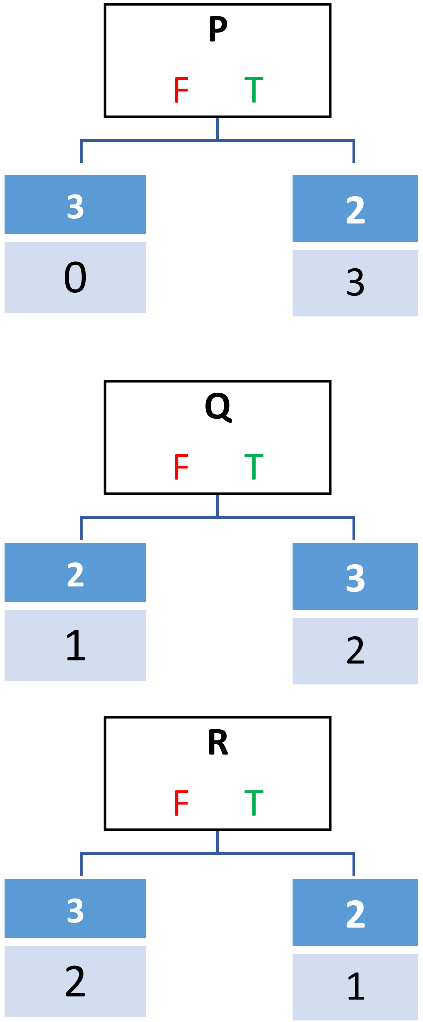
Students

Choosing Attributes: Example

$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right)$$

- $Remainder(P) = \left(\frac{0+3}{8}\right) * B\left(\frac{0}{3}\right) + \left(\frac{3+2}{8}\right) * B\left(\frac{3}{5}\right) = 0.6$
- $Remainder(Q) = \left(\frac{1+2}{8}\right) * B\left(\frac{1}{3}\right) + \left(\frac{2+3}{8}\right) * B\left(\frac{2}{5}\right) = 0.95$
- $Remainder(R) = \left(\frac{2+3}{8}\right) * B\left(\frac{2}{5}\right) + \left(\frac{1+2}{8}\right) * B\left(\frac{1}{3}\right) = 0.95$

➢ We choose P

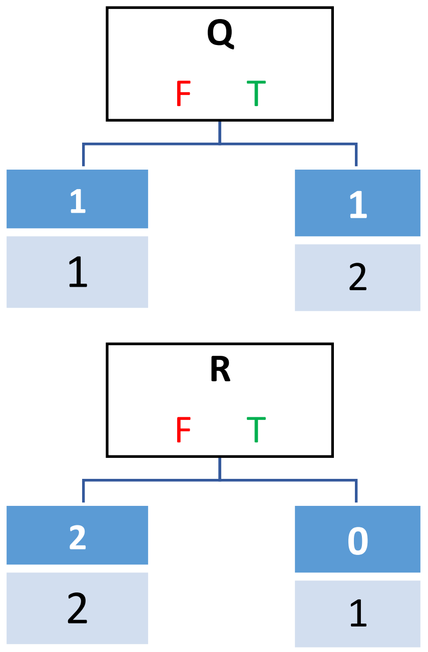


Choosing Attributes: Example

$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right)$$

- $Remainder(Q) = \left(\frac{1+1}{5}\right) * B\left(\frac{1}{2}\right) + \left(\frac{2+1}{5}\right) * B\left(\frac{2}{3}\right) = 0.95$
- $Remainder(R) = \left(\frac{2+2}{5}\right) * B\left(\frac{2}{4}\right) + \left(\frac{1+0}{5}\right) * B\left(\frac{1}{1}\right) = 0.8$

➢ We choose R



Greedy algorithm for learning decision trees

Recursive Algorithm:

1. If the remaining **examples** are **all** positive (or **all** negative), then we are done: we can answer Yes or No.
2. If there are **some** positive and **some** negative examples, then choose the best attribute to split them.
3. If there are **no** examples left, it means that no example has been observed for this combination, and we return a **default value: the plurality classification** of all the examples that were used in constructing the node's parent.
   - take the most frequent class in the parent node and return that as your prediction
4. If there are **no attributes** left, but both positive and negative examples, it means that these examples have exactly the same description, but different classifications. We return a **default value:** the **plurality classification of the remaining examples**.

Students

Learning Decision Trees (ID3)

function DECISION-TREE-LEARNING(*examples*, *attributes*, *parent_examples*) **returns** a tree

if *examples* is empty **then return** PLURALITY-VALUE(*parent_examples*)
else if all *examples* have the same classification **then return** the classification
else if *attributes* is empty **then return** PLURALITY-VALUE(*examples*)
else
    $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, *examples*)
    tree ← a new decision tree with root test A
    for each value v_k of A **do**
        $exs \leftarrow \{e : e \in examples$ **and** $e.A = v_k\}$
        subtree ← DECISION-TREE-LEARNING(*exs*, $attributes - A$, *examples*)
        add a branch to *tree* with label $(A = v_k)$ and subtree *subtree*
    return *tree*