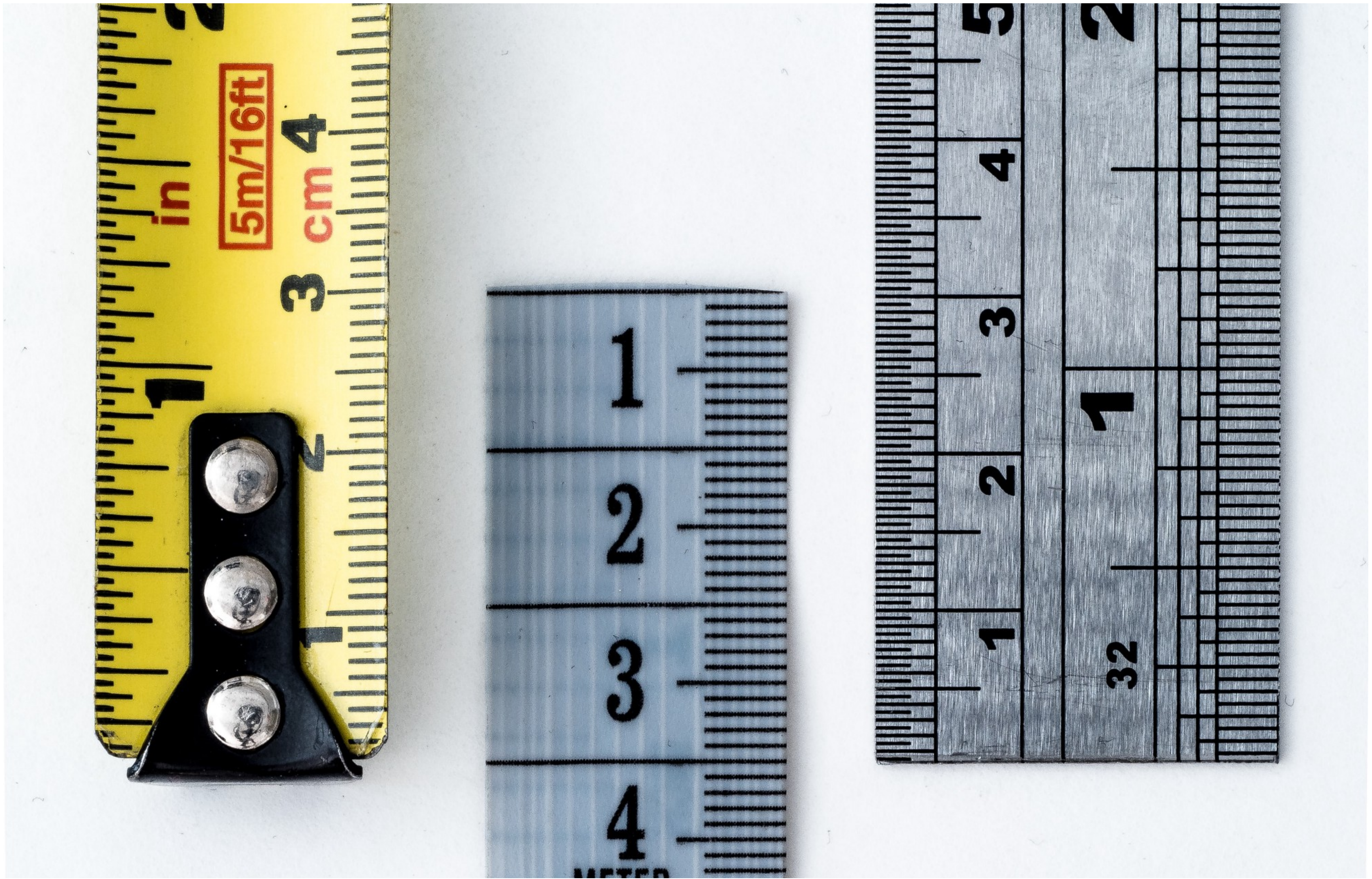
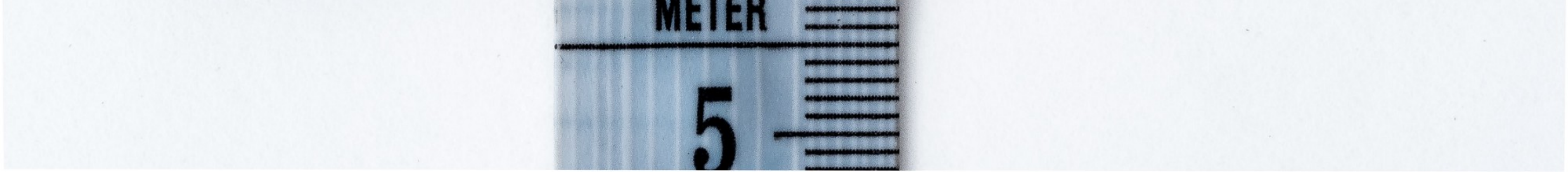
Gini Index vs Information Entropy

Everything you need to know about impurity measurements and information gain

Andrew Hershy Follow
Jul 10, 2019 · 7 min read ★





Source

Introduction:

I recently published a project analyzing the performance between logistic regression and random forest for a multivariate sample. Working on the random forest component, I wanted expand on measures of impurity/information-gain, particularly Gini Index and Entropy. It's not as well known as some other topics in machine learning, but I think it adds some valuable perspective to those wanting to understand the foundational mechanics of how decision trees work.

. . .

Table of Contents

. . .

Background:

Decision trees recursively split features with regard to their target variable's "purity". The entire algorithm is designed to optimize each split on maximizing purity… What is purity? Purity can be thought of as how homogenized the groupings are. You will see in some example below what I mean:

> *If we have 4 red gumballs and 0 blue gumballs, that group of 4 is 100% pure, based on color as the target.*
>
> *If we have 2 red and 2 blue, that group is 100% impure.*
>
> *If we have 3 red and 1 blue, that group is either 75% or 81% pure, if we use Gini or Entropy respectively.*

Why does this matter? Depending on which impurity measurement is used, tree classification results can vary. This can make small (or sometimes large!) impact on your model!

. . .

Gini Index Intuition:

$$Gini = 1 - \sum_j p_j^2$$

Let's start with Gini Index, as it's a bit easier to understand. According to Wikipedia, the goal is to *"measure how often a randomly chosen element from the set would be incorrectly labeled"[1].*

Top highlight

To visualize this, let's go back to the gumball examples. If we decided to arbitrarily label all 4 gumballs as red, how often would one of the gumballs be incorrectly labeled?

4 red and 0 blue:

$$\text{Gini Index} = 1\text{-}(\text{probability\_red}^2+\text{probability\_blue}^2) = 1\text{-}(1^2+0^2) = 0$$

The impurity measurement is 0 because we would never incorrectly label any of the 4 red gumballs here. If we arbitrarily chose to label all the balls 'blue', then our index would still be 0, because we would always incorrectly label the gumballs.

> *The gini score is always the same no matter what arbitrary class you take the probabilities of because they always add to 0 in the formula above.*

A gini score of 0 is the most pure score possible.

2 red and 2 blue:

$$\text{Gini Index} = 1\text{-}(\text{probability\_red}^2+\text{probability\_blue}^2) = 1\text{-}(0.5^2+0.5^2) = 0.5$$

The impurity measurement is 0.5 because we would incorrectly label gumballs wrong about half the time. Because this index is used in binary target variables (0,1), a gini index of 0.5 is the least pure score possible. Half is one type and half is the other. **Dividing gini scores by 0.5 can help intuitively understand what the score represents. 0.5/0.5 = 1, meaning the grouping is as impure as possible (in a group with just 2 outcomes).**

3 red and 1 blue:

$$\text{Gini Index} = 1\text{-}(\text{probability\_red}^2+\text{probability\_blue}^2)=1\text{-}(0.75^2+0.25^2)=0.375$$

The impurity measurement here is 0.375. If we divide this by 0.5 for more intuitive understanding we will get 0.75, which is the probability of incorrectly/correctly labeling.

. . .

Entropy Intuition:

$$Entropy = -\sum_j p_j \log_2 p_j$$

Entropy is more computationally heavy due to the log in the equation. Like gini, The basic idea is to gauge the disorder of a grouping by the target variable. Instead of utilizing simple probabilities, this method takes the log base2 of the probabilities (you can use any log base, however, as long as you're consistent). The entropy equation uses logarithms because of many advantageous properties. The Main advantage is the additive property it provides. These MIT lecture notes will help grasp the concept more clearly (pg8) [2].

Let's visualize how entropy works with the same gumball scenarios:

4 red and 0 blue:

*Entropy = [(probability_red) * log2(probability_red)]—[(probability_blue) * log2(probability of blue)] = [(4/4) * log2(4/4)] -[(0/4) * log2(0/4)] = 0*

Unsurprisingly, the impurity measurement is 0 for entropy as well. This is the max purity score using information entropy.

2 red and 2 blue:

*Entropy = [(probability_red) * log2(probability_red)]—[(probability_blue) * log2(probability of blue)] = [(2/4) * log2(2/4)] -[(2/4) * log2(2/4)] = 1*

The impurity measurement is 1 here, as it's the maximum impurity obtainable.

3 red and 1 blue:

*Entropy = [(probability_red) * log2(probability_red)]—[(probability_blue) * log2(probability of blue)] = [(3/4) * log2(3/4)] -[(1/4) * log2(1/4)] = 0.811*

The purity/impurity measurement is 0.811 here, a bit worse than the gini score.

. . .

Visualization

Let's visualize both the Gini and Entropy curves with some code in python:

Gini:

Below we are making a function to automate gini calculations.

```
#Gini Function
#a and b are the quantities of each class
def gini(a,b):
    a1 = (a/(a+b))**2
    b1 = (b/(a+b))**2
    return 1 - (a1 + b1)
```

Keeping consistent with our gumball theme, let's make a loop that calculates the gini score of any conceivable combination of red and blue gumball floats, adding to 4. We will run 10,000 iterations of the gini function above so we can graph the gini curve later.

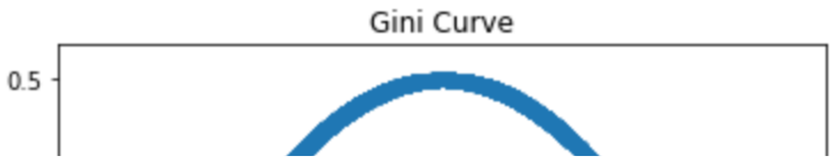
```
#Blank lists
gini_list = []
blue_list = []
red_list = []
blue_prob_list = []

#Looping Gini function on random blue and red float amounts
for x in range (10000):
 blue = random.uniform(0, 4)
 red = abs(4-blue)
 a = gini(red,blue)
 b = blue/(blue+red)
 gini_list.append(a)
 blue_list.append(blue)
 red_list.append(red)
 blue_prob_list.append(b)


#Dataframe of amount of blue, red, Probability of blue, and gini
score
df = pd.DataFrame({"Blue": blue_list, "Red": red_list,"Gini Score":
gini_list, "Probability of Blue": blue_prob_list})
df = df[['Red', 'Blue', 'Probability of Blue', 'Gini Score']]
df
```

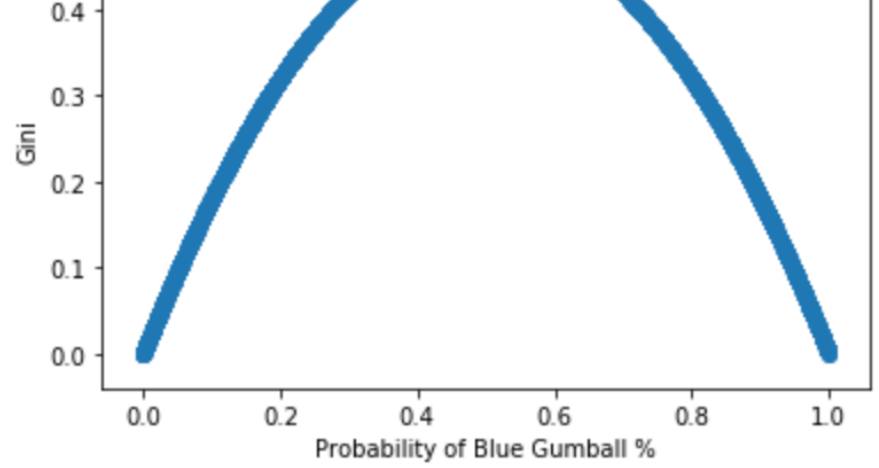
The beginning of the dataframe is below. The other 9,994 rows couldn't fit.

	Red	Blue	Probability of Blue	Gini Score
0	0.575254	3.424746	0.856186	0.246262
1	1.602532	2.397468	0.599367	0.480252
2	2.760943	1.239057	0.309764	0.427621
3	2.640753	1.359247	0.339812	0.448679
4	3.549481	0.450519	0.112630	0.199889
5	0.719211	3.280789	0.820197	0.294948

Now we will plot our curve:

```
plt.scatter(blue_prob_list,gini_list)
plt.xlabel('Probability of Blue Gumball %')
plt.ylabel('Gini')
plt.title('Gini Curve')
```

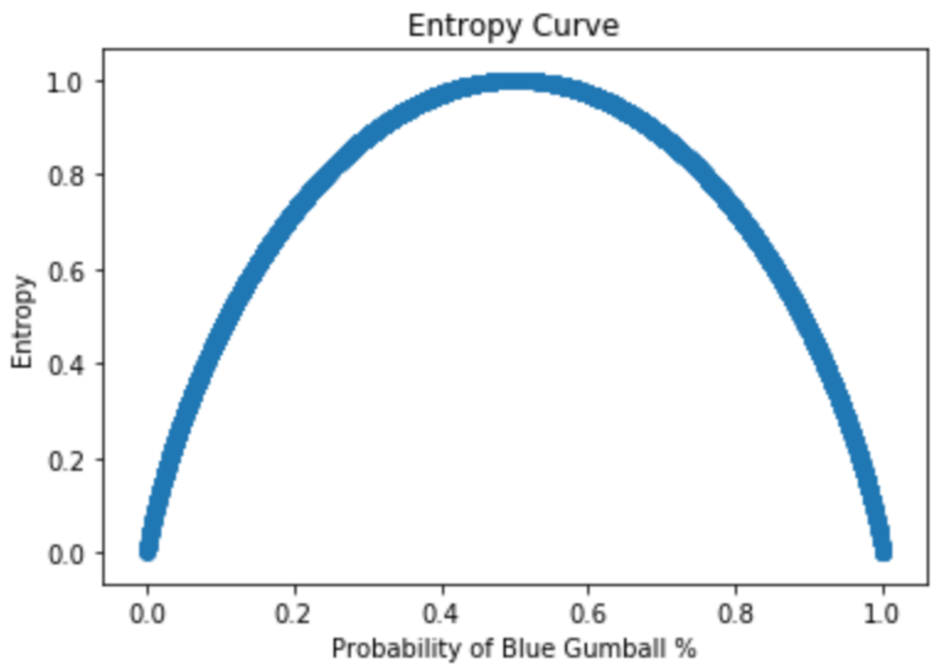




Entropy:

```
#Gini Function
#a and b are the quantities of each class. Base is the log base
input.
def entropy(base,a,b):
    try:
        var =  abs(((a)/(a+b)) * log(((a)/(a+b)),base)) -
(((b)/(a+b)) * log(((b)/(a+b)),base))
        return var
    except (ValueError):
        return 0

#Blank lists
ent_list = []
blue_list = []
red_list = []
blue_prob_list = []

#Loop with log base 2
for x in range (10000):
    blue = random.uniform(0, 4)
    red = abs(4-blue)
    a = entropy(2,red,blue)
    b = blue/(blue+red)
    ent_list.append(a)
    blue_list.append(blue)
    red_list.append(red)
    blue_prob_list.append(b)

df = pd.DataFrame({"Blue": blue_list, "Red": red_list,"Entropy":
ent_list, "Probability of Blue": blue_prob_list})
df = df[['Red', 'Blue', 'Probability of Blue', 'Entropy']]
df
```

	Red	Blue	Probability of Blue	Entropy
0	0.280530	3.719470	0.929867	0.366418
1	2.581253	1.418747	0.354687	0.938184
2	1.799219	2.200781	0.550195	0.992718
3	3.823594	0.176406	0.044101	0.260791
4	0.161097	3.838903	0.959726	0.243548
5	1.971193	2.028807	0.507202	0.999850



Information Gain

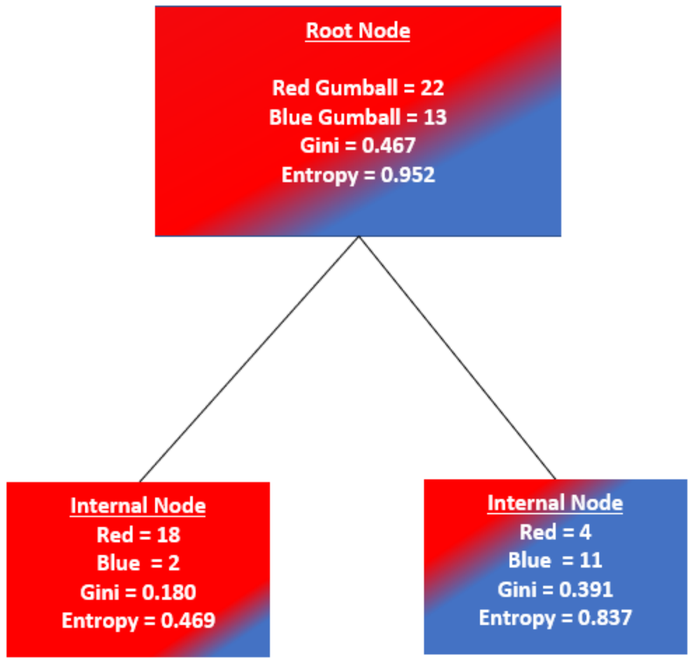
Information gain is why impurity is so important. Once we derive the impurity of the dataset, we can see how much information is gained as we go down the tree and measure the impurity of the nodes.

Information Gain (Parent, Child)

*= Entropy(parent) - [p1(c1) * entropy(c1) + p(c2) * entropy(c2) + …]*

Source [3]

In the example below, we are splitting gumball preference by a particular attribute (let’s say **size** of the gumball). It gives us the parent/child node relationship:



Gini Info Gain (from the equation above)

```
#Defining Gini info gain function:

def gini_info_gain(pa,pb,c1a,c1b,c2a,c2b):
    return (gini(pa,pb))-((((c1a+c1b)/(pa+pb))*gini(c1a,c1b)) +
(((c2a+c2b)/(pa+pb))*gini(c2a,c2b)))
```

```
#Running Function

gini_info_gain(22,13,18,2,4,11)
```

> *= 0.196 Gini Information gain*
>
> *0.196/0.467 = 41.97% Gain*

Entropy Info Gain (from the equation above)

```
#Defining Entropy info gain function:

def entropy_info_gain(base,pa,pb,c1a,c1b,c2a,c2b):
    return (entropy(base,pa,pb))-
((((c1a+c1b)/(pa+pb))*entropy(base,c1a,c1b)) +
(((c2a+c2b)/(pa+pb))*entropy(base,c2a,c2b)))

#Running Function

entropy_info_gain(2,22,13,18,2,4,11)
```

> *= 0.325 Entropy Information gain*
>
> *0.325/0.952 = 34.14% Gain*

Gini has a higher information gain measurement, for this example.

. . .

Final Takeaways:

1. Gini’s maximum impurity is 0.5 and maximum purity is 0
2. Entropy’s maximum impurity is 1 and maximum purity is 0
3. Different decision tree algorithms utilize different impurity metrics: CART uses Gini; ID3 and C4.5 use Entropy. This is worth looking into before you use decision trees /random forests in your model.

*All the code in this publication can be found on my github here

. . .

Sources:

[1] https://en.wikipedia.org/wiki/Decision_tree_learning#Gini_impurity

[2] http://web.mit.edu/6.02/www/f2011/handouts/2.pdf

[3] Provost, F., & Fawcett, T. (2013). *Data Science for Business: What you need to know about data mining and data-analytic thinking*. Köln, CA: O`Reilly.

. . .

Please subscribe if you found this helpful. If you enjoy my content, please check out a few other projects:

Is Random Forest better than Logistic Regression? (a comparison)

Excel vs SQL: A Conceptual Comparison

Predicting Cancer with Logistic Regression in Python

Optimize your Investments using Math and Python

Calculating R-squared from scratch (using python)

Word Clouds in Python: Comprehensive Example

Machine Learning Impurity Classification Decision Tree Data Science