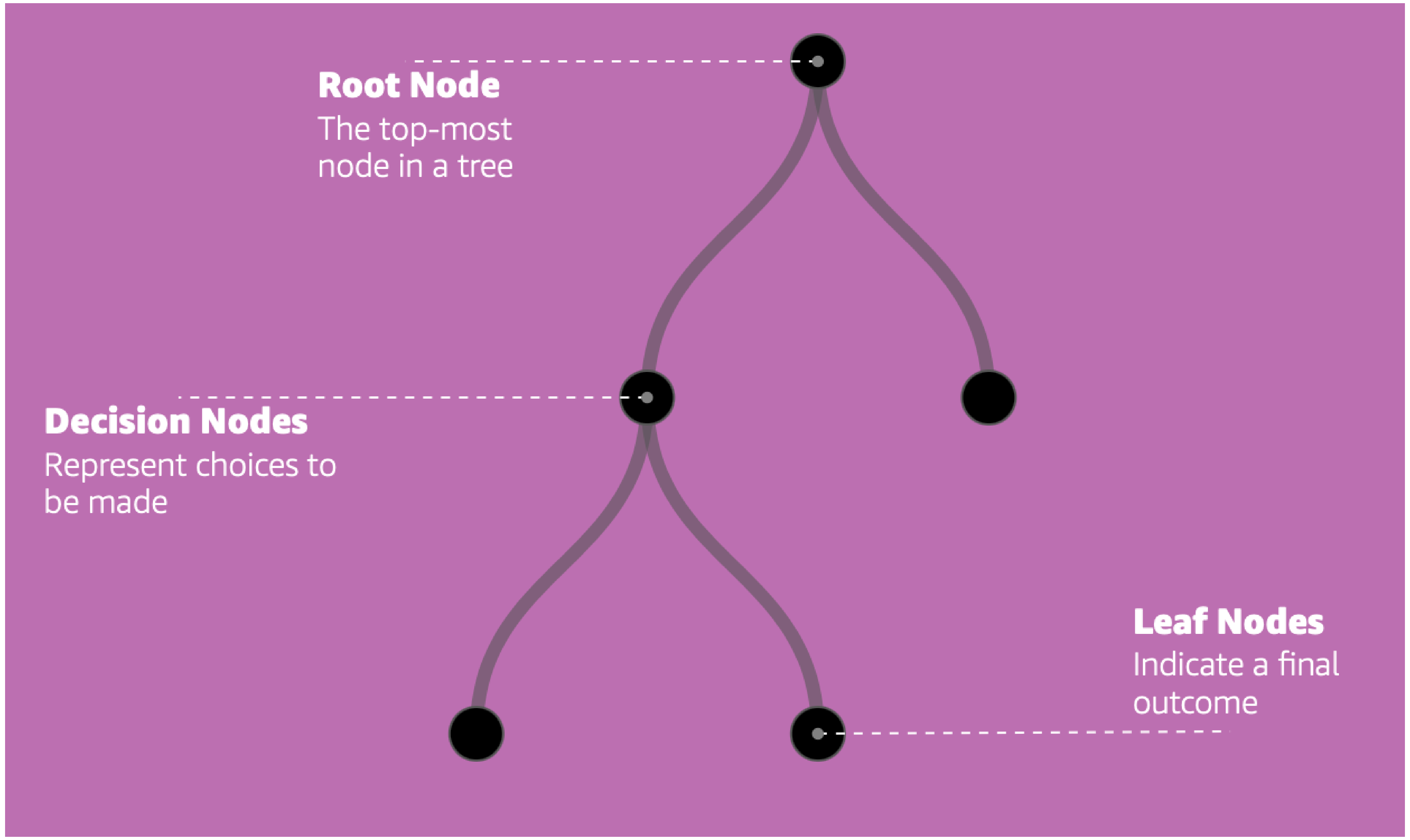
DataC'EPT

# TREE BASED MODELS

by Iheb Chaabane

# DECISION TREES



## DECISION TREES

DECISION TREES ARE WIDELY USED ALGORITHMS FOR **SUPERVISED** MACHINE LEARNING. THEY'RE POPULAR FOR THEIR EASE OF INTERPRETATION AND LARGE RANGE OF APPLICATIONS. THEY WORK FOR BOTH **REGRESSION AND CLASSIFICATION PROBLEMS**.

A DECISION TREE CONSISTS OF :

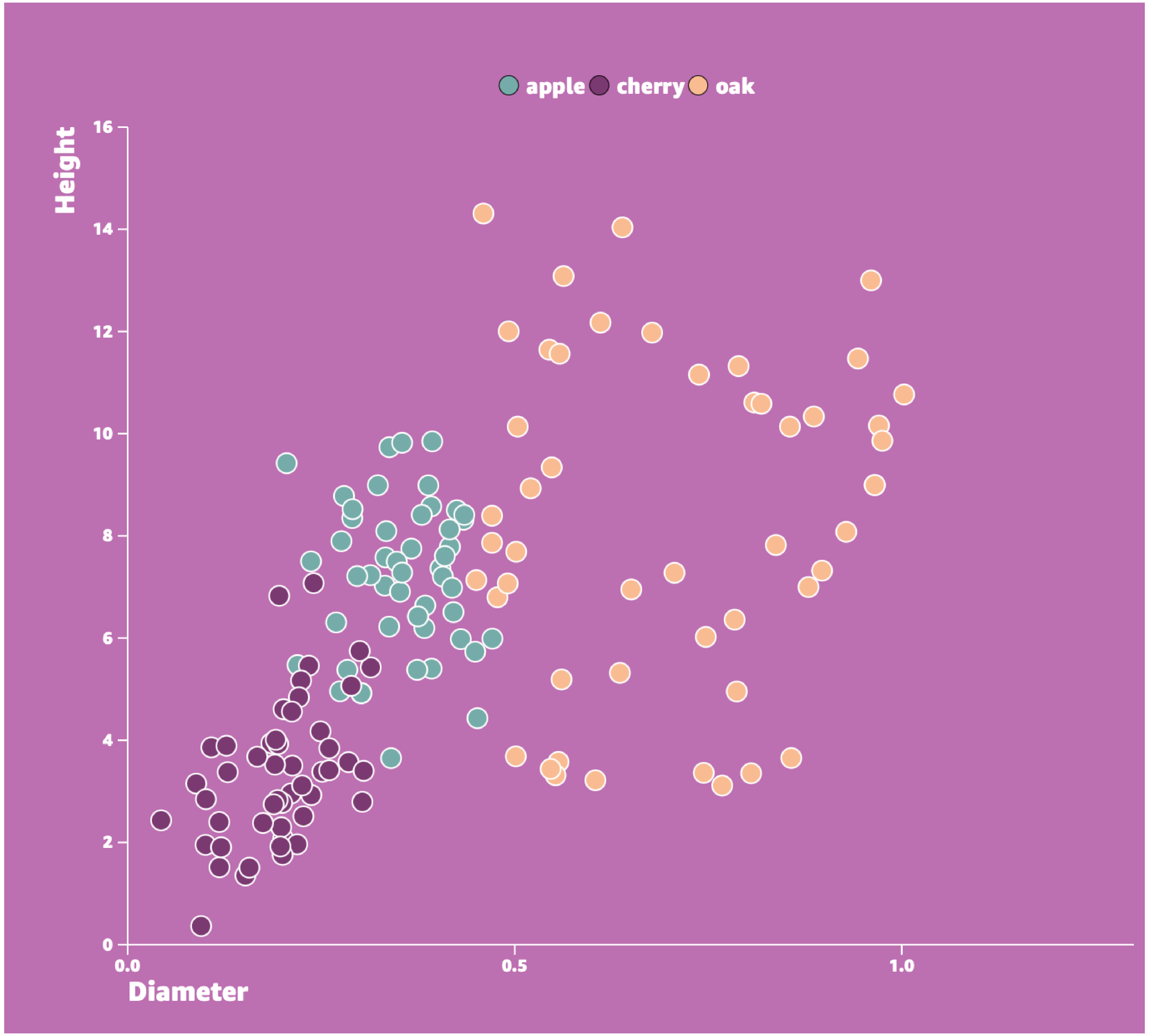
- DECISION NODES, ON SOME DATA SET'S FEATURES.
- EACH DECISION NODE IS SPLIT INTO TWO OR MORE SUBNODES.
- LEAF NODES, ALSO KNOWN AS TERMINAL NODES, REPRESENT **PREDICTION OUTPUTS FOR THE MODEL.**

→ THE RESULTING FLOW-LIKE STRUCTURE IS NAVIGATED VIA CONDITIONAL CONTROL STATEMENTS

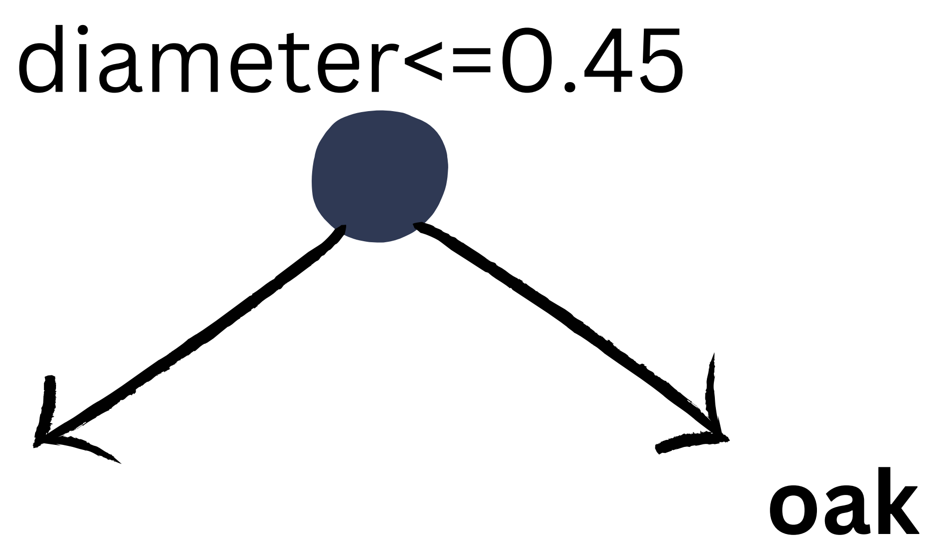
# LET'S BUILD A DECISION TREE AT HIGH LEVEL

GIVEN ONLY THE **DIAMETER** AND **HEIGHT** OF A TREE TRUNK, WE MUST DETERMINE IF IT'S AN **APPLE**, **CHERRY**, OR **OAK** TREE. TO DO THIS, WE'LL USE A DECISION TREE.

3 CATEGORIES : APPLE,CHERRY,OAK
2 FEATURES : DIAMETER,HEIGHT

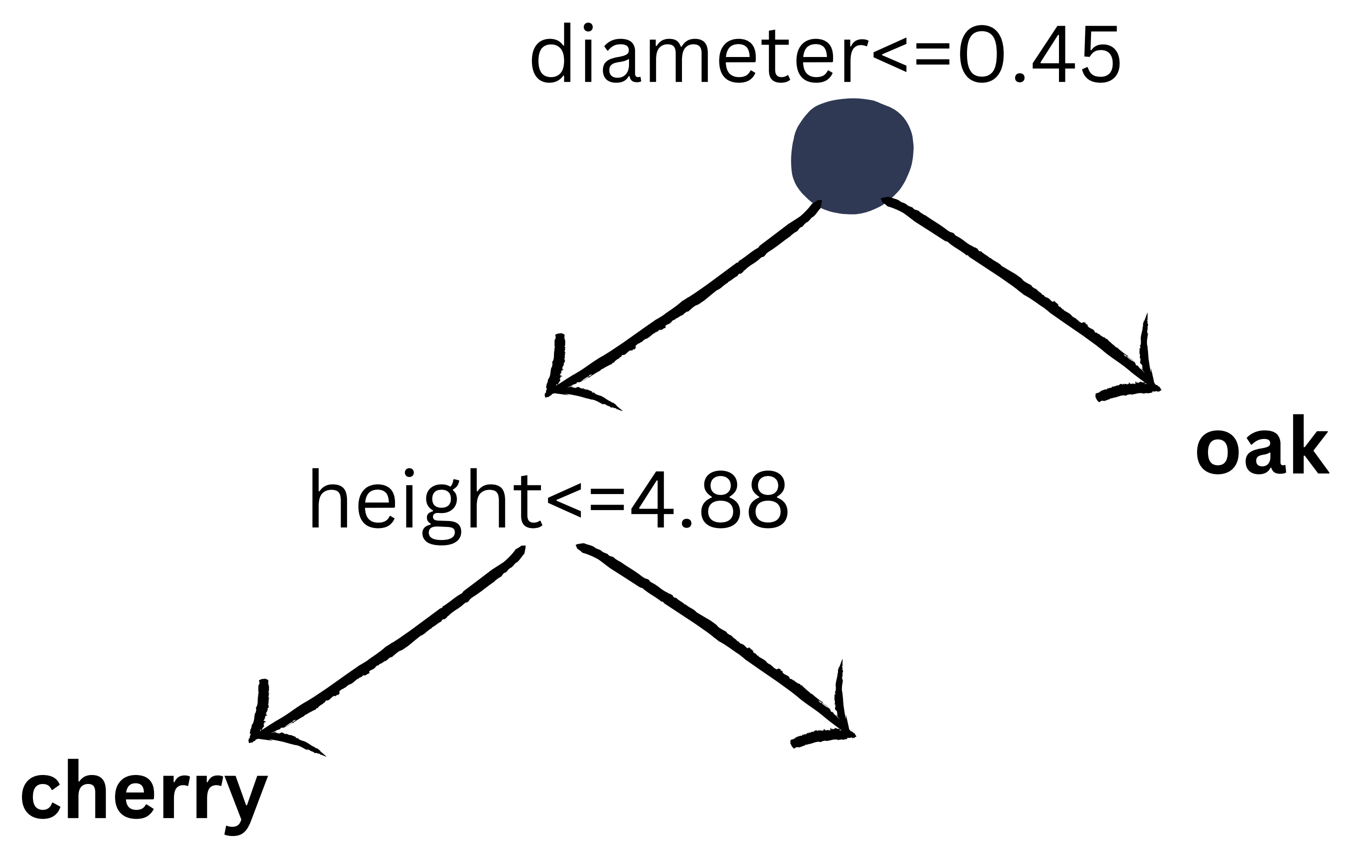

apple
cherry
oak
Height
16
14
12
10
8
6
4
2
0
0.0
0.5
1.0
Diameter

Based on our data set almost every tree with a Diameter ≥ 0.45 is an Oak tree!



Based on our data set almost every tree with a Diameter ≥ 0.45 is an Oak tree!

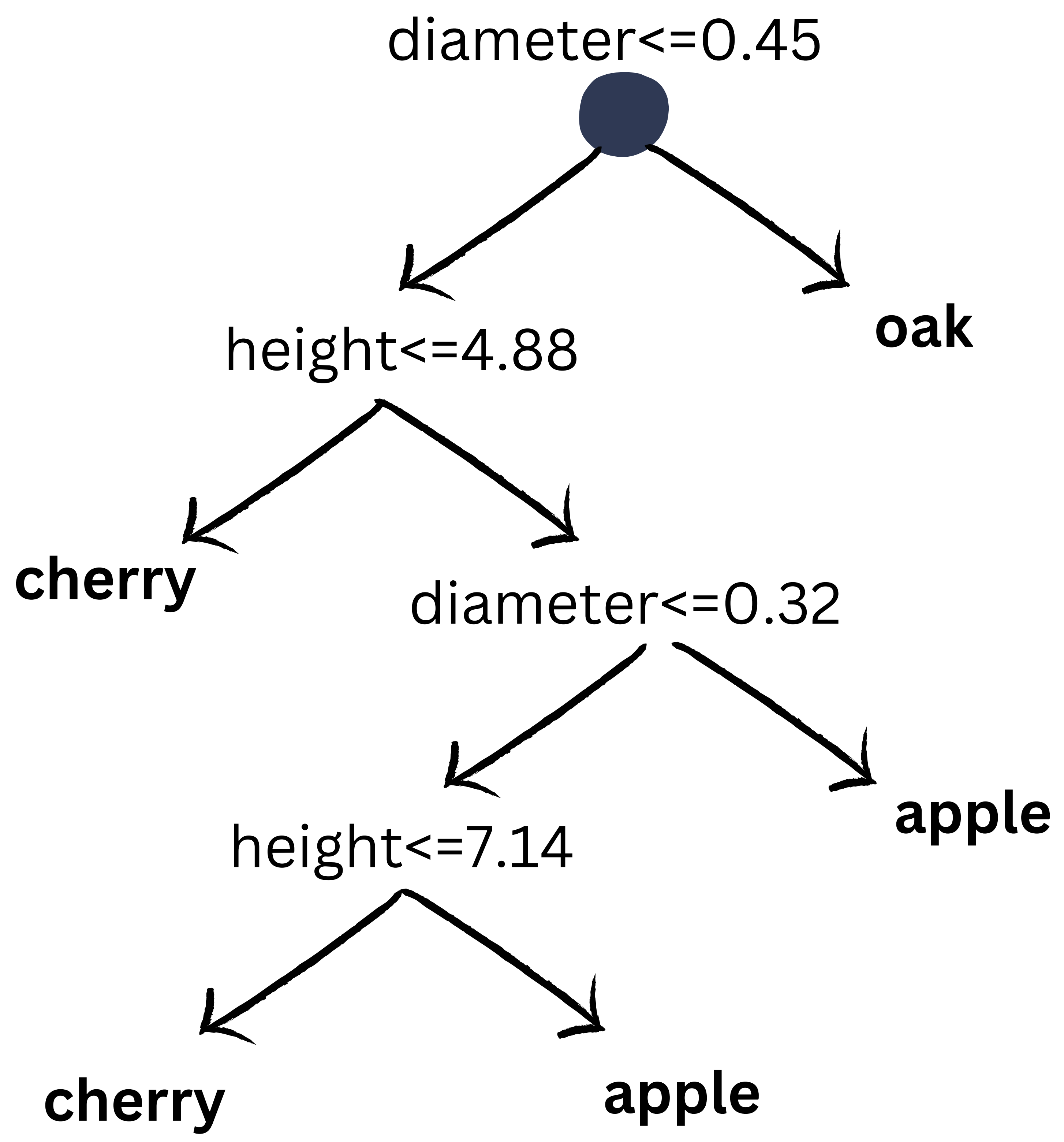
We continue splitting,
We see that creating a new decision nodeat Height ≤ 4.88 leads to a nice section of Cherry trees, so we partition our data there.



Based on our data set almost every tree with a Diameter ≥ 0.45 is an Oak tree!

We continue splitting,
We see that creating a new decision nodeat Height ≤ 4.88 leads to a nice section of Cherry trees, so we partition our data there.

More splits..



## WHERE TO PARTITION?

## ENTROPY

Entropy measures the **amount of information** of some variable or event. We'll make use of it to identify regions consisting of a large number of similar (**pure**) or dissimilar (**impure**) elements.

Given a certain set of events that occur with probabilities (p1,p2,...,pn), the total entropy H can be written as the negative sum of weighted probabilities:

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

## ENTROPY PROPERTIES

- H=0 only if all but one of the pi are **zero**, this one having the value of **1**. Thus the **entropy vanishes** only when **there is no uncertainty** in the outcome.
- H is maximum when all the pi are **equal**. This is the most uncertain, or '**impure**', situation.

THE ENTROPY CAN BE USED TO QUANTIFY THE **IMPURITY** OF A COLLECTION OF LABELED DATA POINTS: A NODE CONTAINING MULTIPLE CLASSES IS IMPURE WHEREAS A NODE INCLUDING ONLY ONE CLASS IS PURE.

## INFORMATION GAIN

Information gain measures an amount the information that we gain. It does so using **entropy**. The idea is to subtract from the entropy of our data before the split the entropy of each possible partition thereafter. We then select the split that yields the largest reduction in entropy, or equivalently, the largest increase in information.

The core algorithm to calculate information gain is called ID3
( https://en.wikipedia.org/wiki/ID3_algorithm)

It's a **recursive procedure** that starts from the root node of the tree and **iterates top-down** on all non-leaf branches in a greedy manner, **calculating at each depth** the difference in entropy:

$$\Delta IG = H_{\text{parent}} - \frac{1}{N} \sum_{\text{children}} N_{\text{child}} \cdot H_{\text{child}}$$

PS:An alternative to the entropy for the construction of Decision Trees is the **Gini impurity**.

## THE NEED TO GO BEYOND DECISION TREES

## PROBLEMS WITH DT

- extremely sensitive to small perturbations in the data.
- unstable
- Overfitters

## SOLUTIONS :

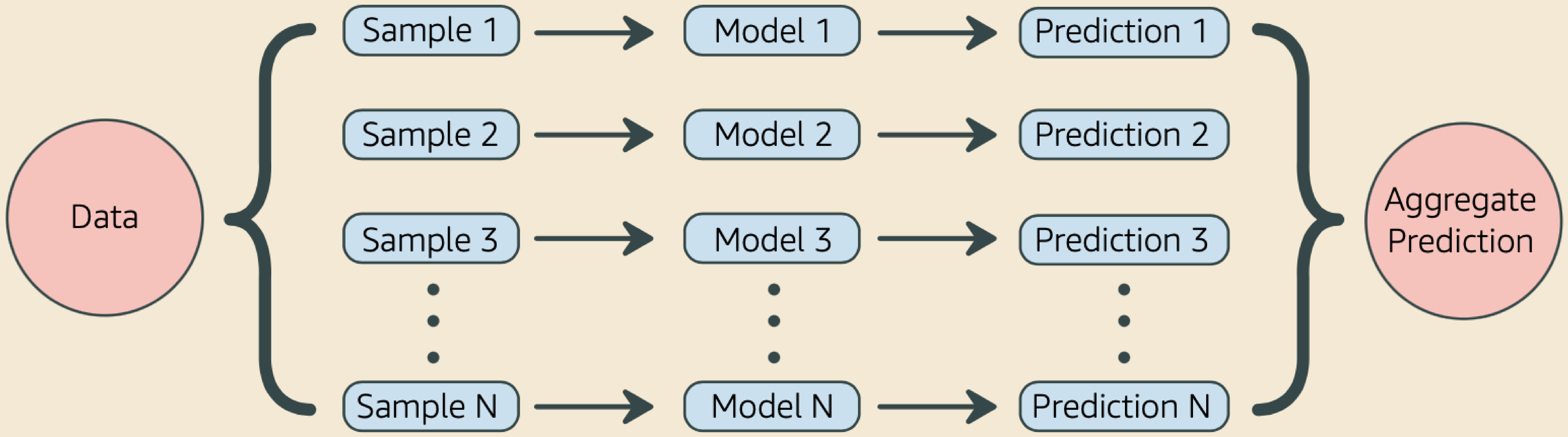
- BAGGING
- BOOSTING

## RANDOM FOREST

**Condorcet's Jury Theorem,** says that if each person is more than 50% correct, then adding more people to vote increases the probability that the majority is correct.

In machine learning, this concept of multiple models working together to come to an aggregate prediction is called **ensemble learning**. It provides the basis for many important machine learning models, including random forests.

## Ensemble Learning



## Bagging Method

One way to produce multiple models that are **different** is to train each model using a different training set. The **Bagging** (Bootstrap Aggregating) method randomly draws a fixed number of samples from the training set with replacement. This means that a data point can be drawn more than once.

According to the random forest inventor , lower correlation between trees results in better performance.

## BOOSTING

### Intuition:

- Take a model f
- Compute the residuals : $\epsilon_i = y_i - f_k\left(\vec{x_i}\right)$
- Fit a model $g_k\left(\vec{x_i}\right) \approx \epsilon_i$
- Define a new model $f_{k+1}(\vec{x}) = f_k(\vec{x}) + g_k(\vec{x})$
- Repeat

The models used are usually **weak learners (**decision tree with small nbr of nodes for example),otherwise overfit will probably occur.

## GRADIENT BOOSTING

### Generalizing:

- let li be the loss per data point $$l_i = \frac{1}{2}(y_i - f_k(\vec{x_i}))^2$$

  our loss is just the sum of li

Then the residuals are : $$\epsilon_i = y_i - f(\vec{x_i}) = -\frac{\partial l_i}{\partial f(\vec{x_i})}$$

We can now see we're doing nothing more the **gradient descent** on the loss function, and with this formulation we could use whatever loss we want according to our problem.

## LIBRARIES IMPLEMENTING GRADIENT BOOSTING

- XGBOOST
- LIGHTGBM
- CATBOOST

## LET'S PRACTICE