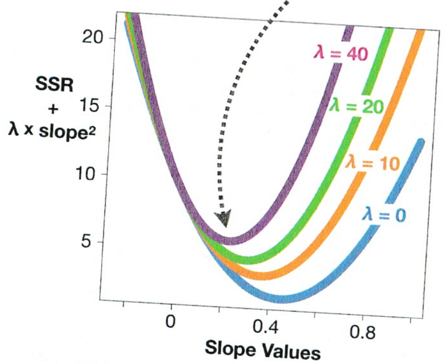
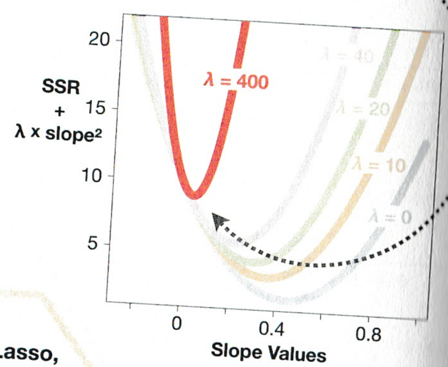
Ridge vs. Lasso Regularization: Details Part 7

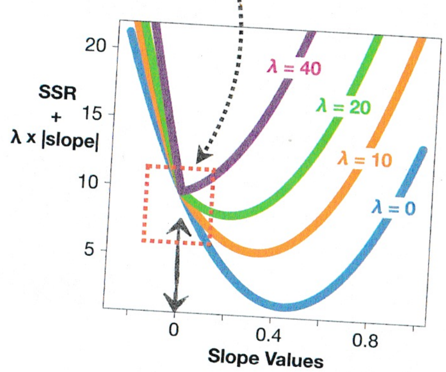
16 In summary, when we increase the **Ridge, Squared, or L2 Penalty** the optimal value for the slope shifts toward **0**, but we retain a nice **parabola** shape, and the optimal slope is never **0** itself.



NOTE: Even if we increase λ all the way to **400**, the **Ridge Penalty** gives us a smooth **red curve** and the lowest point corresponds to a slope value slightly greater than **0**.



17 In contrast, when we increase the **Lasso, Absolute Value**, or **L1 Penalty**, the optimal value for the slope shifts toward **0**, and since we have a kink at **0**, **0** ends up being the optimal slope.



TRIPLE BAM!!!

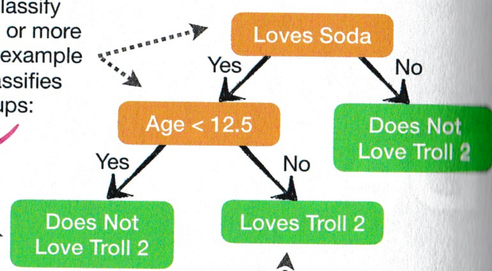
Chapter 10

Decision Trees!!!

Classification and Regression Trees: Main Ideas

1. There are two types of **Trees** in machine learning: trees for **Classification** and trees for **Regression**.

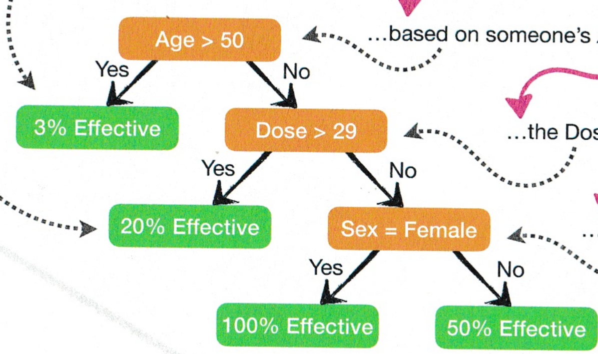
2. **Classification Trees** classify people or things into two or more *discrete* categories. This example **Classification Tree** classifies people into two groups:



...those who *do not* love Troll 2...

...and those who love Troll 2.

3. In contrast, **Regression Trees** try to predict a *continuous* value. This example **Regression Tree** tries to predict how Effective a drug will be...



...based on someone's Age...

...the Dose...

...and their Sex.

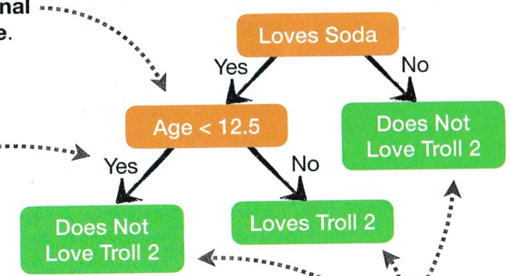
4. In this chapter, we'll cover the **Main Ideas** behind **Classification Trees** and **Regression Trees** and describe the most commonly used methods to build them. But first, it's time for the dreaded...

Terminology Alert!!! Decision Tree Lingo

1. The good news is that **Decision Trees** are pretty simple, and there's not too much terminology.

The very top of the tree is called the **Root Node** or just the **Root**.

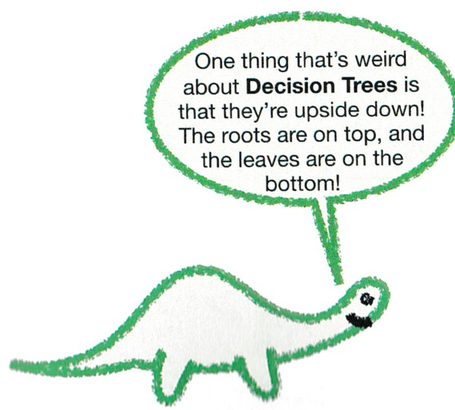
This is called an **Internal Node** or just a **Node**.



The arrows are called **Branches**. In this example, the **Branches** are labeled with **Yes** or **No**, but usually it's assumed that if a statement in a **Node** is ***True***, you go to the *Left*, and if it's ***False***, you go to the *Right*.

These are called **Leaf Nodes** or just **Leaves**.

2. Now that we know the lingo, let's learn about **Classification Trees**!!!



Decision Trees
Part One:

Classification Trees

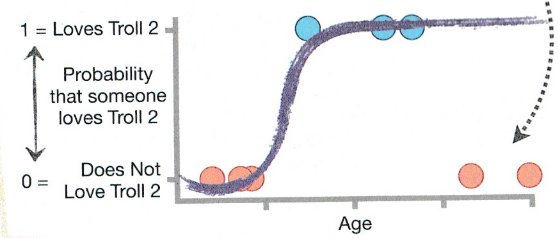
Classification Trees: Main Ideas

1 **The Problem:** We have a mixture of **discrete** and **continuous** data…

Loves Popcorn	Loves Soda	Age	Loves Troll 2
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

…that we want to use to predict if someone loves Troll 2, the **1990** blockbuster movie that was neither about trolls nor a sequel.

Unfortunately, we can't use **Logistic Regression** with these data because when we plot Age vs. Loves Troll 2, we see that fitting an **s-shaped squiggle** to the data would be a terrible idea: both young and old people *do not* love Troll 2, with the people who love Troll 2 in the middle. In this example, an **s-shaped squiggle** will incorrectly classify *all* of the older people.

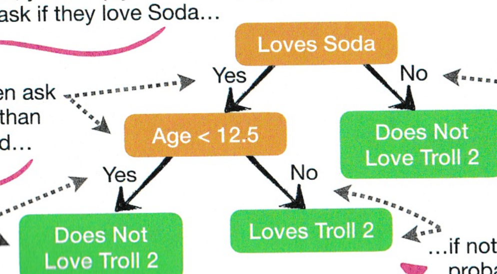


2 **A Solution:** A **Classification Tree**, which can handle all types of data, all types of relationships among the independent variables (the data we're using to make predictions, like Loves Soda and Age), and all kinds of relationships with the dependent variable (the thing we want to predict, which in this case is Loves Troll 2).

Classification Trees are relatively easy to interpret and use. If you meet a new person and want to decide if they love Troll 2 or not, you simply start at the top and ask if they love Soda…

BAM!!!

…if they do, then ask if they're less than **12.5** years old…



…and if they don't love Soda to begin with, they probably *do not* love Troll 2.

…if they are, then they probably *do not* love Troll 2…

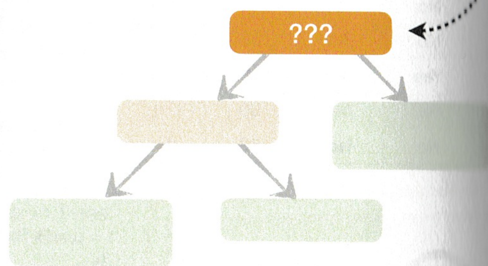
…if not, then they probably love Troll 2…

Building a Classification Tree: Step-by-Step

1 Given this **Training Dataset**, we want to build a **Classification Tree** that uses Loves Popcorn, Loves Soda, and Age to predict whether or not someone will love Troll 2.

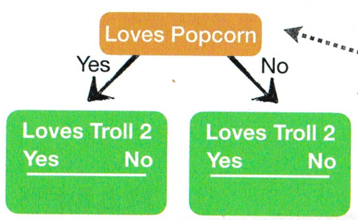
Loves Popcorn	Loves Soda	Age	Loves Troll 2
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

2 The first thing we do is decide whether Loves Popcorn, Loves Soda, or Age should be the question we ask at the very top of the tree.



3 To make that decision, we'll start by looking at how well Loves Popcorn predicts whether or not someone loves Troll 2...

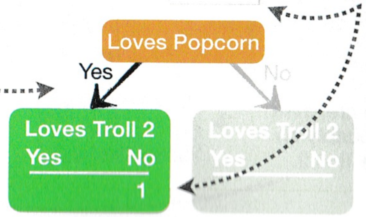
...by making a super simple tree that only asks if someone loves Popcorn and running the data down it.



4 For example, the first person in the **Training Data** loves Popcorn, so they go to the **Leaf** on the left...

Loves Popcorn	Loves Soda	Age	Loves Troll 2
Yes	Yes	7	No

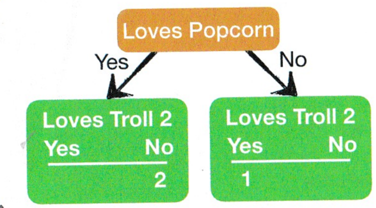
...and because they *do not* love Troll 2, we'll put a **1** under the word **No**.



Building a Classification Tree: Step-by-Step

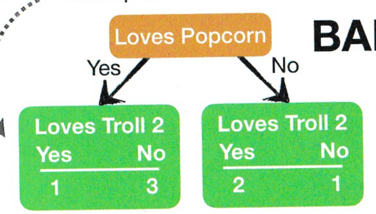
5 The second person also loves Popcorn, so they also go to the **Leaf** on the *left*, and because they do not love Troll 2, we increase **No** to **2**.

Loves Popcorn	Loves Soda	Age	Loves Troll 2
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



6 The third person *does not* love Popcorn, so they go to the **Leaf** on the *right*, but they love Troll 2, so we put a **1** under the word **Yes**.

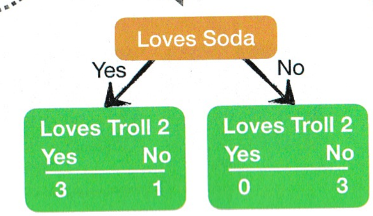
7 Likewise, we run the remaining rows down the tree, keeping track of whether or not each person loves Troll 2 or not.



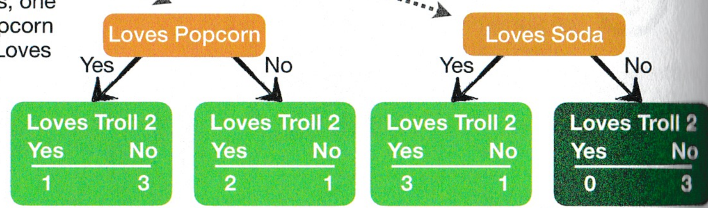
BAM!!!

8 Then we do the same thing for Loves Soda.

Loves Popcorn	Loves Soda	Age	Loves Troll 2
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
...	...	...	...



9 Now, looking at the two little trees, one for Loves Popcorn and one for Loves Soda…



…we see that these three **Leaves** contain mixtures of people who love and do not love Troll 2.

In contrast, this **Leaf** only contains people who *do not* love Troll 2.

TERMINOLOGY ALERT!!!

Leaves that contain mixtures of classifications are called **Impure**.

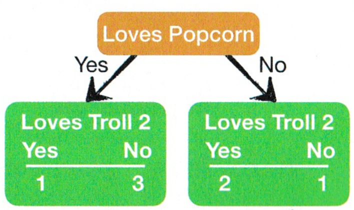
10 Because both **Leaves** in the Loves Popcorn tree are **Impure** and only one **Leaf** in the Loves Soda tree is **Impure**…

…it seems like Loves Soda does a better job classifying who loves and does not love Troll 2, but it would be nice if we could *quantify* the differences between Loves Popcorn and Loves Soda.

11 The good news is that there are several ways to quantify the **Impurity** of **Leaves** and **Trees**.

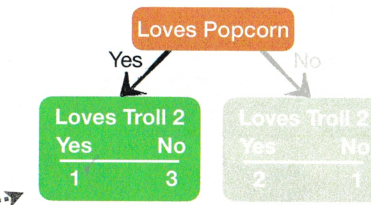
One of the most popular methods is called **Gini Impurity**, but there are also fancy-sounding methods like **Entropy** and **Information Gain**.

In theory, all of the methods give similar results, so we'll focus on **Gini Impurity** since it's very popular and I think it's the most straightforward.



We'll start by calculating the **Gini Impurity** to quantify the **Impurity** in the **Leaves** for loves Popcorn.

12 To calculate the **Gini Impurity** for Loves Popcorn, first we calculate the **Gini Impurity** for each individual **Leaf**. So, let's start by plugging the numbers from the *left* **Leaf** into the equation for **Gini Impurity**.



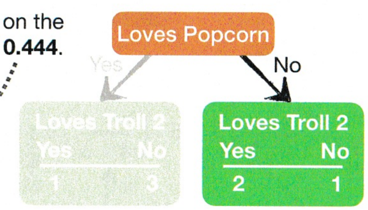
13 **Gini Impurity** for a **Leaf** = 1 - (the probability of "yes")^2 - (the probability of "no")^2

14 For the **Leaf** on the *left*, when we plug the numbers, **Yes = 1**, **No = 3**, and **Total = 1 + 3**, into the equation for **Gini Impurity**, we get **0.375**.

$$= 1 - \left(\frac{\text{The number for Yes}}{\text{The total for the Leaf}}\right)^2 - \left(\frac{\text{The number for No}}{\text{The total for the Leaf}}\right)^2$$

$$= 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2 \boxed{= 0.375}$$

15 For the **Leaf** on the *right*, we get **0.444**.



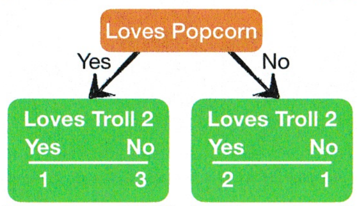
Gini Impurity for a **Leaf** = 1 - (the probability of "yes")^2 - (the probability of "no")^2

$$= 1 - \left(\frac{2}{2+1}\right)^2 - \left(\frac{1}{2+1}\right)^2 \boxed{= 0.444}$$

Building a Classification Tree: Step-by-Step

16 Now, because the **Leaf** on the *left* has **4** people in it…

…and the **Leaf** on the *right* only has **3**, the **Leaves** do not represent the same number of people.



Gini Impurity = 0.375 Gini Impurity = 0.444

So, to compensate for the differences in the number of people in each **Leaf**, the total **Gini Impurity** for Loves Popcorn is the **Weighted Average of the two Leaf Impurities.**

17 Total **Gini Impurity** = weighted average of **Gini Impurities** for the **Leaves**

18 The weight for the *left* **Leaf** is the total number of people in the **Leaf**, **4**…

…divided by the total number of people in both **Leaves**, **7**…

…then we multiply that weight by its associated **Gini Impurity**, **0.375**.

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

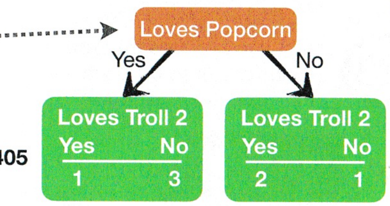
Now we add to that the weight for the *right* **Leaf**, the total number of people in the **Leaf**, **3**, divided by to the total in both **Leaves**, **7**…

…multiplied by the associated **Gini Impurity**, **0.444**…

…and when we do the math, we get **0.405**. **BAM!!!**

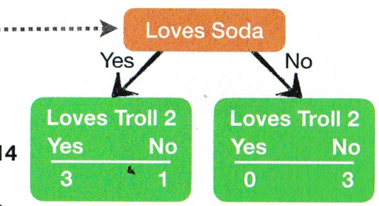
Building a Classification Tree: Step-by-Step

19 Now that we've calculated the **Gini Impurity** for Loves Popcorn, **0.405**…



Gini Impurity for **Loves Popcorn** = **0.405**

…we can follow the same steps to calculate the **Gini Impurity** for Loves Soda, **0.214**.



Gini Impurity for **Loves Soda** = **0.214**

20 The lower **Gini Impurity** for Loves Soda, **0.214**, confirms what we suspected earlier, that Loves Soda does a better job classifying people who love and do not love Troll 2. However, now that we've *quantified* the difference, we no longer have to rely on intuition. Bam!

21 Now we need to calculate the **Gini Impurity** for Age.

Loves Popcorn	Loves Soda	Age	Loves Troll 2
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

However, because Age contains numeric data, and not just **Yes/No** values, calculating the **Gini Impurity** is a little more involved.

Normally, the first thing we do is sort the rows by Age, from lowest to highest, but in this case, the data were already sorted, so we can skip this step.

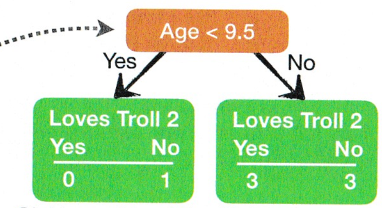
22 The next thing we do is calculate the average Age for all adjacent rows.

Average Age	Age	Loves Troll 2
	7	No
9.5	12	No
15	18	Yes
26.5	35	Yes
36.5	38	Yes
44	50	No
66.5	83	No

23 Lastly, we calculate the **Gini Impurity** values for each average Age.

	Age	Loves Troll 2
	7	No
9.5		
	12	No
15		
	18	Yes
26.5		
	35	Yes
36.5		
	38	Yes
44		
	50	No
66.5		
	83	No

For example, the first average Age is **9.5**, so we use **9.5** as the threshold for splitting the rows into **2** leaves...



Gini Impurity = 0.0   Gini Impurity = 0.5

$$\text{Total Gini Impurity} = \left(\frac{1}{1+6}\right)0.0 + \left(\frac{6}{1+6}\right)0.5$$

$$\boxed{= 0.429}$$

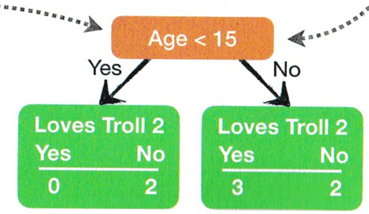
...and when we do the math, we get **0.429**.

24 Ultimately, we end up with a **Gini Impurity** for each potential threshold for Age...

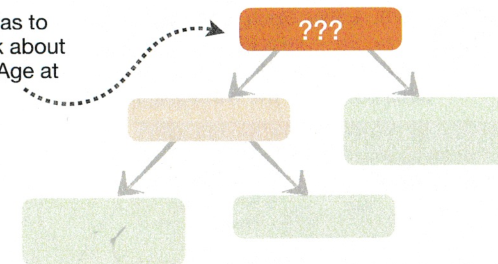
	Age	Loves Troll 2	
	7	No	
9.5			Gini Impurity = 0.429
	12	No	
15			Gini Impurity = 0.343
	18	Yes	
26.5			Gini Impurity = 0.476
	35	Yes	
36.5			Gini Impurity = 0.476
	38	Yes	
44			Gini Impurity = 0.343
	50	No	
66.5			Gini Impurity = 0.429
	83	No	

...and then we identify the thresholds with the lowest **Impurities**, and because the candidate thresholds **15** and **44** are tied for the lowest **Impurity**, **0.343**, we can pick either one for the **Root**. In this case, we'll pick **15**.

BAM!!!

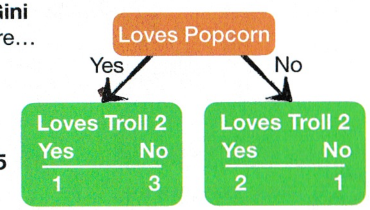


25 Now remember: our first goal was to determine whether we should ask about Loves Popcorn, Loves Soda, or Age at the very top of the tree...

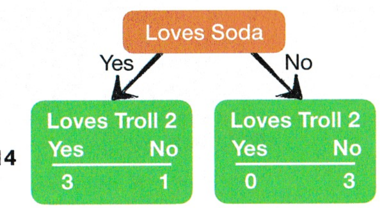


26 ...so we calculated the **Gini Impurities** for each feature...

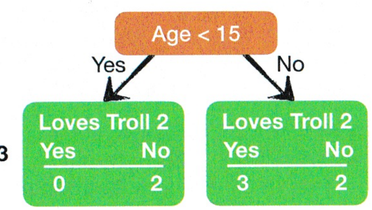
Gini Impurity for **Loves Popcorn** = **0.405**



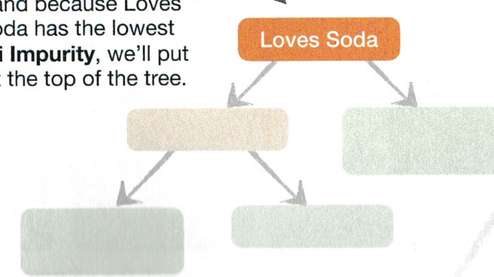
Gini Impurity for **Loves Soda** = **0.214**



Gini Impurity for **Age < 15** = **0.343**



27 ...and because Loves Soda has the lowest **Gini Impurity**, we'll put it at the top of the tree.



BAM!!!

Building a Classification Tree: Step-by-Step

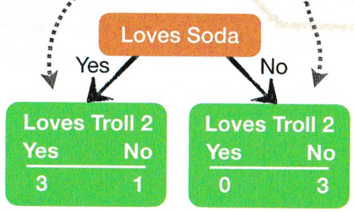
28 With Loves Soda at the top of the tree, the **4** people who love Soda, including **3** who love Troll 2 and **1** who does not, go to the **Node** on the *left*…

Loves Popcorn	Loves Soda	Age	Loves Troll 2
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

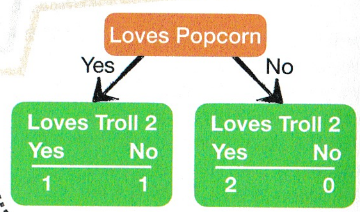
29 …and the **3** people who *do not* love Soda, all of whom *do not* love Troll 2, go to the **Node** on the *right*.

Loves Popcorn	Loves Soda	Age	Loves Troll 2
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

30 Now, because the **Node** on the *left* is **Impure**, we can split the **4** people in it based on Loves Popcorn or Age and calculate the **Gini Impurities**.

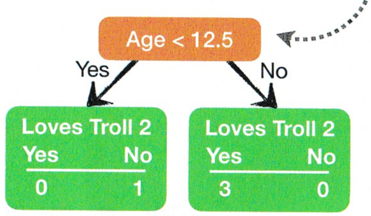


31 When we split the **4** people who love Soda based on whether or not they love Popcorn, the **Gini Impurity** is **0.25**.



Gini Impurity = 0.25

However, when we split the **4** people based on Age **< 12.5**, the **Gini Impurity** is **0**.



Gini Impurity = 0.0

Loves Popcorn	Loves Soda	Age	Loves Troll 2
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

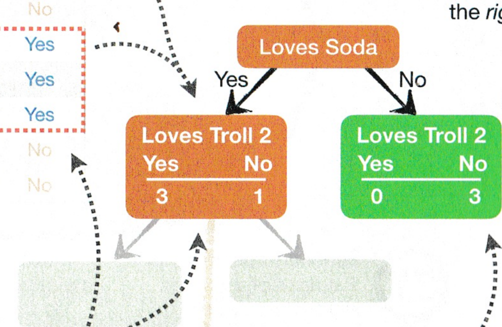
12.5
26.5
36.5

Building a Classification Tree: Step-by-Step

32 Now remember, earlier we put Loves Soda in the **Root** because splitting every person in the **Training Data** based on whether or not they love Soda gave us the *lowest* **Gini Impurity**. So, the **4** people who love Soda went to the *left* **Node**…

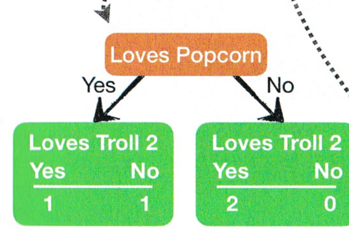
Loves Popcorn	Loves Soda	Age	Loves Troll 2
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

33 …and the **3** people who *do not* love Soda, all of whom *do not* love Troll 2, went to the **Node** on the *right*.

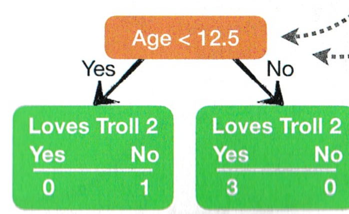


Now, because *everyone* in this **Node** *does not* love **Troll 2**, it becomes a **Leaf**, because there's no point in splitting the people up into smaller groups.

34 In contrast, because the **4** people who love Soda are a mixture of people who *do* and *do not* love Troll 2, we build simple trees with them based on Loves Popcorn and Age…



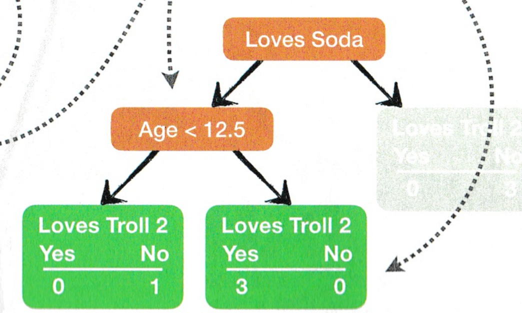
Gini Impurity = 0.25



Gini Impurity = 0.0

35 …and because Age **< 12.5** resulted in the lowest **Gini Impurity**, **0**, we add it to the tree.

And the new **Nodes** are **Leaves** because neither is **Impure**.

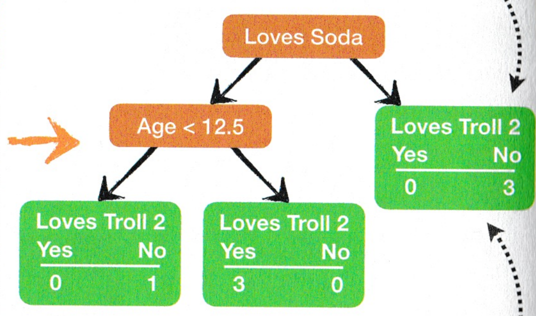


Building a Classification Tree: Step-by-Step

36 At this point, we've created a **Tree** from the **Training Data**.

Loves Popcorn	Loves Soda	Age	Loves Troll 2
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

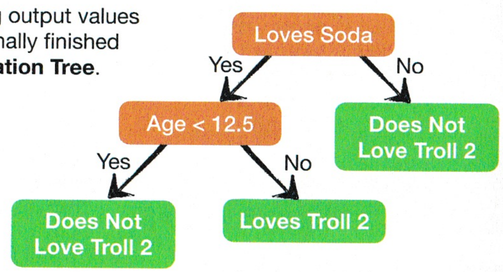
Now the only thing remaining is to assign output values for each **Leaf**.

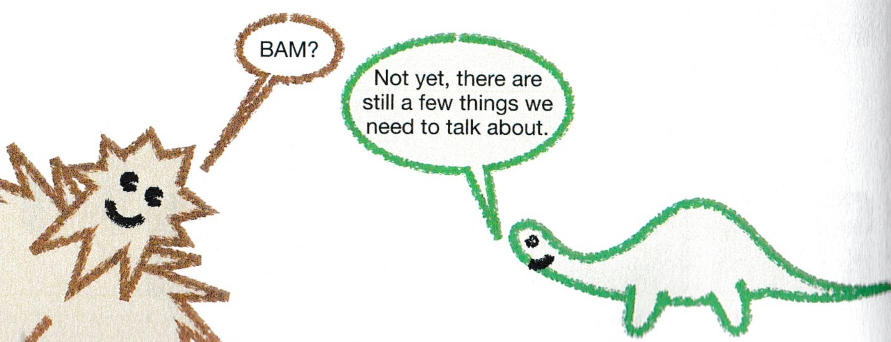


37 Generally speaking, the output of a **Leaf** is whatever category that has the most counts.

In other words, because the majority of the people in this **Leaf** *do not* love Troll 2, its output value is does not love Troll 2.

38 Hooray!!! After assigning output values to each **Leaf**, we've finally finished building a **Classification Tree**.





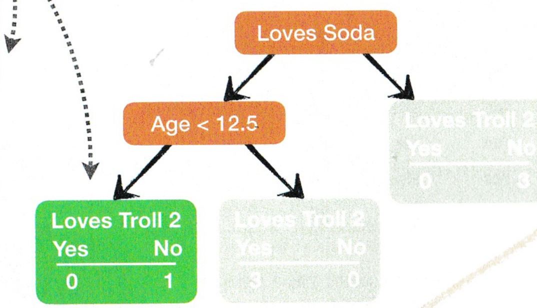
Building a Classification Tree: Step-by-Step

39 When we built this tree, only one person in the **Training Data** made it to this **Leaf**...

...and because so few people in the **Training Data** made it to that **Leaf**, it's hard to have confidence that the tree will do a great job making predictions with future data.

Loves Popcorn	Loves Soda	Age	Loves Troll 2
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

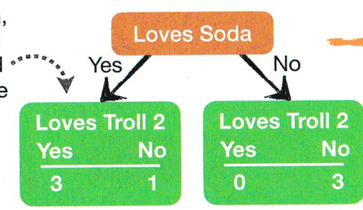
However, in practice, there are two main ways to deal with this type of problem.

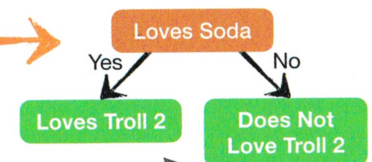


40 One method is called **Pruning**, but we'll save that topic for **The StatQuest Illustrated Guide to Tree-Based Machine Learning!!!**

41 Alternatively, we can put limits on how trees grow, for example, by requiring **3** or more people per **Leaf**.

If we did that with our **Training Data**, we would end up with this tree, and this **Leaf** would be **Impure**...





...but we would also have a better sense of the accuracy of our prediction because we know that only **75%** of the people in the **Leaf** love Troll 2.

NOTE: When we build a tree, we don't know in advance if it's better to require **3** people per **Leaf** or some other number, so we try a bunch, use **Cross Validation,** and pick the number that works best.

ALSO NOTE: Even though this **Leaf** is **Impure**, it still needs an output value, and because most of the people in this **Leaf** love Troll 2, that will be the output value.

BAM!!!

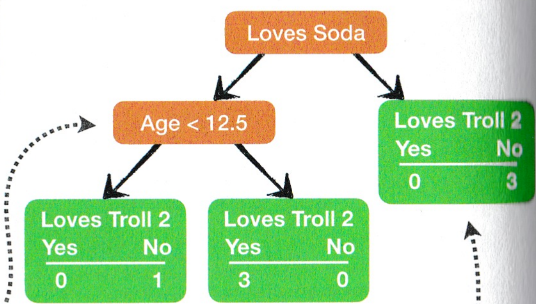
Now let's summarize how to build a **Classification Tree**.

Building a Classification Tree: Summary

1 From the entire **Training Dataset**, we used **Gini Impurity** to select Loves Soda for the **Root** of the tree.

As a result, the **4** people who love Soda went to the *left* and the **3** people who do not love Soda went to the *right*.
BAM!

Loves Popcorn	Loves Soda	Age	Loves Troll 2
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

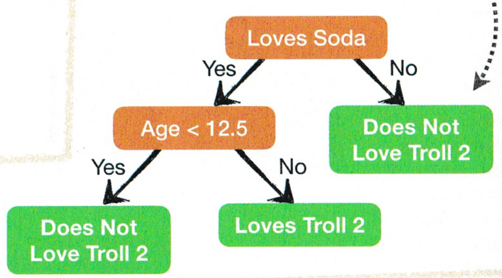


2 Then we used the **4** people who love Soda, which were a mixture of people who *do* and *do not* love Troll 2, to calculate **Gini Impurities** and selected **Age < 12.5** for the next **Node**.

Double BAM!!

Loves Popcorn	Loves Soda	Age	Loves Troll 2
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

3 Then we selected output values for each **Leaf** by picking the categories with the highest counts.



Decision Trees
Part Deux:

Regression Trees