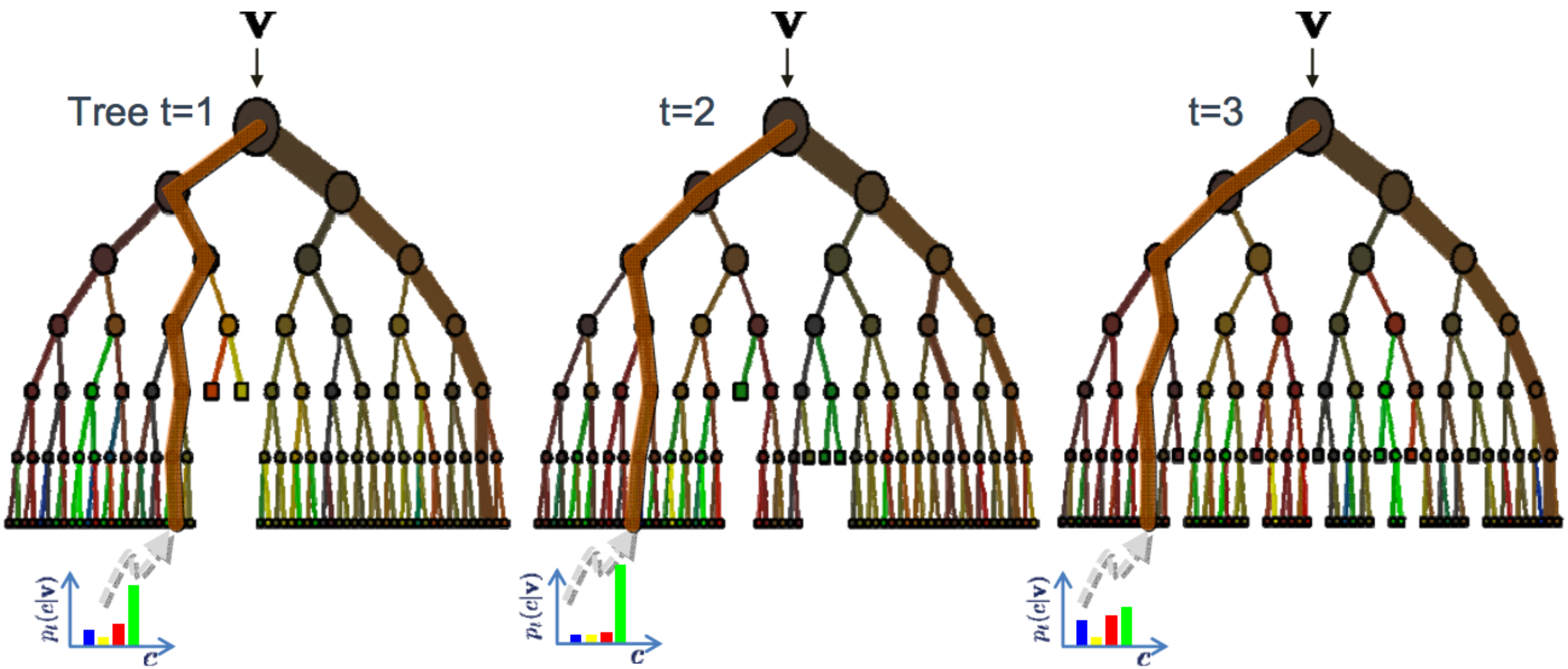
Decision Trees & Random Forests



Attributes

Target attribute

Name	Balance	Age	Employed	Write-off
Mike	$200,000	42	no	yes
Mary	$35,000	33	yes	no
Claudio	$115,000	40	no	no
Robert	$29,000	23	yes	yes
Dora	$72,000	31	no	no

This is one row (example).
Feature vector is: **<Claudio,115000,40,no>**
Class label (value of Target attribute) is **no**

$$\mathit{entropy} = -p_1 \log(p_1) - p_2 \log(p_2) - \cdots$$

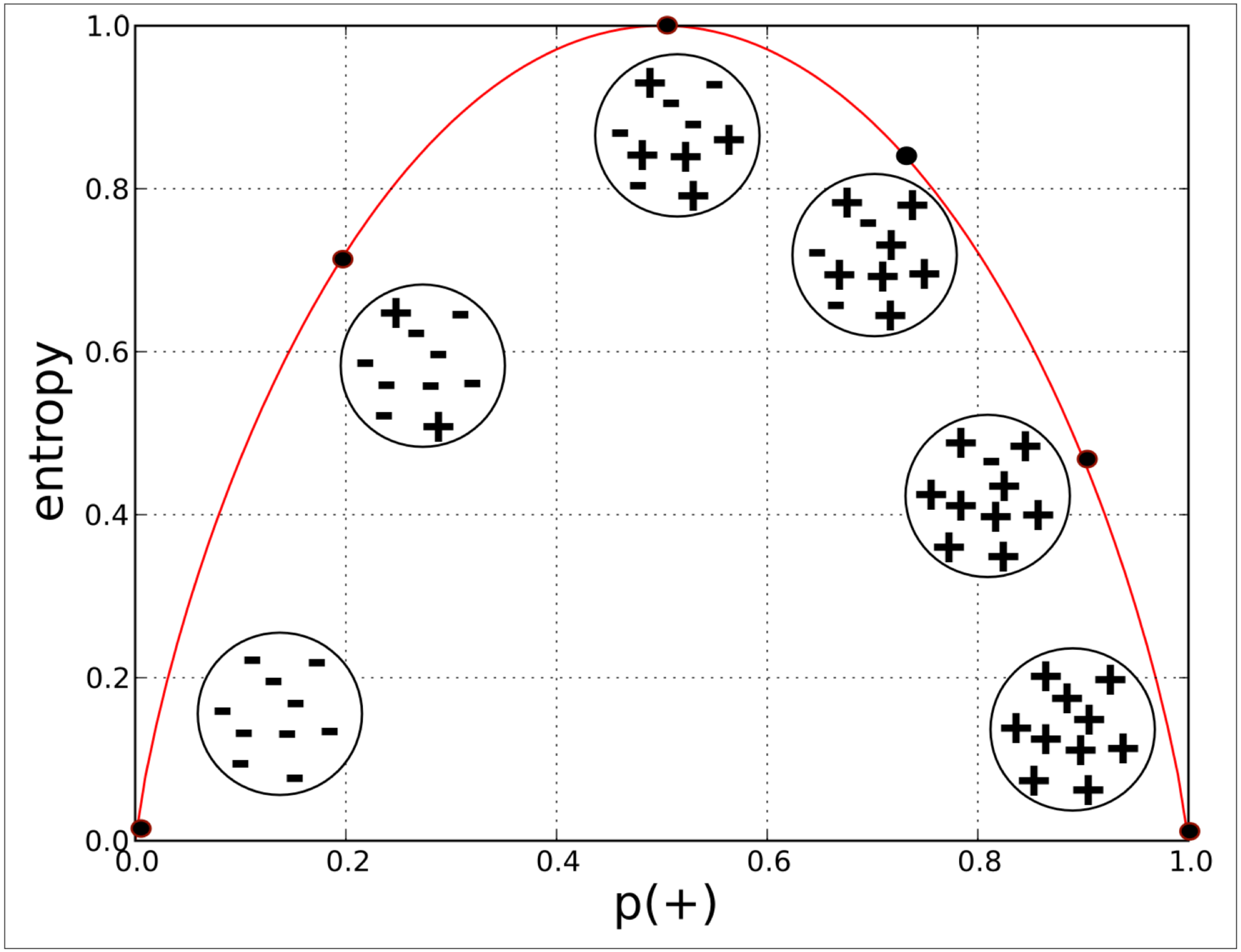
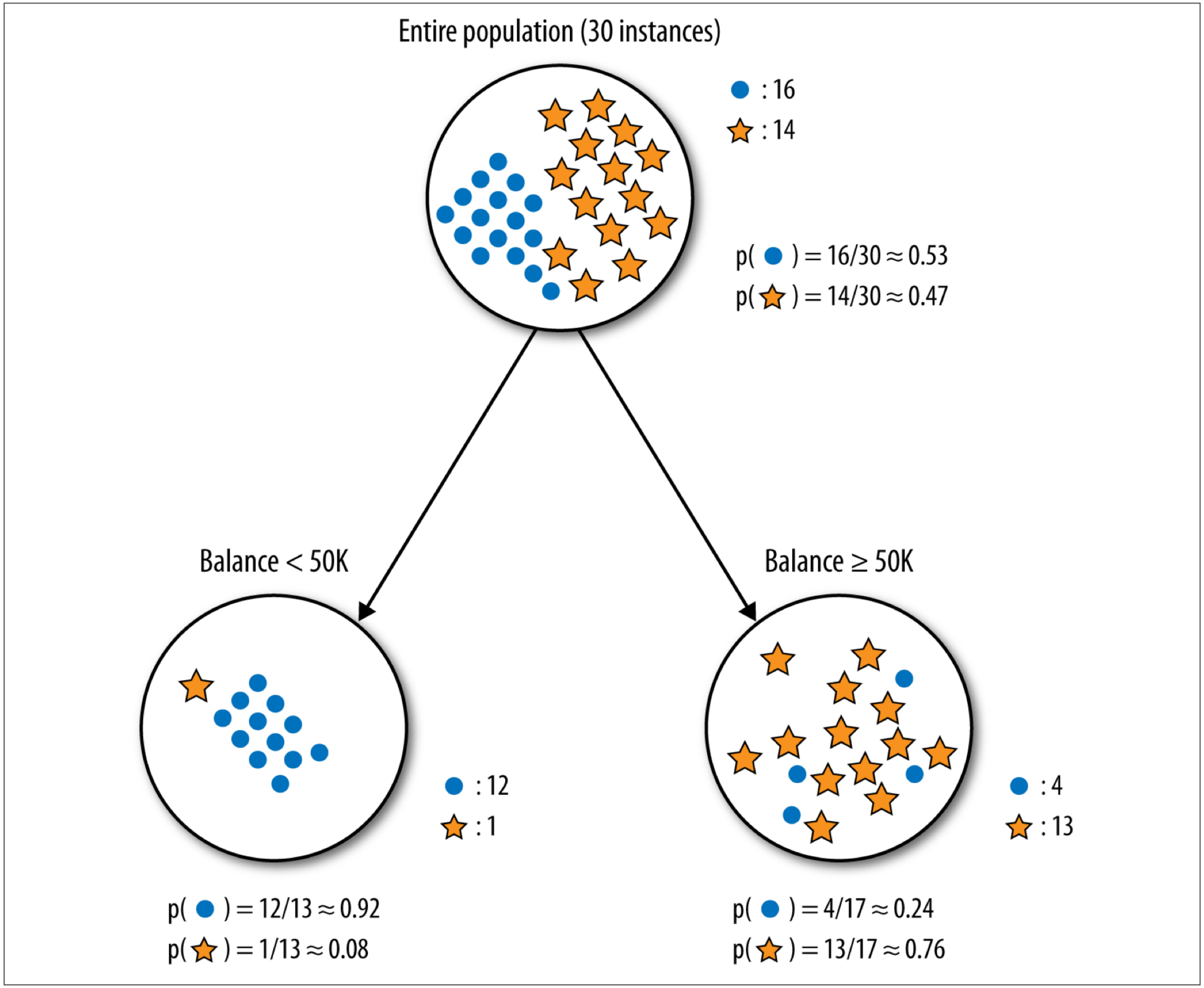


Figure 3-3. Entropy of a two-class set as a function of p(+).

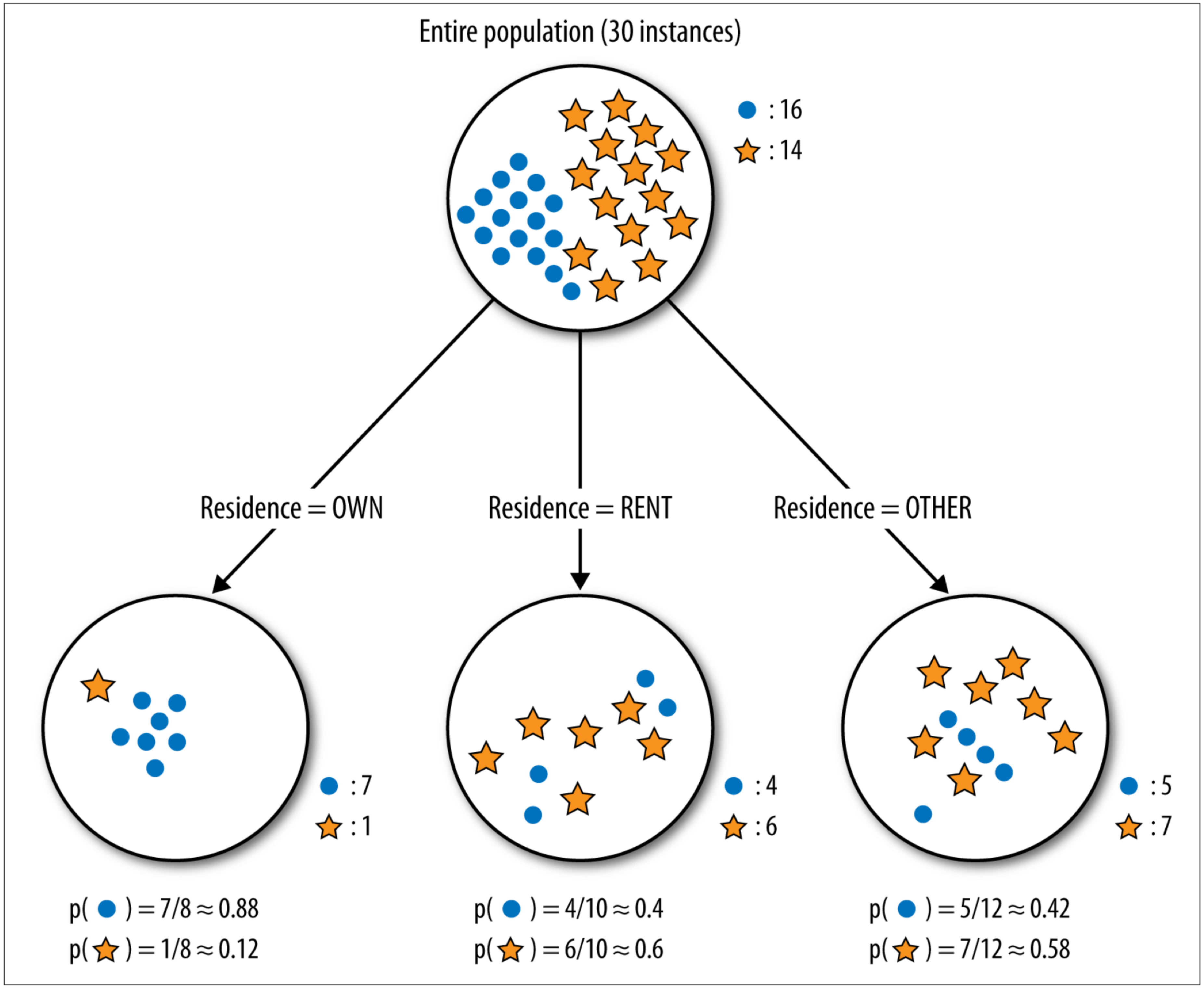
As a concrete example, consider a set S of 10 people with seven of the *non-write-off* class and three of the *write-off* class. So:

$$p(\text{non-write-off}) = 7 / 10 = 0.7$$
$$p(\text{write-off}) = 3 / 10 = 0.3$$

$$\begin{aligned} entropy(S) &= -[0.7 \times \log_2 (0.7) + 0.3 \times \log_2 (0.3)] \\ &\approx -[0.7 \times -0.51 + 0.3 \times -1.74] \\ &\approx 0.88 \end{aligned}$$



Data Science for Business - Foster Provost & Tom Fawcett


Entire population (30 instances)
● : 16
☆ : 14
Residence = OWN
Residence = RENT
Residence = OTHER
● : 7
☆ : 1
● : 4
☆ : 6
● : 5
☆ : 7
p(●) = 7/8 ≈ 0.88
p(☆) = 1/8 ≈ 0.12
p(●) = 4/10 ≈ 0.4
p(☆) = 6/10 ≈ 0.6
p(●) = 5/12 ≈ 0.42
p(☆) = 7/12 ≈ 0.58

Split on Balance > 50K

$$\begin{aligned} IG &= entropy(parent) - [\,p(\text{Balance} < 50\text{K}) \times entropy(\text{Balance} < 50\text{K}) \\ &\qquad + p(\text{Balance} \geq 50\text{K}) \times entropy(\text{Balance} \geq 50\text{K})] \\ &\approx 0.99 - [0.43 \times 0.39 + 0.57 \times 0.79] \\ &\approx 0.37 \end{aligned}$$

Split on Residence

$$\begin{aligned} entropy(parent) &\approx 0.99 \\ entropy(\text{Residence=OWN}) &\approx 0.54 \\ entropy(\text{Residence=RENT}) &\approx 0.97 \\ entropy(\text{Residence=OTHER}) &\approx 0.98 \\ IG &\approx 0.13 \end{aligned}$$

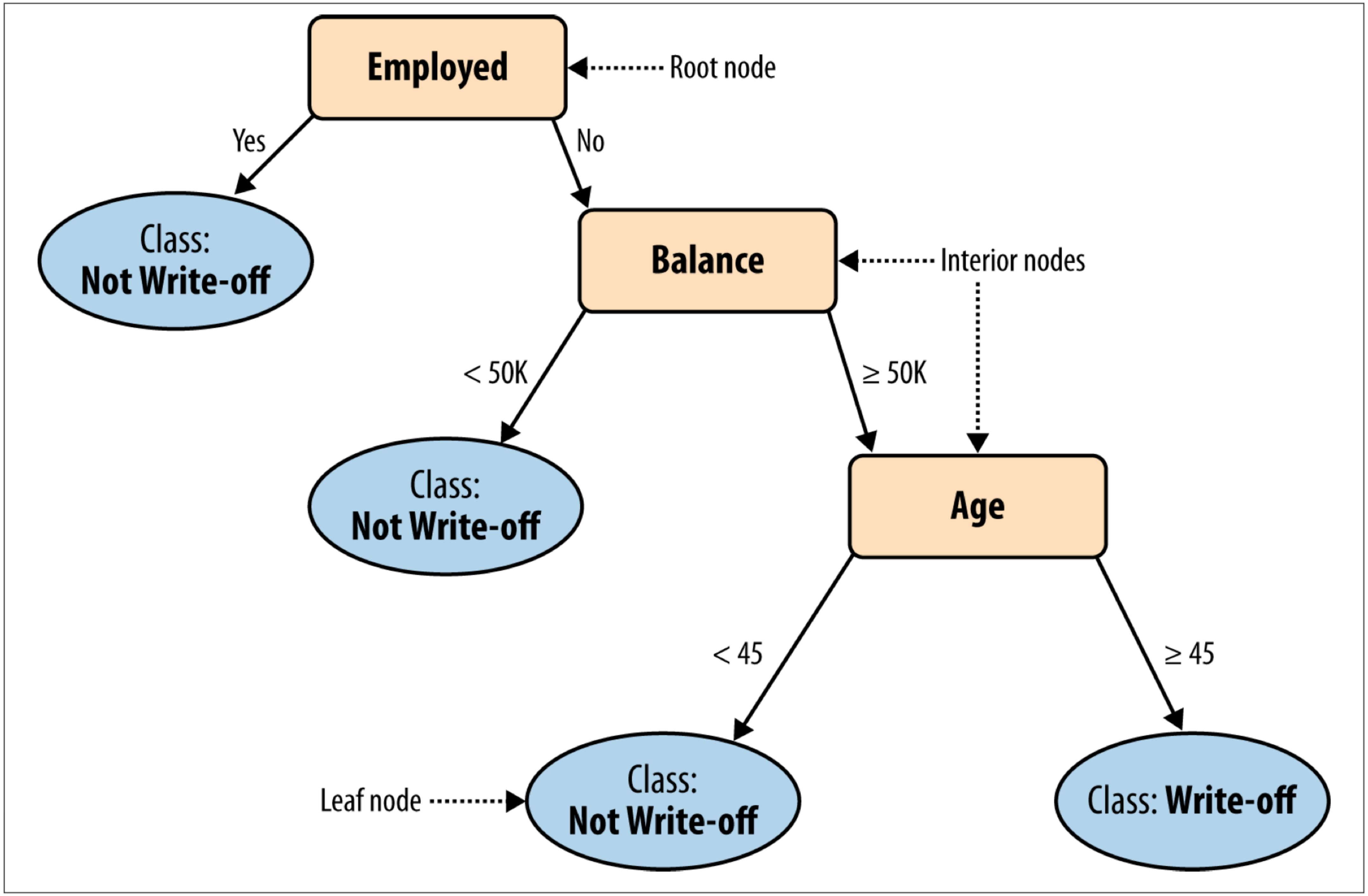


Figure 3-10. A simple classification tree.

Bias and Variance

- We train a model with the goal of fitting it correctly to the data
- When a model can't express a complex function, it risks **underfitting** the data, and we say it has high **bias**
    - E.g. doing a linear regression through a sine wave
- When a model can express a complex function, it risks **overfitting** the data, and we say it has high **variance**
    - E.g. a decision tree, which can represent arbitrarily complex functions

For a formal definition of bias and variance, see Thomas Dietterich's paper on the subject

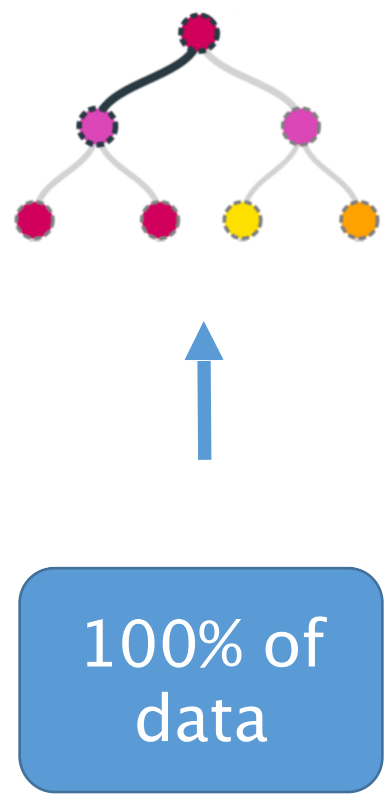
Decision trees

- Can represent complex functions
  - But they are prone to overfitting; they have high variance
  - A single tree can be led astray by outliers
- We can address this problem by:
  - Taking several **bootstrap samples** from the original data set
  - Training a decision tree on each sample
  - For classification, trees vote on the class
  - For regression, average the results from the different trees
- Goal: Get the expressiveness of a decision tree, with less overfitting

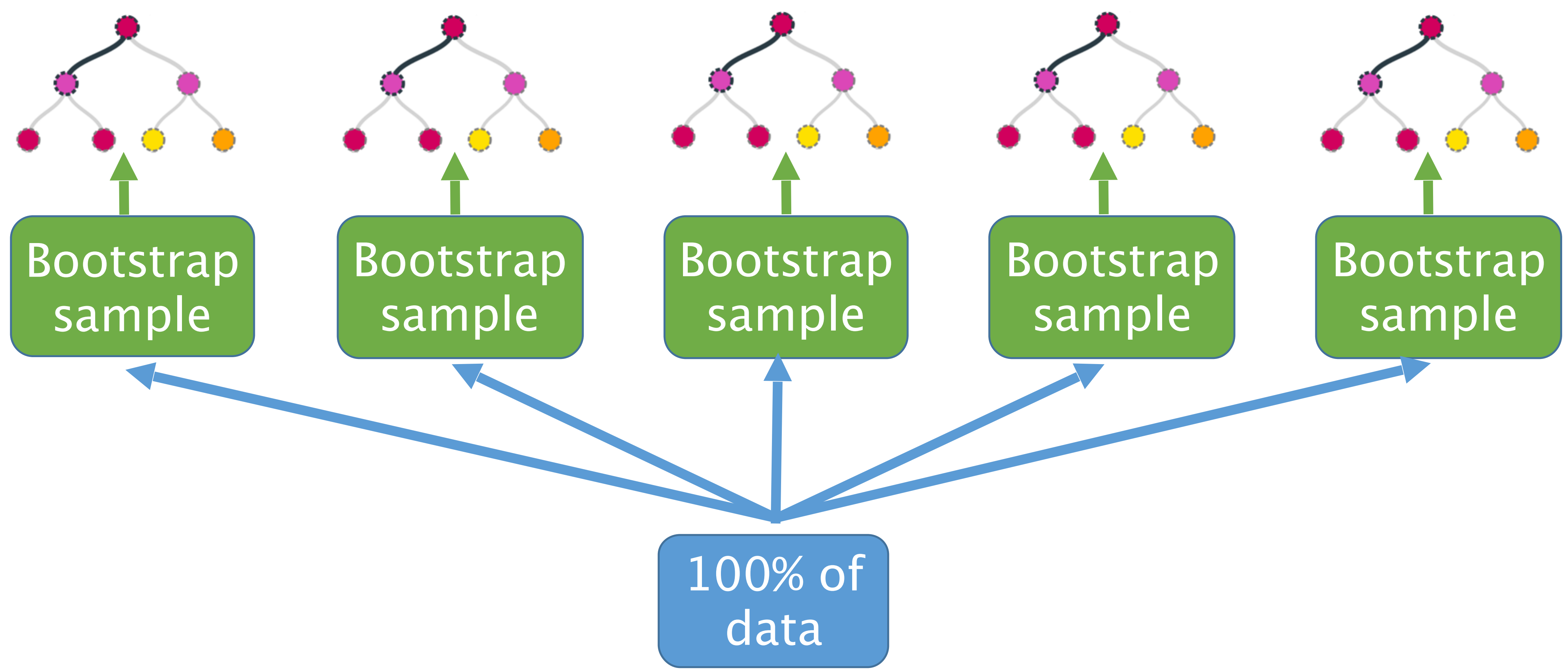
What's a Bootstrap Sample?

- A bootstrap sample has two characteristics:
  - Sampled **with replacement**
  - Number of items in the sample is equal to the number of instances in the dataset
- Because we sample with replacement, an instance can be selected more than once, or not at all
- On average, a bootstrap sample will contain 63.2% of the instances in the original dataset
  - 36.8% of the time, an instance is selected 0 times

Single tree



Bootstrap Aggregating ("Bagging")



Outlier Protection

- Suppose you have a "Joker" in your data set, an outlier that is utterly useless for prediction
- If you train a **single** decision tree on all of the data, the Joker can mess up your predictions
- If you use an **ensemble** of trees trained on bootstrap samples, then the Joker has to clear two hurdles
  - First, it has to make it in to each bootstrap sample (only a 63.2% chance)
  - Second, after clearing this first hurdle, it has to mess up enough models to impact the vote on the class

Feature Importances