Decision Tree

BUS 212: Spring 2023

Yeabin Moon

Lecture 18: Motivation

Tree-based methods

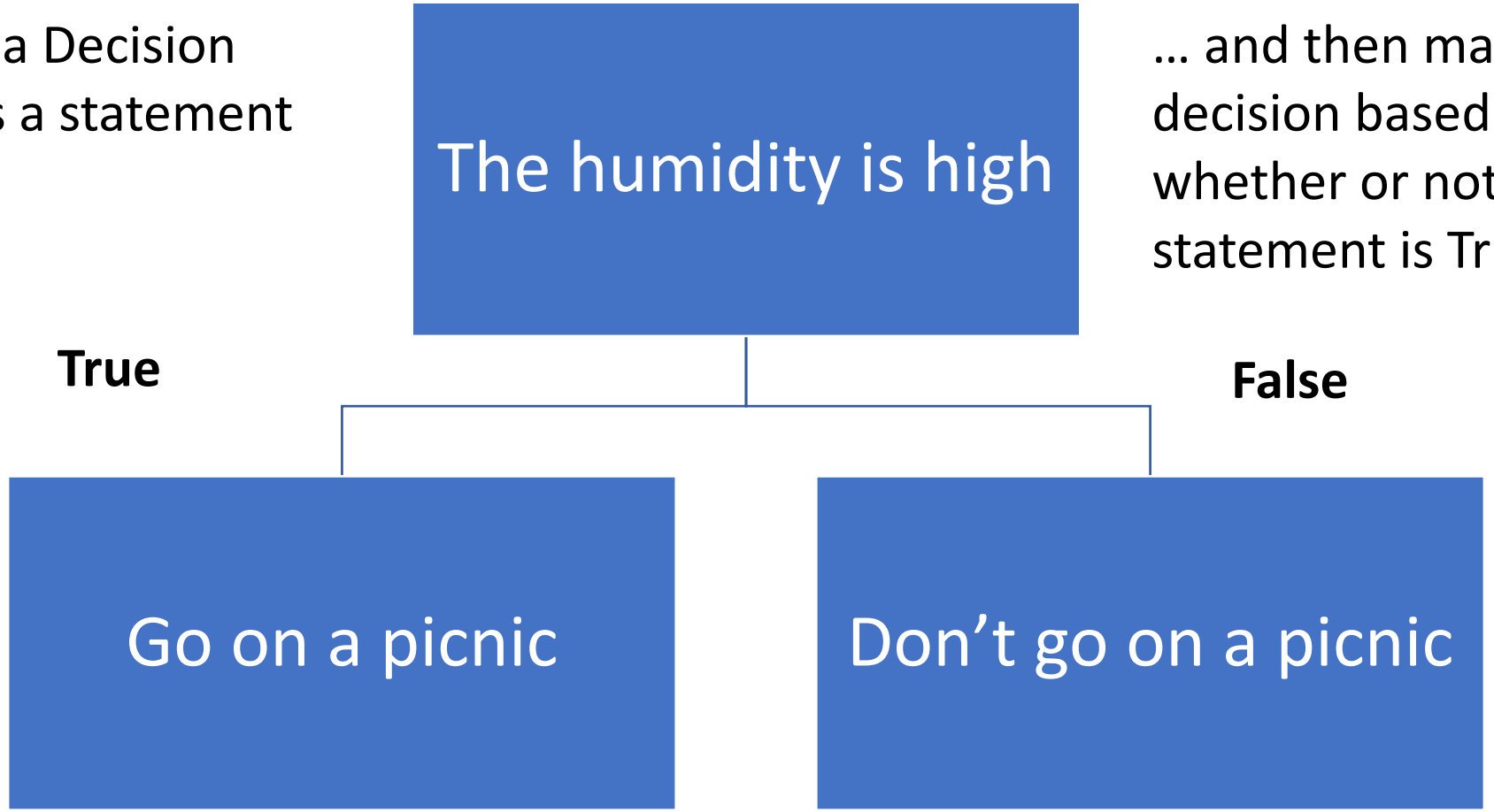
- For the 5 lectures, we will spend time learning on the Tree-based methods
- The methods are a class of machine learning algorithms
  - Classification
  - Regression
- Easy to interpret and visualize and work well even on multioutput tasks
  - Some say XGBoost is the best

Learning Goals

- Today, we will study some motivational examples
- Next 2 lectures focus on Decision Trees
- Then, next 2 lectures address ensemble method and random forests
  - Why boosting is so popular?

Example

In general, a Decision Tree makes a statement

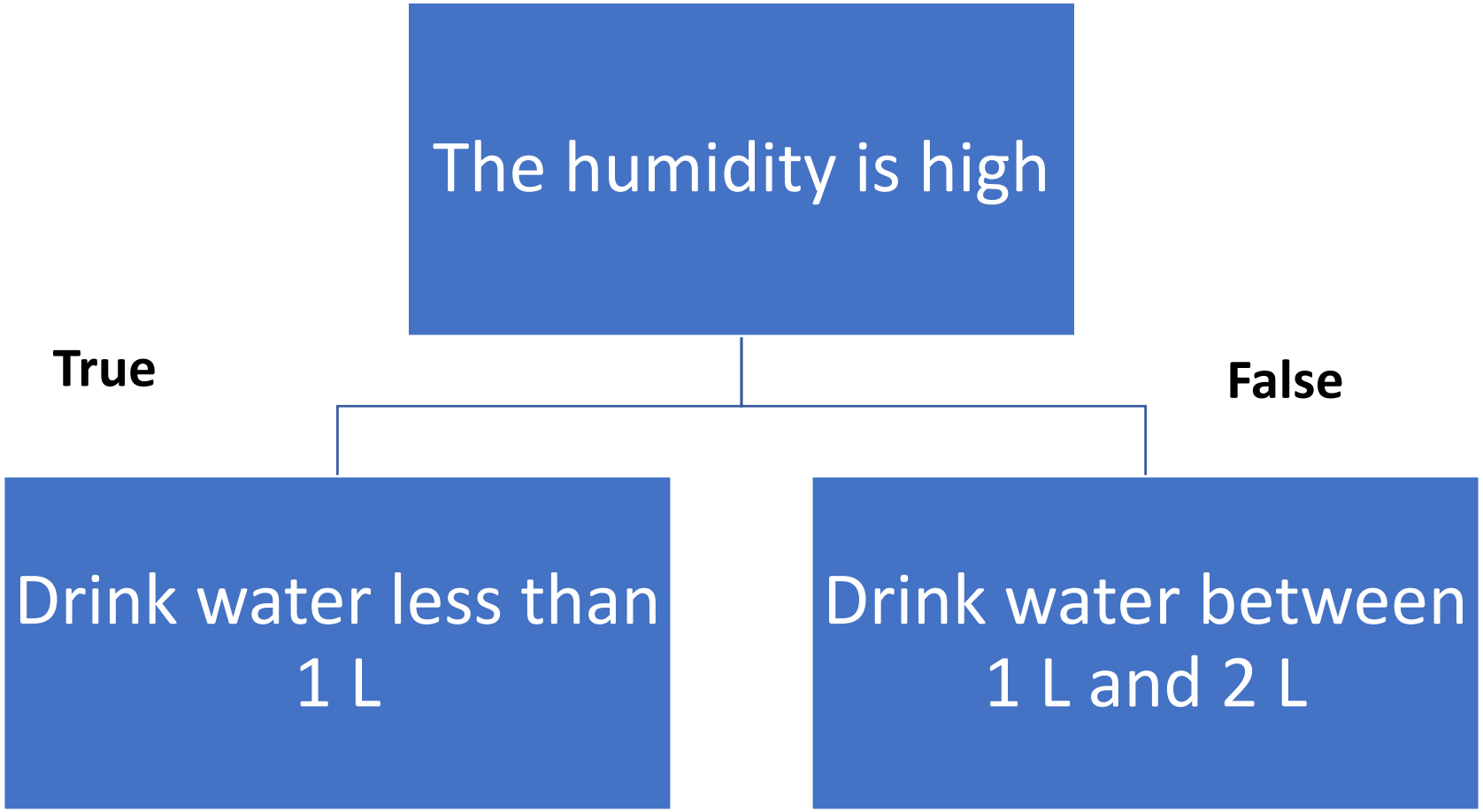


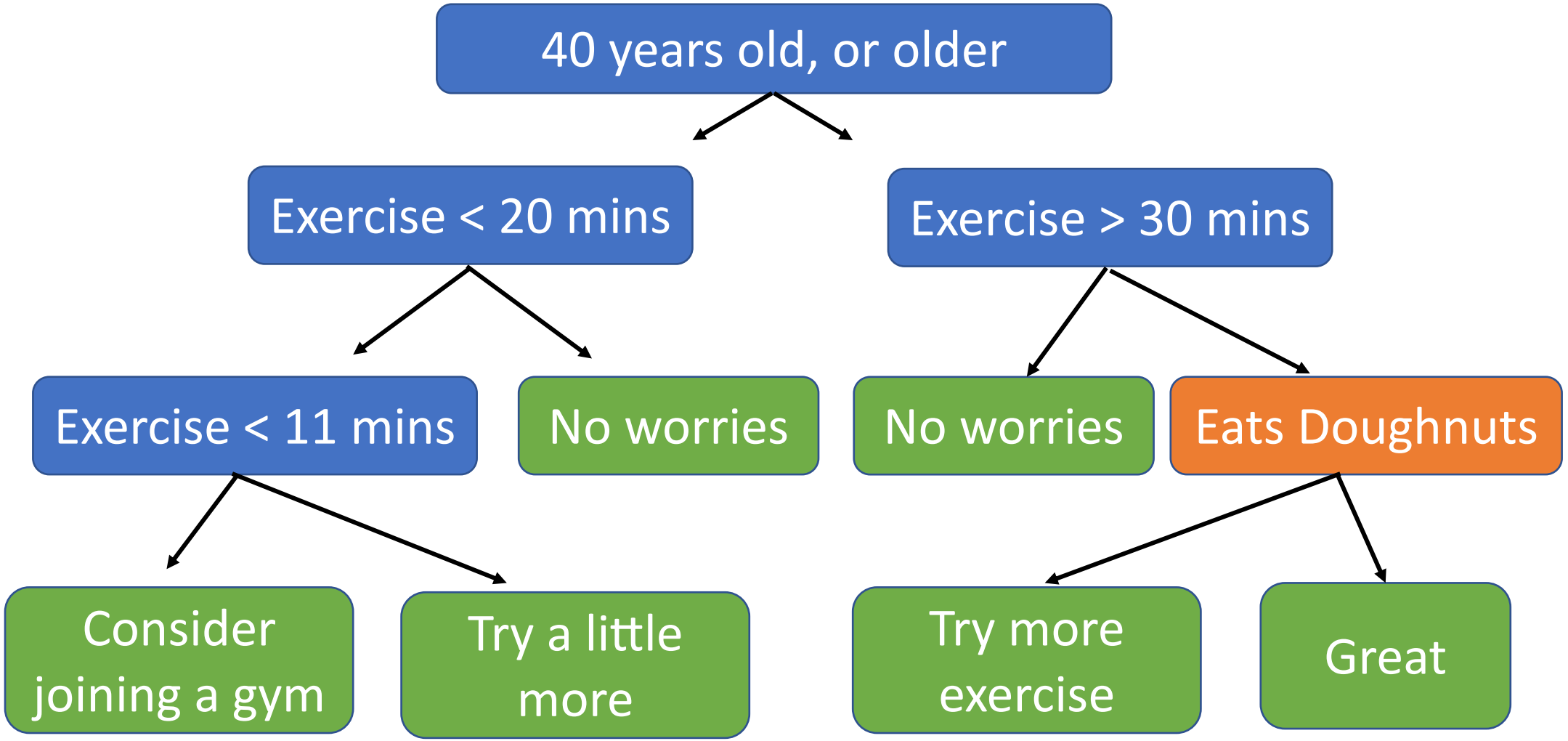
… and then makes a decision based on whether or not the statement is True or False

Decision Tree

- When a Decision Tree classifies things into categories
  - It’s called a **Classification Tree**
- When a Decision Tree predicts numeric values
  - It’s called a **Regression Tree**

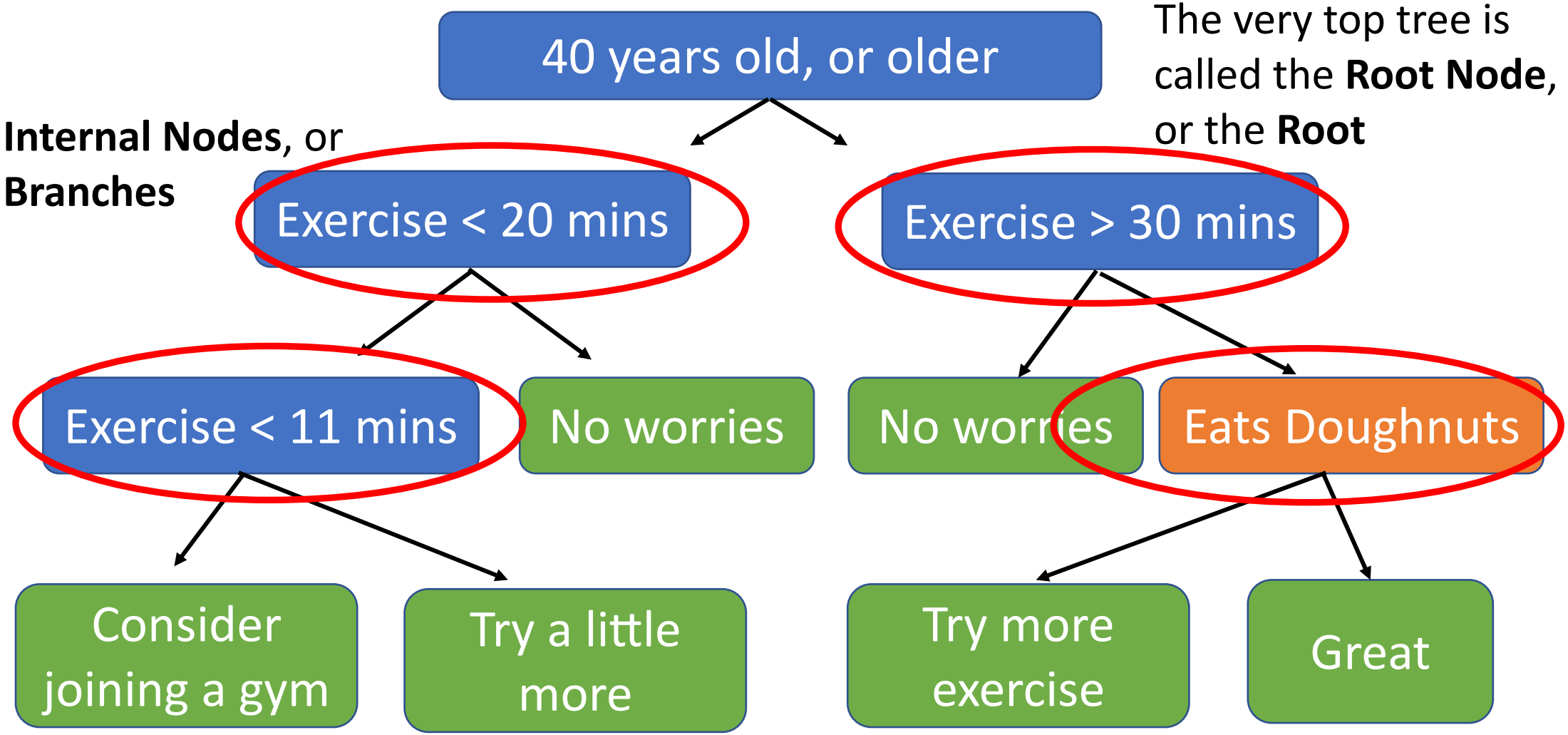
Regression Tree?




40 years old, or older
Exercise < 20 mins
Exercise > 30 mins
Exercise < 11 mins
No worries
No worries
Eats Doughnuts
Consider joining a gym
Try a little more
Try more exercise
Great

More complicated classification tasks?

- Usually, it is just assumed that if a statement is **True**, you go to the **Left**
- Let's Just go through
- It combines numeric data with yes/no data
  - Ok to mix data types in the same tree
- Also notice that tree asks about **Exercising** multiple times
  - … and the amount of time Exercising isn't always the same
  - So numeric thresholds can be different for the same data
- Lastly, the final classifications can be repeated



The terminal nodes are called **Leaf Nodes**, or **Leaves**

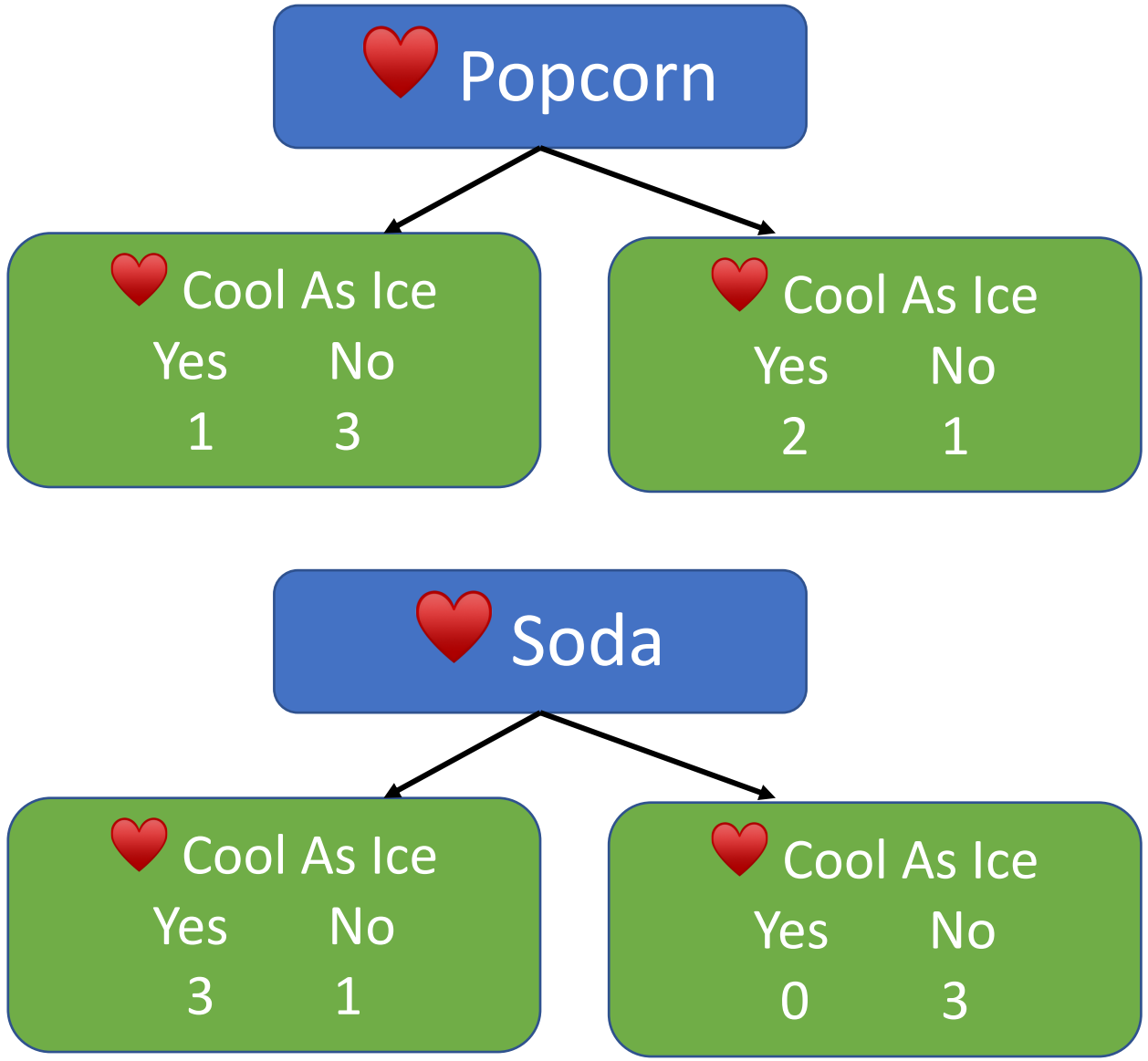
How to build the tree from raw data?

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

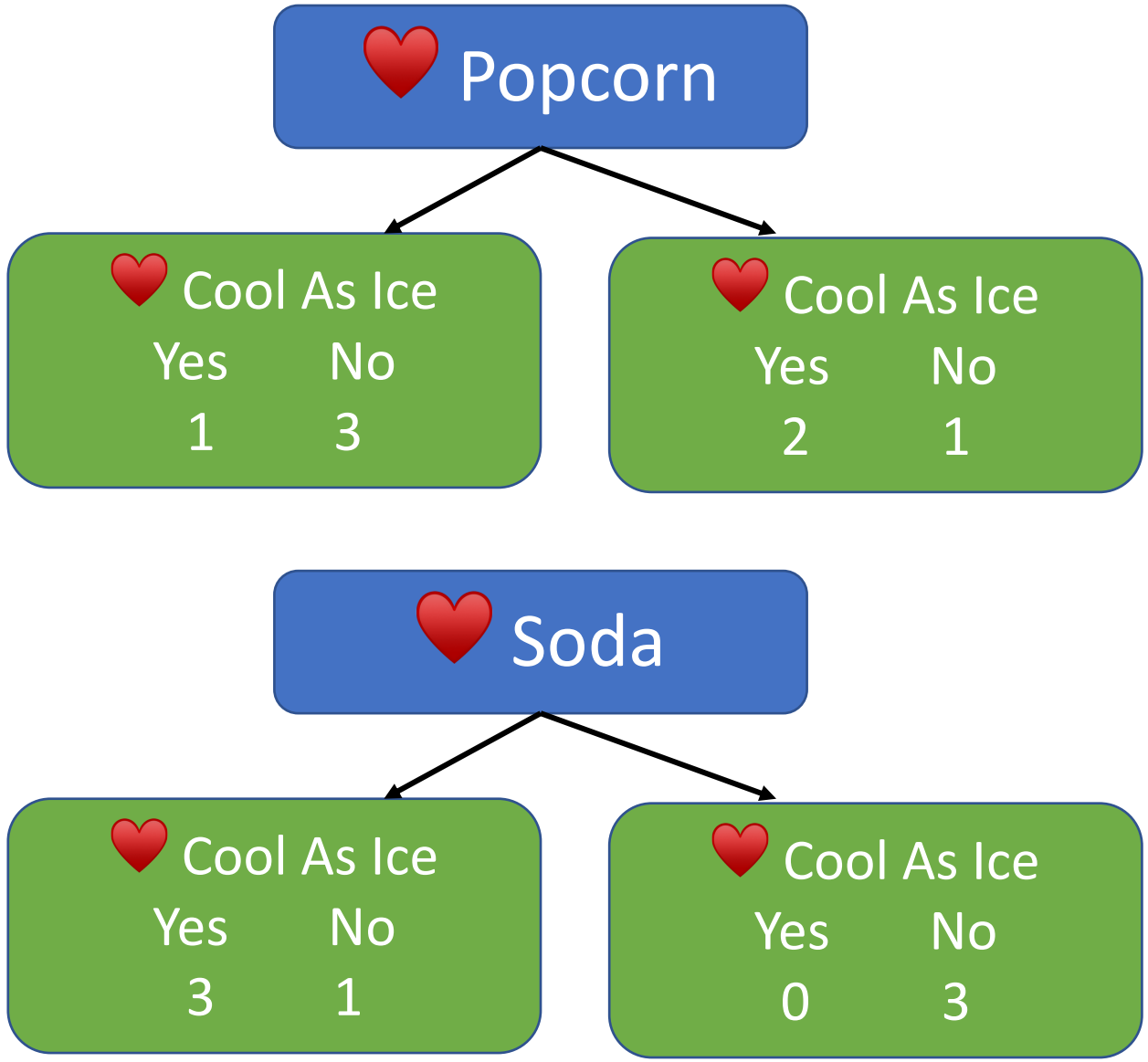
Build Classification Tree

- Use the first three column to predict whether or not someone **Loves Cool As Ice**
- Now, pretend you've never seen the tree before
- Let's see how to build a tree starting with just data
- The first thing we do is to decide
  - Whether **Loves Popcorn, Loves Soda, or Age** should be the question we ask at the top of the tree
- Let's start with Loves Popcorn

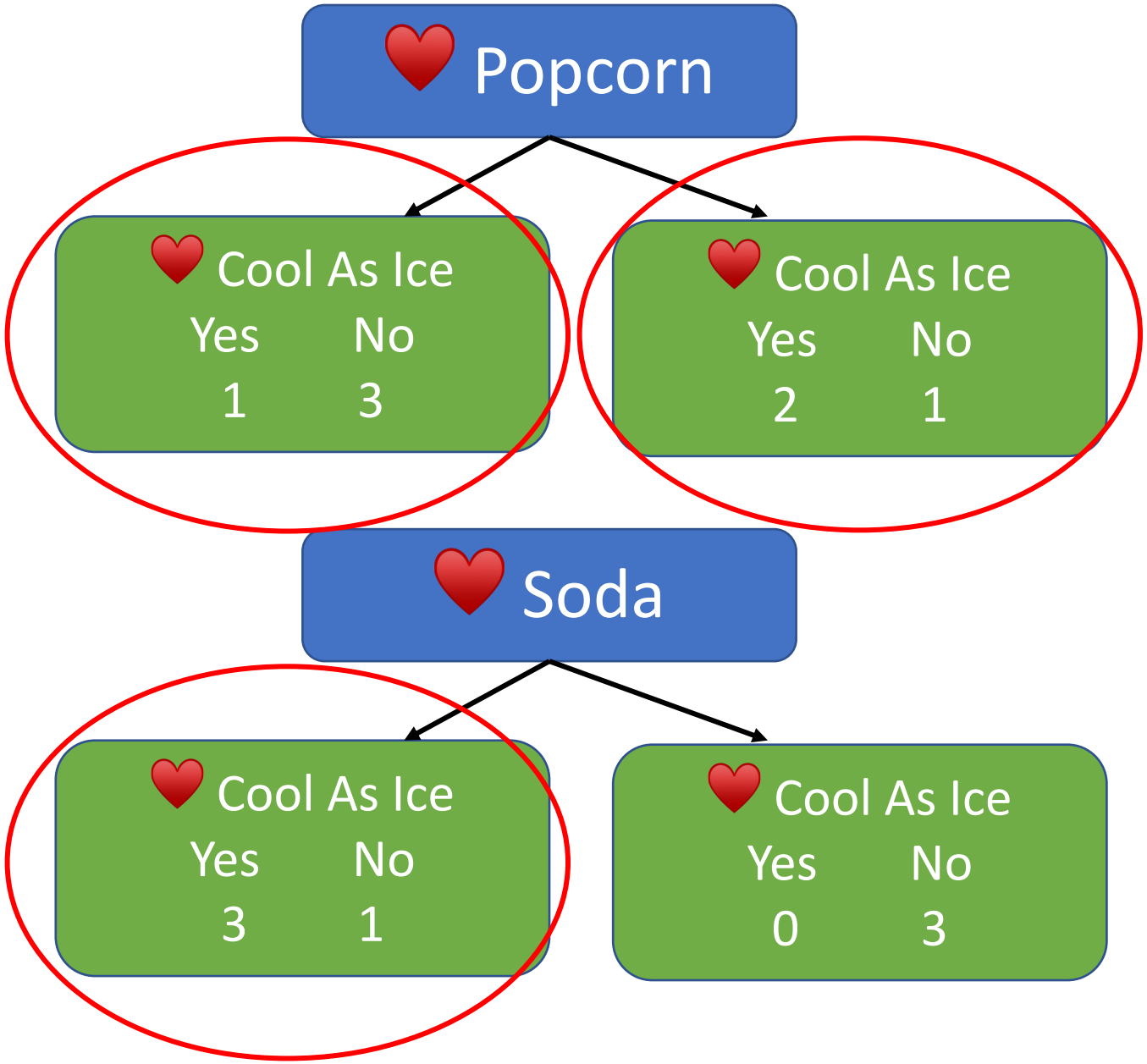
Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



Looking at the two little trees, we see that neither one does a perfect job predicting who will and who will not **Love Cool As Ice**

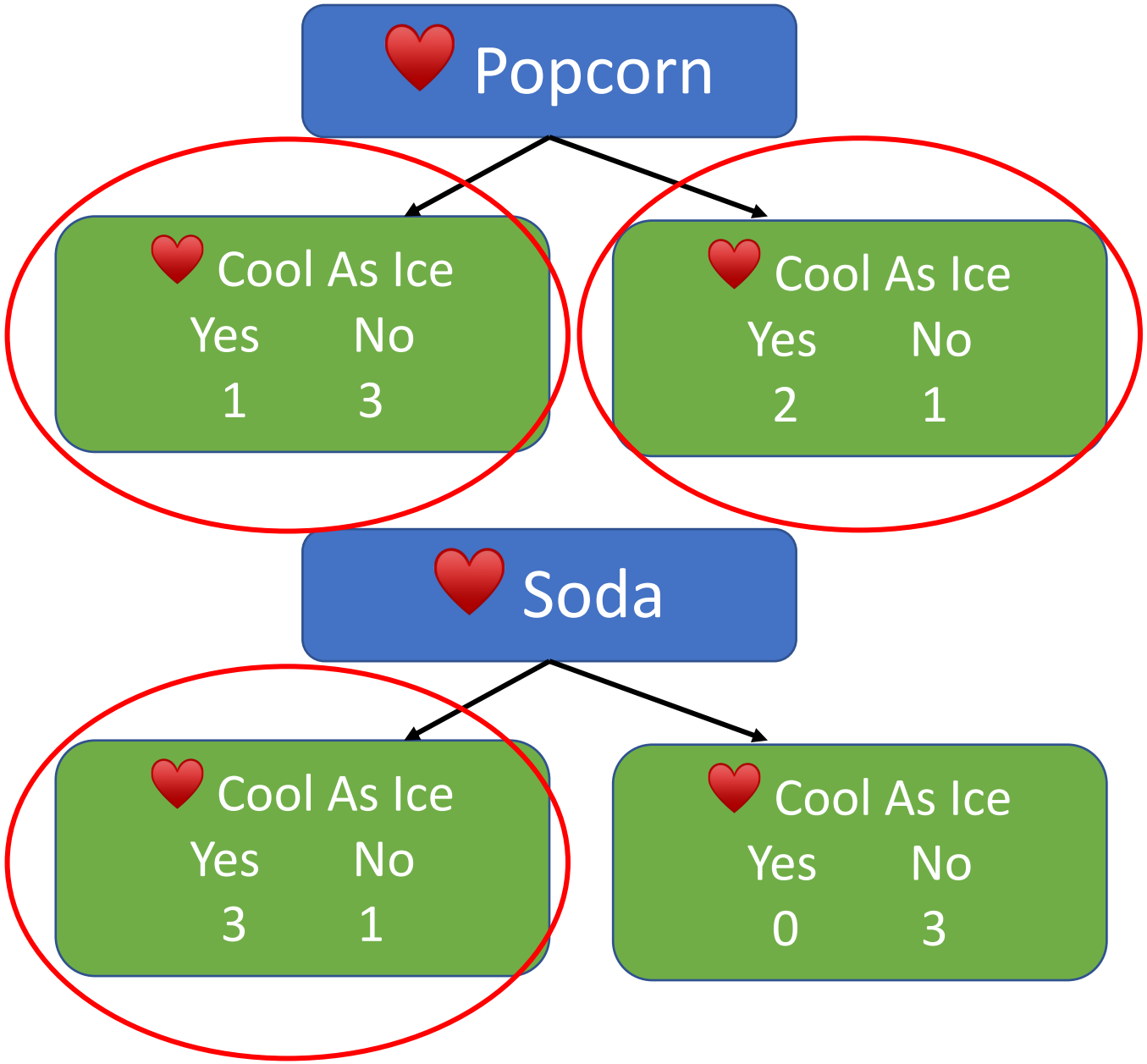


Specifically, these three Leaves contain mixtures of people that do and do not **Love Cool As Ice**

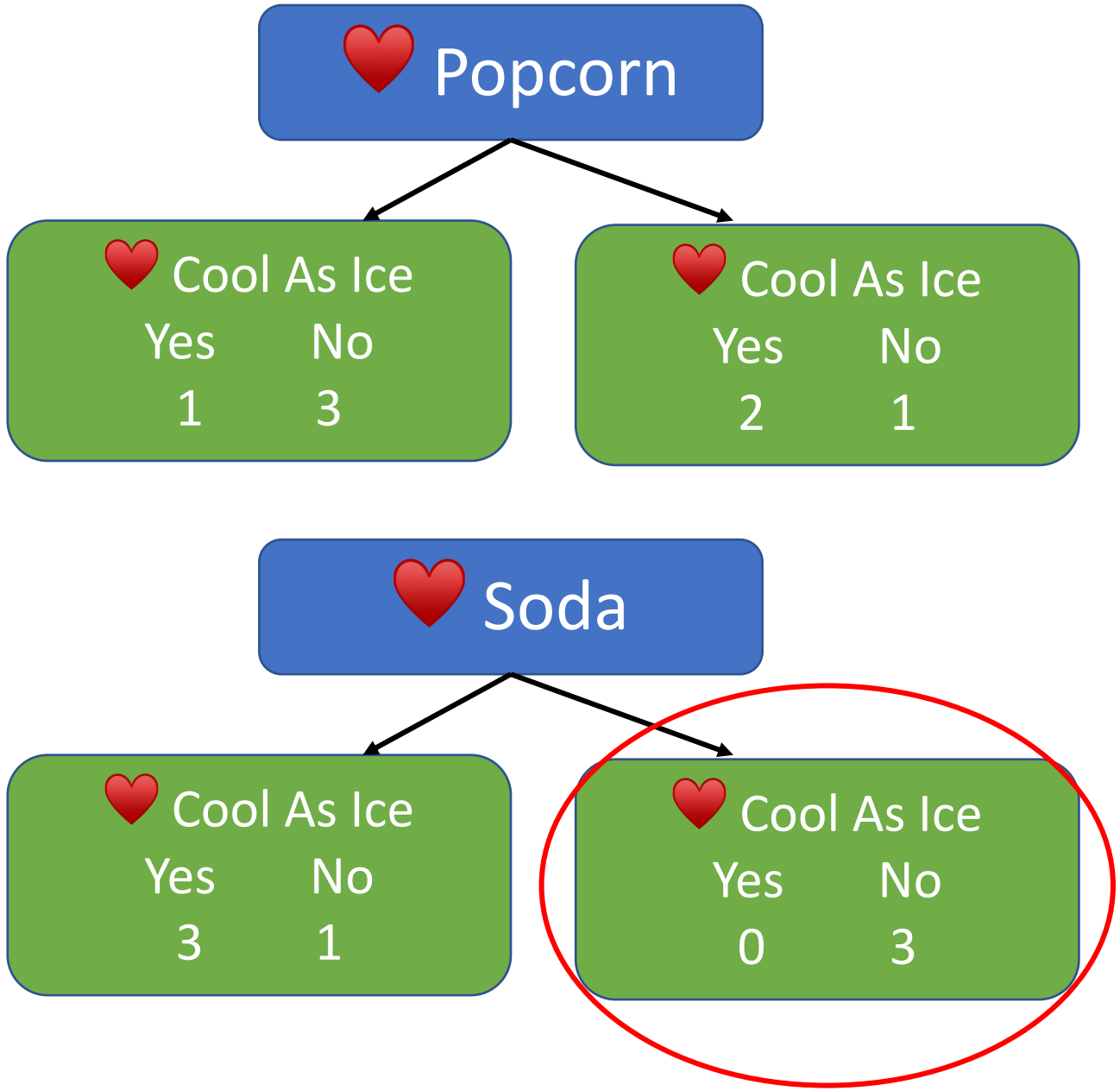


Because these 3 Leaves all contain a mixture of people who do and do not **Love Cool As Ice**…

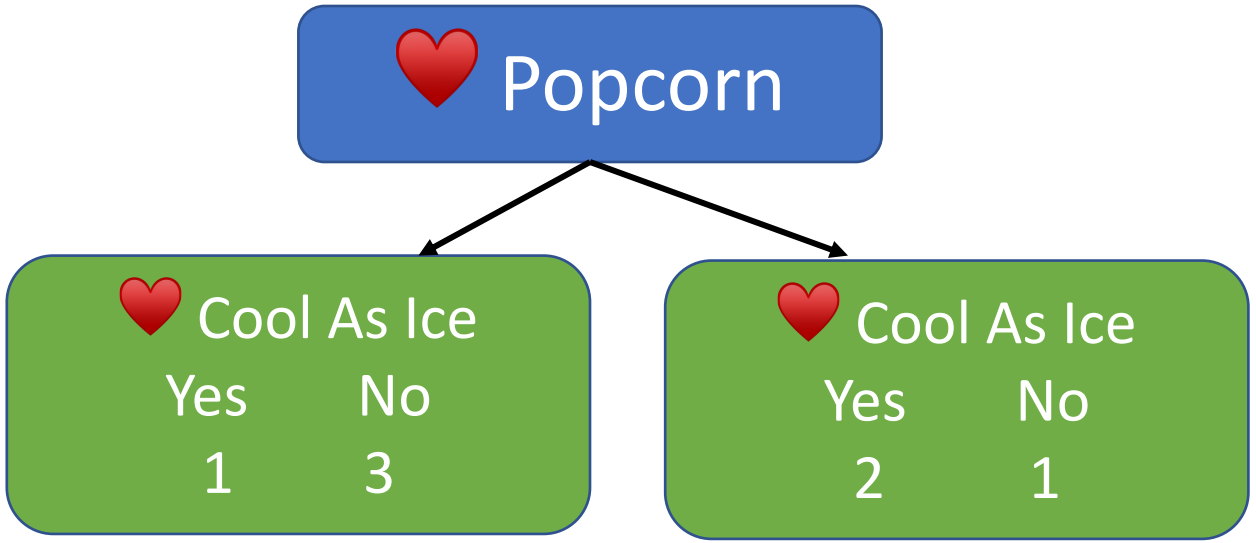
They called **Impure**



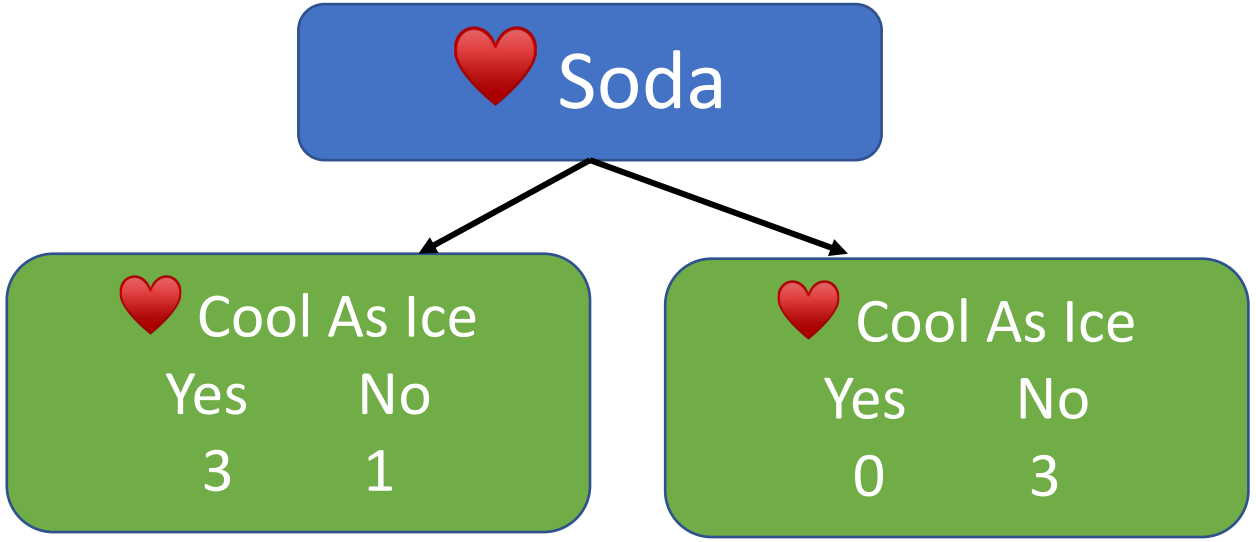
In contrast, this **Leaf** only contains people who do not **Love Cool As Ice**



Because both **Leaves** in the **Loves Popcorn** tree are **Impure…**



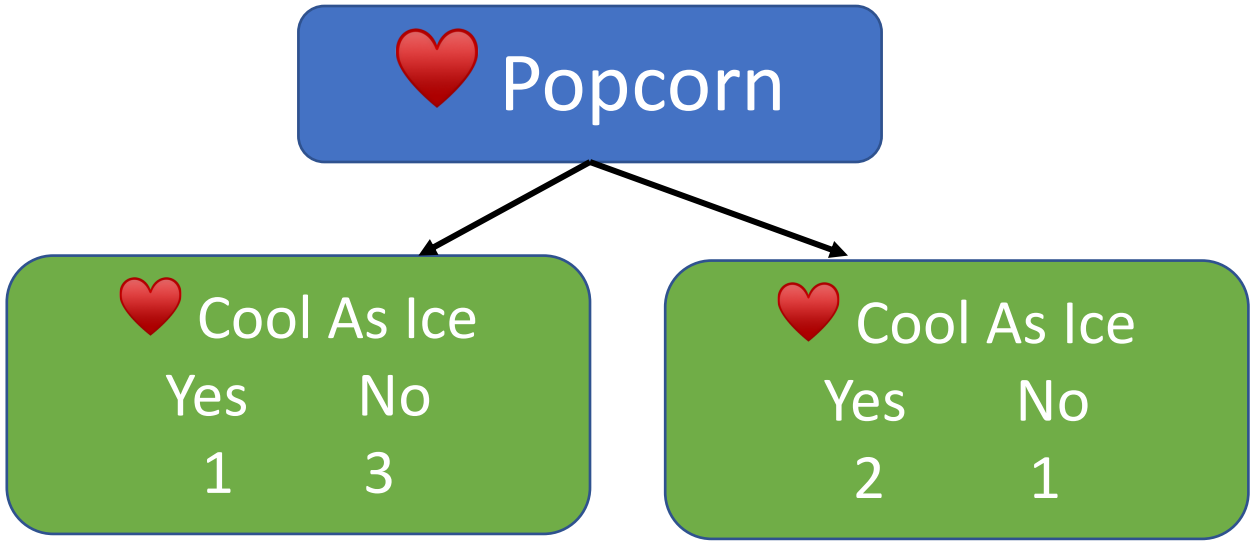
…it seems like **Loves Soda** does a better job predicting who will and will not **Love Cool As Ice**…



Quantify Impurity

- Of course, it would be nice if we could quantify the differences between **Loves Popcorn** and **Loves Soda**
- The good news is that there are several ways to quantify the **Impurity** of the leaves
- One of the most popular methods is called **Gini Impurity**
  - **Entropy** and **Information Gain** are competing candidates
  - Numerically, the methods are quite similar
- Let's focus on Gini Impurity

To calculate the **Gini Impurity** for **Loves Popcorn**, we start by calculating the **Gini Impurity** for the individual **Leaves**



Gini Impurity for the **left** Leaf = $1 - (\text{Prob. } \text{“}Yes\text{”})^2 - (\text{Prob. } \text{“}No\text{”})^2$

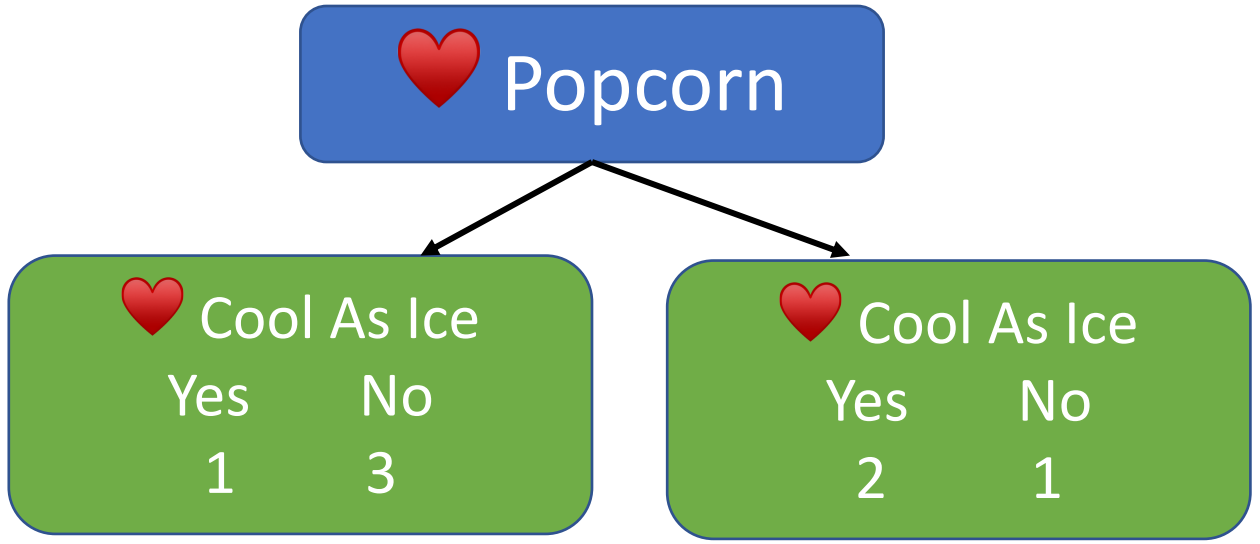
$$= 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2$$

$$= 0.375$$

Gini Impurity for the **right** Leaf = $1 - \left(\frac{2}{2+1}\right)^2 - \left(\frac{1}{2+1}\right)^2$

$$= 0.444$$

Note that the Leaves do not represent the same number of people



Gini Impurity is 0.375

Gini Impurity is 0.444

Thus, the total **Gini Impurity** is the **Weighted Average** of the **Leaf Impurities**

$$\text{Total Gini Impurity} = \left(\frac{4}{4+3}\right) 0.375 + \left(\frac{3}{4+3}\right) 0.444 = 0.405$$

So the **Gini Impurity** for **Loves Popcorn** is 0.405

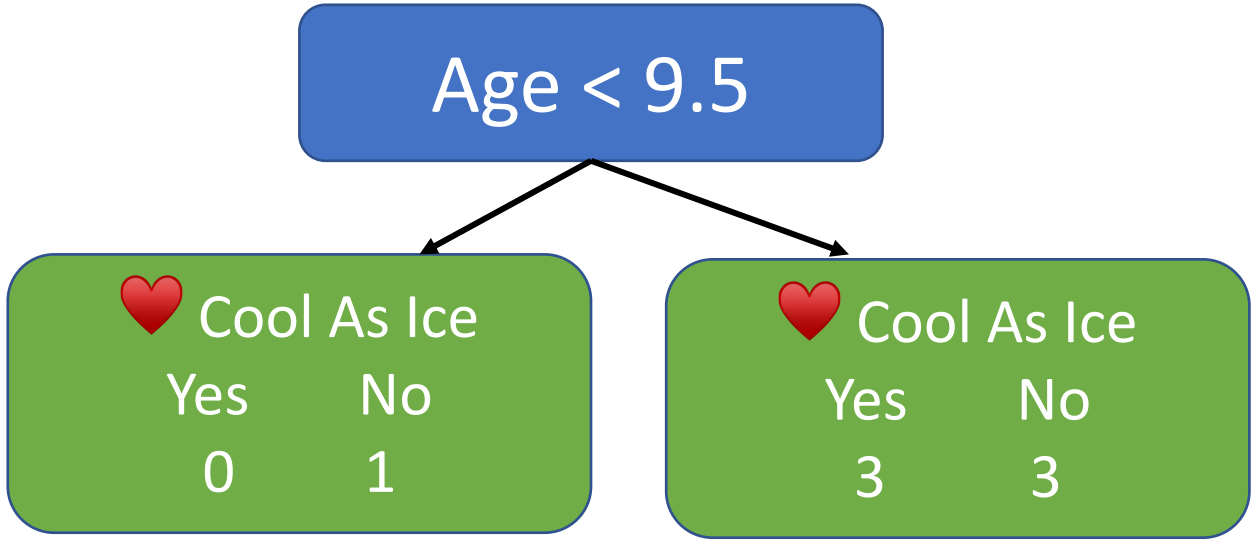
Likewise, the **Gini Impurity** for **Loves Soda** is 0.214

Gini Impurity for the continuous variable

- Now we need to calculate the Gini Impurity for Age
- Because Age contains numeric data, calculating the Gini Impurity is a little more involved
- The first thing we do is sort the rows by Age, from lowest value to highest value
- Then we calculate the average Age for all adjacent people

	Age	Loves Cool As Ice
9.5	7	No
15	12	No
26.5	18	Yes
36.5	35	Yes
44	38	Yes
66.5	50	No
	83	No

E.g. to calculate the Gini Impurity for the first value

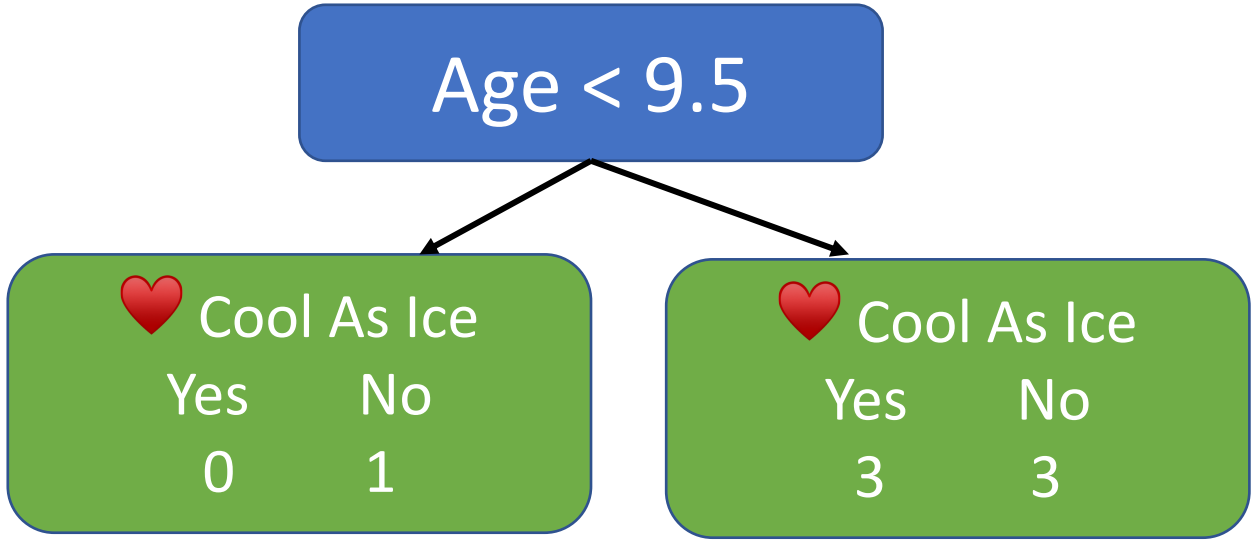


$$\text{Left Gini Impurity} = 1 - \left(\frac{0}{0+1}\right)^2 - \left(\frac{1}{0+1}\right)^2 = 0$$

$$\text{Right Gini Impurity} = 1 - \left(\frac{3}{3+3}\right)^2 - \left(\frac{3}{3+3}\right)^2 = 0.5$$

	Age	Loves Cool As Ice
9.5	7	No
	12	No
15		
	18	Yes
26.5		
	35	Yes
36.5		
	38	Yes
44		
	50	No
66.5		
	83	No

E.g. to calculate the Gini Impurity for the first value



Total Gini Impurity $= \left(\frac{1}{1+6}\right) 0 + \left(\frac{6}{1+6}\right) 0.5 = 0.429$

	Age	Loves Cool As Ice
	7	No
9.5		
	12	No
15		
	18	Yes
26.5		
	35	Yes
36.5		
	38	Yes
44		
	50	No
66.5		
	83	No

Gini Impurity = 0.429

Gini Impurity = 0.343

Gini Impurity = 0.476

Gini Impurity = 0.476

Gini Impurity = 0.343

Gini Impurity = 0.429

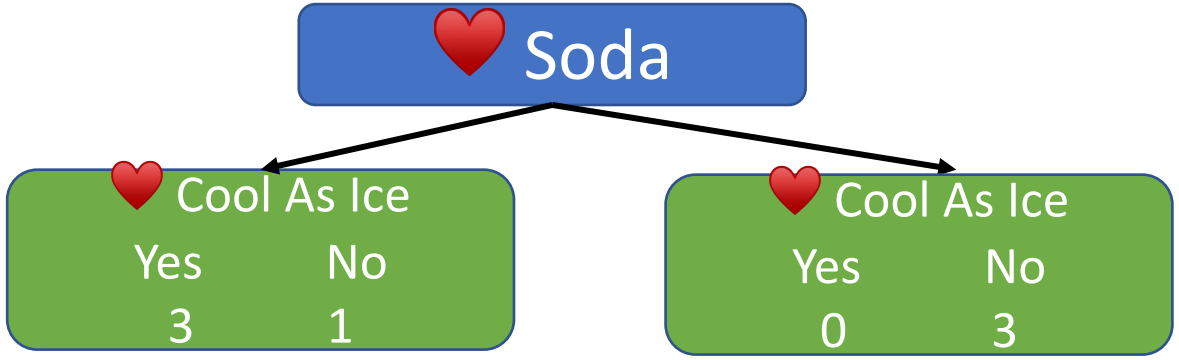
These two candidate thresholds are tied for the lowest **Impurity**, **0.343**

So we can pick either one. Let's pick 15.

Decide the Root

- Remember that we are comparing Gini Impurity values for Age, Loves Popcorn and Loves Soda
  - To decide which feature should be at the very top of the tree
- Gini Impurity for Loves Popcorn is 0.405
- Gini Impurity for Loves Soda is **0.214**
- Gini Impurity for Age <15 is 0.343
- Because **Loves Soda** has the **lowest** the Gini Impurity overall
  - So we put **Loves Soda** at the top of the tree

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

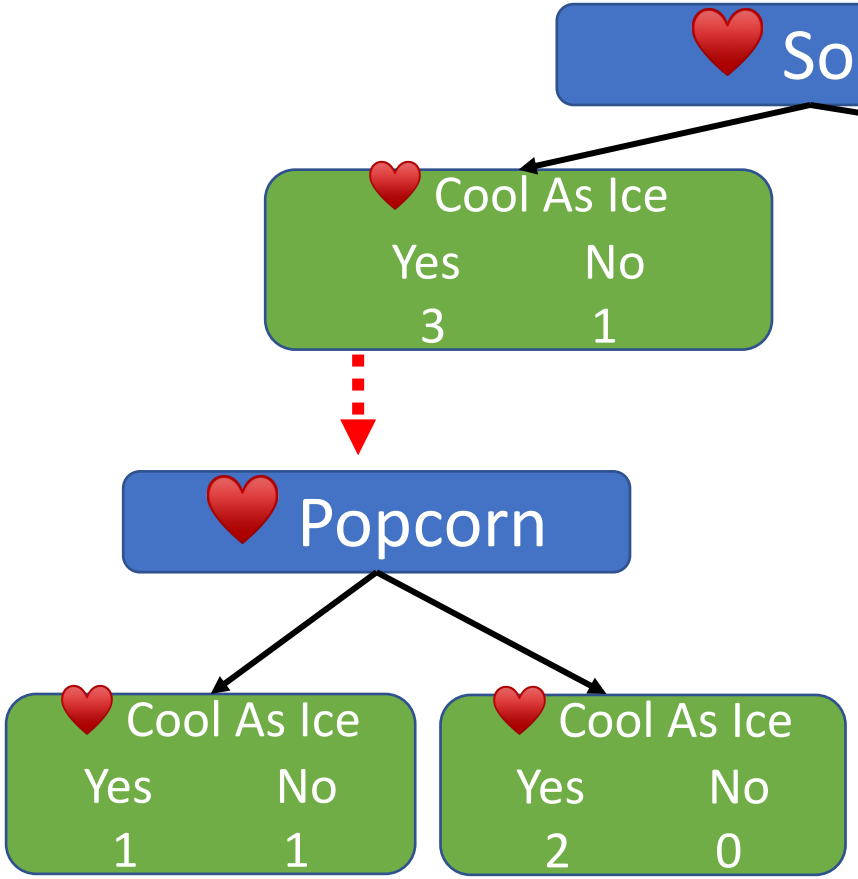


Now let's focus on the **Node** on the left

This is Impure

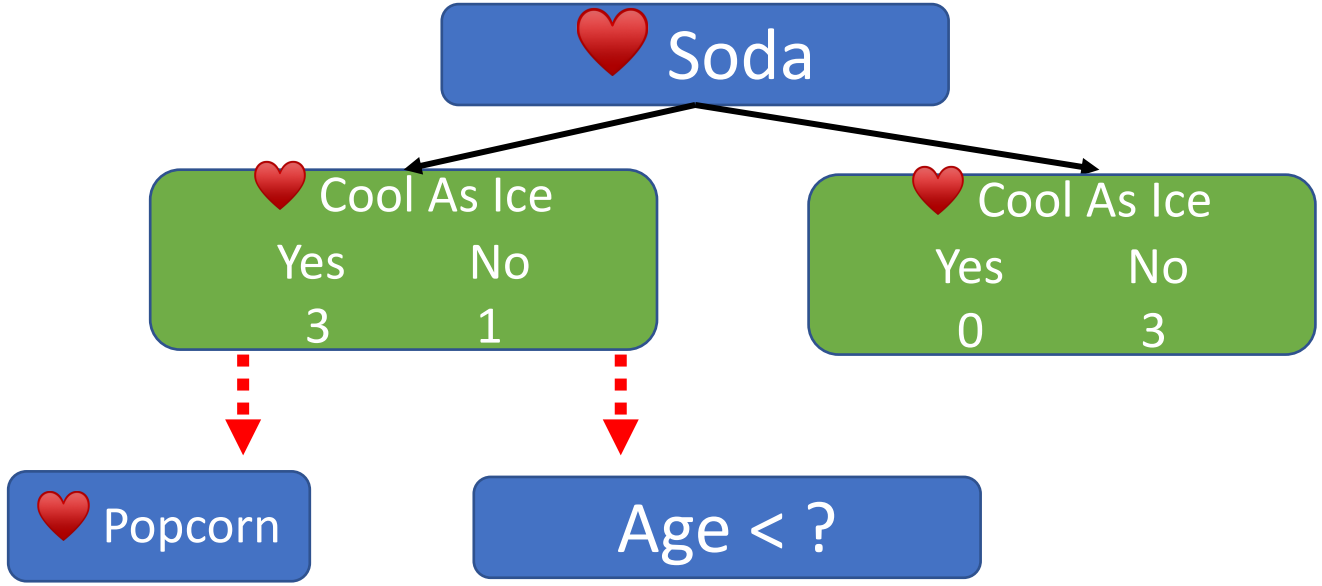
So let's see if we can reduce the **Impurity** by splitting the people that **Love Soda** based on **Loves Popcorn** or **Age**

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



The total Gini Impurity for this split is 0.25

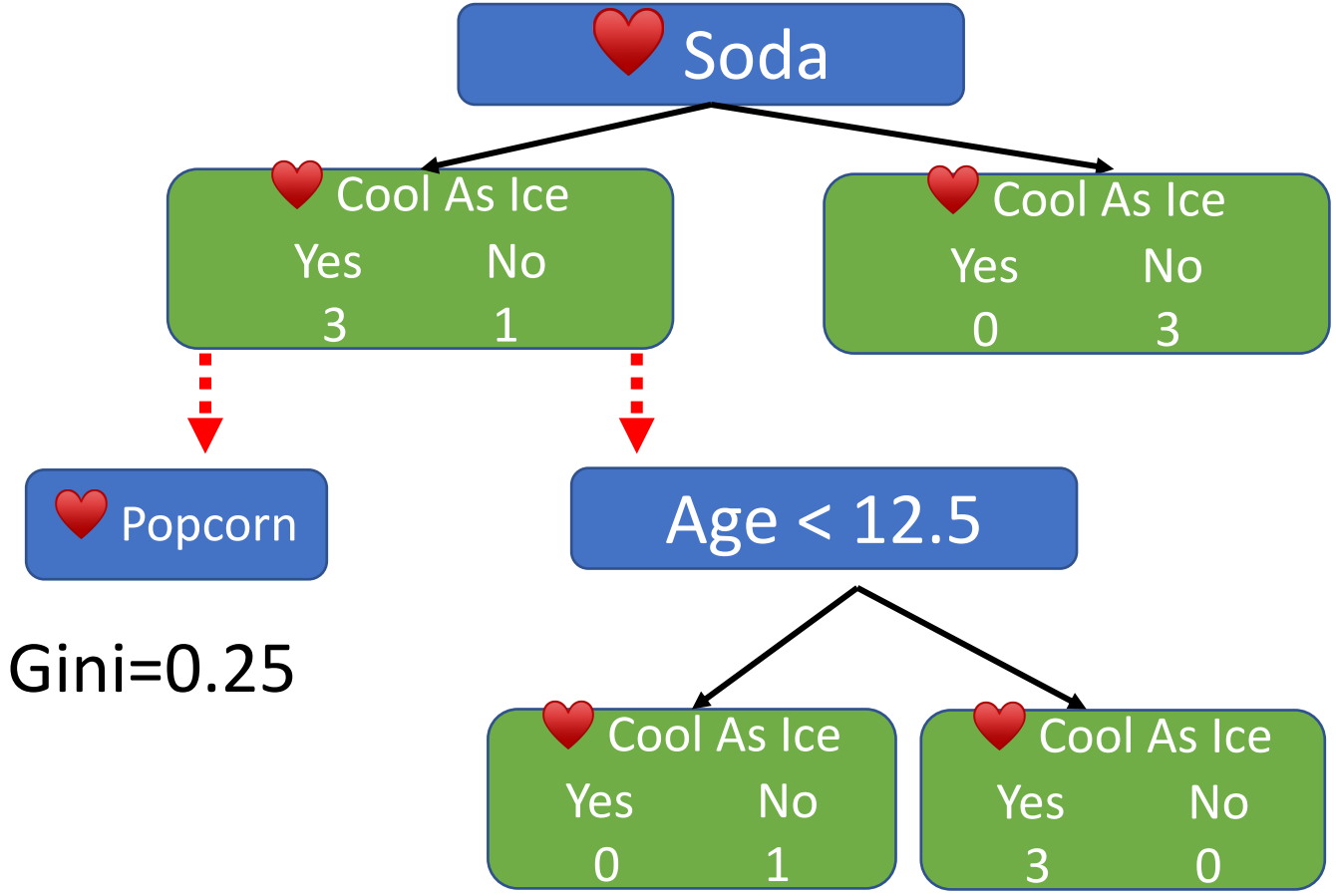
Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



Gini=0.25

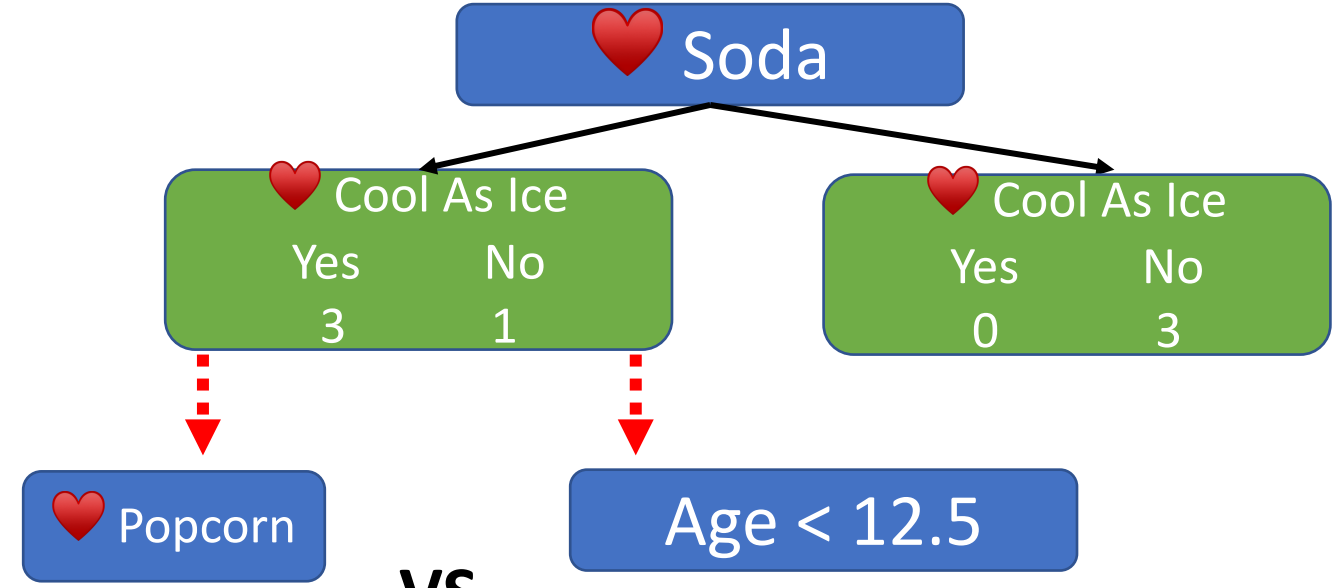
…only this time we only consider the **Ages** of people who **Love Soda**
As before, set 12.5, 26.5, 36.5 and calculate Impurities
…and **Age<12.5** gives us the **lowest Impurity**, **0**

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



Gini=0.25

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



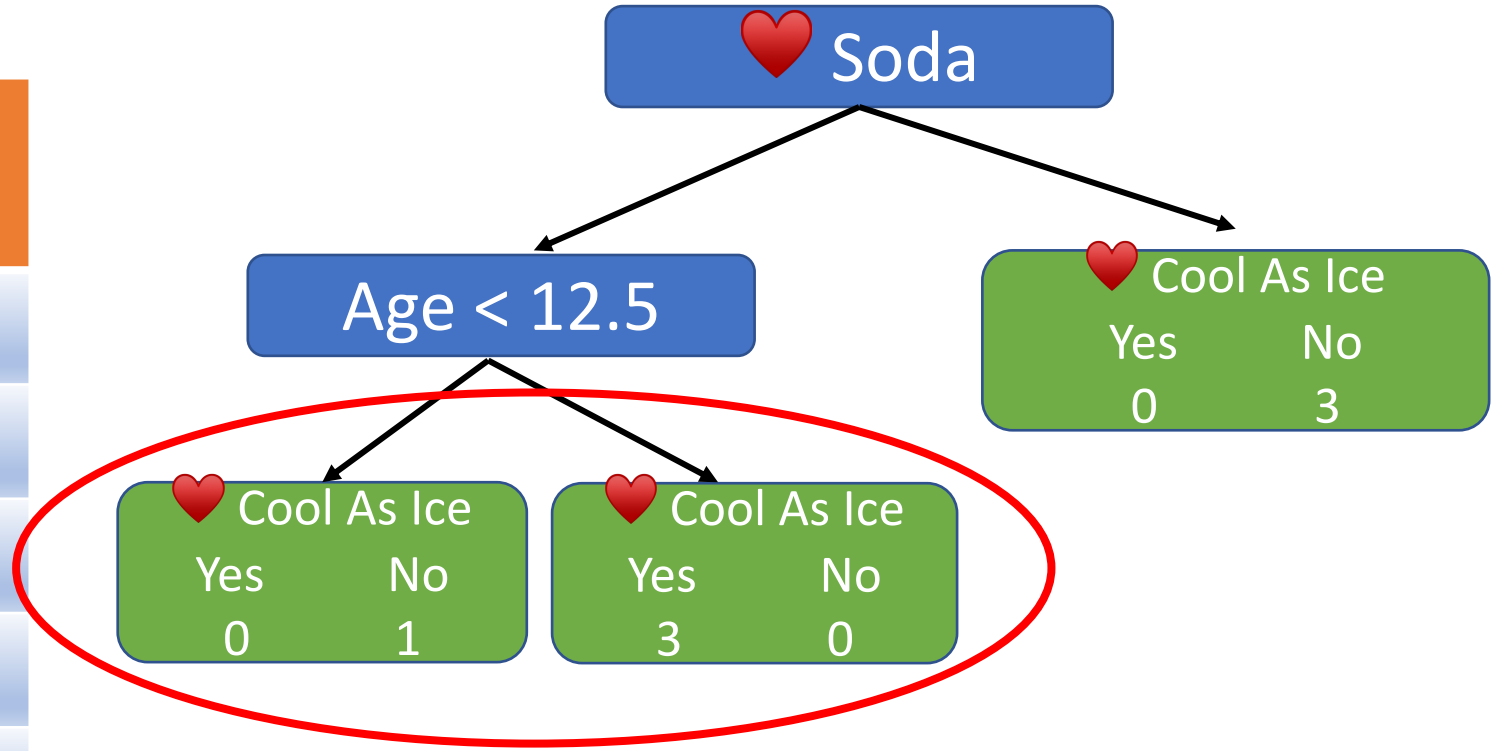
VS.

Gini=0.25

Gini=0

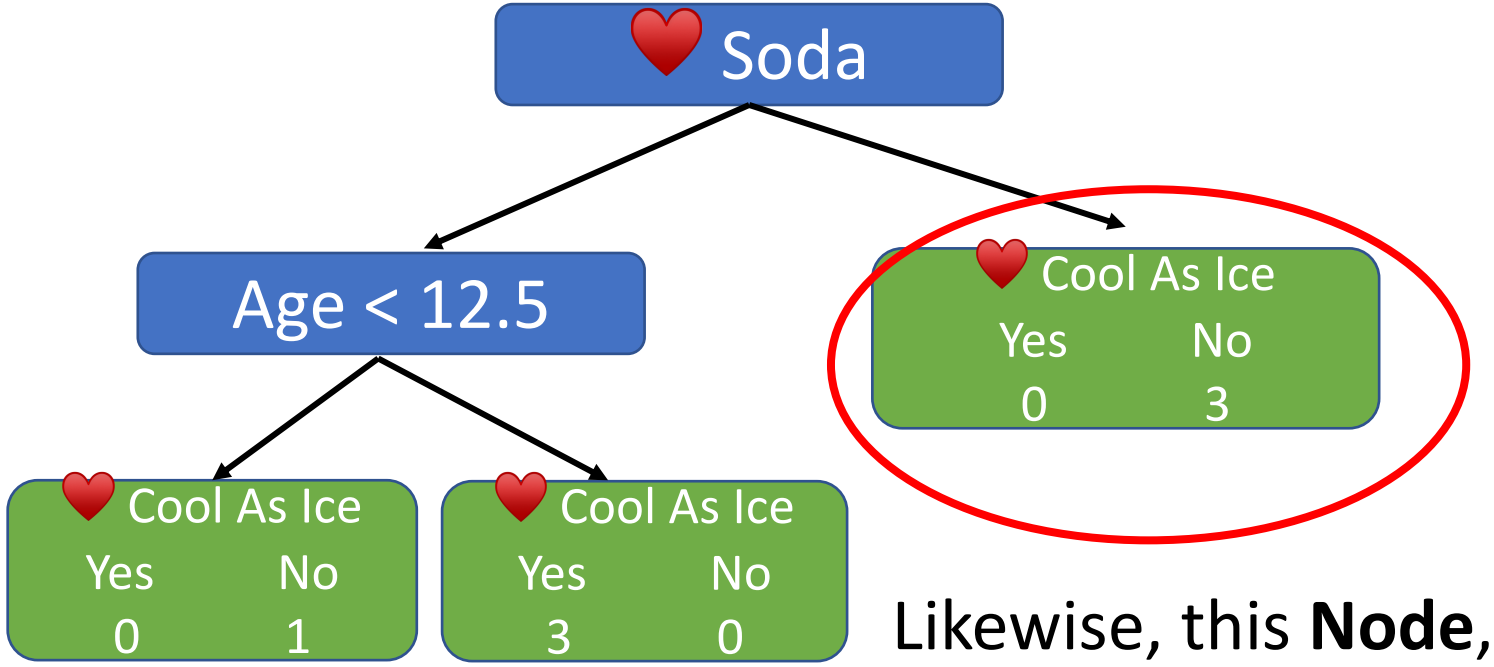
Now, because 0 is less than 0.25, we will use **Age < 12.5** to split this **Node** into **Leaves**.

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

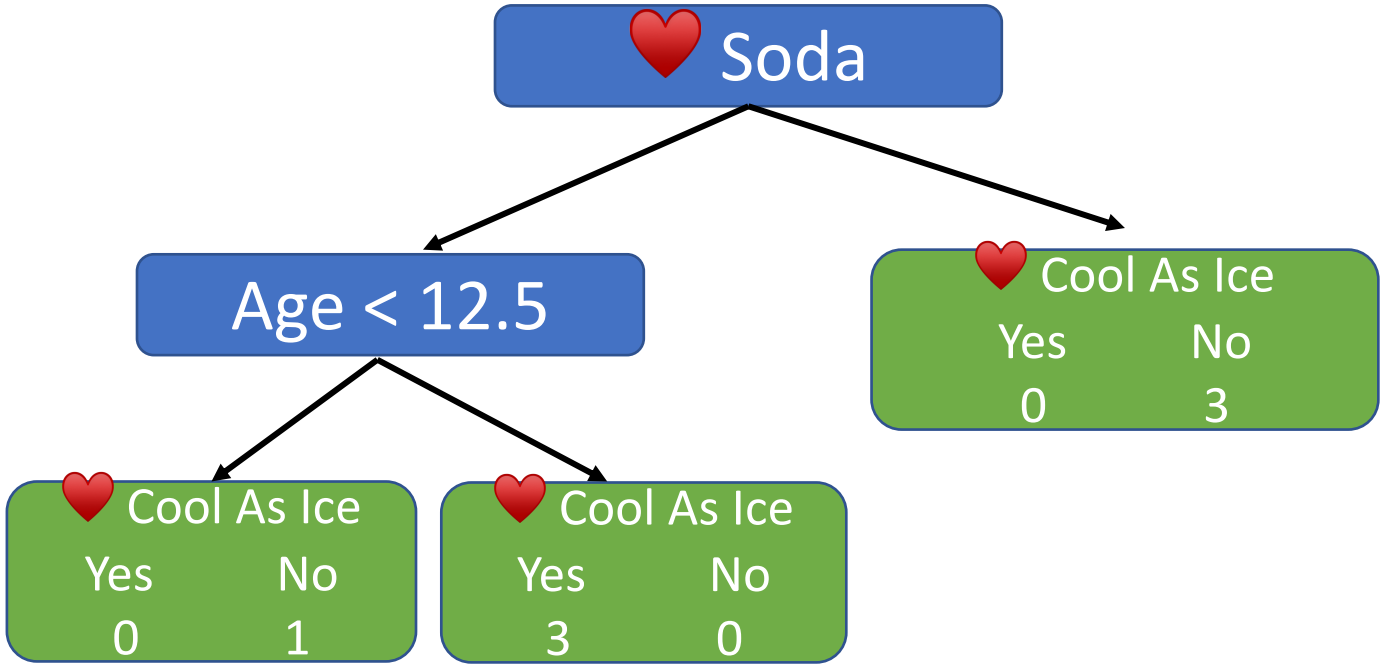


Note: These are **Leaves** because there is no reason to continue splitting these people into smaller groups

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



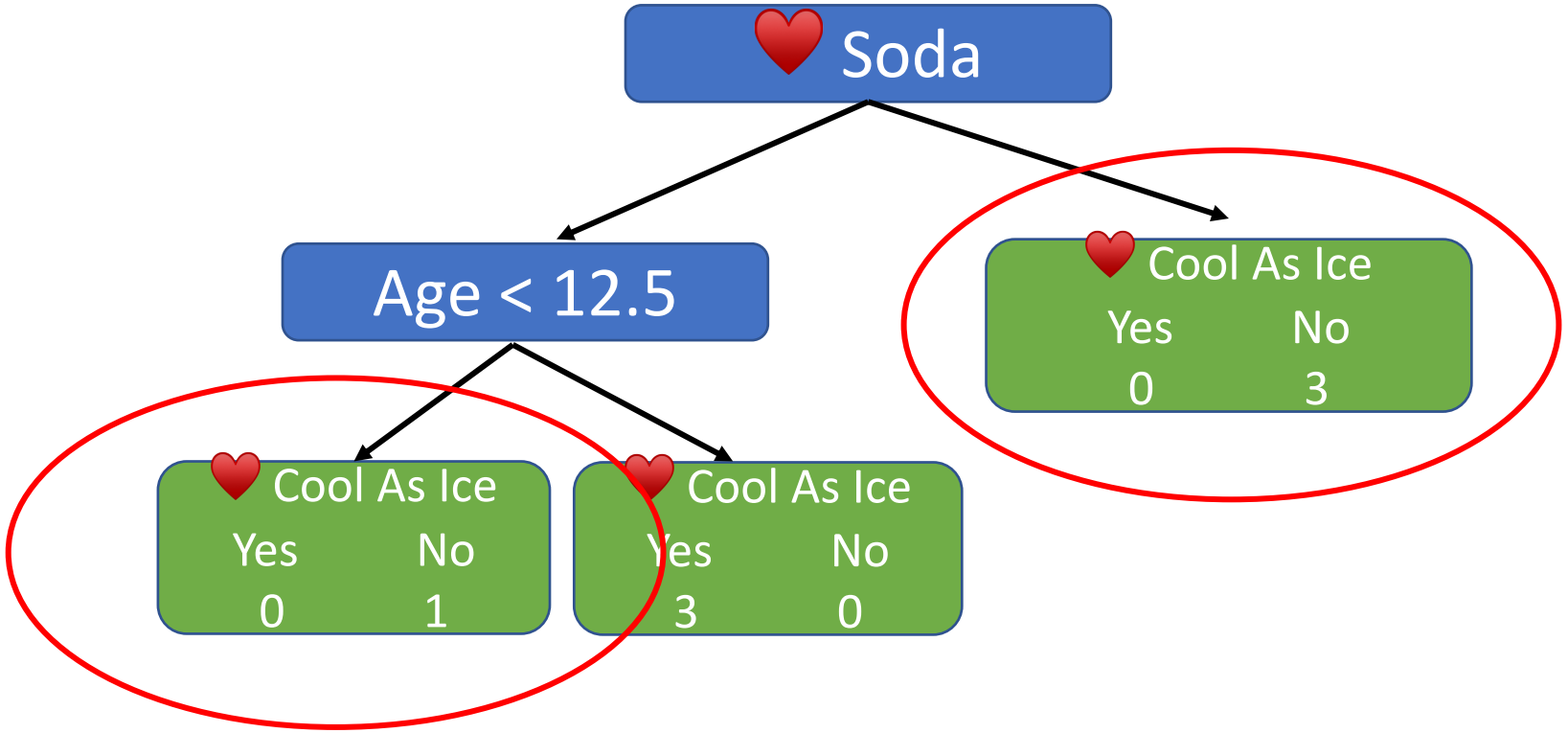
Likewise, this **Node**, consisting of the **3** people who do not **Love Soda**, is also a **Leaf**



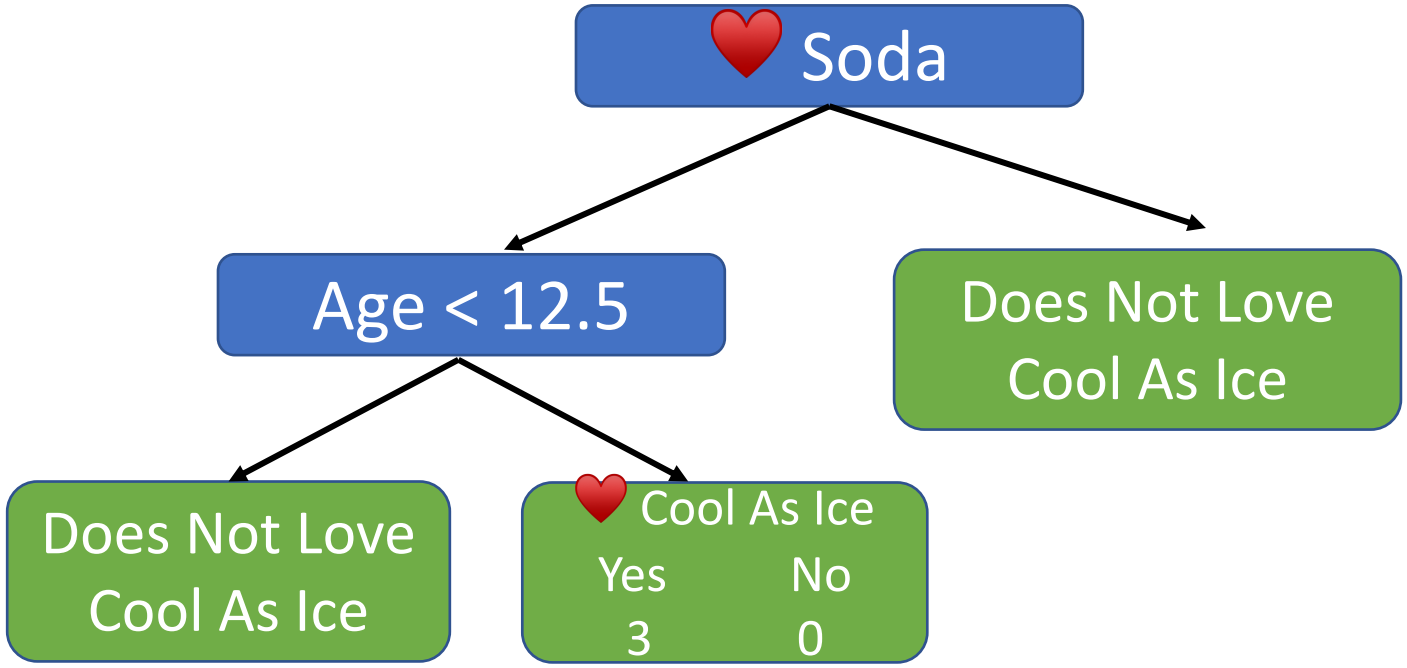
Now there is just one last thing we need to do before we are done building this tree

We need to assign output values for each **Leaf**

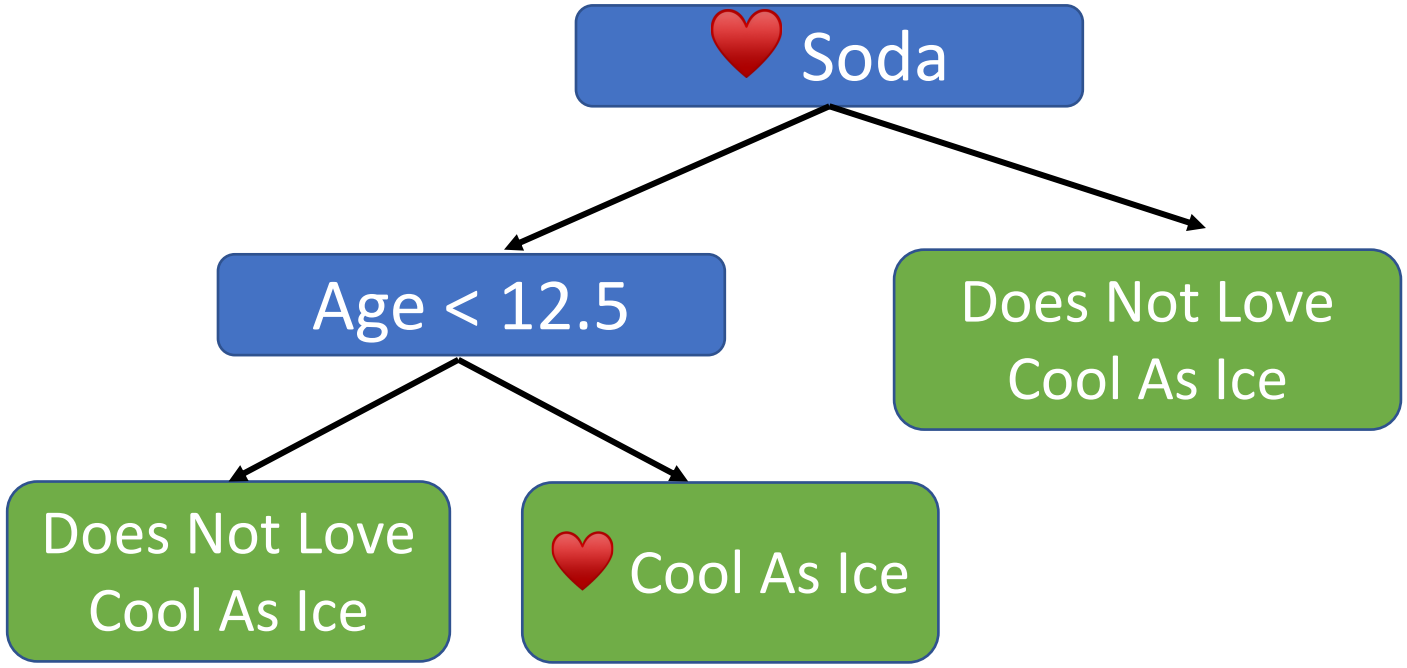
Generally speaking, the output of a **Leaf** is whatever category that has the most votes



Because the majority of the people in these **Leaves** do **not Love Cool As Ice**



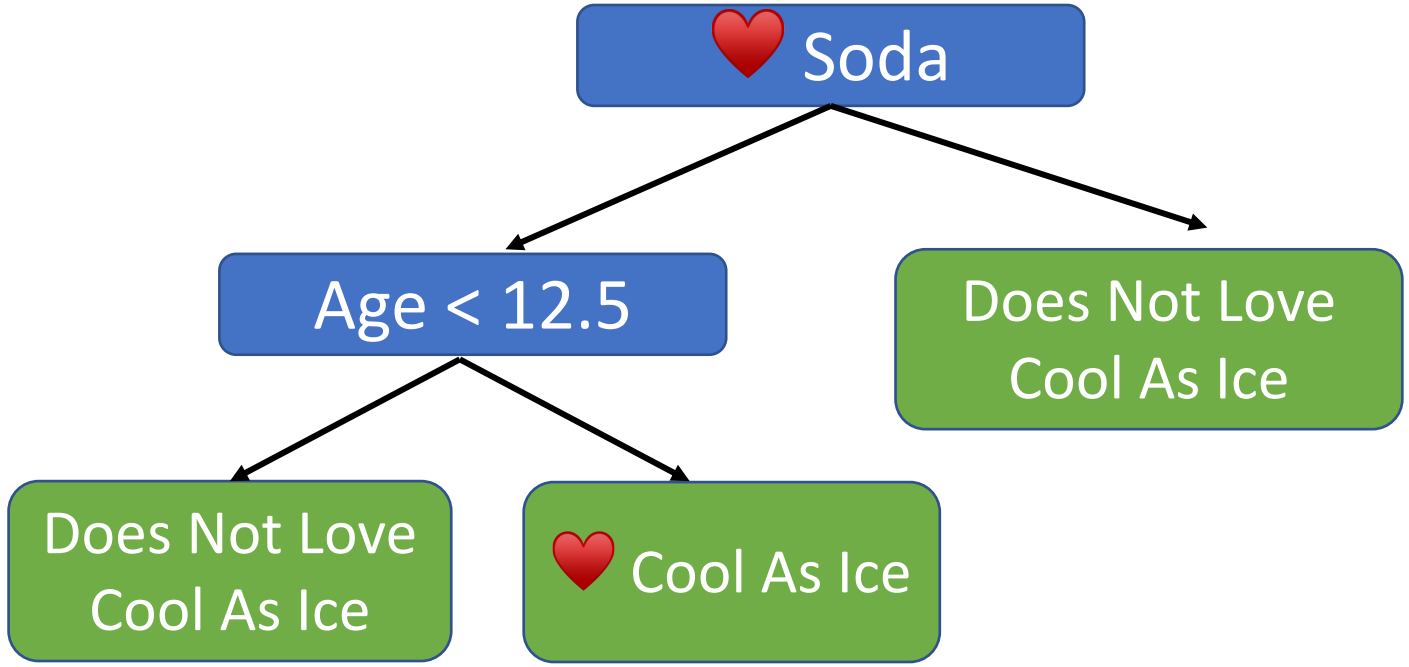
The output values are **Does Not Love Cool As Ice**



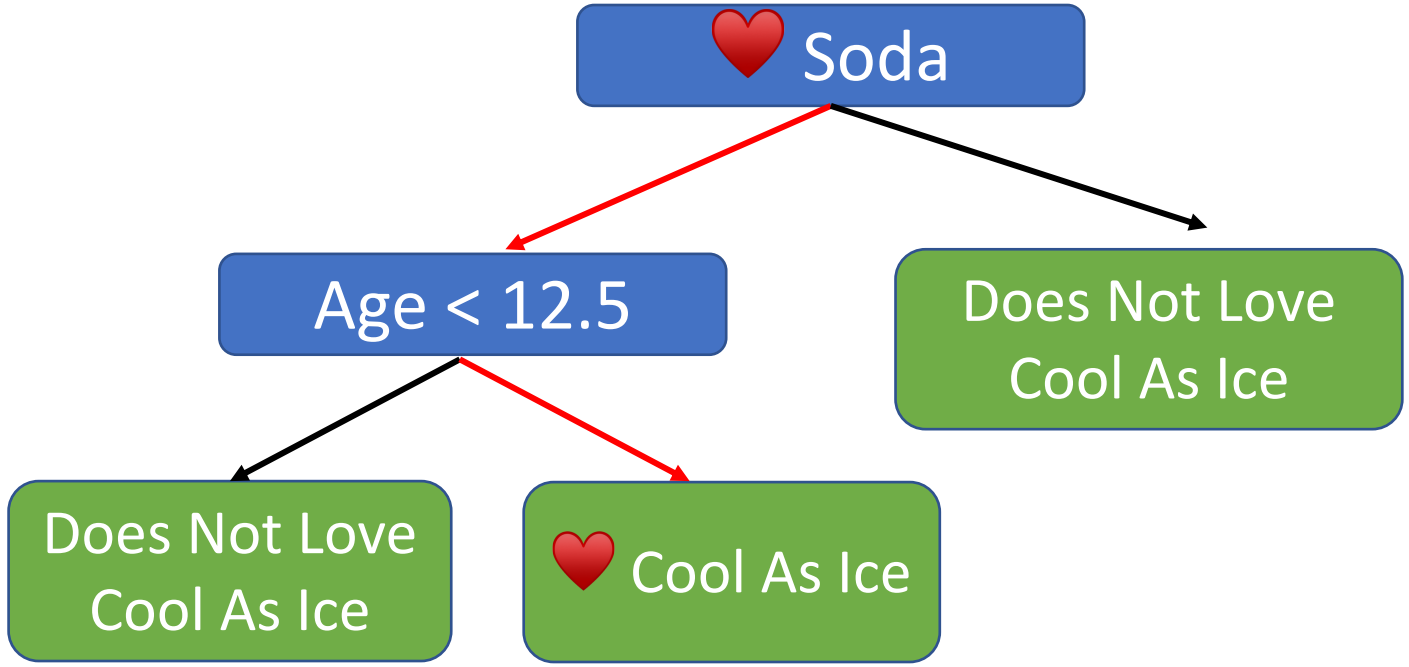
We finished building a **Tree** from this data

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	15	???

Suppose we want to predict if they will Love Cool As Ice

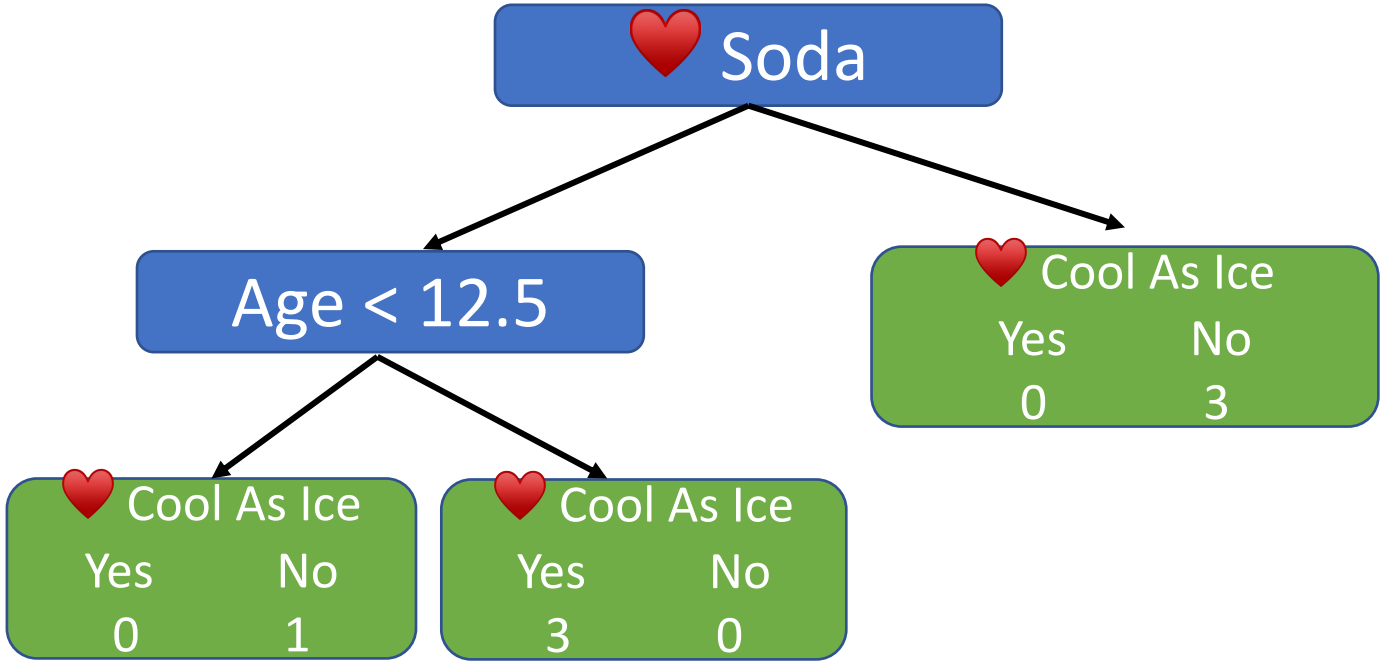


Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	15	???



Suppose we want to predict if they will Love Cool As Ice

Let’s discuss one technical detail

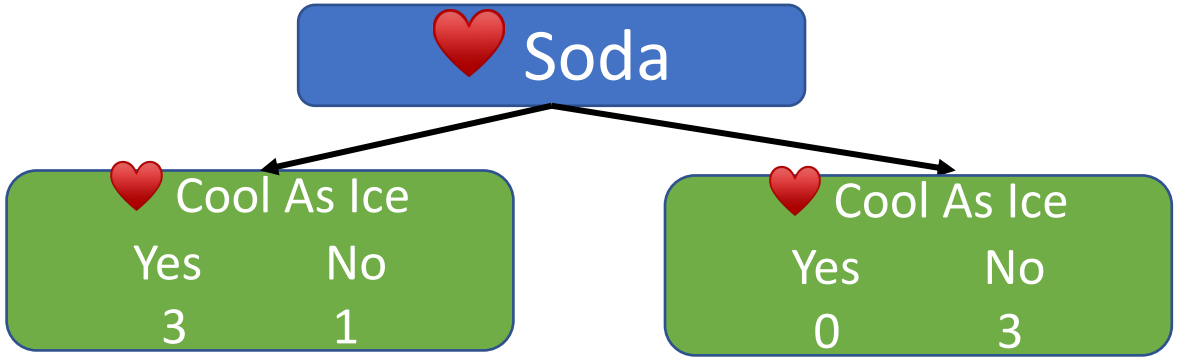


Remember, when we built this tree, only one person in the original dataset made it to this **Leaf**

Because so few people made it to this **Leaf**, it's hard to have confidence that it will do a great job making predictions with future data

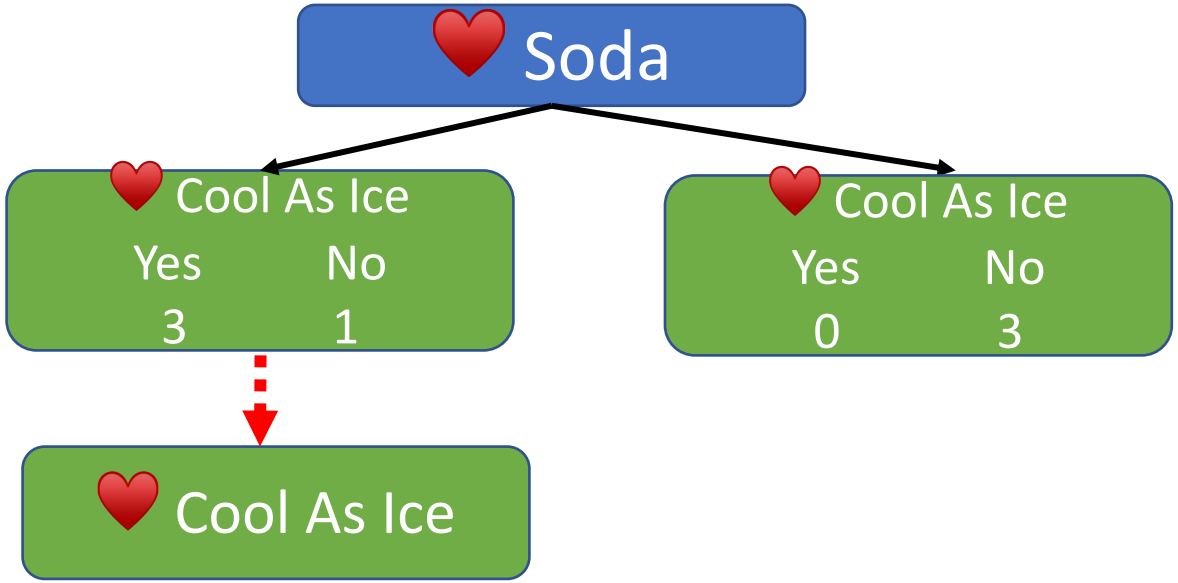
Overfitting problem

- It’s possible that we have **Overfit** the data, here
- In practice, there are two main ways to deal with this problem
  - One method is called **Pruning**
  - Alternatively, we can put limits on how trees grow
    - E.g. by requiring 3 or more people per leaf



Now we end up with an **Impure leaf**

But also a better sense of the accuracy of our prediction, because we know that only 75 % of the people in the **Leaf Loved Cool As Ice**



Even when a **Leaf** is **Impure** we still need an output value to make a classification

And since most of the people in this leaf Love Cool As Ice, that will be the output value

Cross Validation

- When we build a tree, we don’t know in advance if it is better to require 3 people per leaf or some other number
- So we test different values with Cross Validation and pick the one that works best