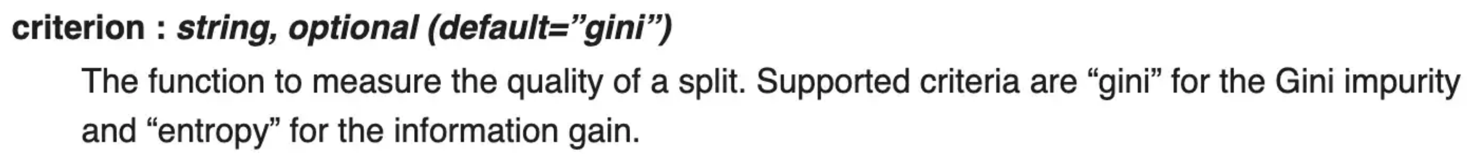
A Simple Explanation of Gini Impurity

What Gini Impurity is (with examples) and how it's used to train Decision Trees.

MARCH 29, 2019

If you look at the documentation for the DecisionTreeClassifier class in scikit-learn, you'll see something like this for the `criterion` parameter:



criterion : ***string, optional (default="gini")***

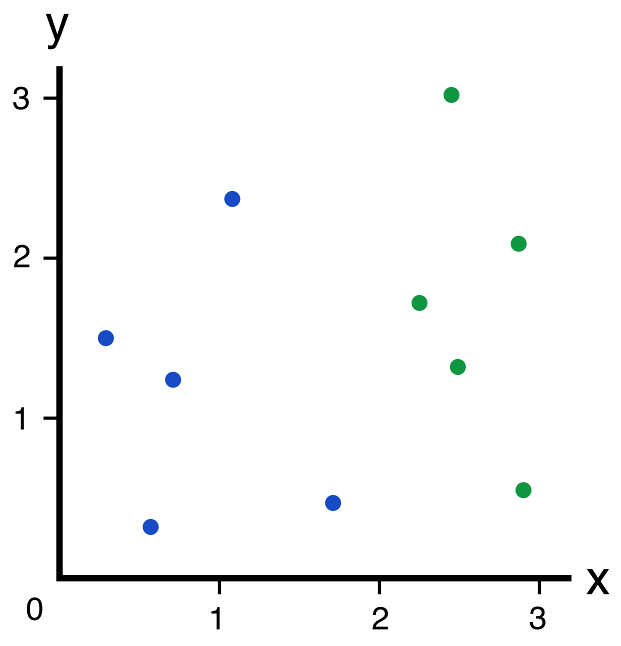
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

The RandomForestClassifier documentation says the same thing. Both mention that the default criterion is "gini" for the **Gini Impurity**. What is that?!

> *TLDR: Read the Recap.*

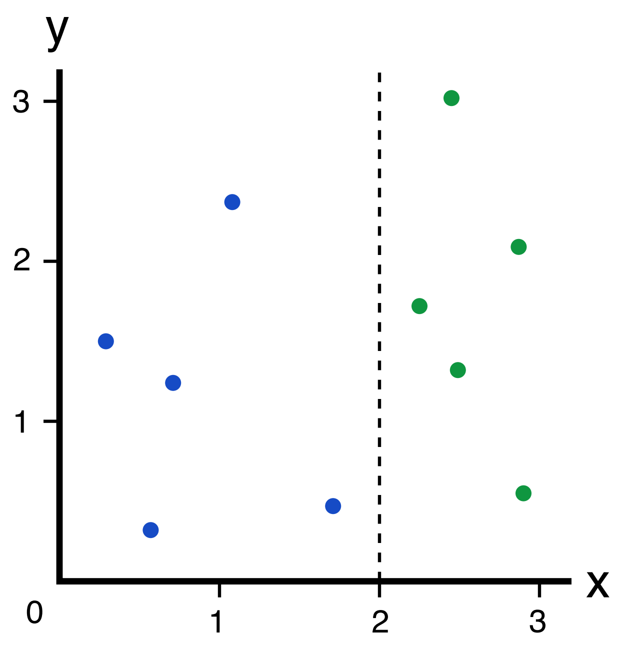
Decision Trees 🌲

Training a decision tree consists of iteratively splitting the current data into two branches. Say we had the following datapoints:



Right now, we have 1 branch with 5 blues and 5 greens.

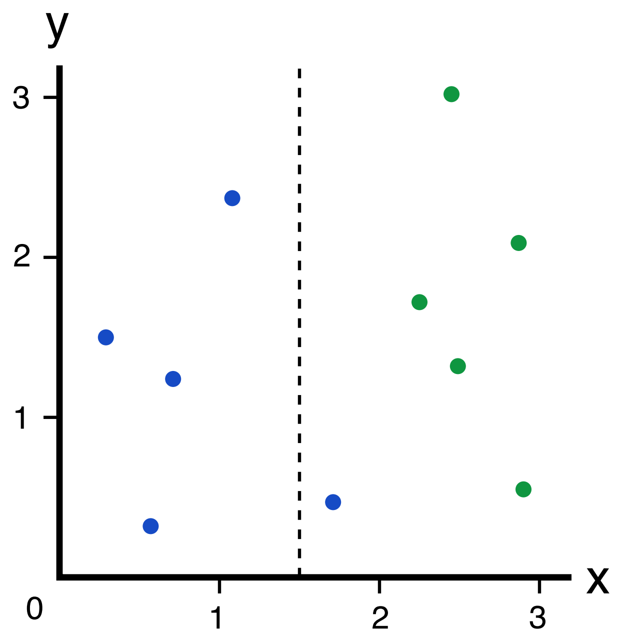
Let's make a split at $x = 2$:



This is a **perfect** split! It breaks our dataset perfectly into two branches:

- Left branch, with 5 blues.
- Right branch, with 5 greens.

What if we'd made a split at $x = 1.5$ instead?



This imperfect split breaks our dataset into these branches:

- Left branch, with 4 blues.
- Right branch, with 1 blue and 5 greens.

It's obvious that this split is worse, but **how can we quantify that?**

Being able to measure the quality of a split becomes even more important if we add a third class, reds . Imagine the following split:

- Branch 1, with 3 blues, 1 green, and 1 red.

- Branch 2, with 3 greens and 1 red.

Compare that against this split:

- Branch 1, with 3 blues, 1 green, and 2 reds.
- Branch 2, with 3 greens.

Which split is better? It’s no longer immediately obvious. We need a way to **quantitatively evaluate** how good a split is.

Gini Impurity

This is where the Gini Impurity metric comes in.

Suppose we

1. Randomly pick a datapoint in our dataset, then
2. **Randomly classify it according to the class distribution in the dataset**. For our dataset, we’d classify it as blue $\frac{5}{10}$ of the time and as green $\frac{5}{10}$ of the time, since we have 5 datapoints of each color.

What’s the probability we classify the datapoint incorrectly? The answer to that question is the Gini Impurity.

Example 1: The Whole Dataset

Let’s calculate the Gini Impurity of our entire dataset. If we randomly pick a datapoint, it’s either blue (50%) or green (50%).

Now, we randomly classify our datapoint according to the class distribution. Since we have 5 of each color, we classify it as blue 50% of the time and as green 50% of the time.

What's the probability we classify our datapoint **incorrectly**?

Event	Probability
Pick Blue, Classify Blue ✓	25%
Pick Blue, Classify Green ❌	25%
Pick Green, Classify Blue ❌	25%
Pick Green, Classify Green ✓	25%

We only classify it incorrectly in 2 of the events above. Thus, our total probability is 25% + 25% = 50%, so the Gini Impurity is $\boxed{0.5}$.

The Formula

If we have C total classes and $p(i)$ is the probability of picking a datapoint with class i, then the Gini Impurity is calculated as

$$G = \sum_{i=1}^{C} p(i) * (1 - p(i))$$

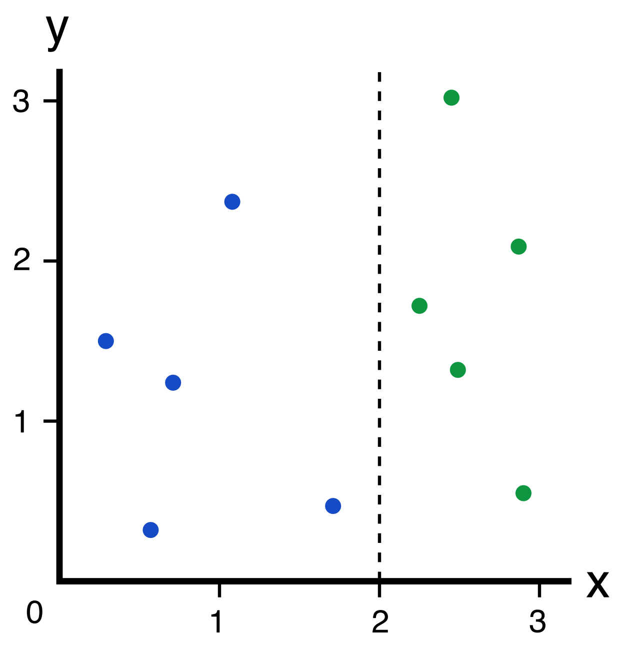
For the example above, we have $C = 2$ and $p(1) = p(2) = 0.5$, so

$$\begin{aligned} G &= p(1) * (1 - p(1)) + p(2) * (1 - p(2)) \\ &= 0.5 * (1 - 0.5) + 0.5 * (1 - 0.5) \\ &= \boxed{0.5} \end{aligned}$$

which matches what we calculated!

Example 2: A Perfect Split

Let’s go back to the perfect split we had. What are the Gini Impurities of the two branches after the split?



Left Branch has only blues, so its Gini Impurity is

$$G_{left} = 1 * (1 - 1) + 0 * (1 - 0) = \boxed{0}$$

Right Branch has only greens, so its Gini Impurity is

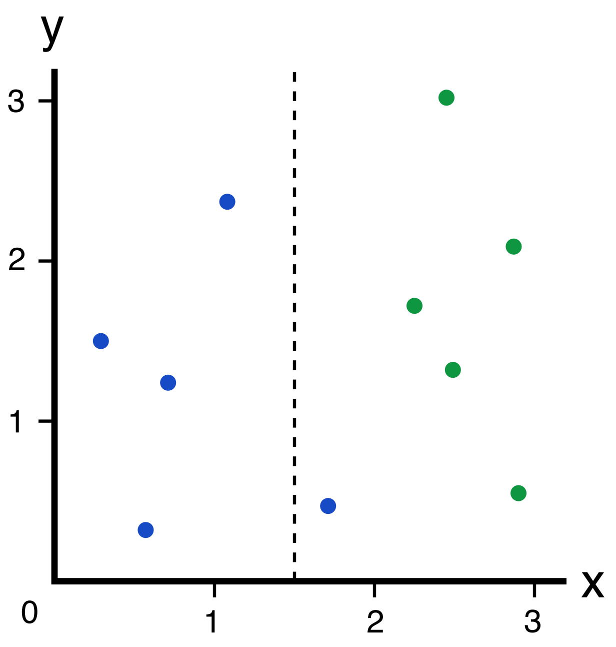
$$G_{right} = 0 * (1 - 0) + 1 * (1 - 1) = \boxed{0}$$

Both branches have 0 impurity! The perfect split turned a dataset with 0.5 impurity into 2 branches with 0 impurity.

A Gini Impurity of 0 is the lowest and best possible impurity. It can only be achieved when everything is the same class (e.g. only blues or only greens).

Example 3: An Imperfect Split

Finally, let's return to our imperfect split.



Left Branch has only blues, so we know that $G_{left} = \boxed{0}$.

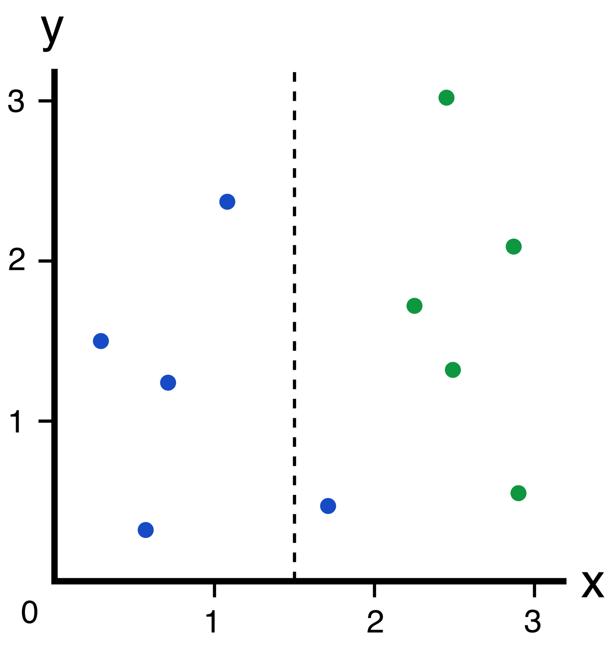
Right Branch has 1 blue and 5 greens, so

$$\begin{aligned} G_{right} &= \frac{1}{6} * (1 - \frac{1}{6}) + \frac{5}{6} * (1 - \frac{5}{6}) \\ &= \frac{5}{18} \\ &= \boxed{0.278} \end{aligned}$$

Picking The Best Split

It's finally time to answer the question we posed earlier: **how can we quantitatively evaluate the quality of a split?**

Here's the imperfect split yet again:



We've already calculated the Gini Impurities for:

- Before the split (the entire dataset): 0.5
- Left Branch: 0
- Right Branch: 0.278

We'll determine the quality of the split by **weighting the impurity of each branch by how many elements it has**. Since Left Branch has 4 elements and Right Branch has 6, we get:

$$(0.4 * 0) + (0.6 * 0.278) = 0.167$$

Thus, the amount of impurity we've "removed" with this split is

$$0.5 - 0.167 = \boxed{0.333}$$

I'll call this value the Gini Gain. This is what's used to pick the best split in a decision tree! **Higher Gini Gain = Better Split**. For example, it's easy to verify that the Gini Gain of the perfect split on our dataset is $0.5 > 0.333$.

Recap

Gini Impurity is the probability of *incorrectly* classifying a randomly chosen element in the dataset if it were randomly labeled *according to the class distribution* in the dataset. It's calculated as

$$G = \sum_{i=1}^{C} p(i) * (1 - p(i))$$

where C is the number of classes and $p(i)$ is the probability of randomly picking an element of class i.

When training a decision tree, the best split is chosen by **maximizing the Gini Gain**, which is calculated by subtracting the weighted impurities of the branches from the original impurity.

Want to learn more? Check out my explanation of Information Gain, a similar metric to Gini Gain, or my guide Random Forests for Complete Beginners.

This blog is open-source on Github.

Tags: Machine Learning | For Beginners | Decision Trees | Random Forests

YOU MIGHT ALSO LIKE

Machine Learning for Beginners: An Introduction to Neural Networks

January 13, 2026

A simple explanation of how they work and how to implement one from scratch in Python.

Random Forests for Complete Beginners

January 13, 2026

The definitive guide to Random Forests and Decision Trees.

Victor Zhou @victorczhou

Software Engineer. I blog about web development, machine learning, and more topics.

SHARE THIS POST

Facebook Twitter LinkedIn Reddit