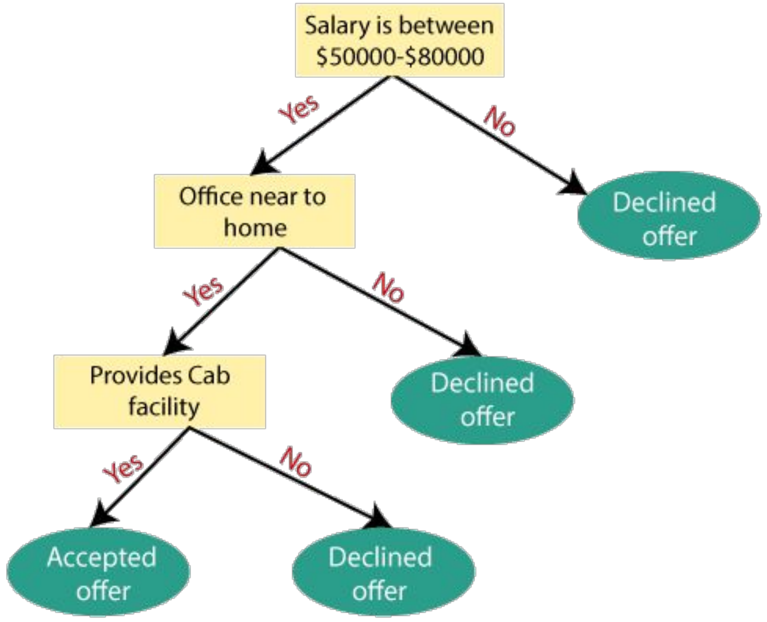
# Introduction to Decision Trees

THE KNOWLEDGE HOUSE

this model won't accept a higher salary???

## Agenda - Schedule

1. Decision Trees Intuition
2. Formal Decision Trees
3. Information & Entropy
4. Break
5. Decision Trees Lab



*How do you make decisions?*

## Agenda - Goals

- Understand the formulation of a decision tree
- Understand the different measures of information loss
- Implement decision trees in Python

## Review

## Review Question

When applying Bayes Theorem to the Naive Bayes classifier, why do we **remove** the division operation from our calculation?

$$P(H|E) = \frac{P(H)\ P(E|H)}{P(E)} \longrightarrow \hat{y} = \text{argmax}\ P(Y)\ P(X1|Y)\ P(X2|Y) \ldots P(Xn|Y)$$

A) **Impossible**. You must keep the division

B) Removing the division is necessary as it is too computationally expensive

C) Removing the division does **not** impact the **relationship of inequalities amongst confidence of classes**

## Review Question

As we increase our “alpha” in laplace smoothing, what occurs to all of our probabilities in the multinomial naive bayes classifier.

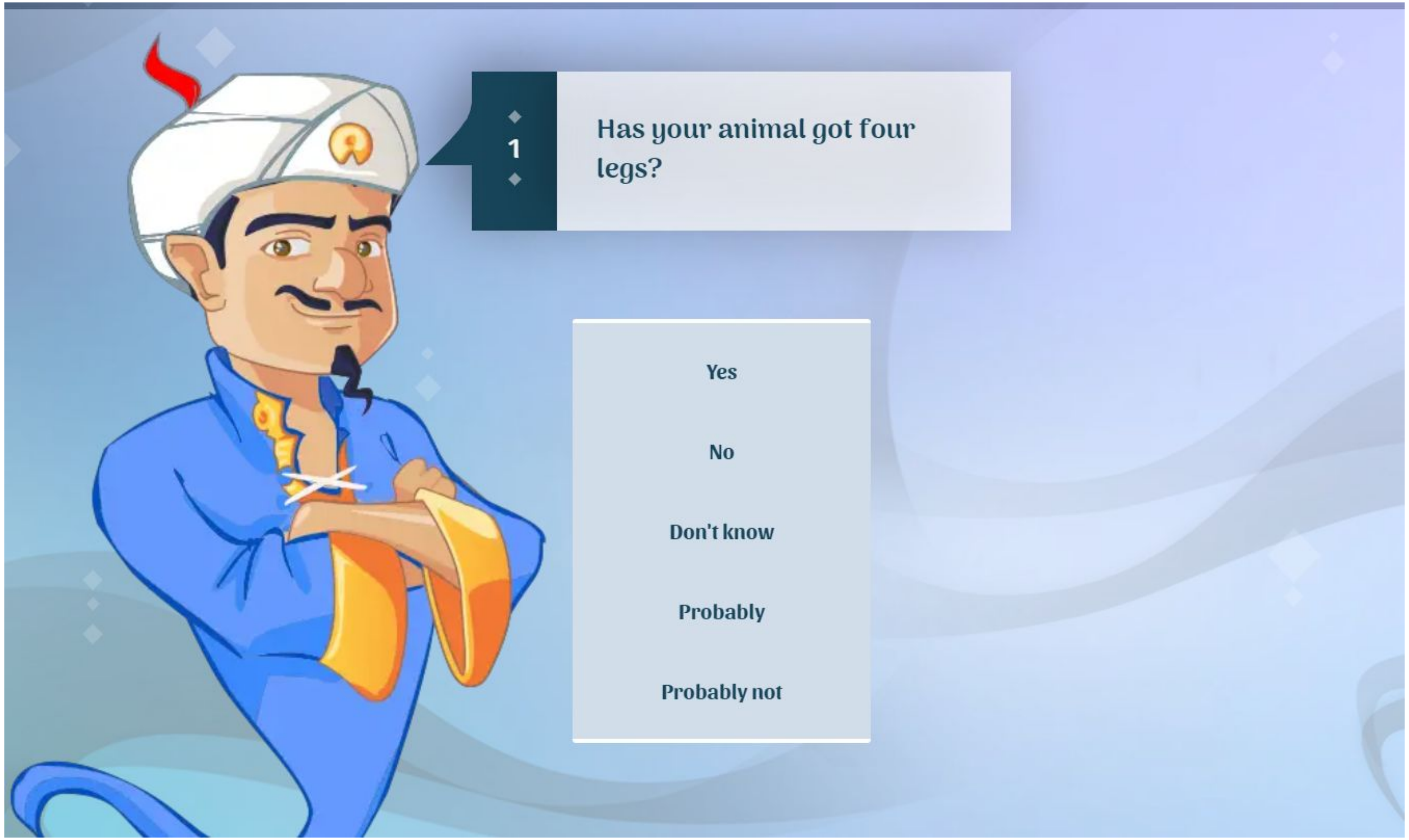
A) Probabilities shrink to 0

B) Probabilities increase to 1

C) Probabilities become uniformly distributed

D) Probabilities become normally distributed

## Decision Trees



Before we go any further, let's check out akinator to see how a decision tree works "in theory." Keep in mind that this program is not actually a decision tree.

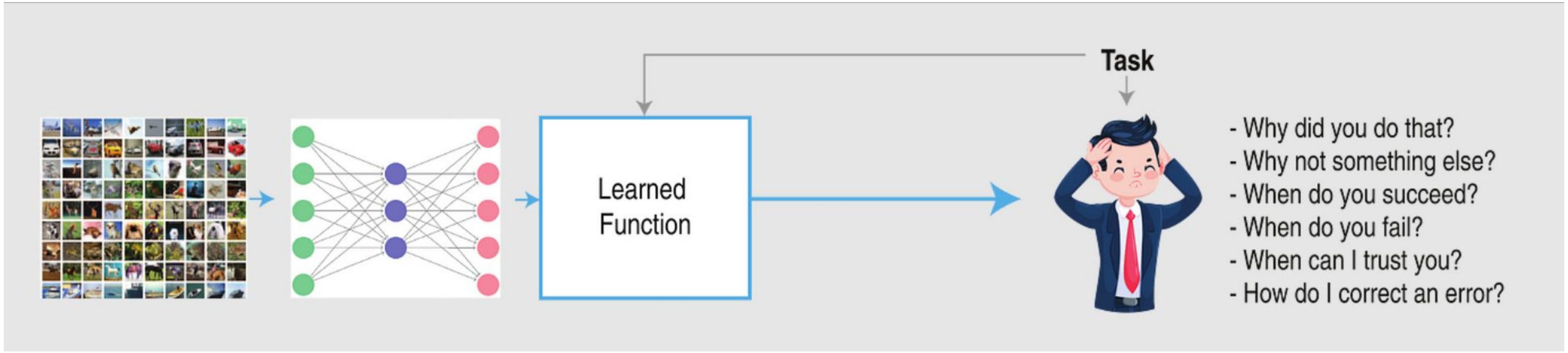
## Decision Tree

Something to consider when training a machine learning algorithm is:

*Why did my algorithm decide to do xyz?*

The last thing we want to do, is to treat our software/algorithm as an **unknowable system.**

This might not apply to the current algorithms that we use, but as we increase **the amount of maths that occurs in the back-end**, the model becomes less understandable.

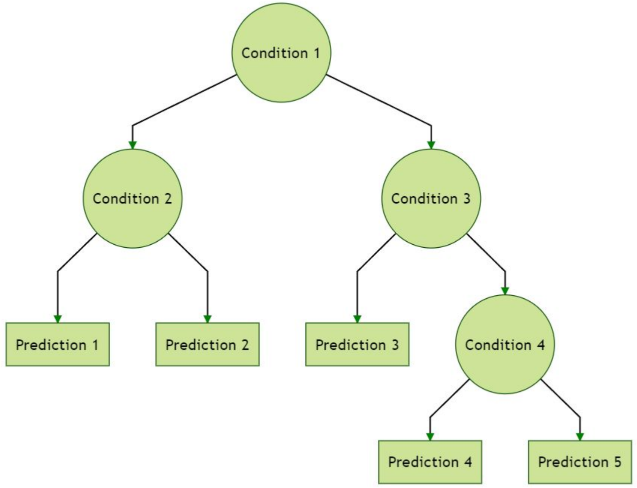


This will be especially true when we get to neural networks with “hidden layers.”
https://medium.com/@prsangeetha/exploring-the-power-of-explainable-ai-xai-in-the-era-of-artificial-intelligence-e8d87e3391eb

## Decision Trees (DT’s)

While decision trees are not necessarily XAI, they do offer a perspective that could be easily interpreted by a human.

DT is a **non-parametric supervised learning multiclass classifier** (could be used for regression as well). Much like kNN and Naive Bayes, we do not fit coefficients, but rather calculate classes based on some iterative rules-based approach.



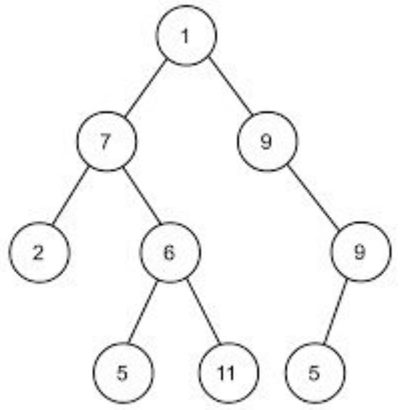
## An Aside - Trees

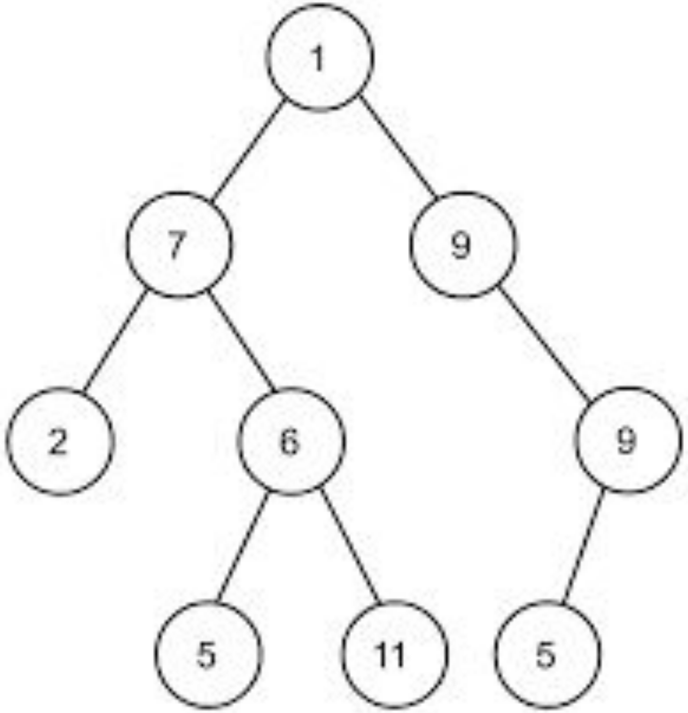
Let's recall what a **tree** is in formal computer science.

A tree, much like a stack or queue is a way to **organize data.**

We group related pieces of information together using the following terms:

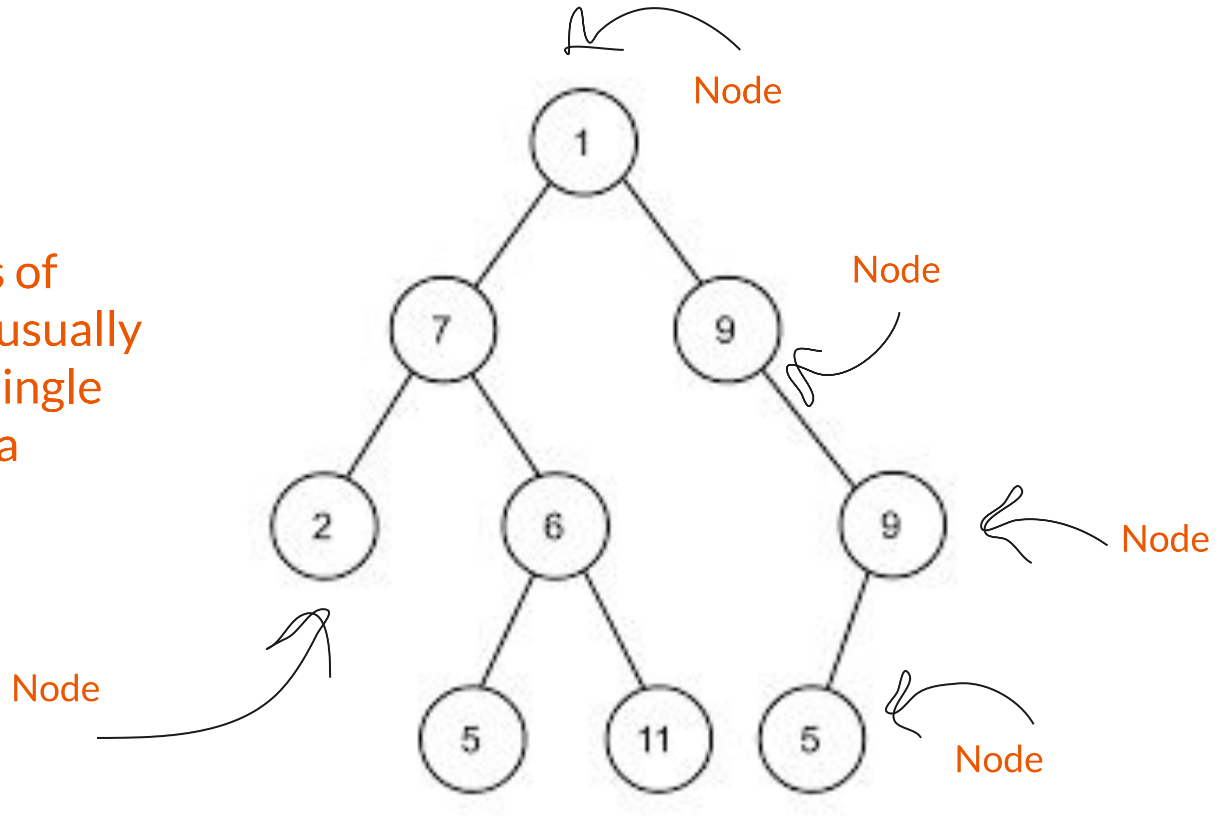
- *Nodes*
- *Edges*
- *Leaves*





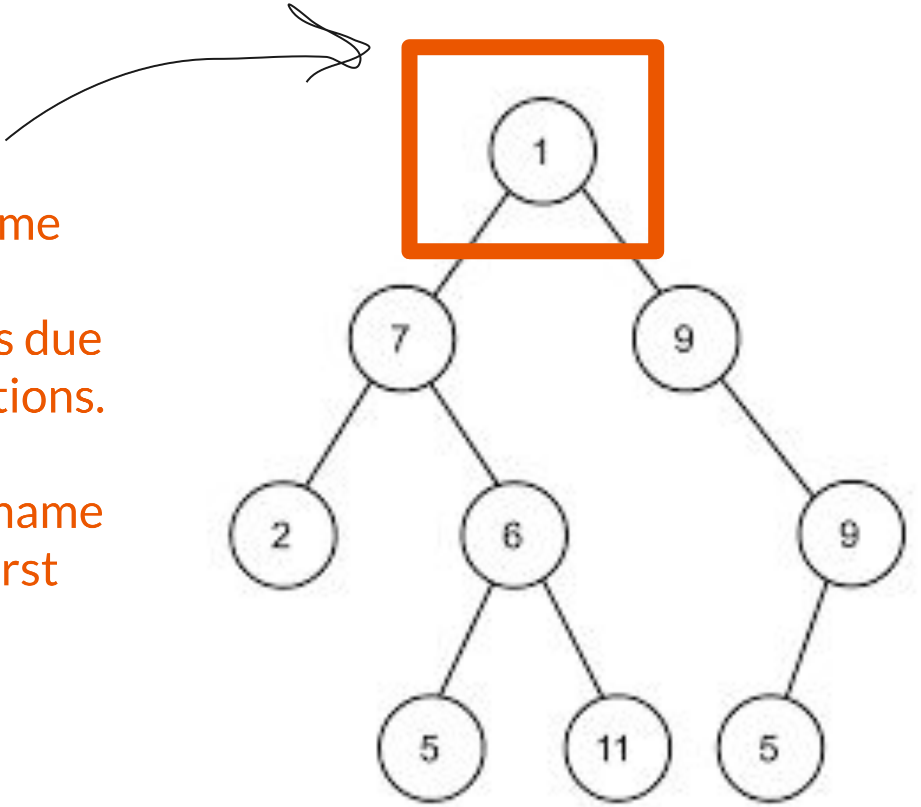
Let’s identify these discrete components before going through the decision tree algorithm.

**Nodes**:
Basic units of trees that usually contain a single unit of data

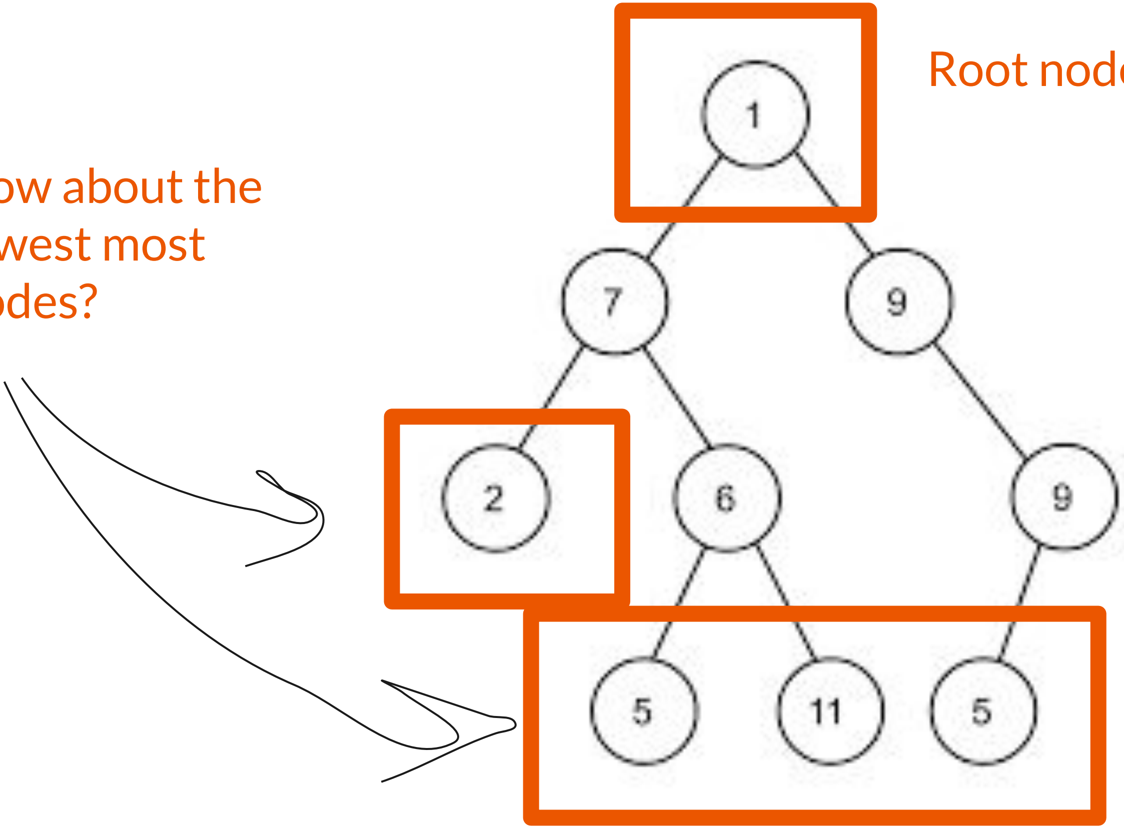


However, some nodes attain special labels due to their positions.

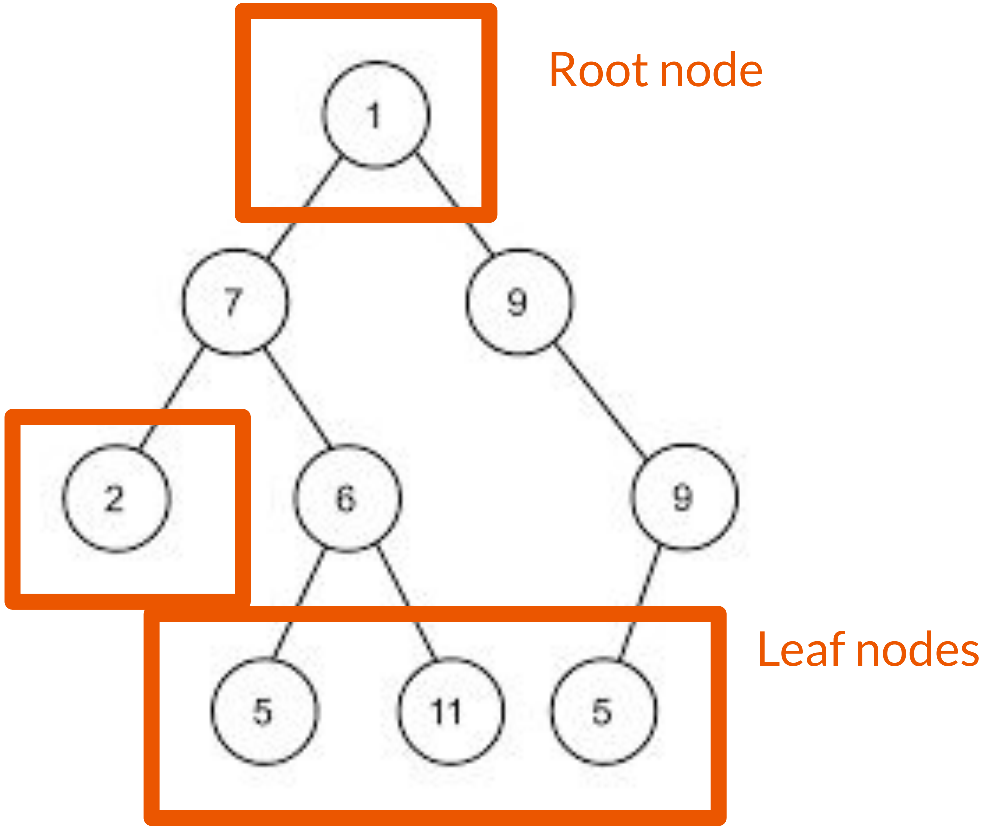
What is the name of the very first node?



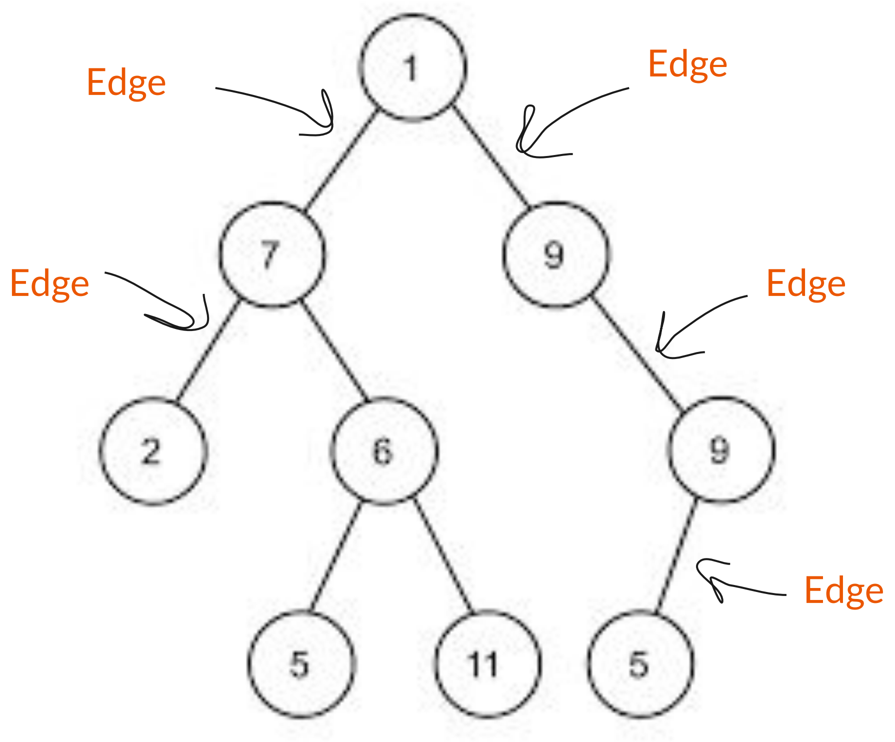
How about the lowest most nodes?



Hence the data structure name "tree"



**Edges**:
Connections between nodes that provide **direct paths**.



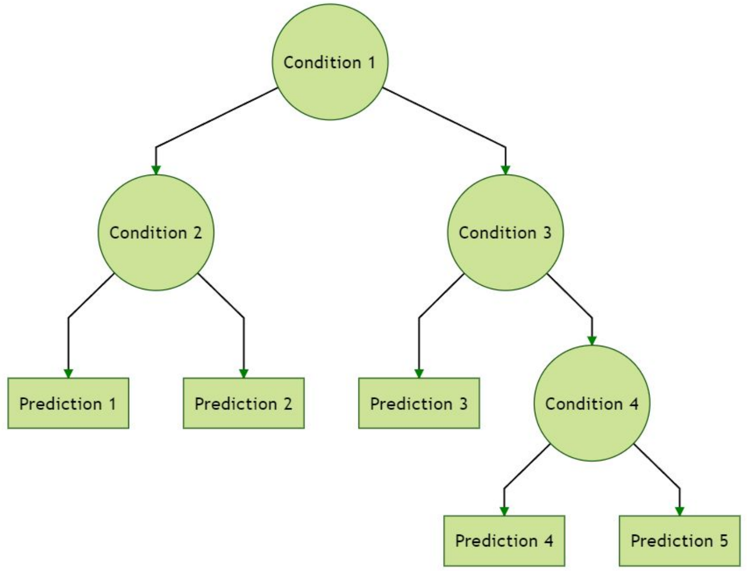
To succeed in a software engineering interview, you will have to be able to answer programmatic questions regarding trees. This will **not** be our focus for the time being.

## Decision Trees (DT's)

Instead of considering discrete units of data, we instead construct a decision tree:

- Conditionals (*True or False statements*) at **root node** and **internal nodes**
- **Edges** are known as **branches**
- Discrete predictions in **leaf nodes**

**Let's take a look at a pseudo-march-madness dataset to see how this is created.**



We're making an assumption that ONLY one basketball teams stats matter during a game, which is a terrible assumption to make.

However this is for simplicity's sake.

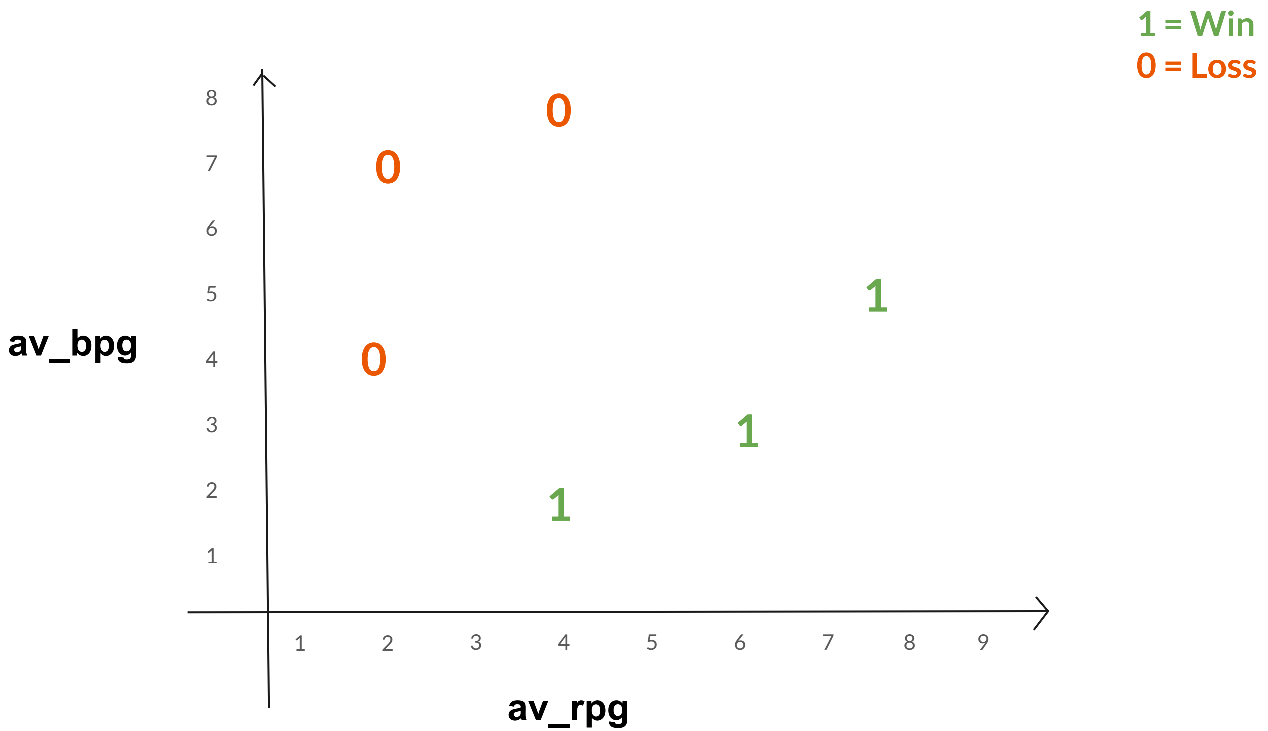
| av_rpg | av_bpg | won |
|---|---|---|
| 8 | 5 | 1 |
| 4 | 2 | 1 |
| 4 | 8 | 0 |
| 2 | 7 | 0 |
| 6 | 3 | 1 |
| 2 | 4 | 0 |

*rpg: "rebounds per game" for that specific season

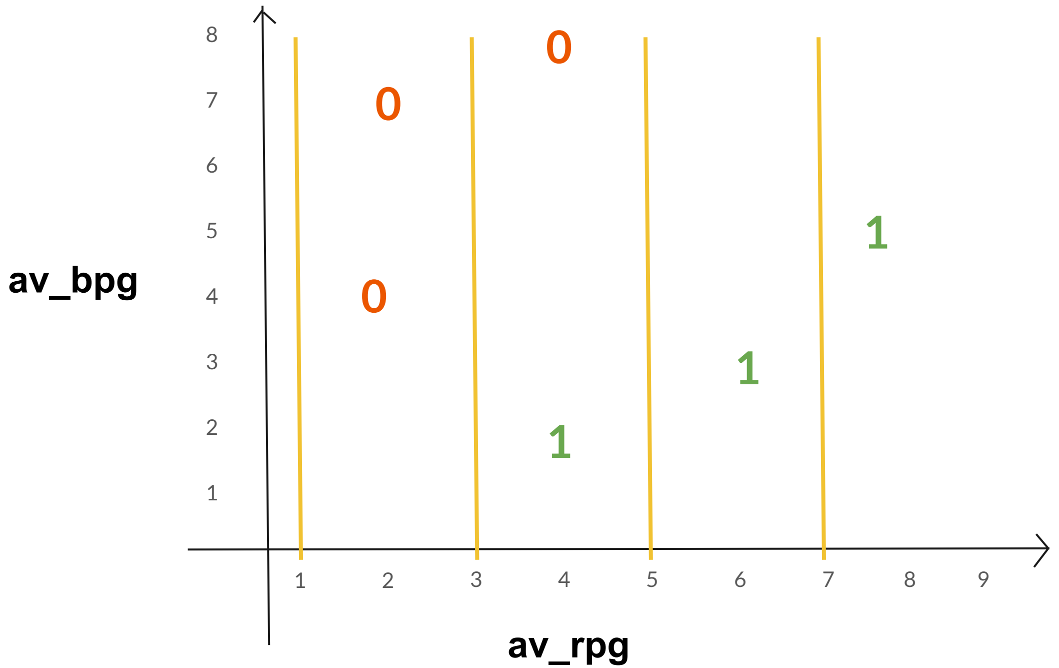
* bpg: "blocks per game" for that specific season

Each sample (row) describes the stats of one team for an arbitrary upcoming basketball game in 2024

For example, in the first sample might be "**Gonzaga**" and the second sample might be "**UvA.**"

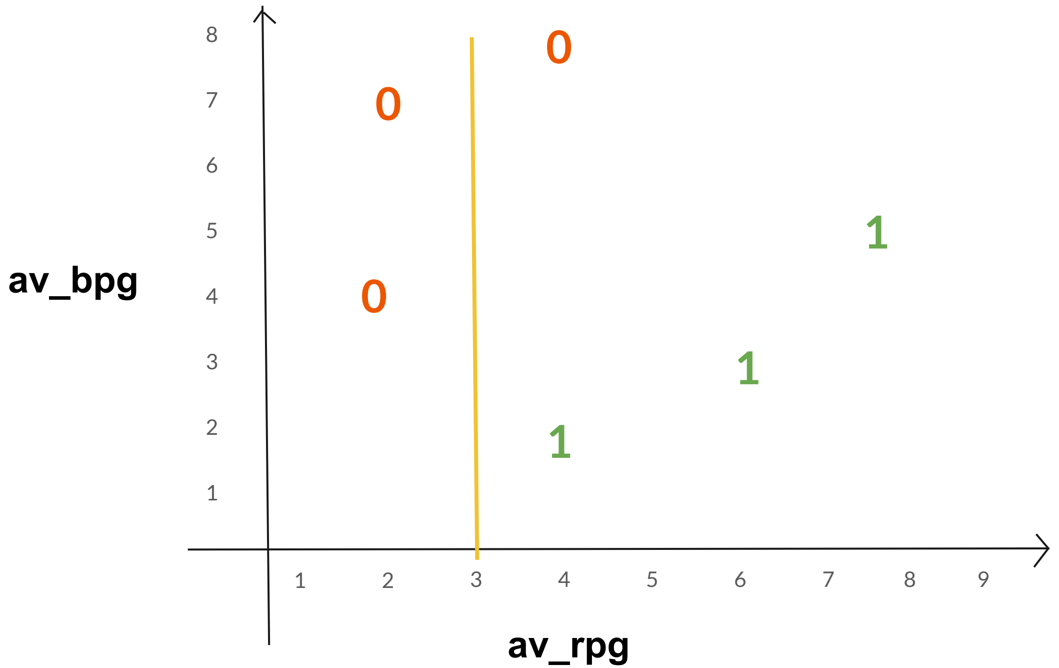


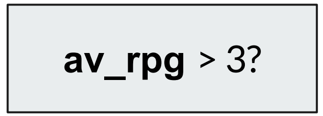
Even before we get to creating a decision tree on this data, what do you notice about this dataset? Which other models can we use to represent this decision boundary?



To begin our exploration of decision trees, let's consider how we can make predictions on this dataset by viewing the regions that our classes of data-points exist in.

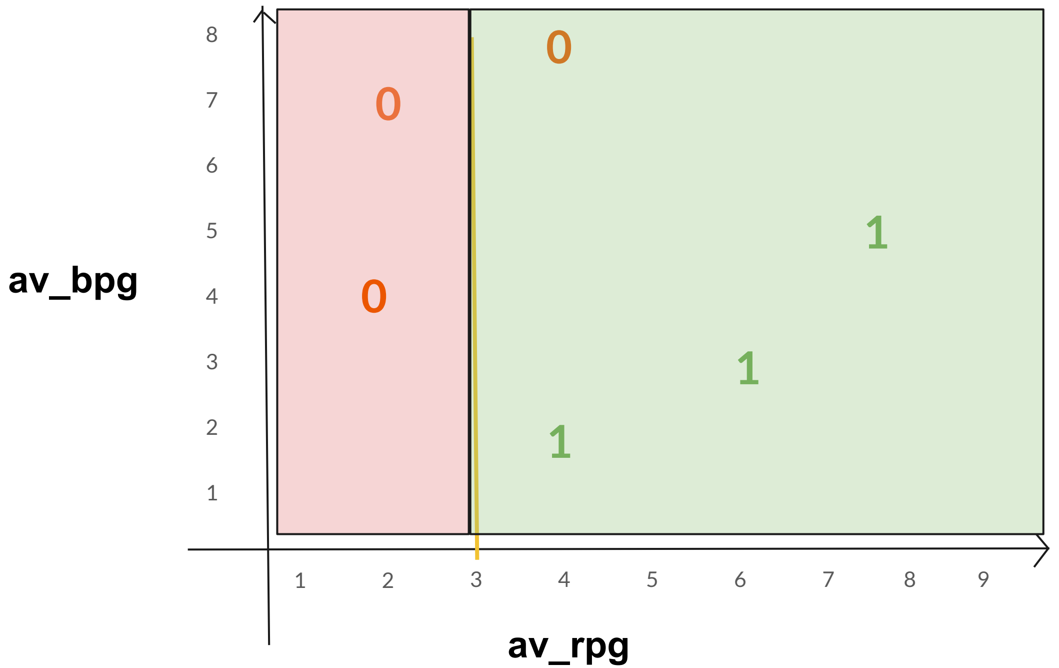
First off, which **av_rpg**-value are all of our 1's **greater than?** For now, **ignore** the misclassified 0's, we only want to capture as many 1's as possible.

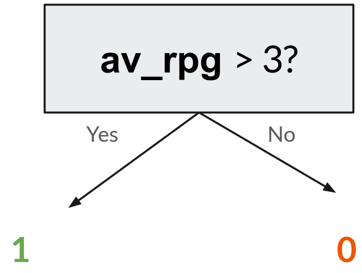




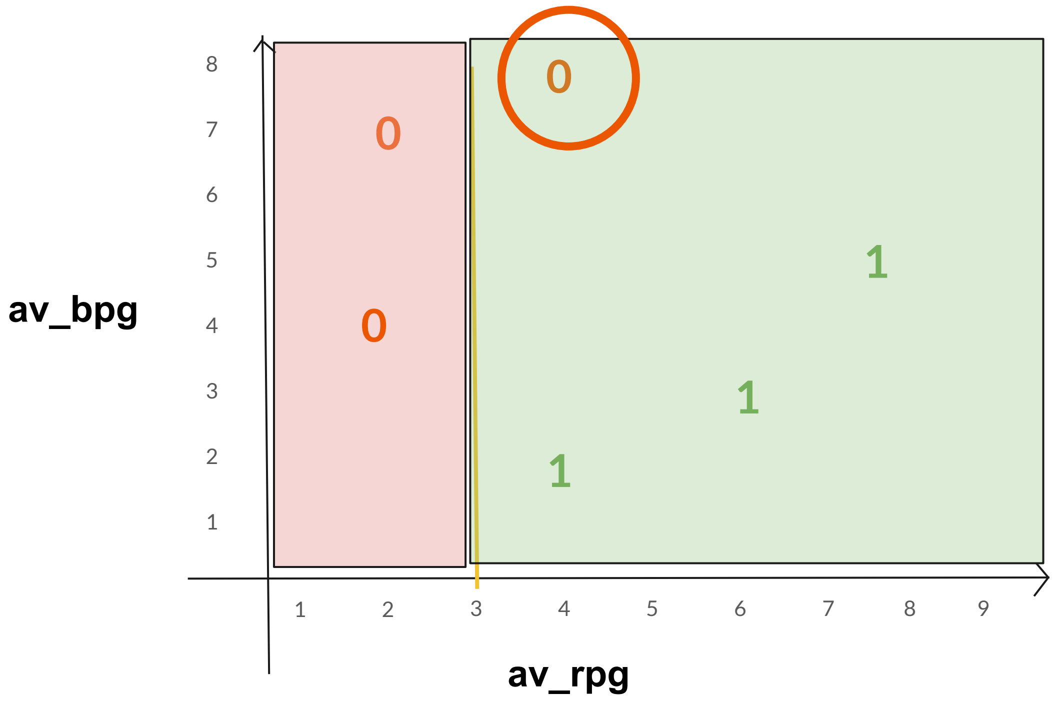
Seems like all 1's are located **above (>) av_rpg=3.**

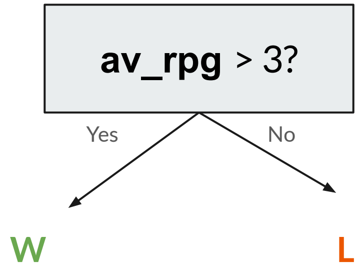
Let's make note of that using a "conditional node."



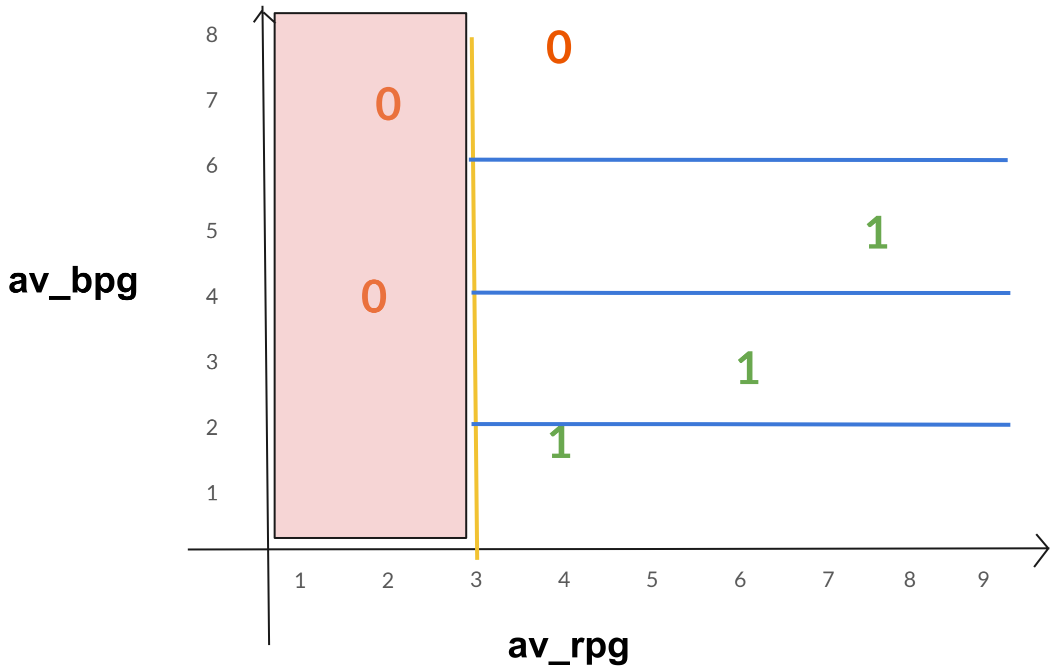


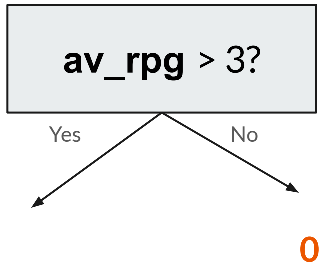
Can we confidently state that **all** samples **team1_rpg > 3** should be classified as **Wins**, or are there present misclassifications?





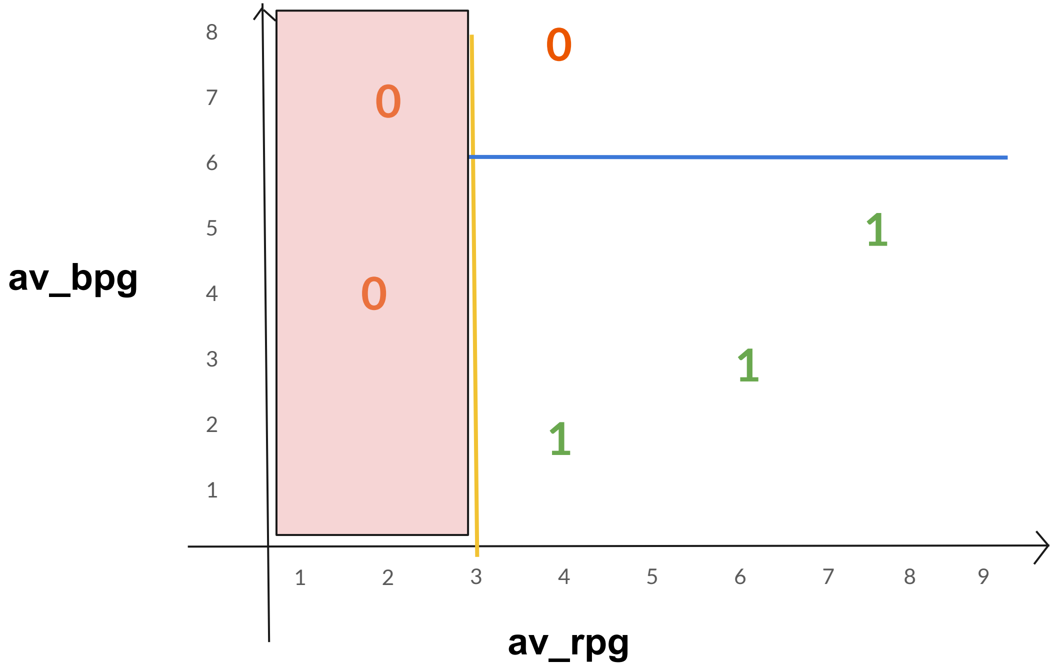
While all samples < 3 are accurately predicted as 0, is there another split we can make for the region defined as **av_rpg > 3** to further refine our predictions?

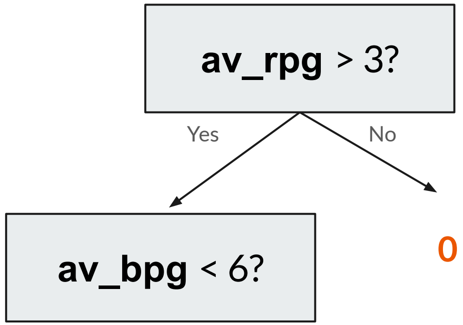




*Note: once again these visualizations are rough approximations. We will see more precise language later

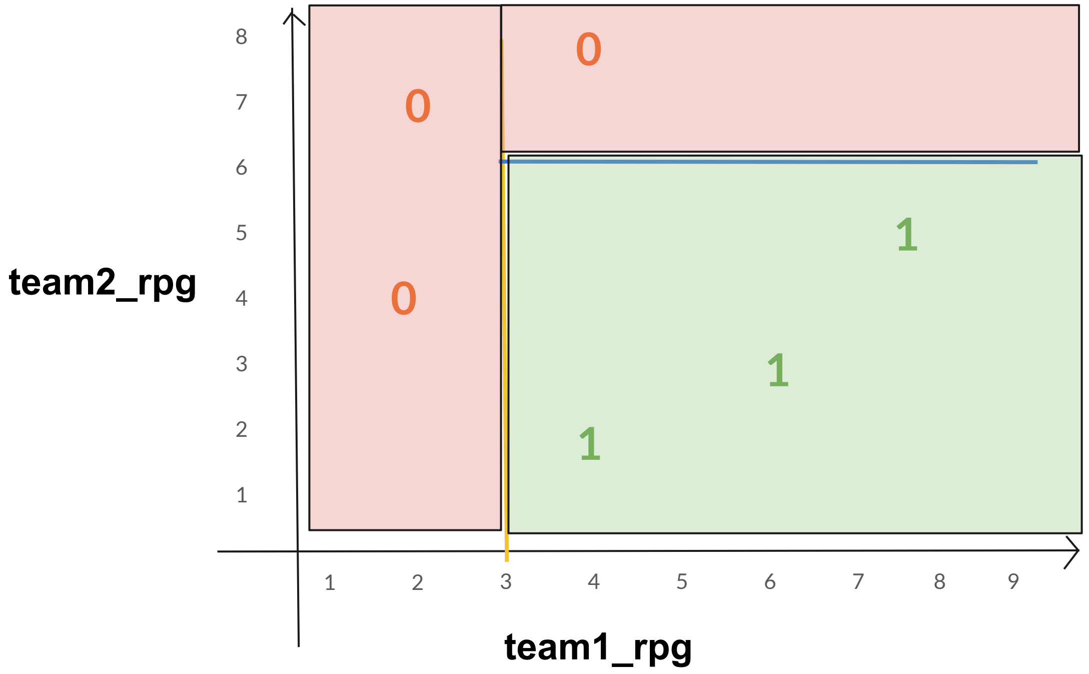
Let’s consider another split along our **av_bpg** axis Which **av_bpg** -value are all of our 1’s **less than**?

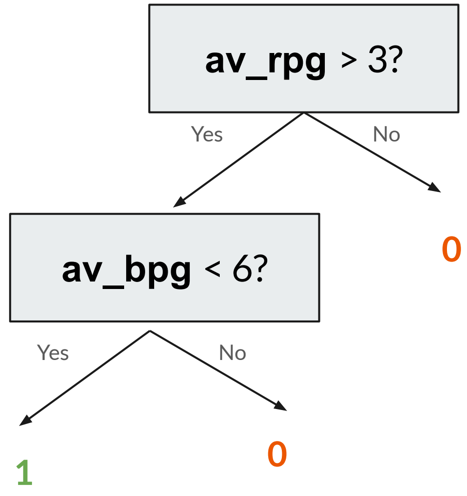




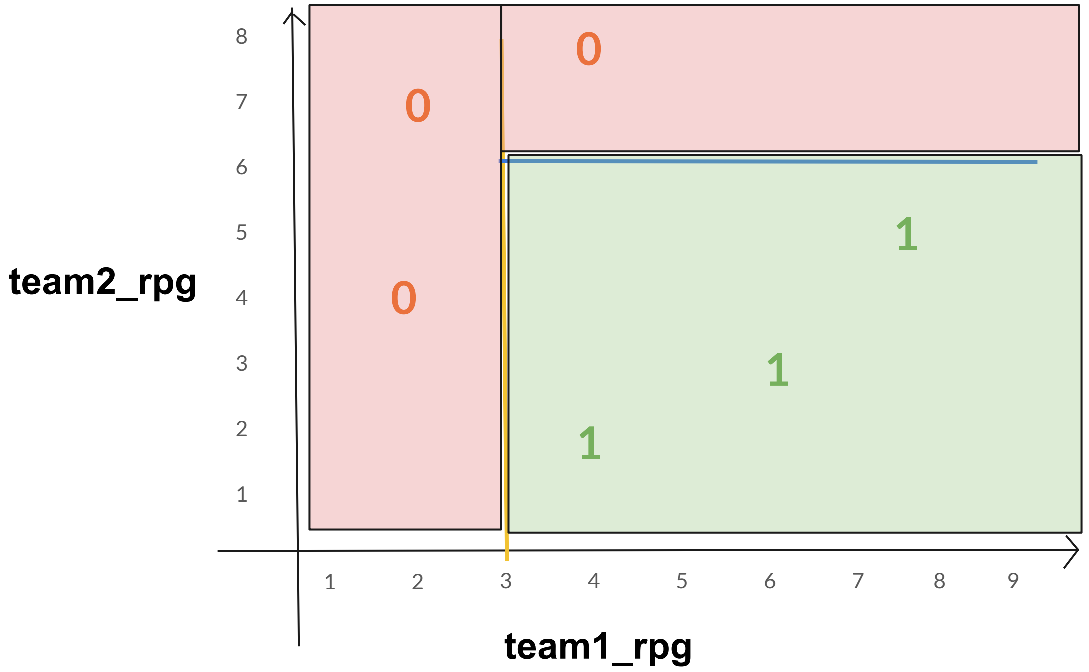
Seems like all 1's are located **below (<) av_bpg =6.**

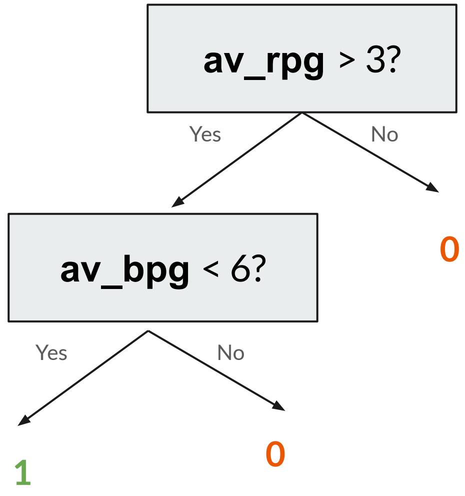
Let's make note of that using another "conditional node."





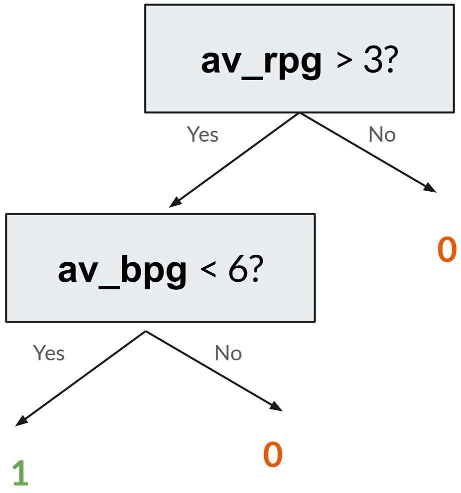
Now that we've considered this split along the y-axis, have we sufficiently **learned** the regions which best model our dataset?





YES! You’ve just successfully built a Decision Tree! Notice how this structure is sufficient to capture the model of the data, while also being **simple enough to explain to a human.**

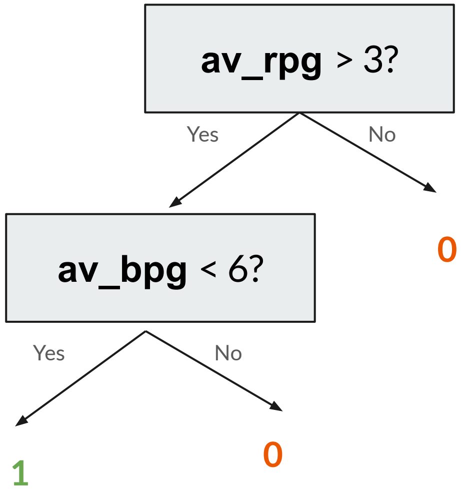
| av_rpg | av_bpg | pred | won |
|---|---|---|---|
| 5 | 8 | | ??? |
| 2 | 1 | | ??? |



Let’s say we’ve formed this DT and now want to utilize it to figure out which team to **bet on in two upcoming basketball games** (test samples).

Let’s take a look at the first sample.

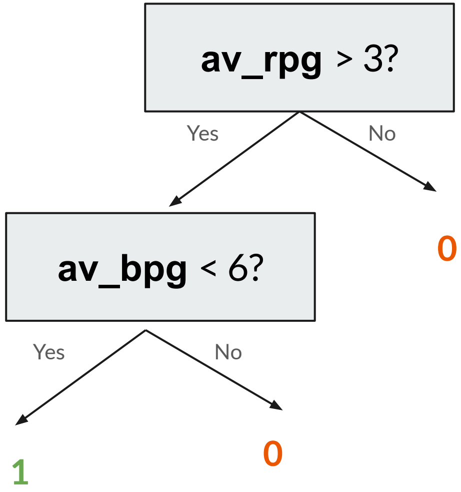
| av_rpg | av_bpg | pred | won |
|---|---|---|---|
| 5 | 8 |  | ??? |
| 2 | 1 |  | ??? |



Let’s follow along with our decision tree **until we reach a predicted class.** **The first node tells us to check the value of av_rpg.**

Which branch do we take according to the **av_rpg >3** node?

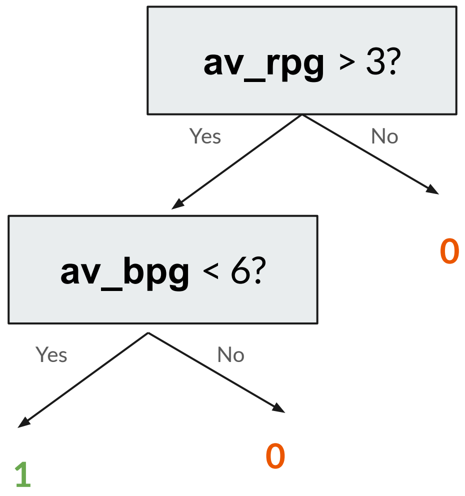
| av_rpg | av_bpg | pred | won |
|---|---|---|---|
| 5 | 8 | | ??? |
| 2 | 1 | | ??? |



As **av_rpg > 3** evaluates to true, we take the "Yes" branch and move towards the second node.

**Now we check the av_bpg value of this new test sample**. Which branch do we take according to the second **av_bpg** < 6 node?

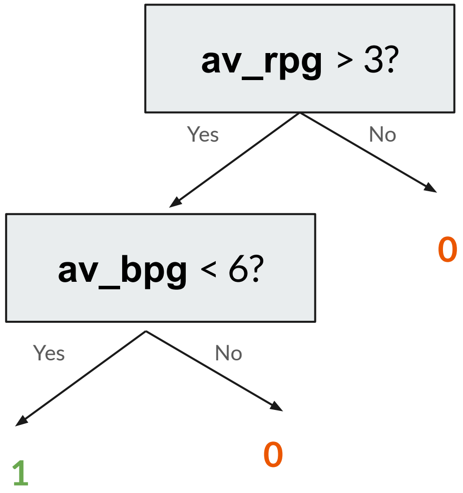
| av_rpg | av_bpg | pred | won |
|---|---|---|---|
| 5 | 8 |  | ??? |
| 2 | 1 |  | ??? |



As **av_bpg < 6** evaluates to false (or No), we must classify this as a 0!

Should you bet on team1 in this instance?

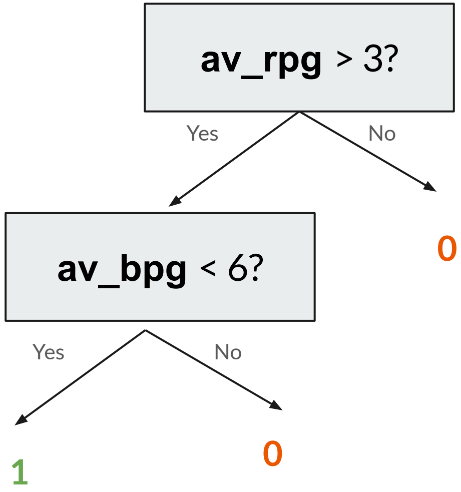
| av_rpg | av_bpg | pred | won |
|---|---|---|---|
| 5 | 8 | **0** | ??? |
| 2 | 1 | | ??? |



According to this decision tree, **no.** (Once again, this is a limited example, as the probability of a loss is also dependent on the stats of the other team, which we are ignoring)

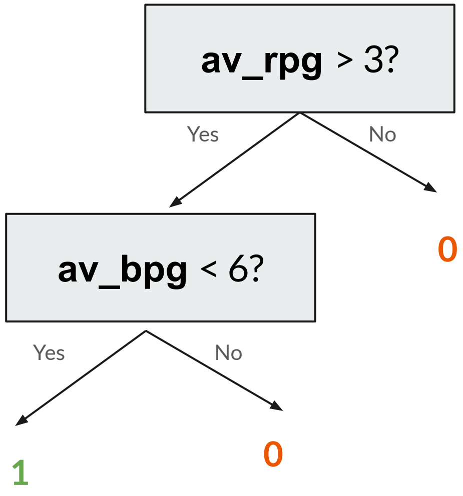
Let's take a look at the next sample.

| av_rpg | av_bpg | pred | won |
|---|---|---|---|
| 5 | 8 | **0** | ??? |
| 2 | 1 | | ??? |



Which branch do we take according to the **av_rpg >3** node for the second sample?

| av_rpg | av_bpg | pred | won |
|---|---|---|---|
| 5 | 8 | **0** | ??? |
| 2 | 1 | **0** | ??? |



Seems like we classify this as a "**0**" as well.

Let's refine this algorithm by taking a look at the metrics that a decision tree uses.

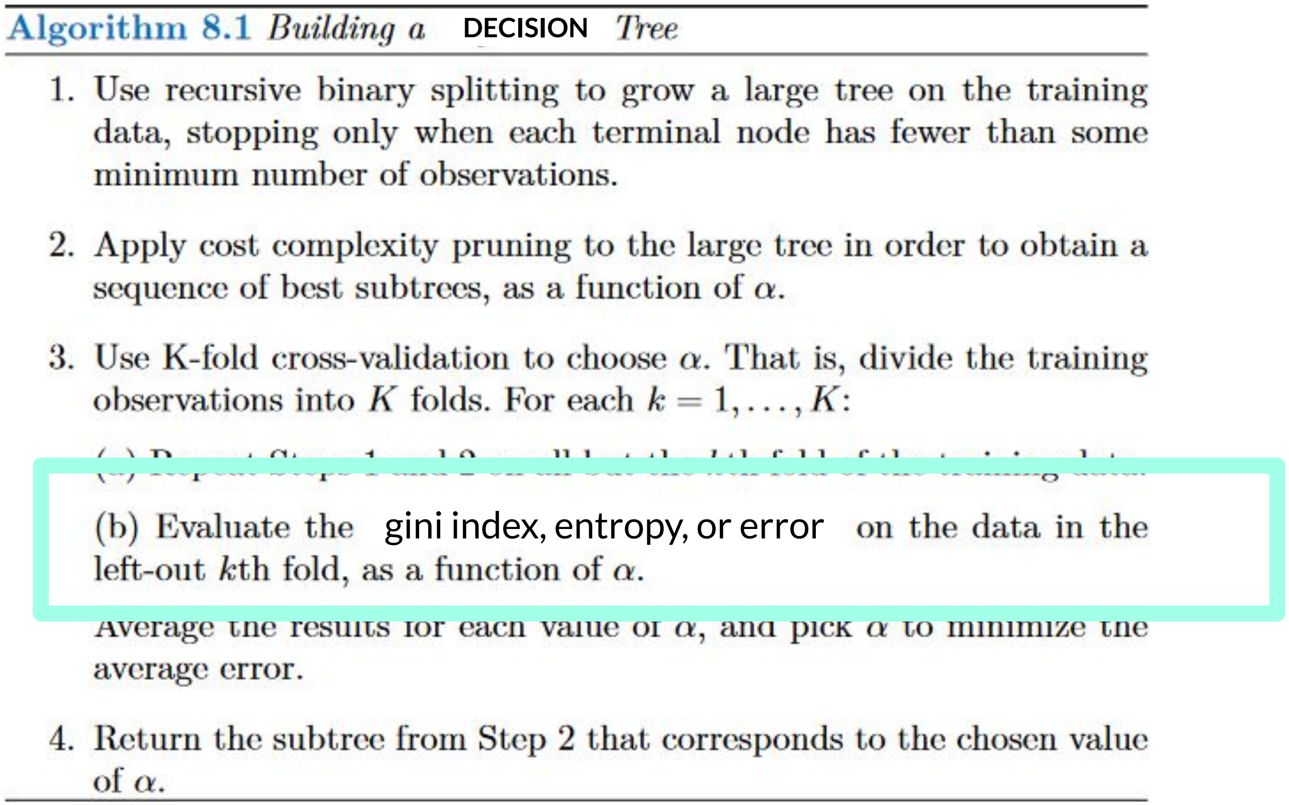
## Decision Trees

In order to find necessary “splits” in our data, we **eyeballed** our data points and named subsequent inequalities.

As we’ve learned so far, **mathematics** provides the necessary precision to **formalize these ideas**.

Let’s take a closer look as to how trees make decisions on splits.

## Gini & Entropy



```
Algorithm 8.1 Building a DECISION Tree
1. Use recursive binary splitting to grow a large tree on the training
   data, stopping only when each terminal node has fewer than some
   minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a
   sequence of best subtrees, as a function of $\alpha$.
3. Use K-fold cross-validation to choose $\alpha$. That is, divide the training
   observations into $K$ folds. For each $k = 1, \ldots, K$:
     (a) Repeat Steps 1 and 2 on all but the $k$th fold of the training data.
     (b) Evaluate the gini index, entropy, or error on the data in the
         left-out $k$th fold, as a function of $\alpha$.
   Average the results for each value of $\alpha$, and pick $\alpha$ to minimize the
   average error.
4. Return the subtree from Step 2 that corresponds to the chosen value
   of $\alpha$.
```

To “**grow” a tree**, we utilize a few techniques that originate from computer science (and that are actually related to the development of **Deep Blue**), however we will abstract away most steps and simply focus on the concept of “node purity”

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

To evaluate the **purity of a node** (aka the error-rate) of a conditional node, we can utilize the gini index. A **large gini index** indicates a high rate of misclassifications, while a **small gini index** indicates that a node *predominantly contains observations from a single class.*

Ex: 0 → all elements underneath a node belong to a certain class

Let’s break down this equation before working through our previous dataset.

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

Calculate the probability that a sample belongs to a class (squared)

Sum each probability squared, and subtract from 1

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

Calculate the probability that a sample belongs to a class (squared)

Sum each probability squared, and subtract from 1

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

As we could imagine, if a collection of samples all belong to one class, we would get a probability of 100 (1).

As 1 - 1 = 0, this indicates a "pure node."

Let's see how this applies to our toy dataset.

Calculate the probability that a sample belongs to a class (squared)

| av_rpg | av_bpg | won |
|---|---|---|
| 2 | 7 | 0 |
| 2 | 4 | 0 |
| 4 | 2 | 1 |
| 4 | 8 | 0 |
| 6 | 3 | 1 |
| 8 | 5 | 1 |

To calculate *Gini Impurity* we will first begin analyzing a single predictor, such as “**av_rpg**.” Let’s view adjacent averages of our sorted **av_rpg** values

| av_rpg | av_bpg | won |
|---|---|---|
| 2 | 7 | 0 |
| 2 | 4 | 0 |
| 4 | 2 | 1 |
| 4 | 8 | 0 |
| 6 | 3 | 1 |
| 8 | 5 | 1 |

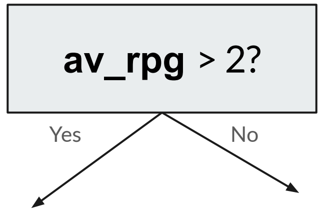
*(2+2) / 2 = 2*

*(2+4) / 2 = 3*

*(4+4) / 2 = 4*

*(4+6) / 2 = 5*

*(6+8) / 2 = 7*



We will try all these adjacent averages to **calculate the "*purest*" splits available**, these will determine our *optimal splits*. Let's first try a split of **av_rpg > 2**

| av_rpg | av_bpg | won |
|---|---|---|
| 2 | 7 | 0 |
| 2 | 4 | 0 |
| 4 | 2 | 1 |
| 4 | 8 | 0 |
| 6 | 3 | 1 |
| 8 | 5 | 1 |

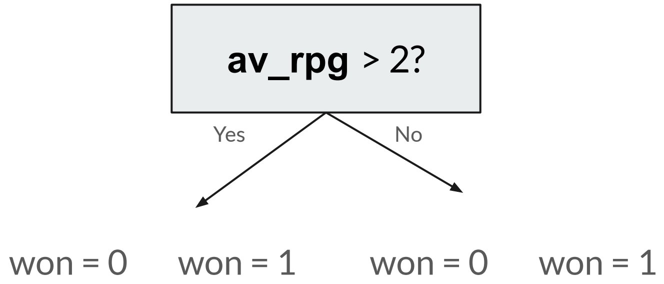
*(2+2) / 2 = 2*

*(2+4) / 2 = 3*

*(4+4) / 2 = 4*

*(4+6) / 2 = 5*

*(6+8) / 2 = 7*



How many samples belong to the "**won=1**" class where **av_rpg > 2**? How many samples belong to the "**won=0**" class where **av_rpg>2**?

What about for the case where **av_rpg > 2 is False**?

| av_rpg | av_bpg | won |
|---|---|---|
| 2 | 7 | 0 |
| 2 | 4 | 0 |
| 4 | 2 | 1 |
| 4 | 8 | 0 |
| 6 | 3 | 1 |
| 8 | 5 | 1 |

*(2+2) / 2 = 2*

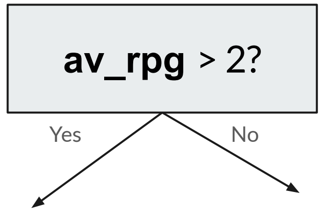
*(2+4) / 2 = 3*

*(4+4) / 2 = 4*

*(4+6) / 2 = 5*

*(6+8) / 2 = 7*

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



| won = 0 | won = 1 | won = 0 | won = 1 |
|---|---|---|---|
| **1** | **3** | **2** | **0** |

1 - ((1/4)^2 + (3/4)^2)    1 - ((2/2)^2 + (0/2)^2)

Using these frequencies, let's calculate our probabilities (**ratios of classes**) and subsequent gini impurity.

What is the gini impurity of the "True" split, what about the "No" split???

| av_rpg | av_bpg | won |
|---|---|---|
| 2 | 7 | 0 |
| 2 | 4 | 0 |
| 4 | 2 | 1 |
| 4 | 8 | 0 |
| 6 | 3 | 1 |
| 8 | 5 | 1 |

*(2+2) / 2 = 2*

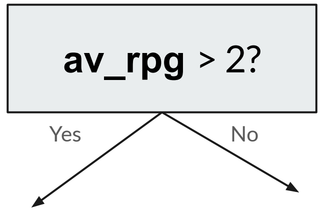
*(2+4) / 2 = 3*

*(4+4) / 2 = 4*

*(4+6) / 2 = 5*

*(6+8) / 2 = 7*

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



| won = 0 | won = 1 | won = 0 | won = 1 |
|---|---|---|---|
| 1 | 3 | 2 | 0 |

Gini = 0.375

Gini = 0

**Looks like we have a "pure" split (0) for the right-leaf and a gini impurity of 0.375 for the left-leaf.**

| av_rpg | av_bpg | won |
|---|---|---|
| 2 | 7 | 0 |
| 2 | 4 | 0 |
| 4 | 2 | 1 |
| 4 | 8 | 0 |
| 6 | 3 | 1 |
| 8 | 5 | 1 |

*(2+2) / 2 = 2*

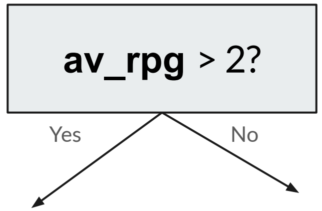
*(2+4) / 2 = 3*

*(4+4) / 2 = 4*

*(4+6) / 2 = 5*

*(6+8) / 2 = 7*

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



| won = 0 | won = 1 | won = 0 | won = 1 |
|---|---|---|---|
| **1** | **3** | **2** | **0** |

*Total Gini = (2/6)*0 + (4/6)*0.375*

Let's combine these impurities using the **weighted average of both leaves**.
Consider that the left-leaf has 4 samples, and the right-leaf has 2 samples.

| av_rpg | av_bpg | won |
|---|---|---|
| 2 | 7 | 0 |
| 2 | 4 | 0 |
| 4 | 2 | 1 |
| 4 | 8 | 0 |
| 6 | 3 | 1 |
| 8 | 5 | 1 |

*(2+2) / 2 = 2*

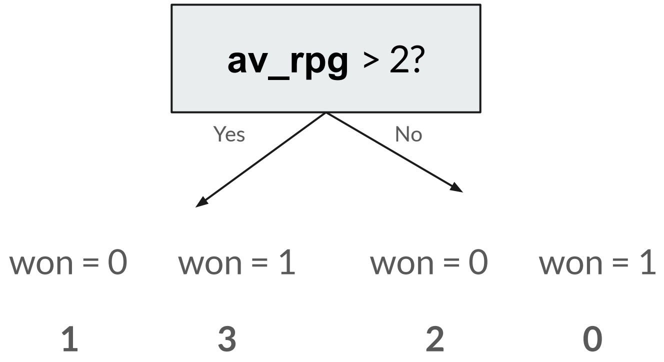
*(2+4) / 2 = 3*

*(4+4) / 2 = 4*

*(4+6) / 2 = 5*

*(6+8) / 2 = 7*

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

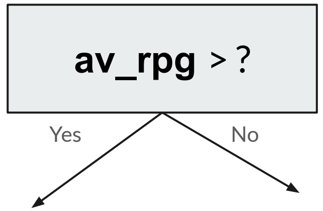


*Total Gini = 0.25*

This gives us a total impurity score of 0.25. We want to select the split that results in the lowest gini impurity possible. **Let's continue this calculation for each adjacent average.**

| av_rpg | av_bpg | won |
|---|---|---|
| 2 | 7 | 0 |
| 2 | 4 | 0 |
| 4 | 2 | 1 |
| 4 | 8 | 0 |
| 6 | 3 | 1 |
| 8 | 5 | 1 |

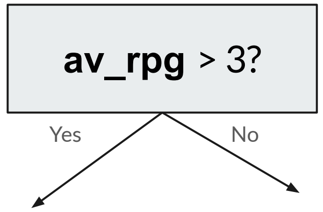
| | |
|---|---|
| *(2+2) / 2 = 2* | ***Gini = 0.25*** |
| *(2+4) / 2 = 3* | ***Gini = 0.25*** |
| *(4+4) / 2 = 4* | ***Gini = 0.25*** |
| *(4+6) / 2 = 5* | ***Gini = 0.25*** |
| *(6+8) / 2 = 7* | ***Gini = 0.40*** |



Observing the following gini impurities, which split is the **least impure**?

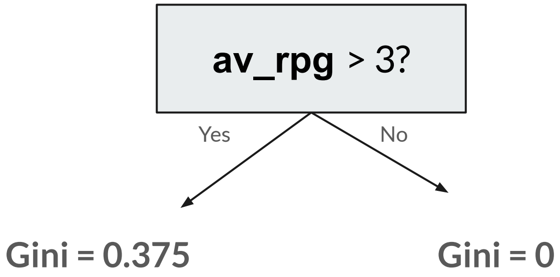
| av_rpg | av_bpg | won |
|---|---|---|
| 2 | 7 | 0 |
| 2 | 4 | 0 |
| 4 | 2 | 1 |
| 4 | 8 | 0 |
| 6 | 3 | 1 |
| 8 | 5 | 1 |

| | |
|---|---|
| *(2+2) / 2 = 2* | *Gini = 0.25* |
| *(2+4) / 2 = 3* | *Gini = 0.25* |
| *(4+4) / 2 = 4* | *Gini = 0.25* |
| *(4+6) / 2 = 5* | *Gini = 0.25* |
| *(6+8) / 2 = 7* | *Gini = 0.40* |



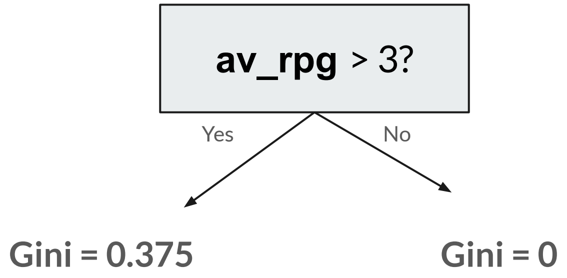
Since this dataset is a simple thought-experiment, we get non-interesting impurities. **Anyways let's just choose "3" since that's what we got previously.**

| av_rpg | av_bpg | won |
|---|---|---|
| 2 | 7 | 0 |
| 2 | 4 | 0 |
| 4 | 2 | 1 |
| 4 | 8 | 0 |
| 6 | 3 | 1 |
| 8 | 5 | 1 |



As we still have impure splits, for the "Yes" branch, we will continue our evaluation for **gini impurity for additional predictors until we have impurities of 0** (or the lowest possible val)

| av_rpg | av_bpg | won |
|---|---|---|
| 2 | 7 | 0 |
| 2 | 4 | 0 |
| 4 | 2 | 1 |
| 4 | 8 | 0 |
| 6 | 3 | 1 |
| 8 | 5 | 1 |



- How do we know when to stop creating splits?
- What's the risk of creating too many splits?
- How do we know which predictor to start from?

A few questions regarding this algorithm should be coming to surface, most of these will be addressed tomorrow.

# Decision Trees (DT's)

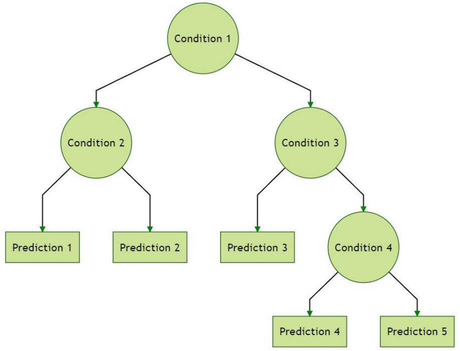
In summary, DT's are a **non-parametric multiclass supervised learning algorithm** that creates optimal splits based on measures of **information loss**.

Pros

- Easy to **explain & visualize**
- No need to **normalize data** (most of the time)

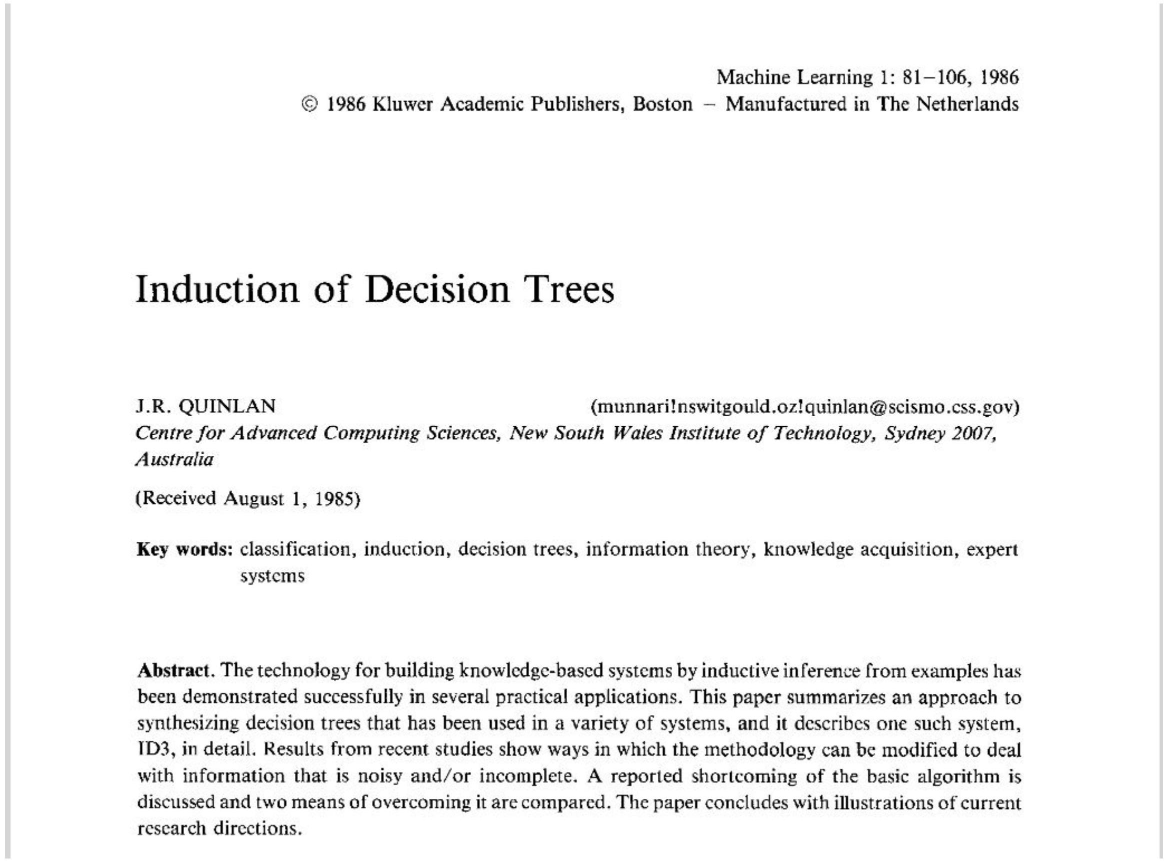
Cons

- **Non-robust** classifier
- Need to store **entire training dataset** for prediction
- Poor **predictive accuracy**



However, they usually **display the worst predictive capabilities** out of all algorithms we've seen so far. We will see how we can improve upon trees next week.

Note that trees are not limited to just one level! Instead we could potentially have a “deep” tree.



Machine Learning 1: 81–106, 1986
© 1986 Kluwer Academic Publishers, Boston – Manufactured in The Netherlands

Induction of Decision Trees

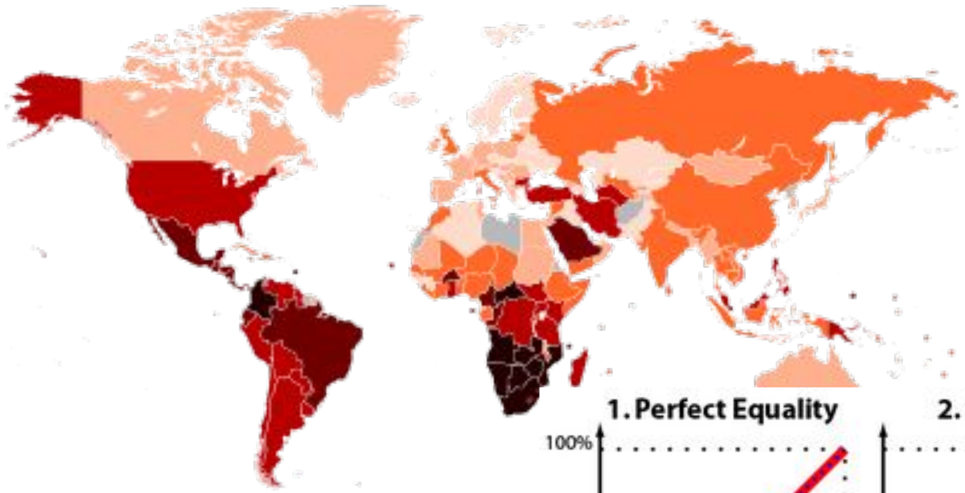
J.R. QUINLAN (munnari!nswitgould.oz!quinlan@scismo.css.gov)
*Centre for Advanced Computing Sciences, New South Wales Institute of Technology, Sydney 2007, Australia*

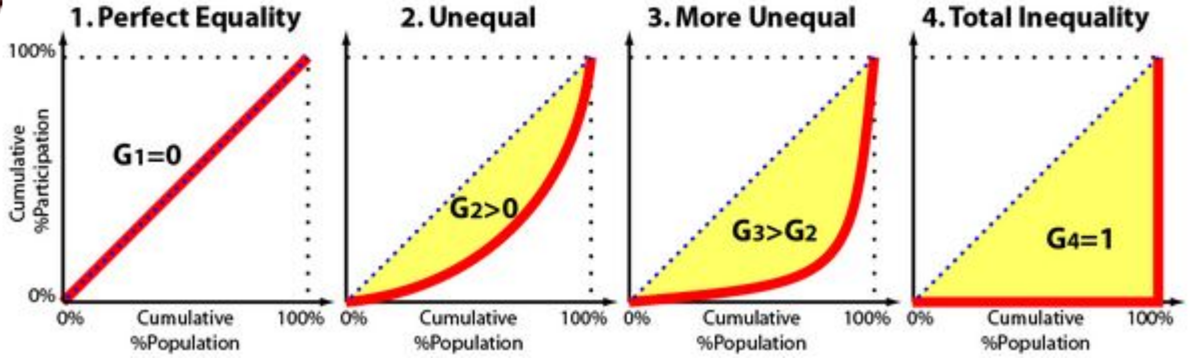
(Received August 1, 1985)

**Key words:** classification, induction, decision trees, information theory, knowledge acquisition, expert systems

**Abstract.** The technology for building knowledge-based systems by inductive inference from examples has been demonstrated successfully in several practical applications. This paper summarizes an approach to synthesizing decision trees that has been used in a variety of systems, and it describes one such system, ID3, in detail. Results from recent studies show ways in which the methodology can be modified to deal with information that is noisy and/or incomplete. A reported shortcoming of the basic algorithm is discussed and two means of overcoming it are compared. The paper concludes with illustrations of current research directions.

The origins of the decision tree might reach as far as the 1950's, but the 1986 paper provides some of the earliest documentation of this technique: https://link.springer.com/article/10.1007/BF00116251





Fun fact: the gini coefficient is also used to measure social inequality in the world

## End of Class Announcements

## Lab (Due 7/23)

*Zurich, Switzerland*

You are a data scientist working for a Zurich-based international bank called Caishen. The company announced in an all-hands meeting that they are aiming to develop a classification algorithm **that can identify 99% of all fraudulent activity within customer-facing bank accounts.**

For this project, you will use a **dataset of 1 million bank transactions to create a classifier** that will detect if fraudulent activity has occurred for a transaction.

Your second checkpoint will be to complete your EDA (of template files) by **7/16.**

## Wednesday

### Review

- *What questions do you have about all classification algos so far?*
- *What differentiates all of these algorithms?*

***What is the benefit of working in an ensemble?***