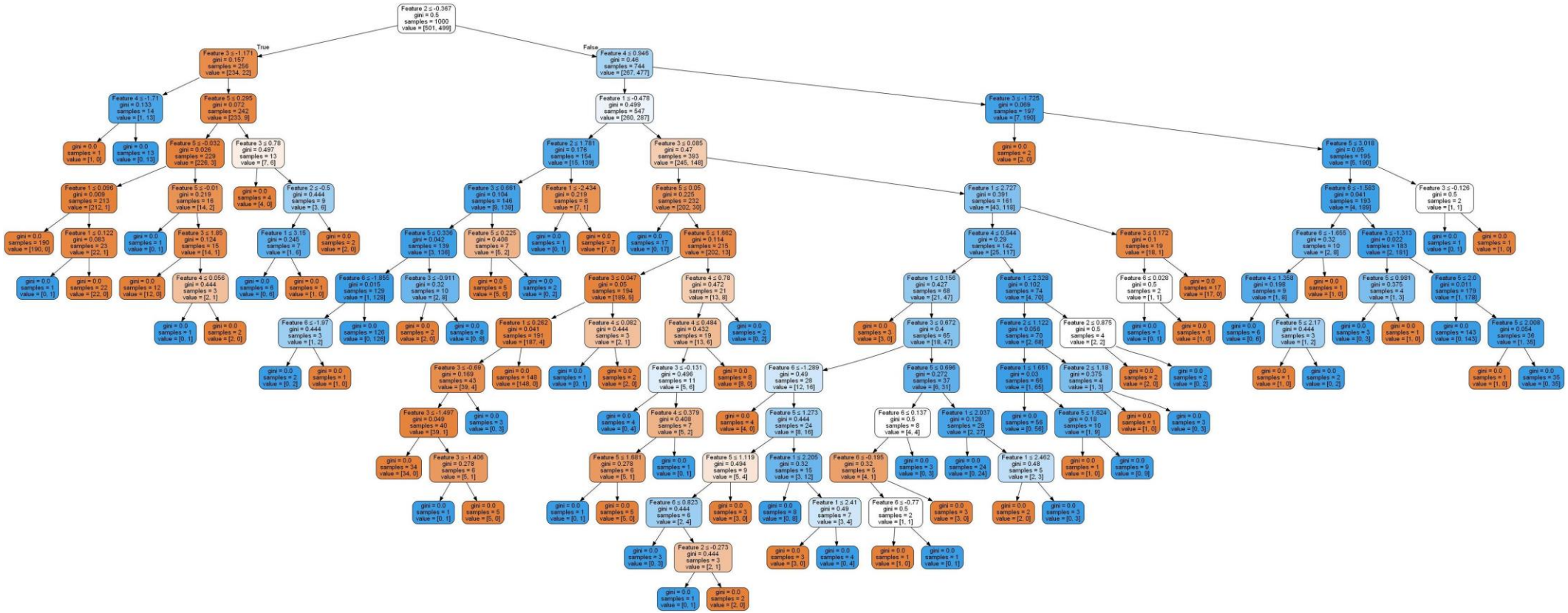
# CSE 445
# Lecture 12

Decision Tree (Part 1)

# Decision tree

- Classification and regression trees are machine-learning methods for constructing prediction models from data
- The models are obtained by recursively partitioning the data space and fitting a simple prediction model within each partition
- As a result, the partitioning can be represented graphically as a decision tree
- Classification trees are designed for dependent variables that take a finite number of unordered values, with prediction error measured in terms of misclassification cost
- Regression trees are for dependent variables that take continuous or ordered discrete values, with prediction error typically measured by the squared difference between the observed and predicted values

# Decision Tree cassifier



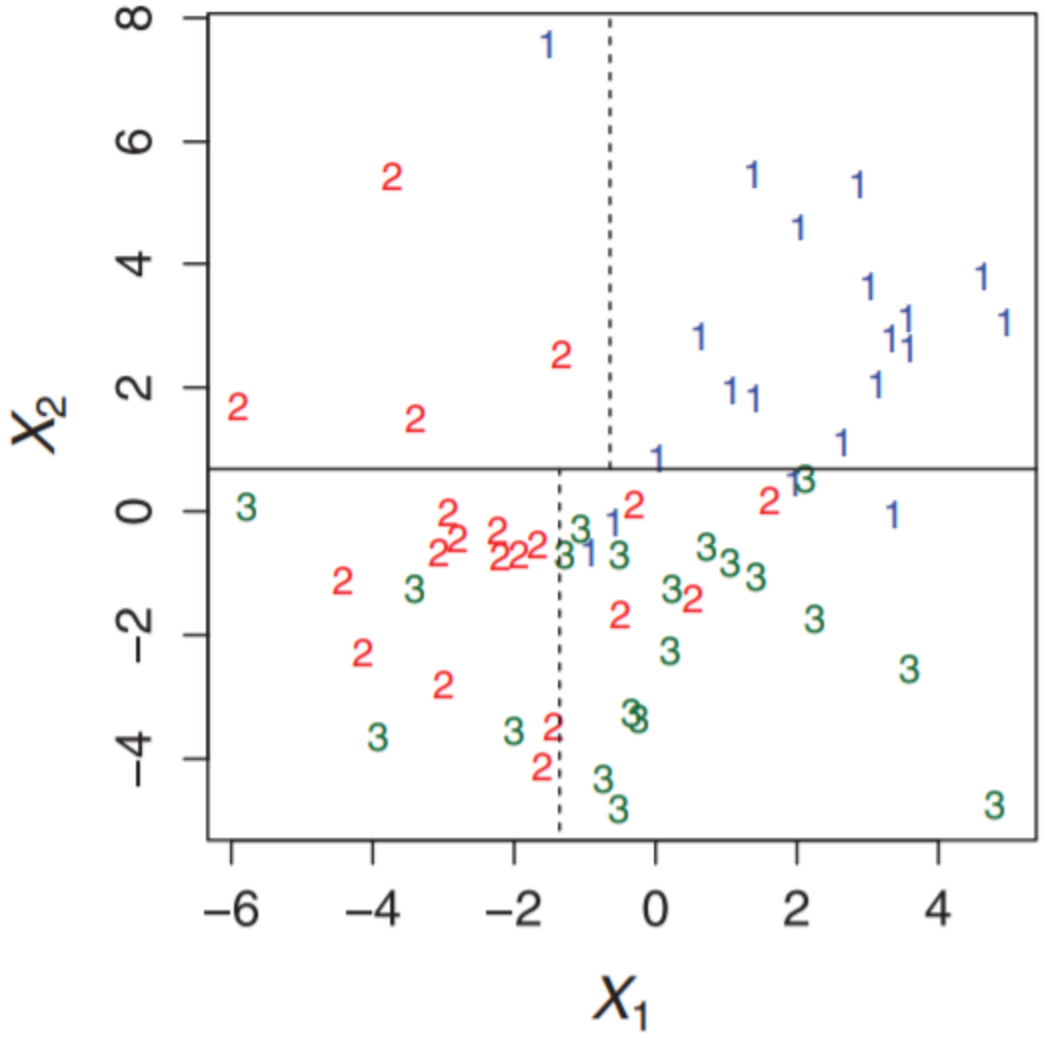
# Decision tree

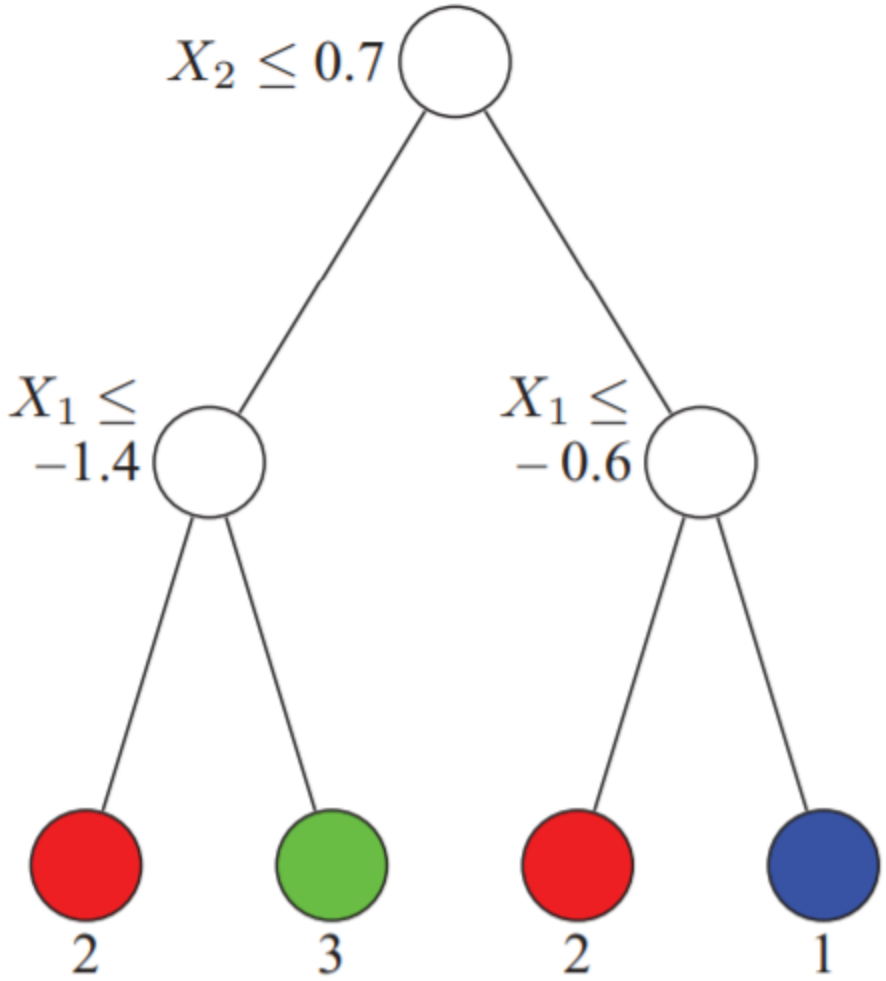
- In a classification problem, we have a training sample of n observations on a class variable Y that takes values 1, 2, ... , k, and p predictor variables, X1,..., Xp
  - Our goal is to find a model for predicting the values of Y from new X values
- In theory, the solution is simply a partition of the X space into k disjoint sets, A1, A2,..., Ak, such that:
  - The predicted value of Y is j if X belongs to Aj , for j = 1, 2,..., k
- If the X variables take ordered values, two classical solutions are linear discriminant analysis and nearest neighbor classification
- These methods yield sets Aj with piecewise linear and nonlinear, respectively, boundaries that are not easy to interpret if p is large

# An example

- Classification tree methods yield rectangular sets $A_j$ by recursively partitioning the data set one X variable at a time
- This makes the sets easier to interpret.
- The figure on the next slide gives an example wherein there are three classes and two X variables
  - The left panel plots the data points and partitions and the right panel shows the corresponding decision tree structure
- A key advantage of the tree structure is its applicability to any number of variables, whereas the plot on its left is limited to at most two.

# Decision tree example



# Decision tree algorithms

- Pseudocode for tree construction by exhaustive search
  - Start at the root node
  - For each X, find the set S that minimizes the sum of the node impurities in the two child nodes and choose the split $\{X* \in S*\}$ that gives the minimum overall X and S
  - If a stopping criterion is reached, exit. Otherwise, apply step 2 to each child node in turn
- In high level English words in the next slide

# Decision tree

- We start at the root and split the data based on the features
- We repeat this splitting procedure at each child node until the leaves are pure
- This means that the training examples at each node all belong to the same class
- In practice, this can result in a very deep tree with many nodes, which can easily lead to overfitting
- We typically want to **prune** the tree by setting a limit for the maximal depth of the tree

# Cost function (Objective function)

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

- The IG is simply the difference between the impurity of the parent node and the sum of the child node impurities
- The lower the impurities of the child nodes, the larger the information gain
- To keep the solution tractable, that is to reduce the combinatorial search space, most implementations use binary decision trees
- In that case, the above IG formula transform to the following

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

# Impurity measurement: Entropy

$$I_H(t) = -\sum_{i=1}^{c} p(i|t)\ log_2\ p(i|t)$$

- Here $p(i|t)$ – the proportion of the examples that belong to class $i$ for a particular node $t$
- The entropy is 0 if all examples at a node belong to the same class
- The entropy is maximal if we have uniform class distribution
- In a binary class setting, the entropy is 0 if $p(i = 1|t) = 1$ or $p(i = 0|t) = 0$
- If the classes are distributed uniformly with $p(i = 1|t) = 0.5$ and $p(i = 0|t) = 0.5$, the entropy is 1
- Therefore, we can say that the entropy criterion attempts to maximize the mutual information in the tree

# Impurity measurement: Gini impurity

$$I_G(t) = \sum_{i=1}^{c} p(i|t)\big(1 - p(i|t)\big) = 1 - \sum_{i=1}^{c} p(i|t)^2$$

- It tries to minimize the probability of misclassification
- Gini impurity is maximal if the classes have equal probabilities
  - For example, in a binary class setting

$$I_G(t) = 1 - \sum_{i=1}^{c} 0.5^2 = 0.5$$

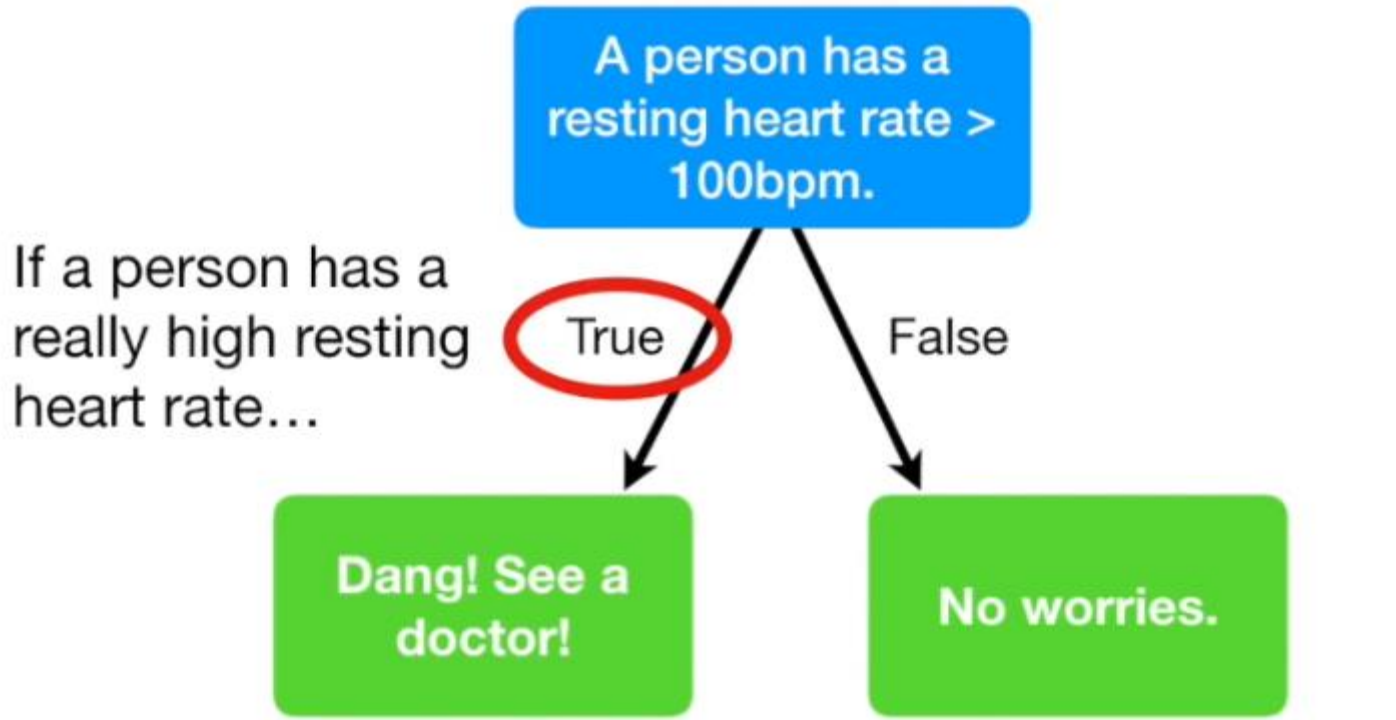
- However in practice Gini impurity and entropy performs almost the same

# Impurity measurement: Classification error

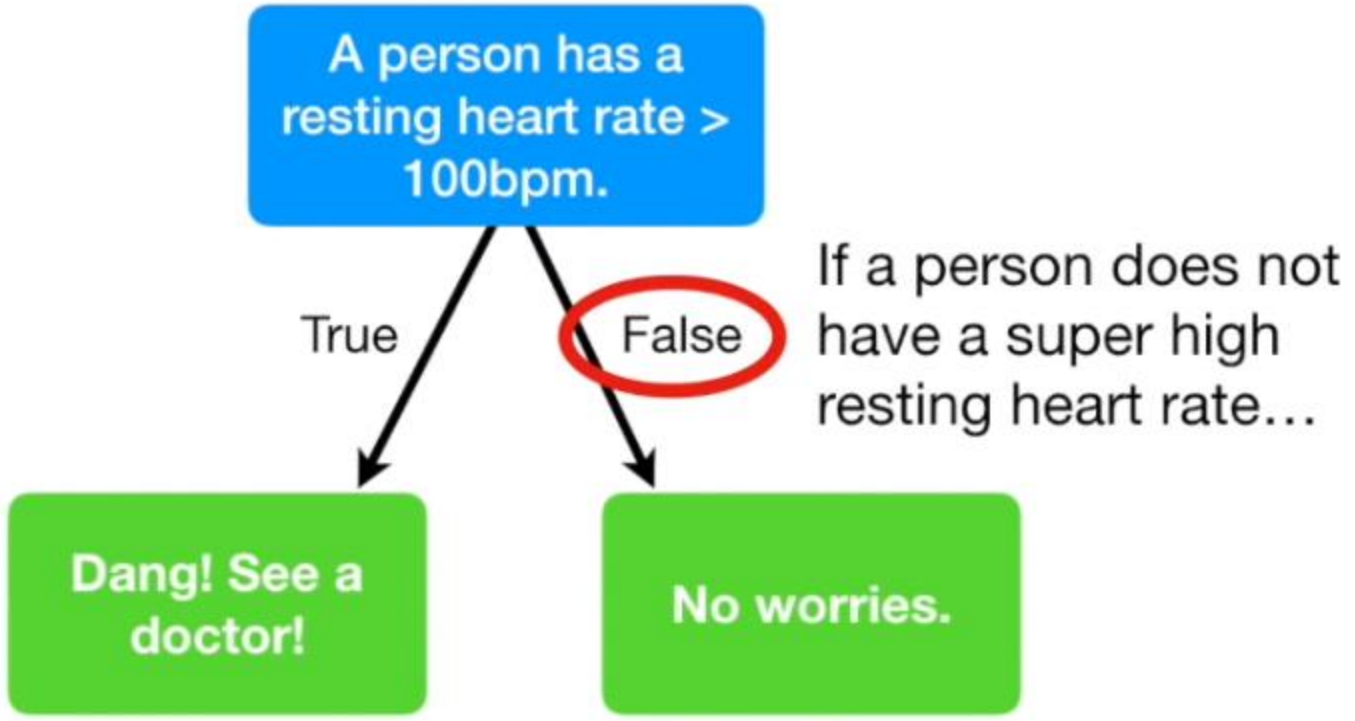
$$I_E(t) = 1 - \max\{ p(i|t)\}$$

- It looks promising
- But it is manly used for pruning the tree for building the tree
- The main issue  - it is less sensitive to the class probabilities of the nodes

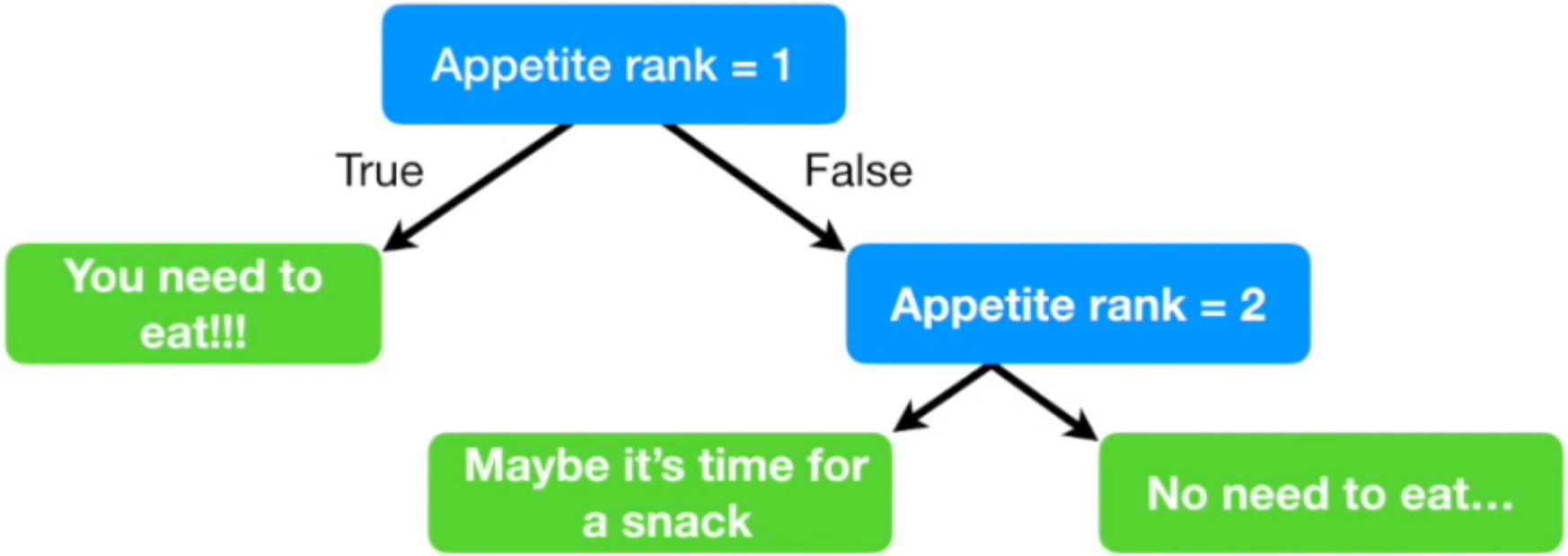
# Building a Decision Tree



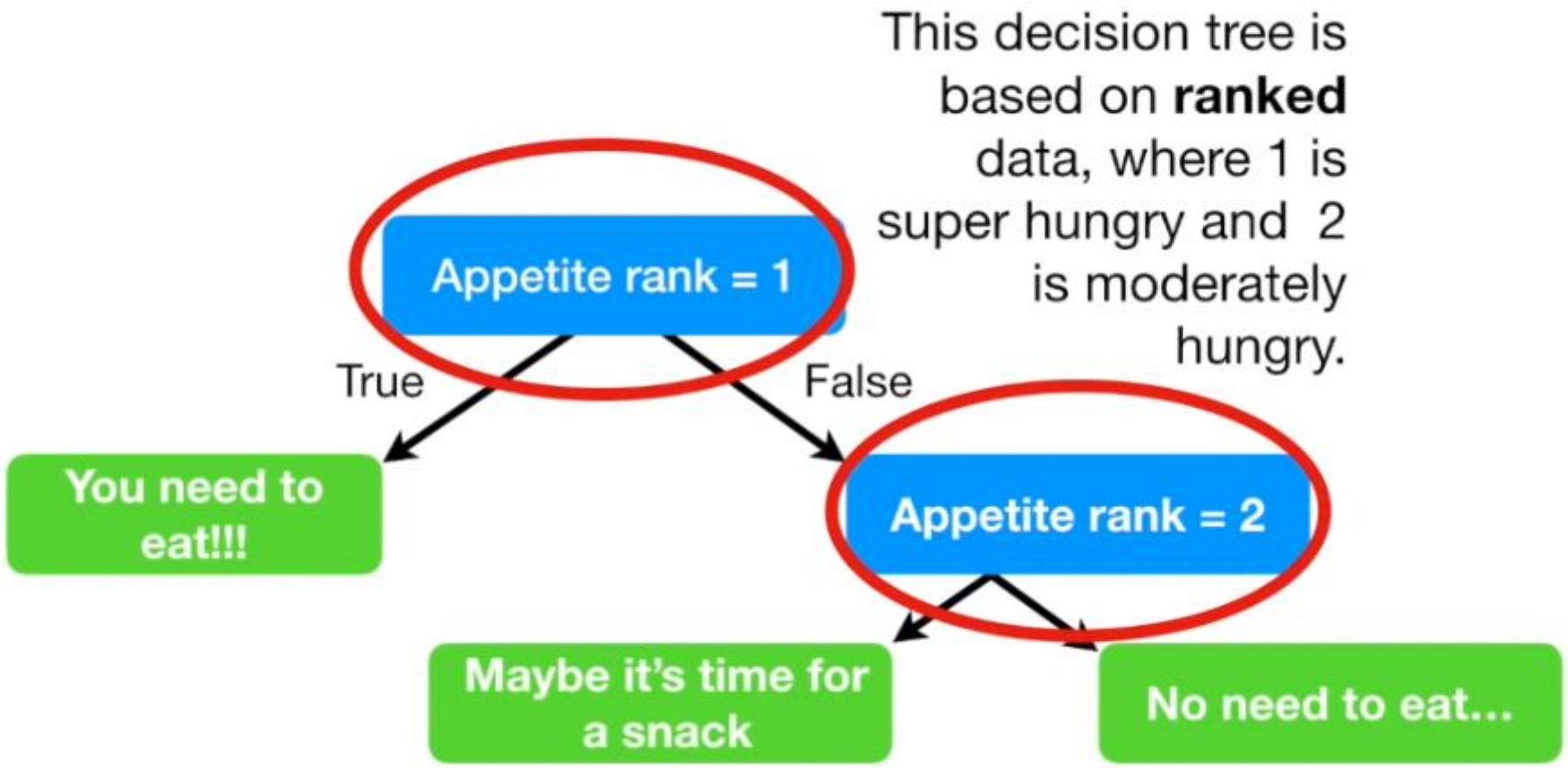
# Building a Decision Tree



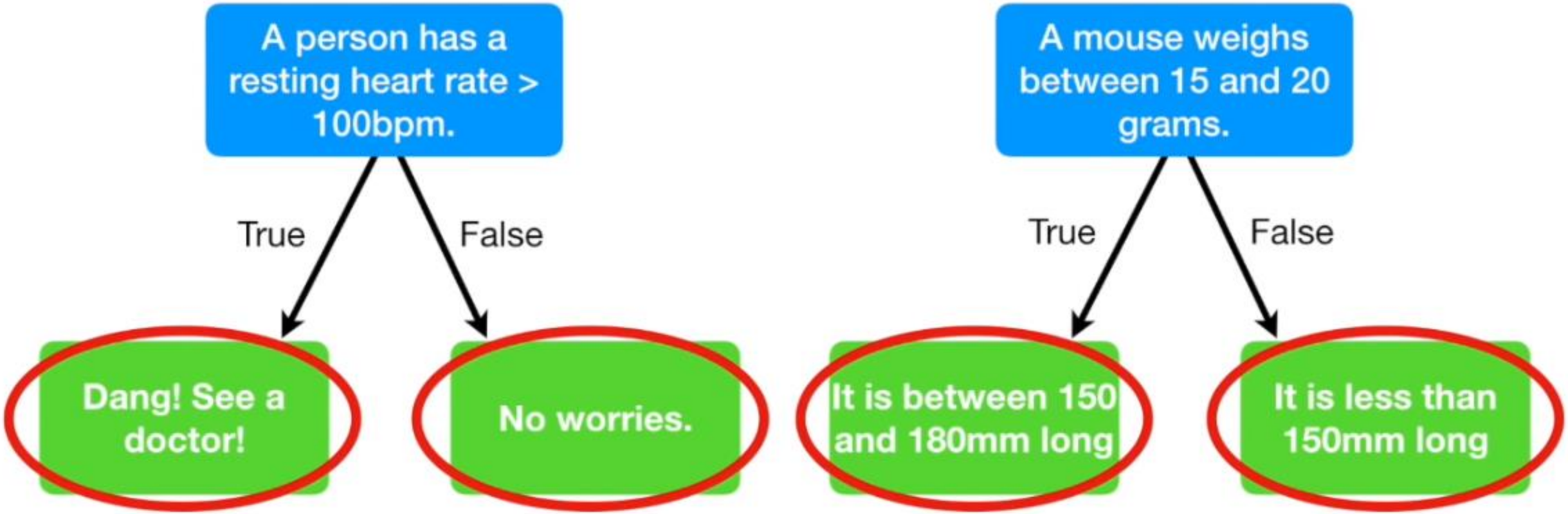
# Building a Decision Tree



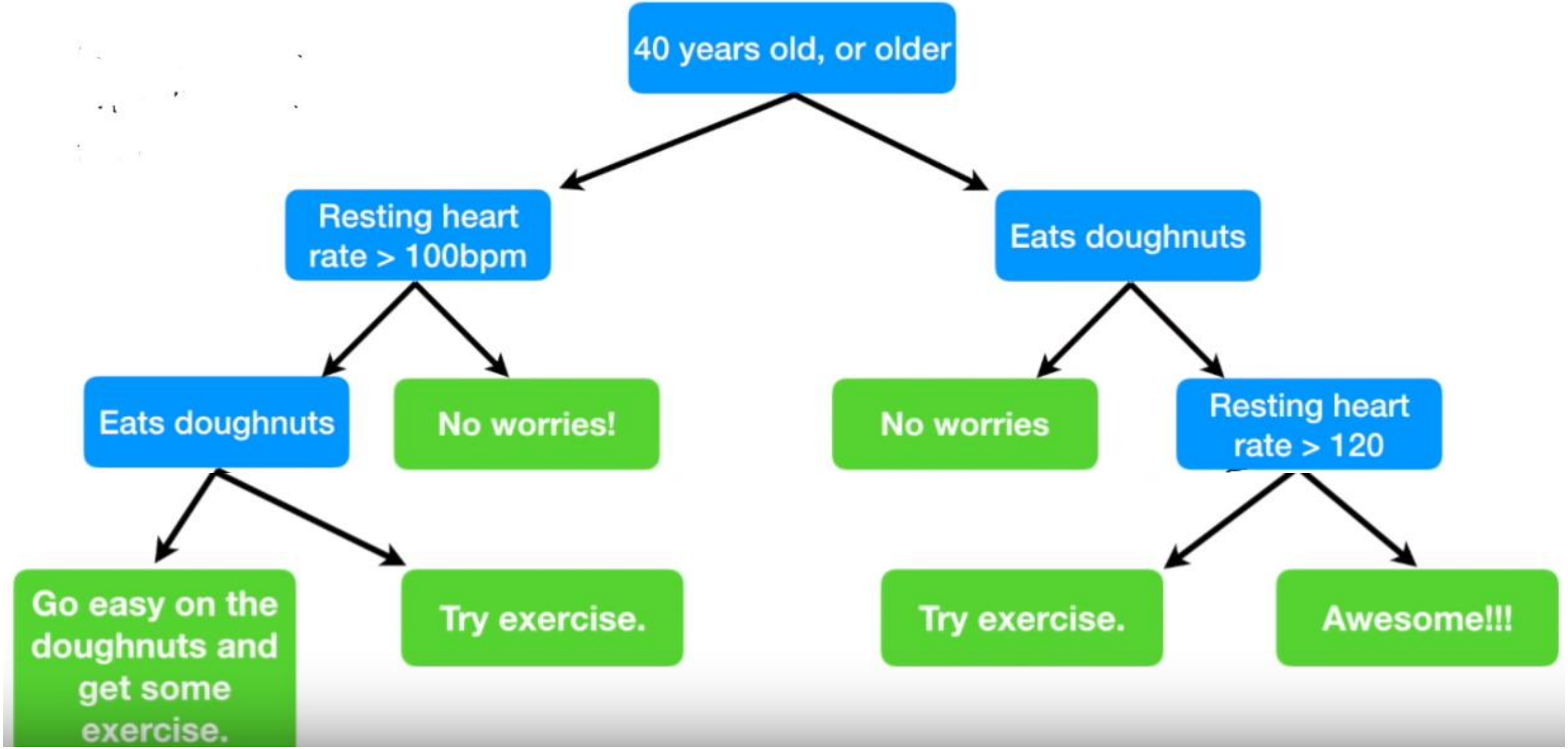
# Building a Decision Tree



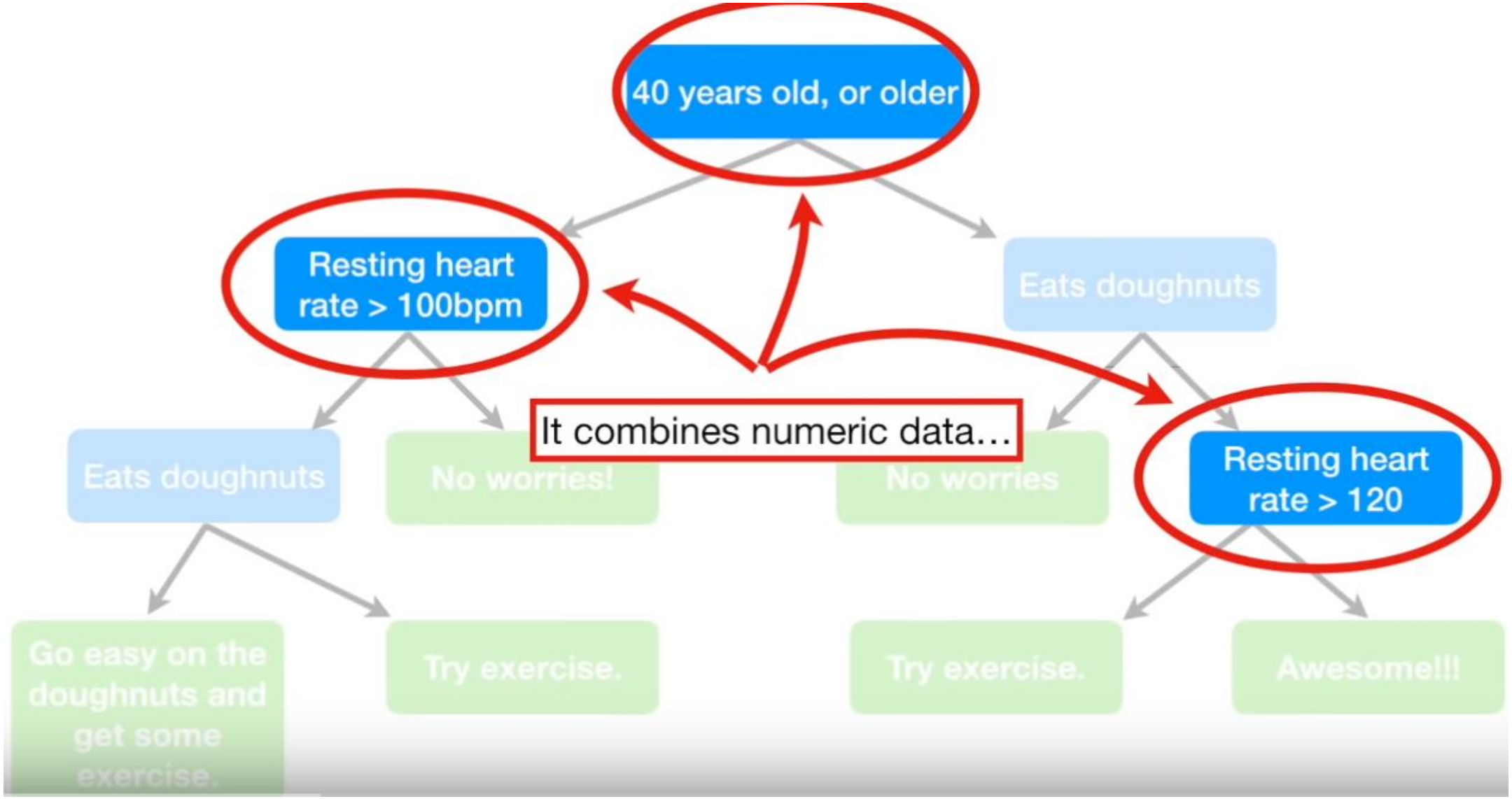
# Building a Decision Tree



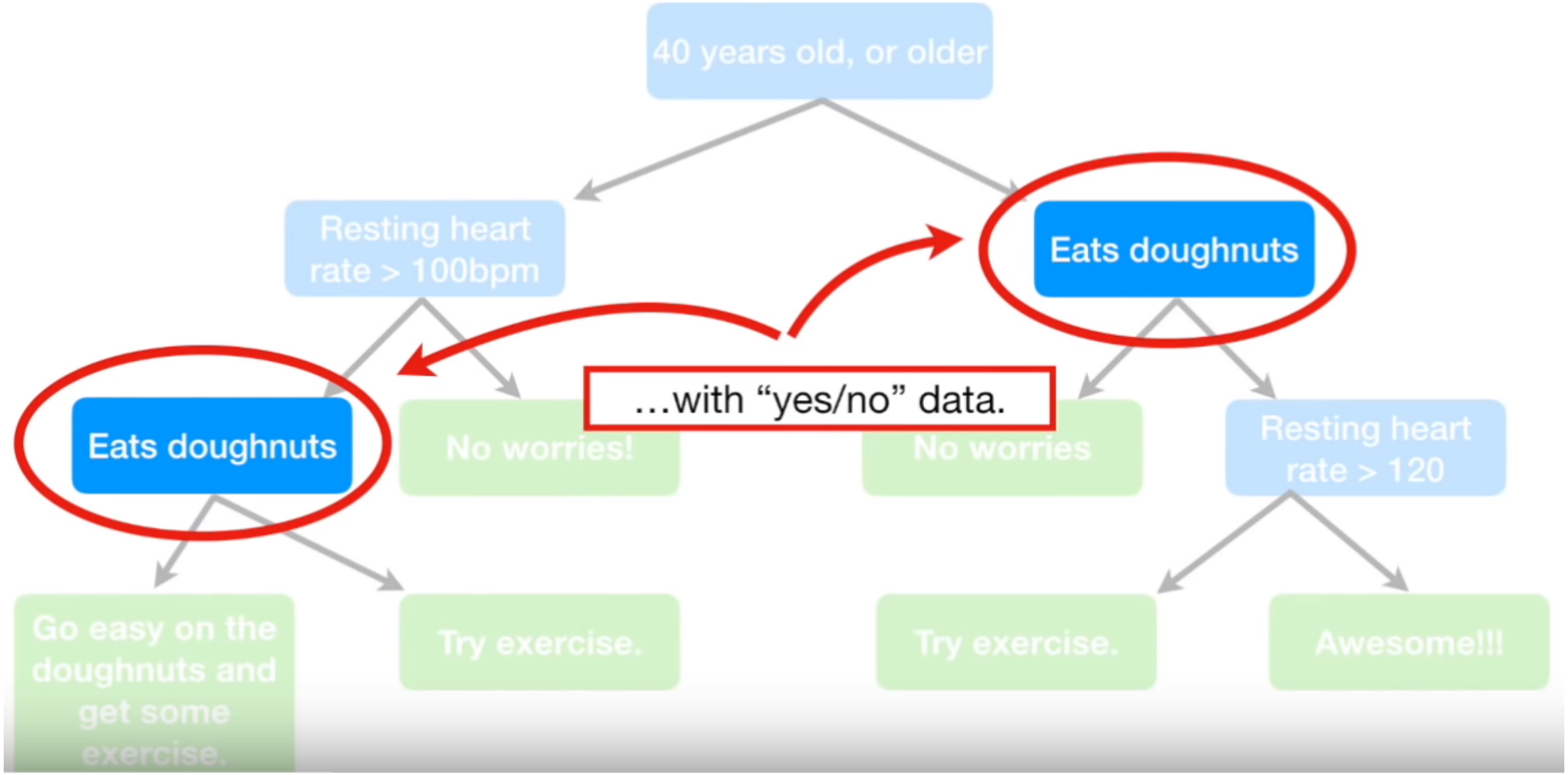
# Building a Decision Tree



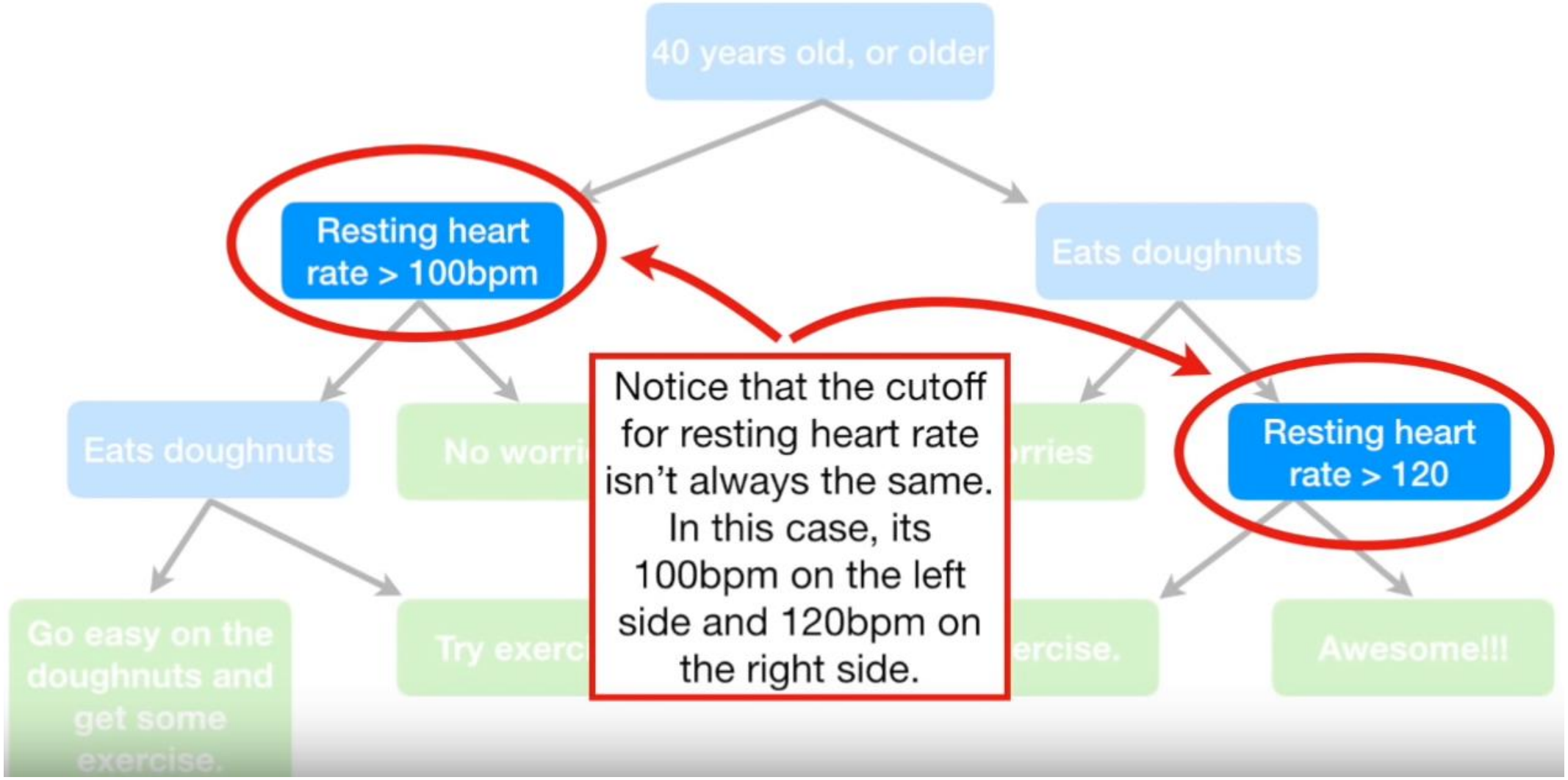
# Building a Decision Tree



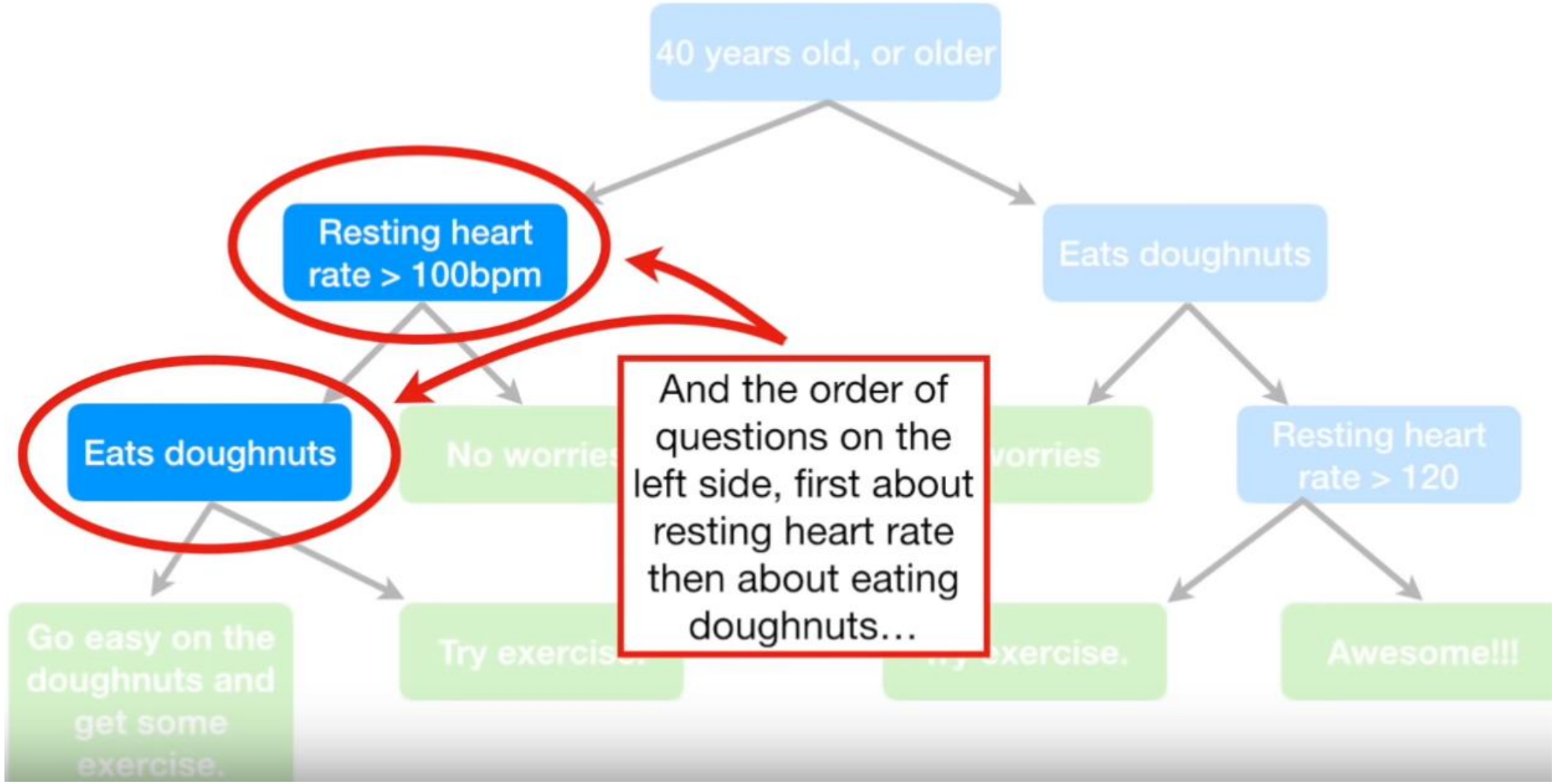
# Building a Decision Tree



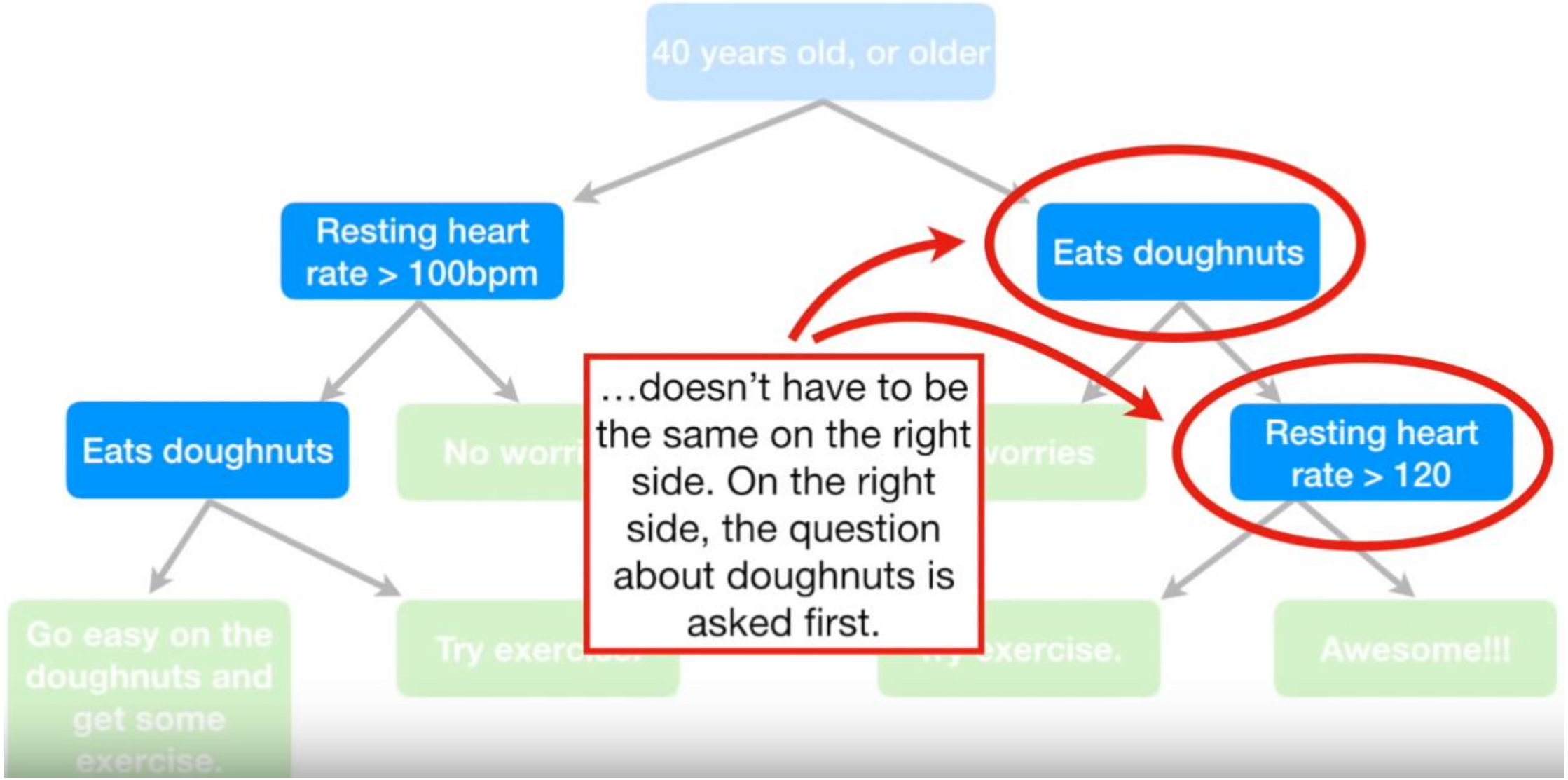
# Building a Decision Tree



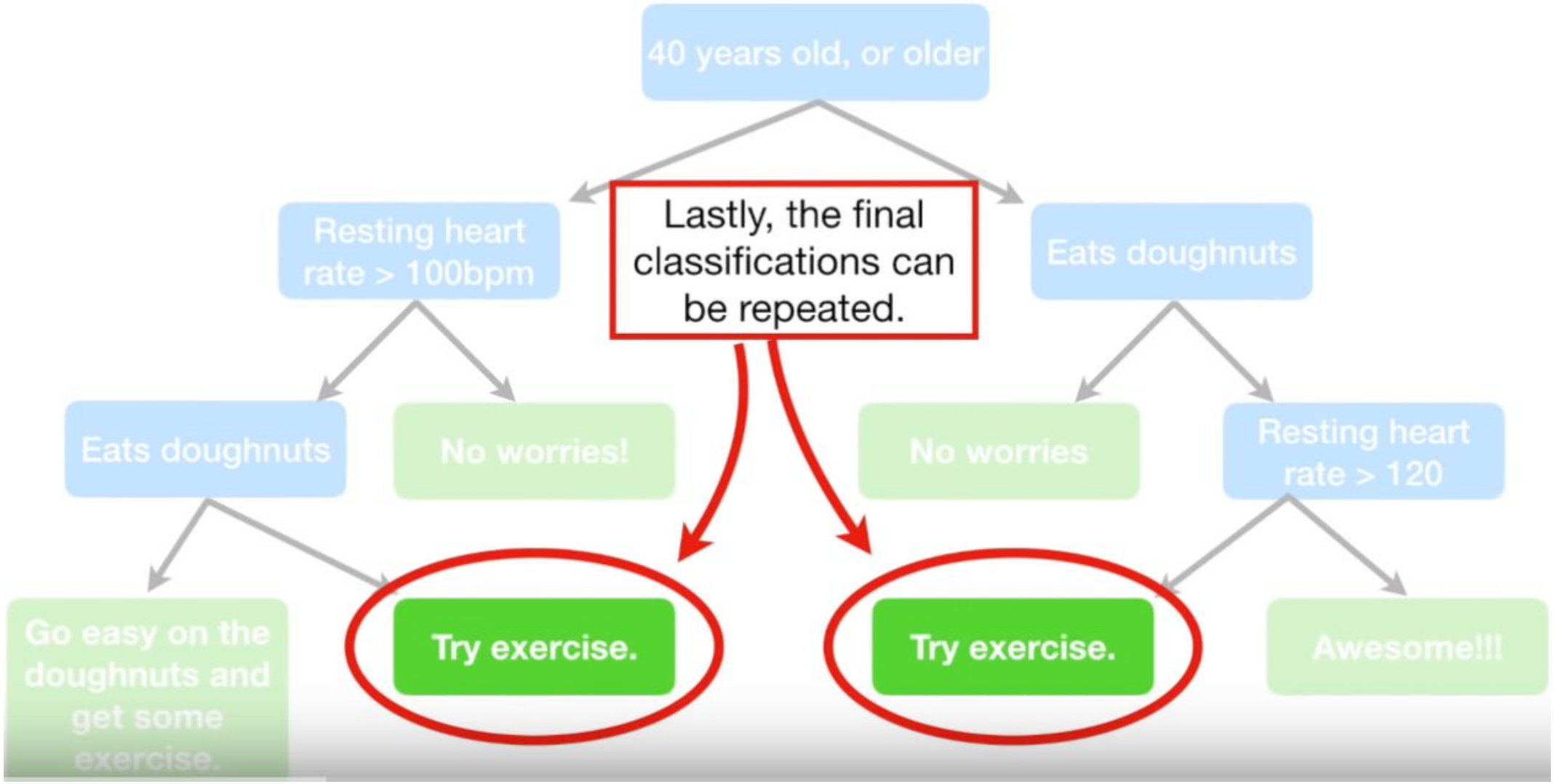
# Building a Decision Tree



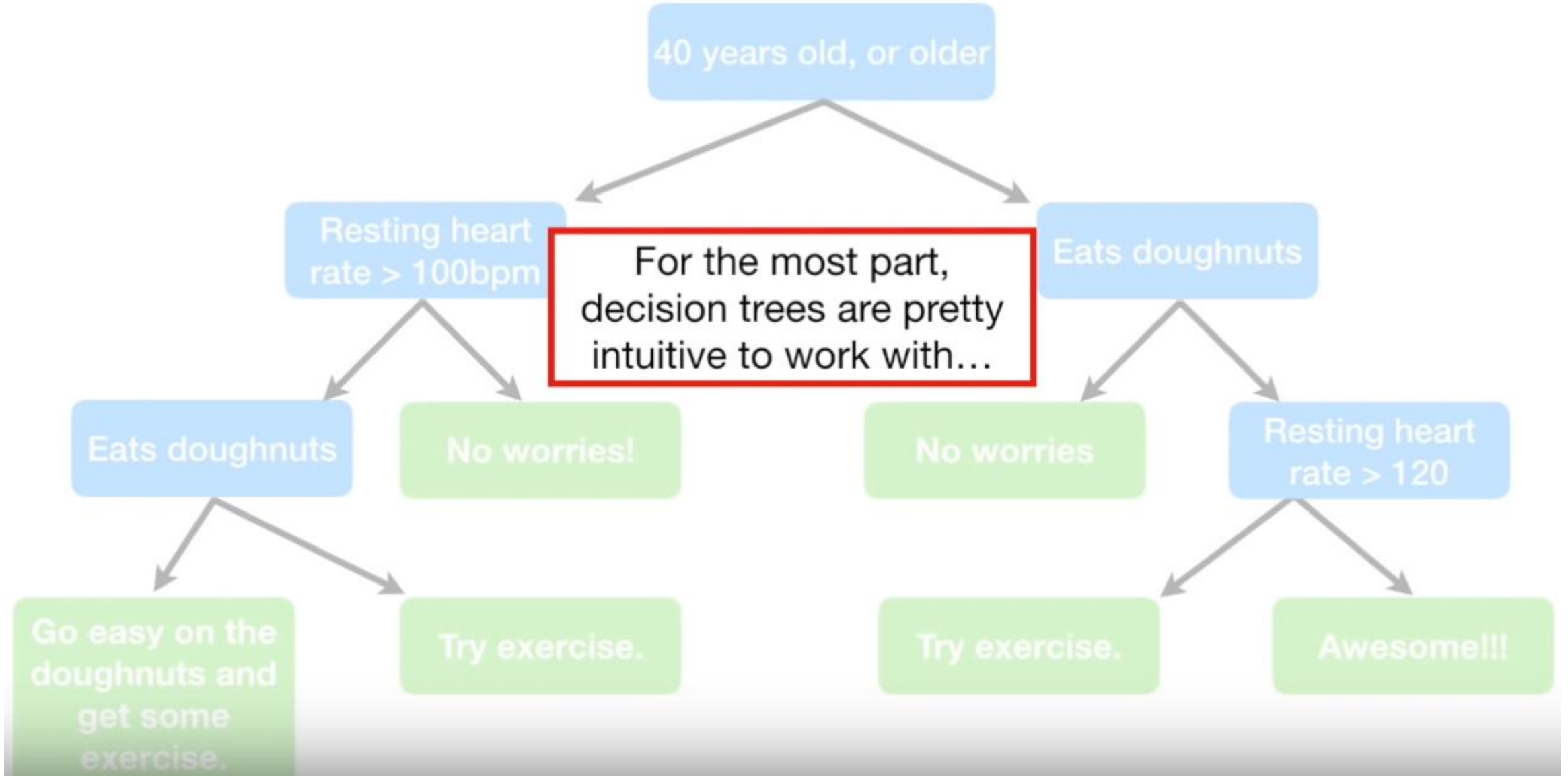
# Building a Decision Tree



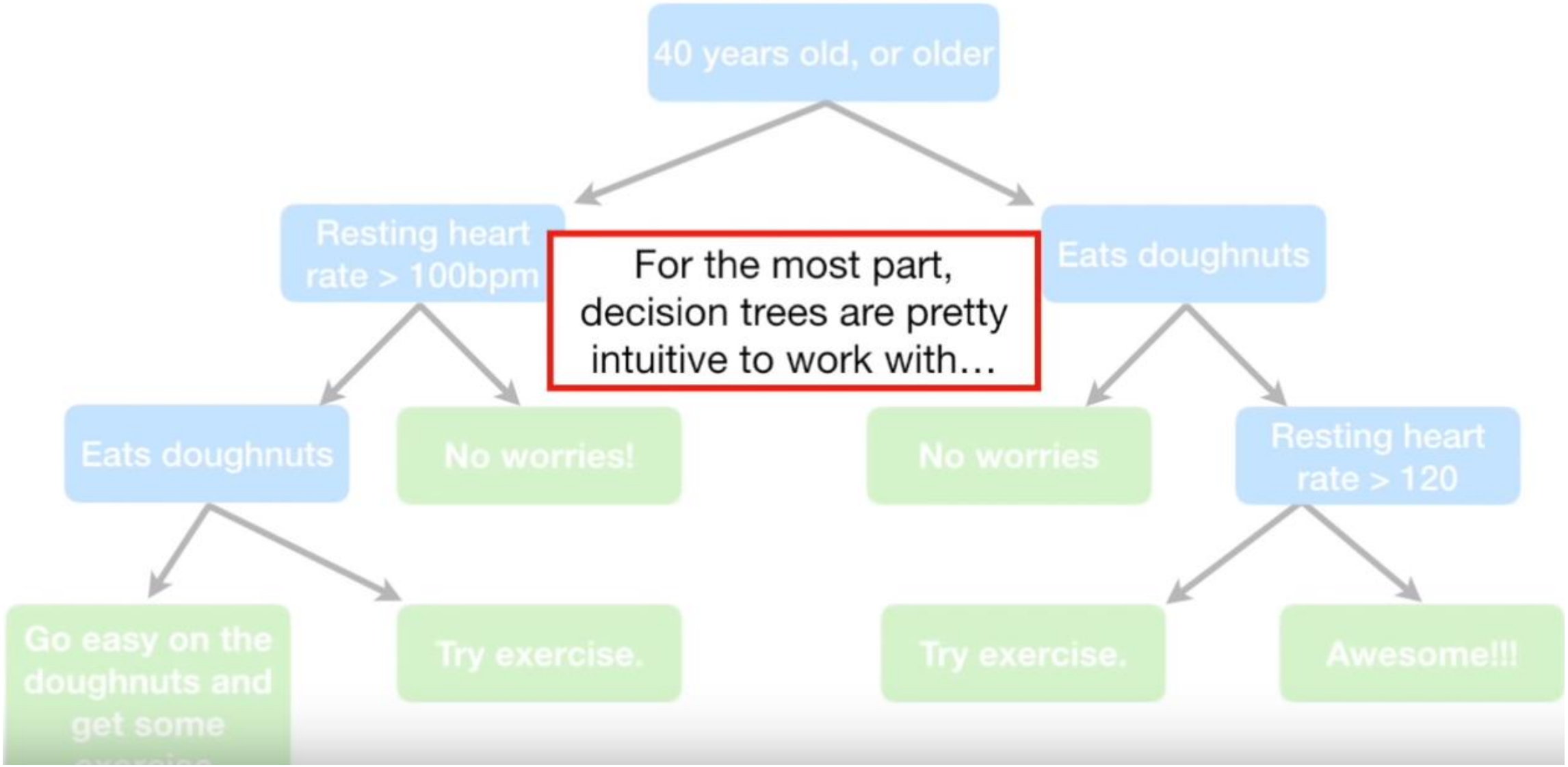
# Building a Decision Tree



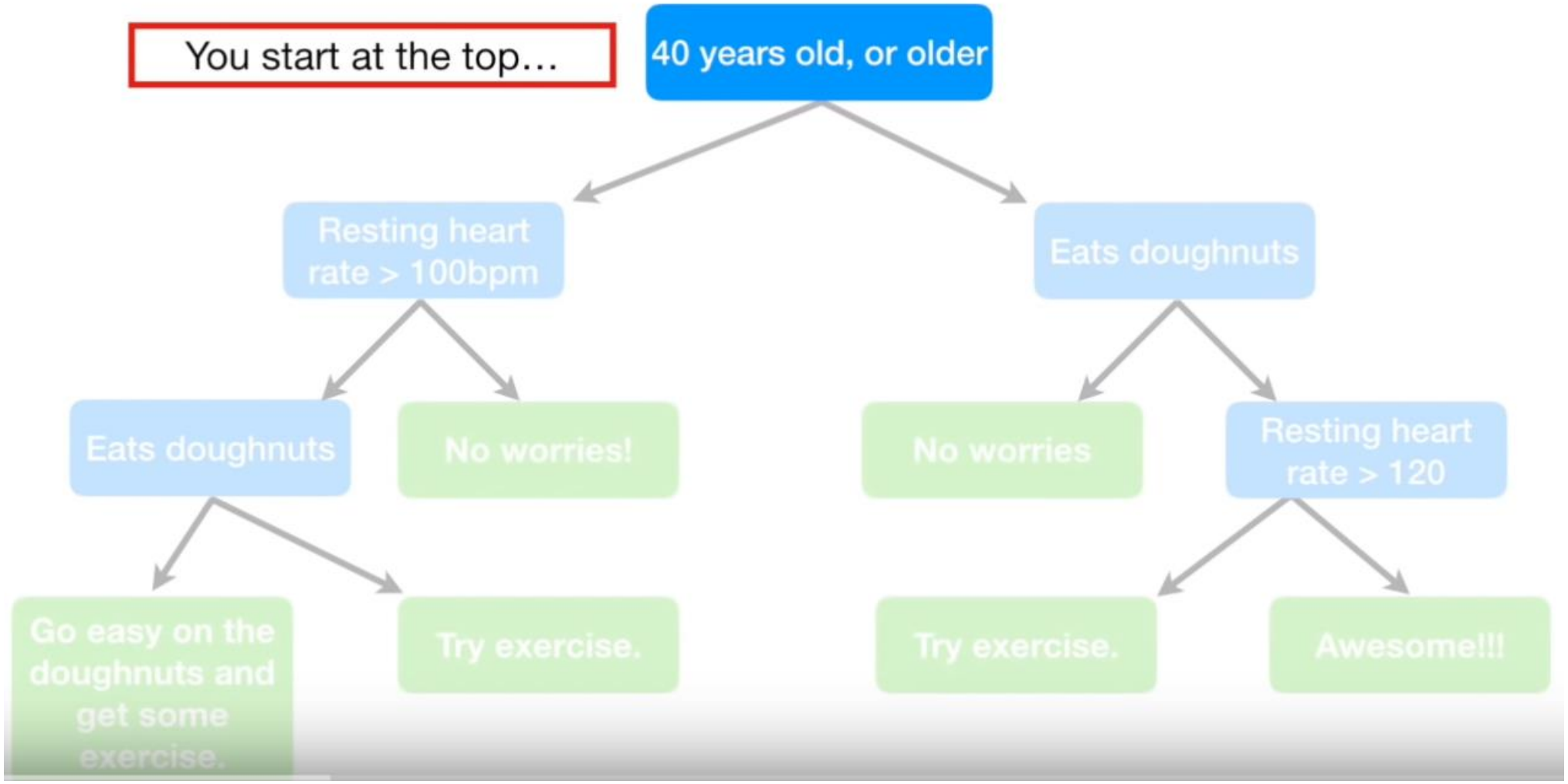
# Building a Decision Tree



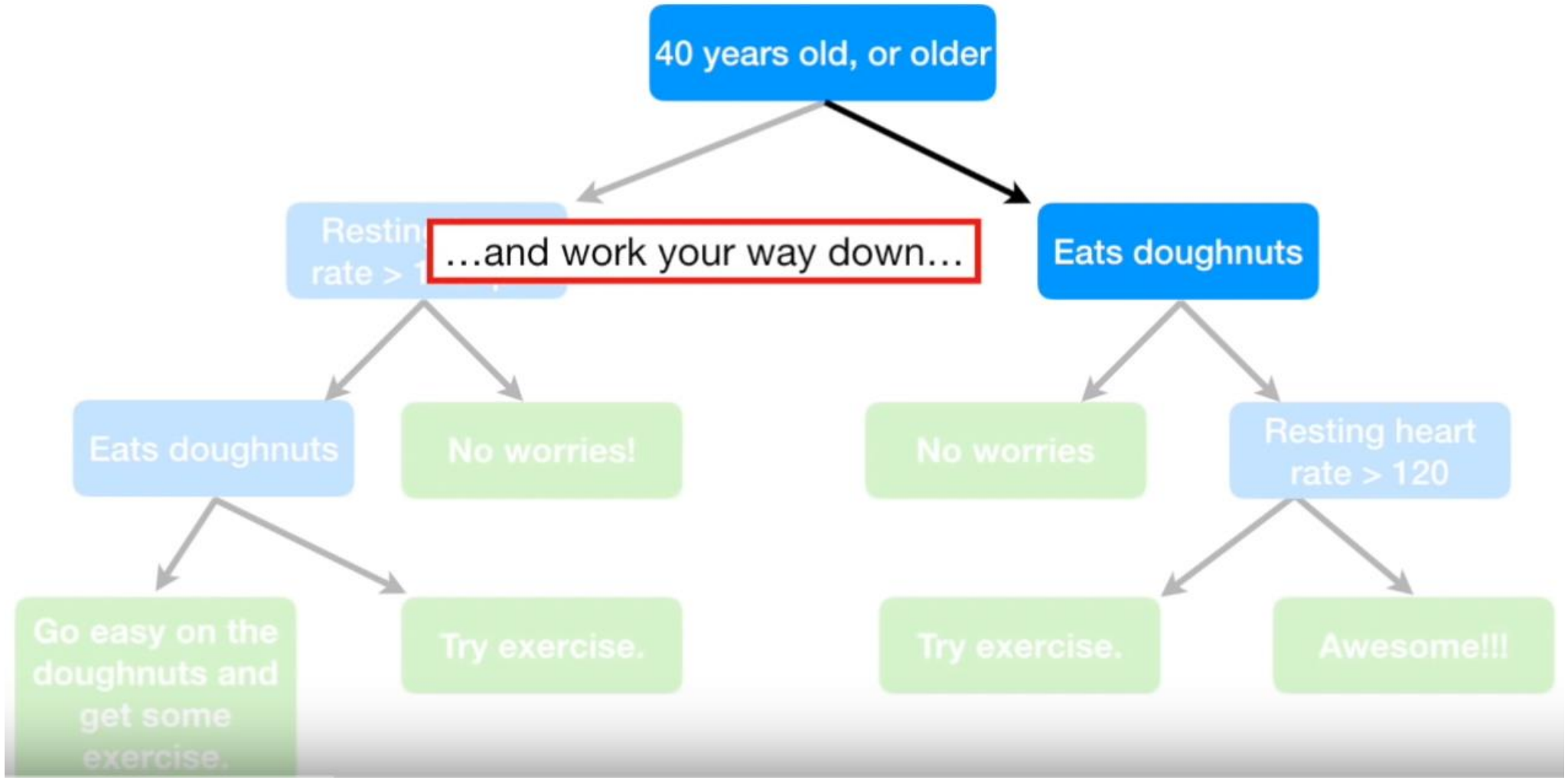
# Building a Decision Tree



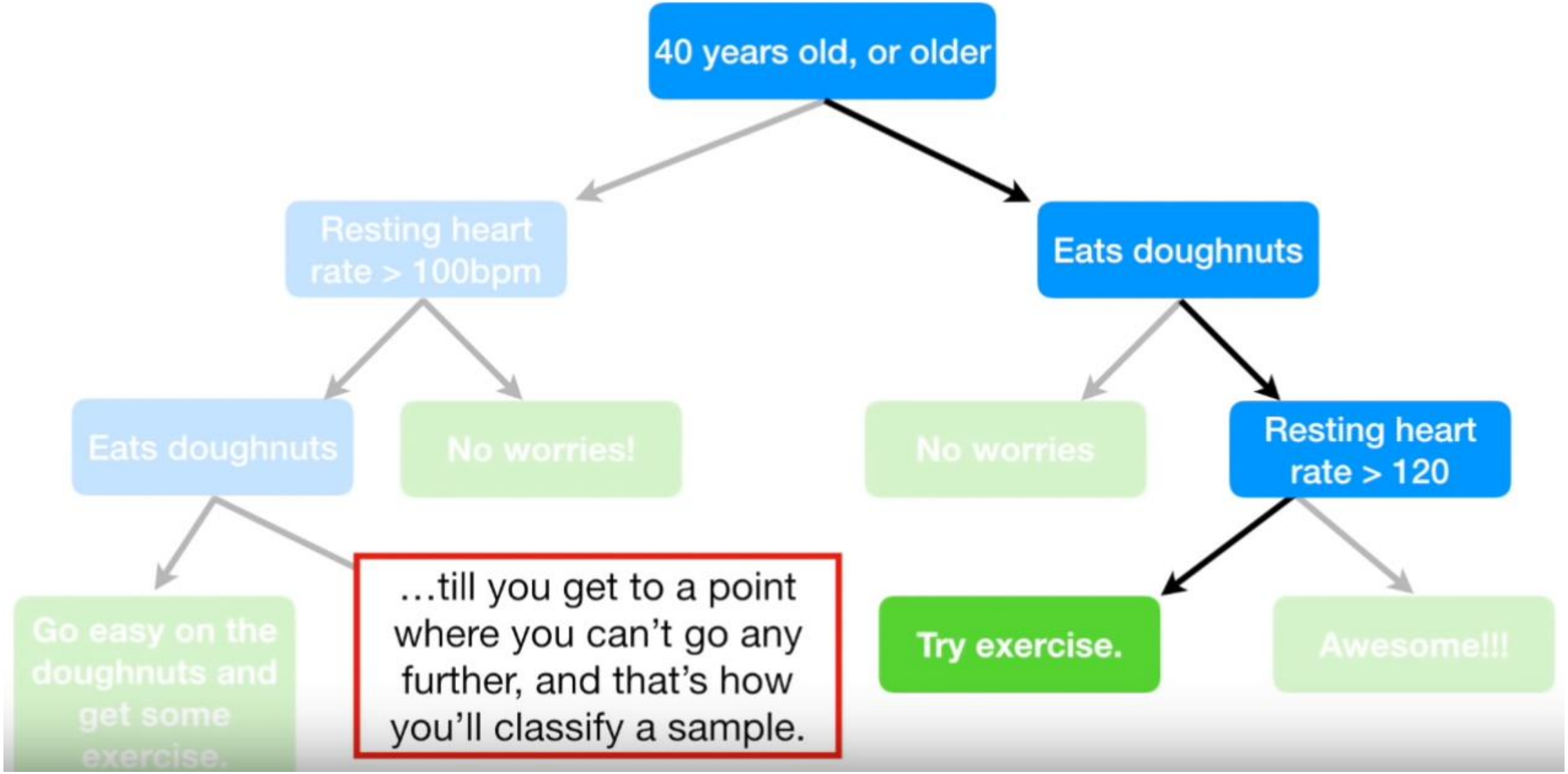
# Building a Decision Tree



# Building a Decision Tree



# Building a Decision Tree



# Building a Decision Tree

**Parent** of a node $c$ is the immediate predecessor node.

**Children** of a node $c$ are the immediate successors of $c$, equivalently nodes which have $c$ as a parent.

**Root node** is the top node of the tree; the only node without parents.

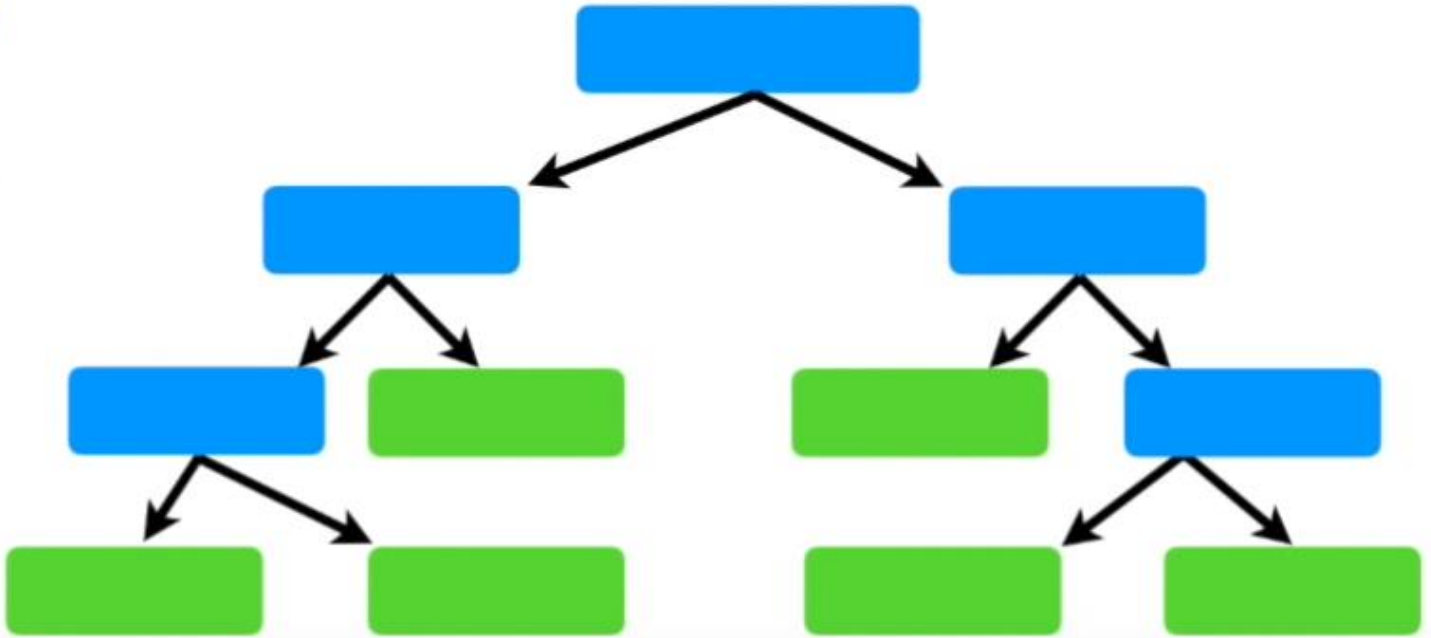
**Leaf nodes** are nodes which do not have children.

**A $K$-ary tree** is a tree where each node (except for leaf nodes) has $K$ children. Usually working with binary trees ($K = 2$).

**Depth** of a tree is the maximal length of a path from the root node to a leaf node.

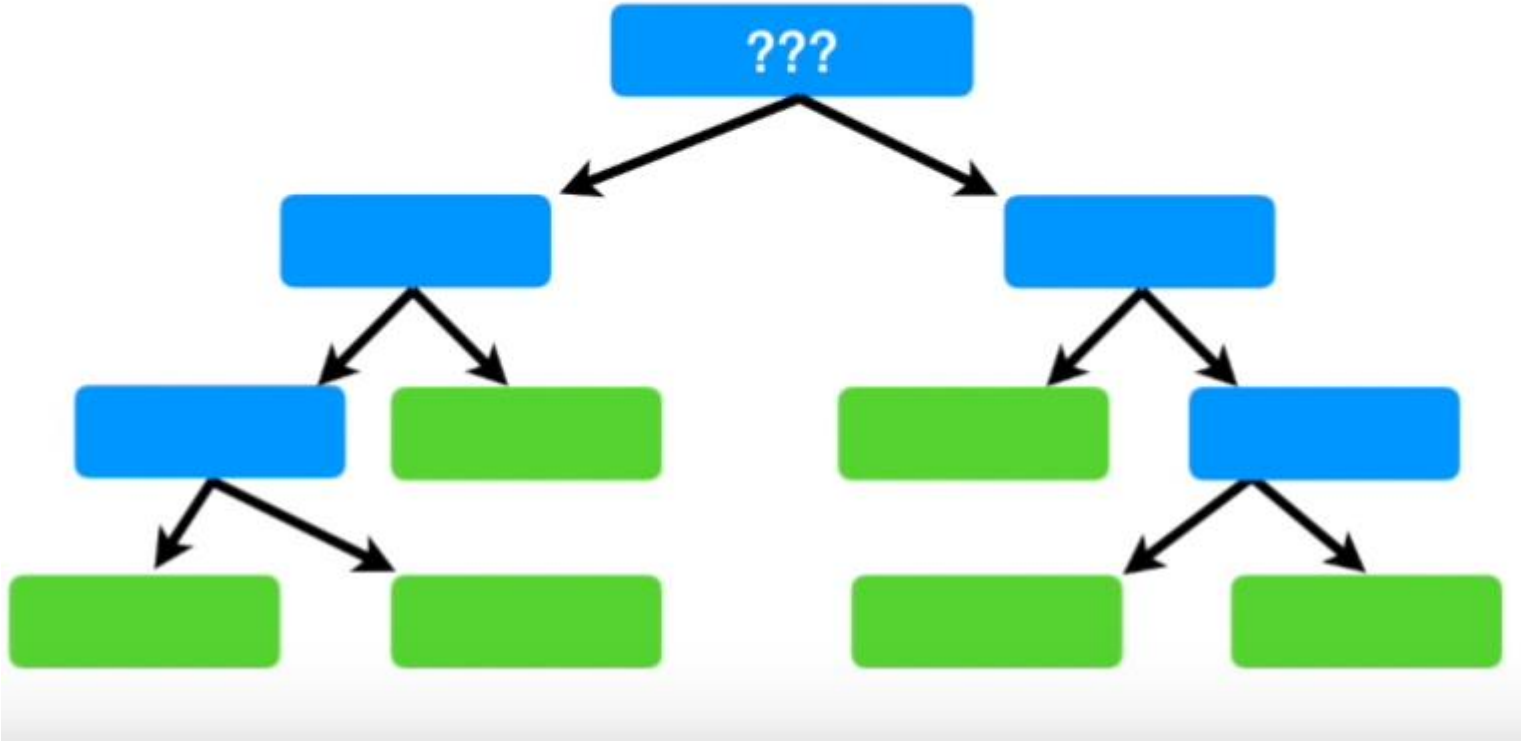
# Building a Decision Tree

| Chest Pain | Good Blood Circulation | Blocked Arteries | Heart Disease |
|---|---|---|---|
| No | No | No | No |
| Yes | Yes | Yes | Yes |
| Yes | Yes | No | No |
| Yes | No | ??? | Yes |

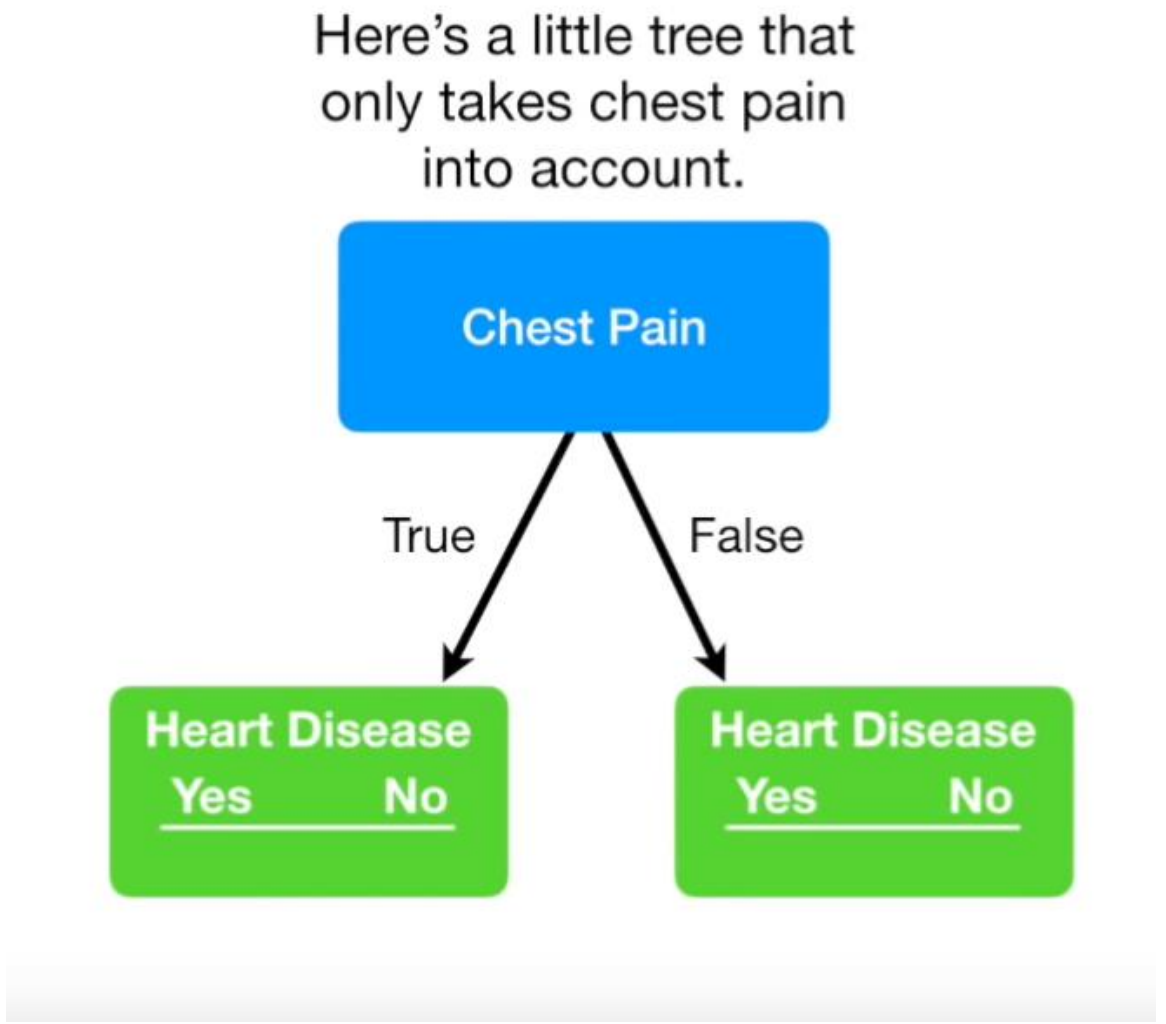


# Building a Decision Tree

The first thing we want to know is whether **Chest Pain**, **Good Blood Circulation** or **Blocked Arteries** should be at the very top of our tree.



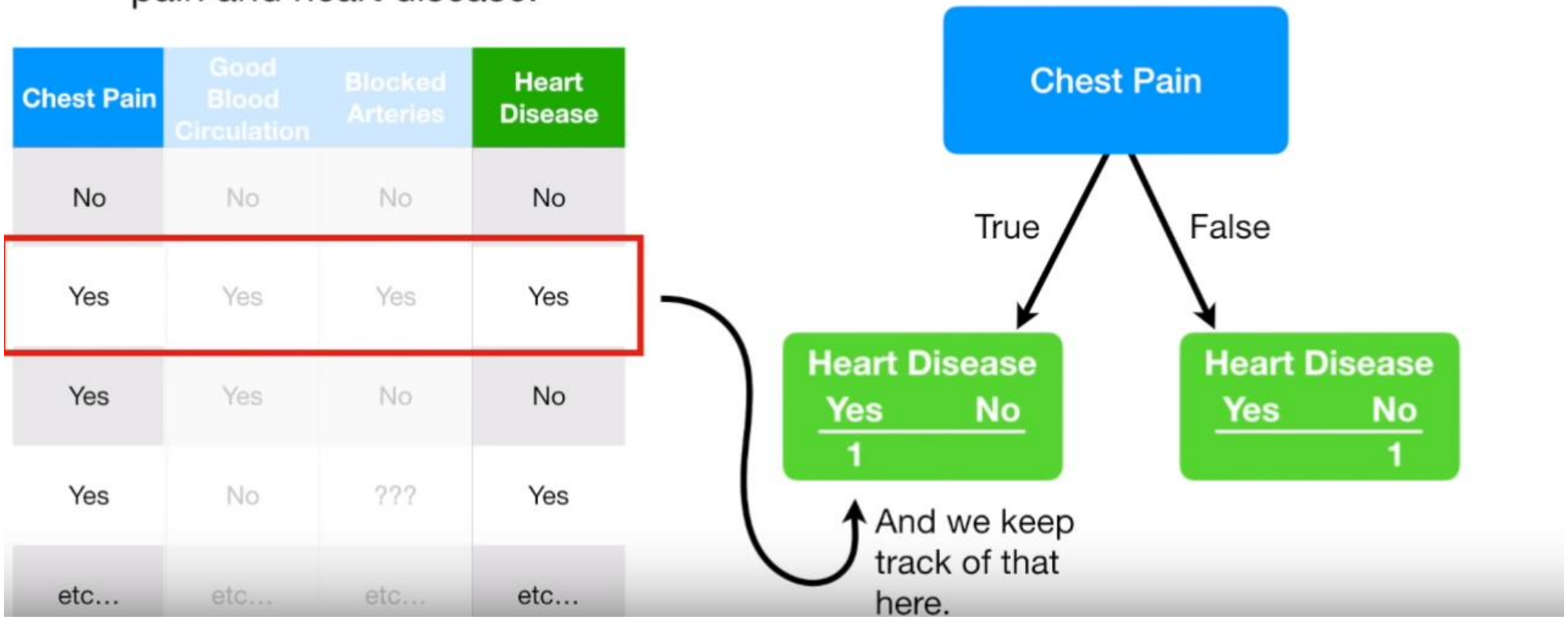
# Building a Decision Tree



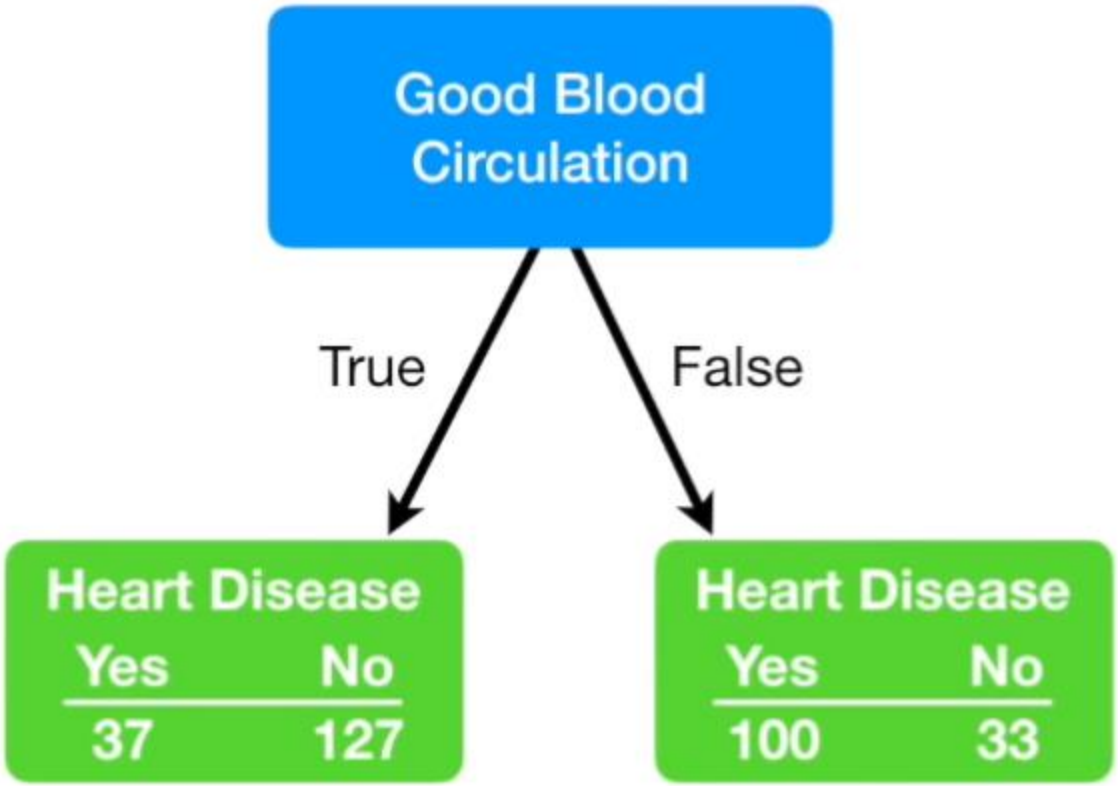
# Building a Decision Tree

The 2nd patient has chest pain and heart disease.

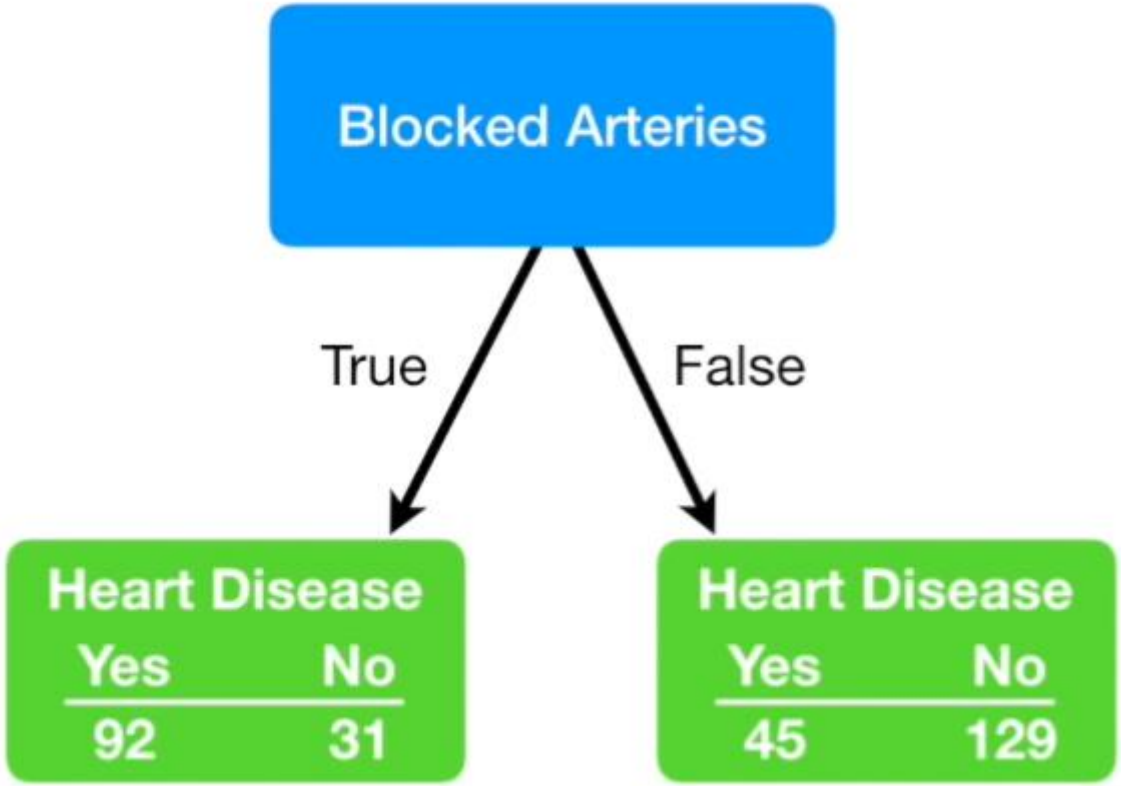
| Chest Pain | Good Blood Circulation | Blocked Arteries | Heart Disease |
|---|---|---|---|
| No | No | No | No |
| Yes | Yes | Yes | Yes |
| Yes | Yes | No | No |
| Yes | No | ??? | Yes |
| etc... | etc... | etc... | etc... |



# Building a Decision Tree

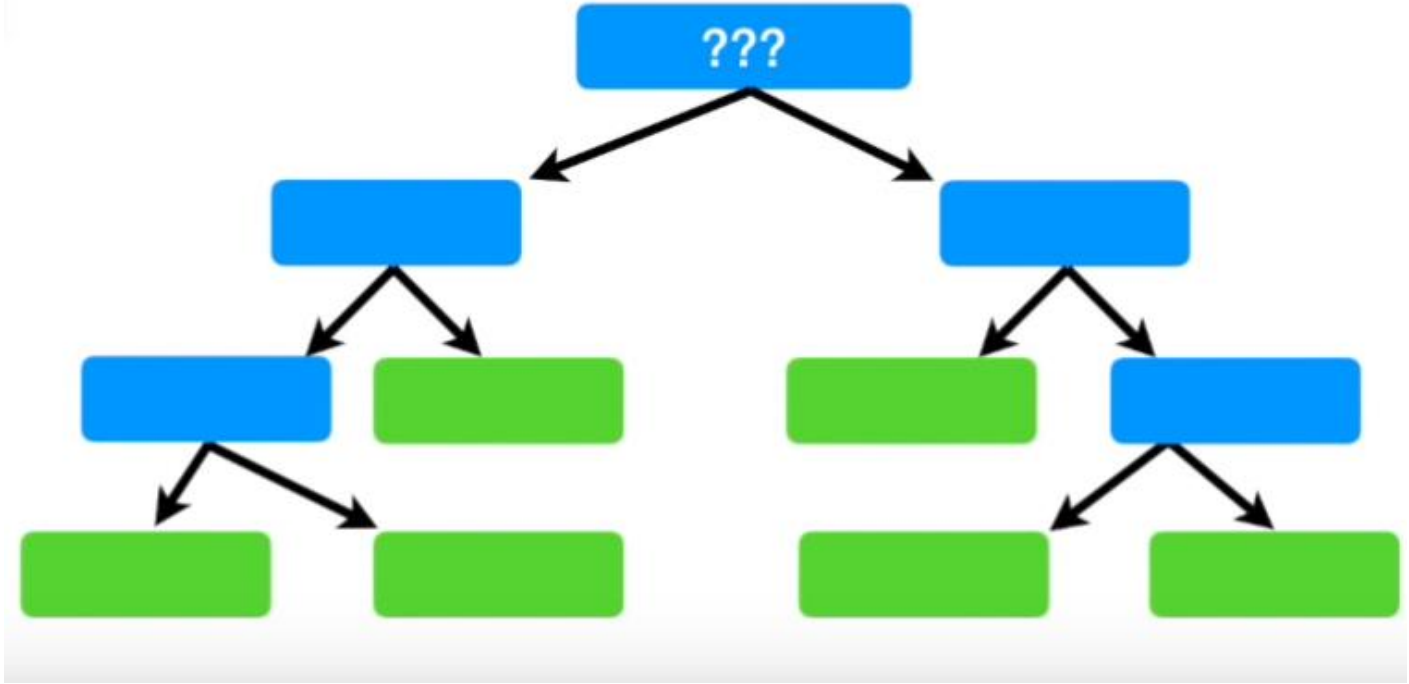


# Building a Decision Tree

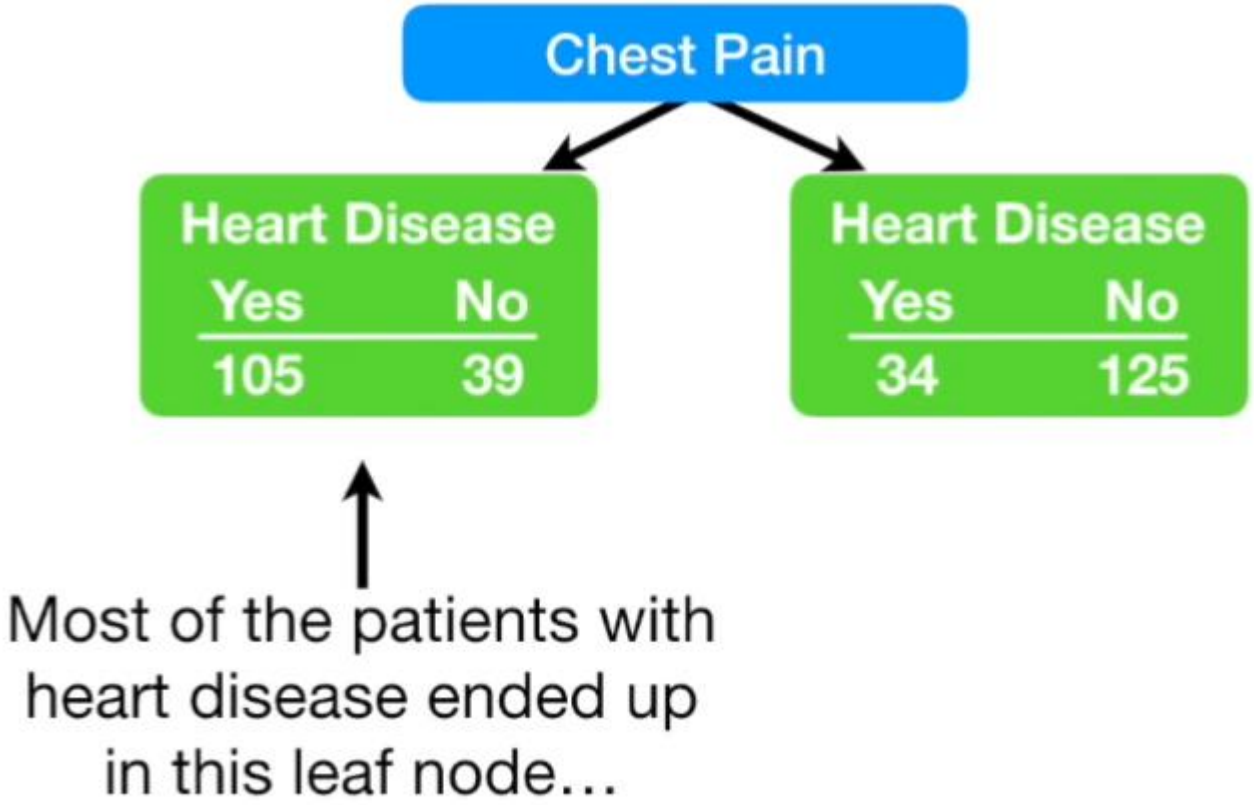


# Building a Decision Tree

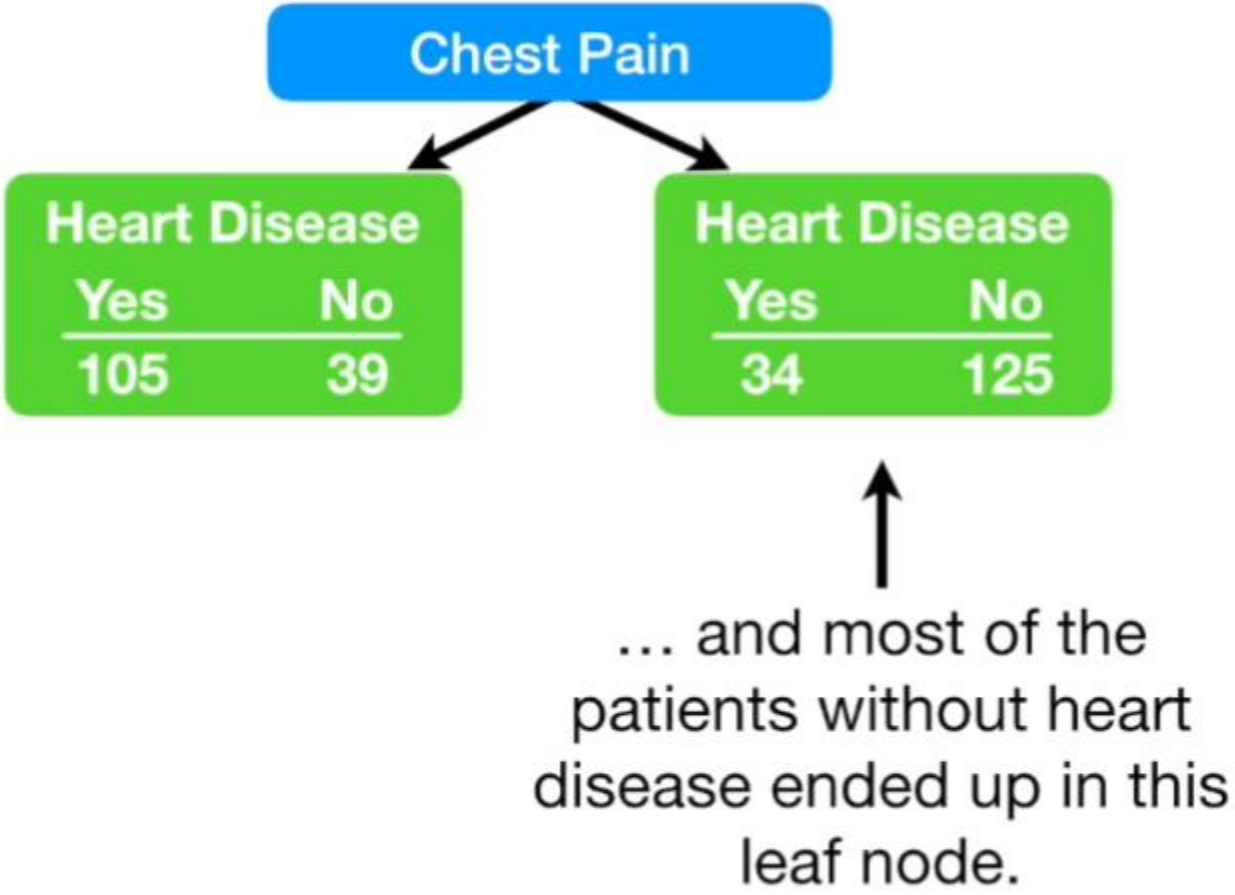
Remember the goal is to decide whether **Chest Pain**, **Good Blood Circulation** or **Blocked Arteries** should be the first thing in our decision tree (aka **The Root Node**).



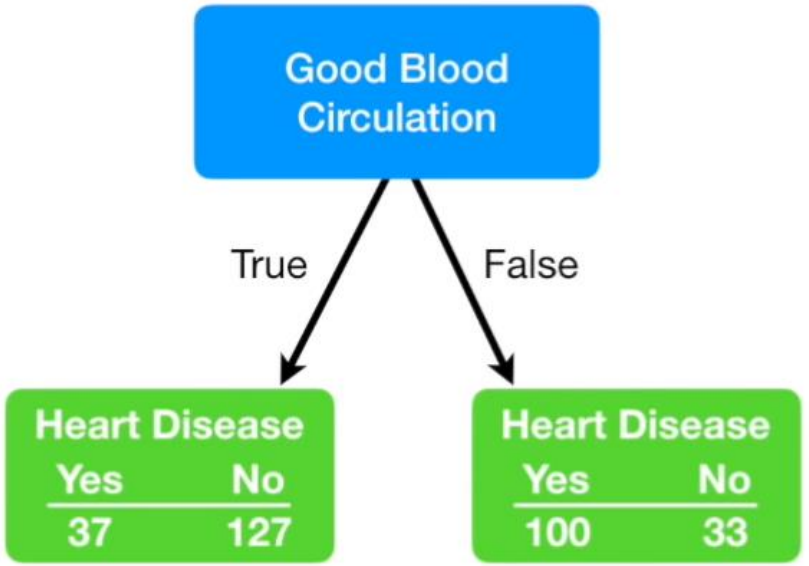
# Building a Decision Tree

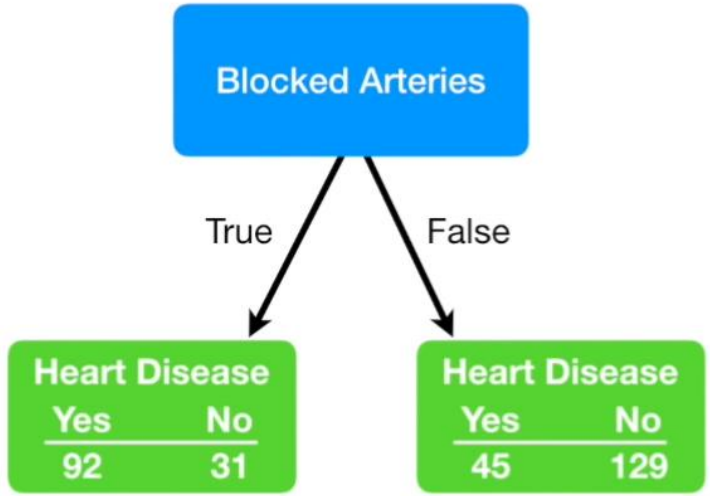


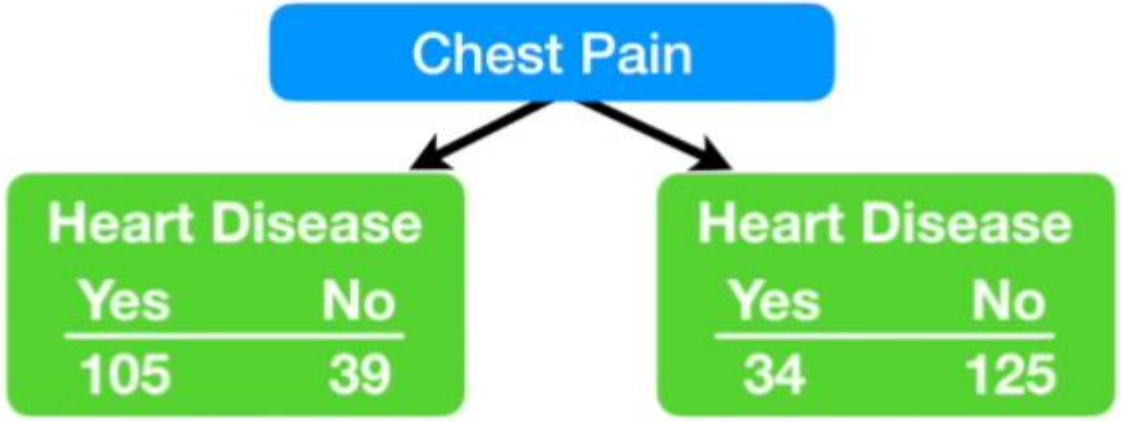
# Building a Decision Tree



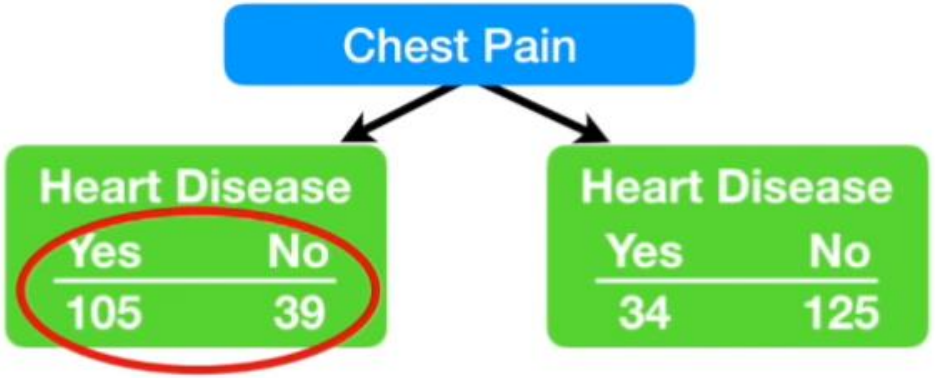
# Building a Decision Tree



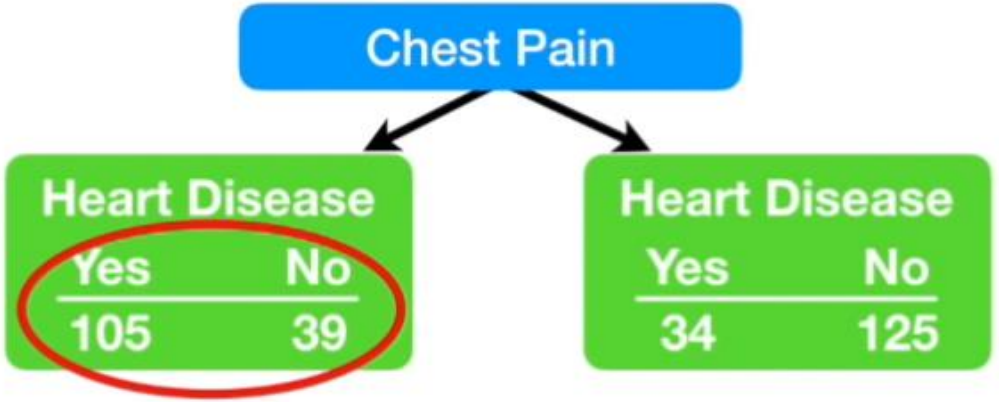
# Building a Decision Tree




For this leaf, the Gini impurity = 1 - (the probability of "yes")$^2$ - (the probability of "no")$^2$

$$= 1 - \left(\frac{105}{105 + 39}\right)^2$$

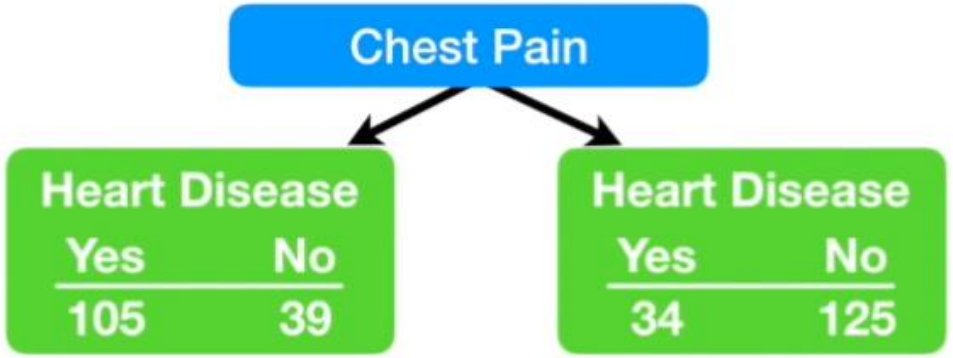
# Building a Decision Tree




For this leaf, the Gini impurity = 1 - (the probability of "yes")$^2$ - (the probability of "no")$^2$

$$= 1 - \left(\frac{105}{105 + 39}\right)^2 - \left(\frac{39}{105 + 39}\right)^2$$

# Building a Decision Tree

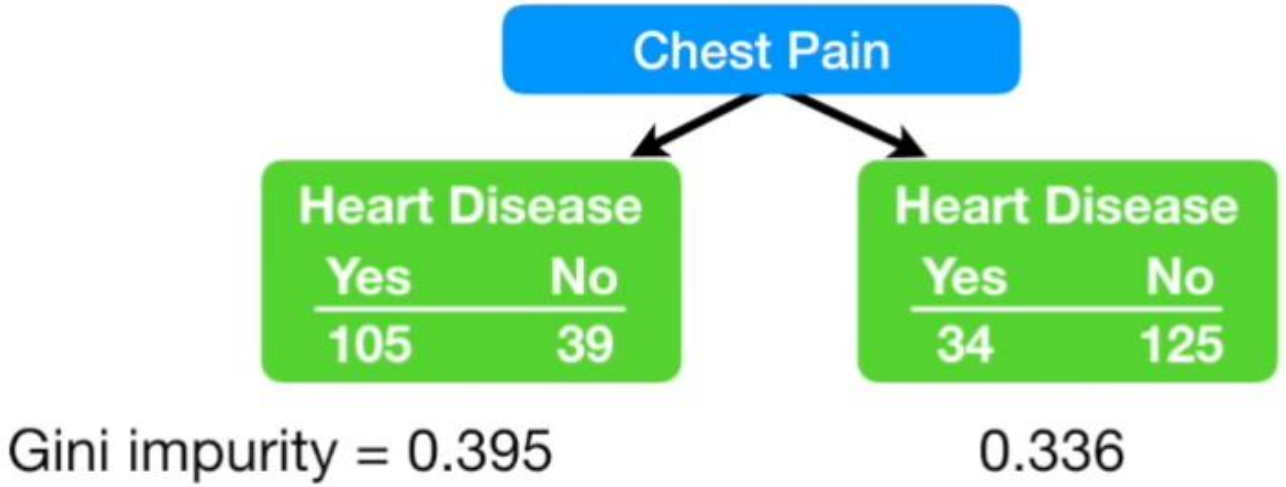



For this leaf, the Gini impurity = 1 - (the probability of "yes")$^2$ - (the probability of "no")$^2$

$$= 1 - \left(\frac{105}{105 + 39}\right)^2 - \left(\frac{39}{105 + 39}\right)^2$$

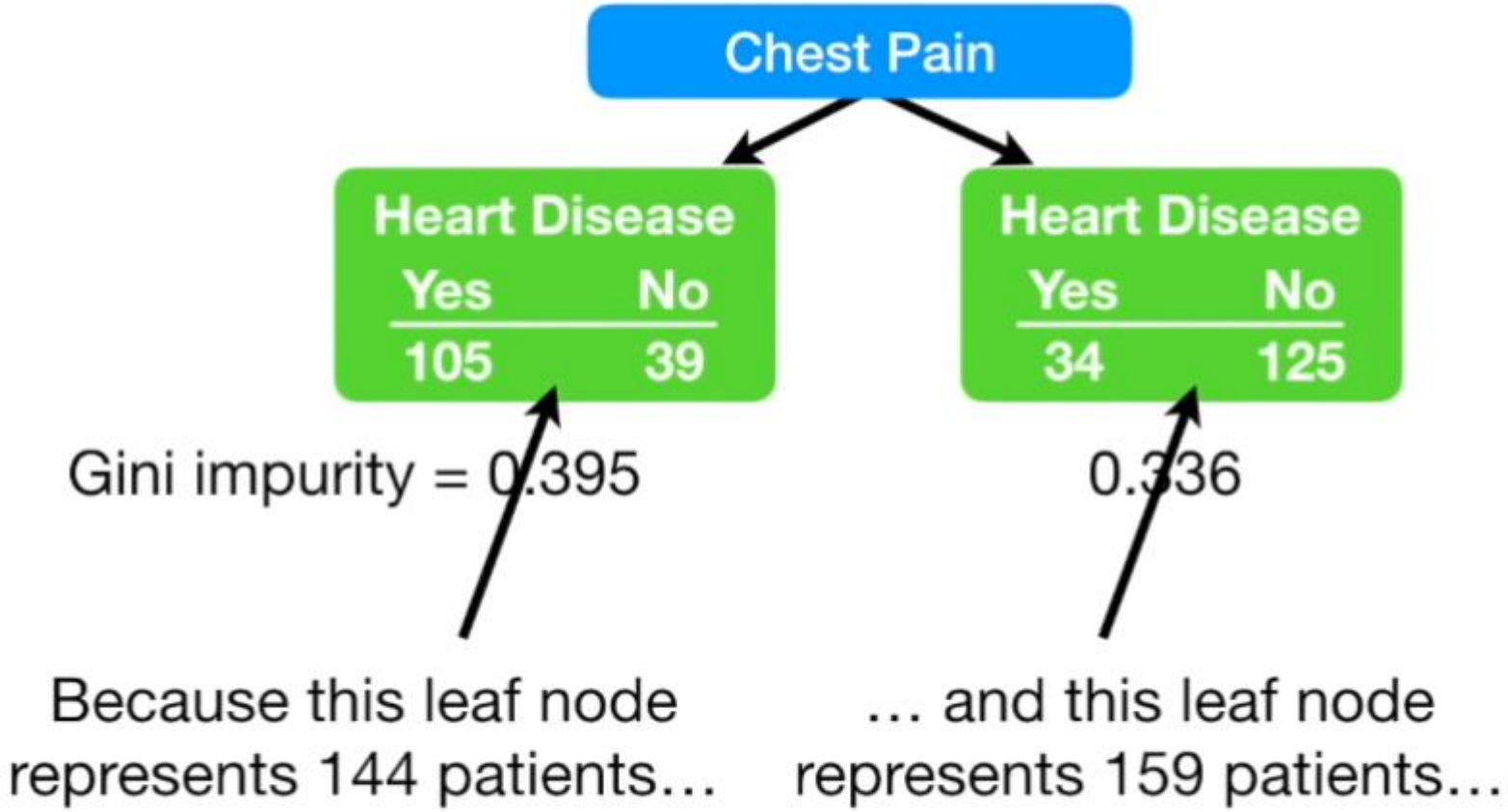
$$= 0.395$$

# Building a Decision Tree

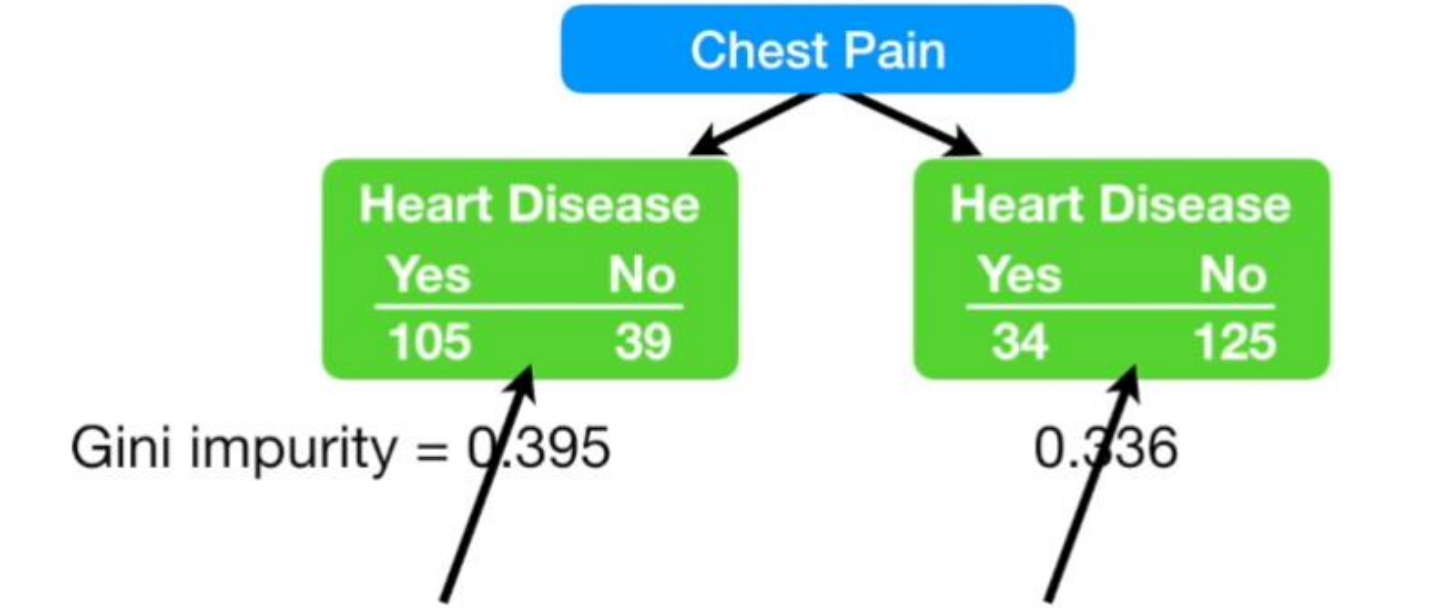



Now that we have measured the Gini impurity for both leaf nodes, we can calculate the total Gini impurity for using Chest Pain to separate patients with and without heart disease.

# Building a Decision Tree

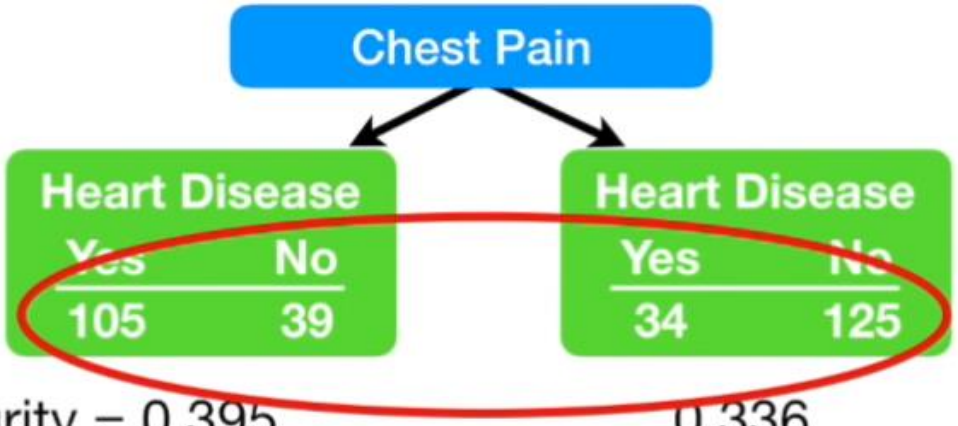


# Building a Decision Tree




Thus, the total Gini impurity for using Chest Pain to separate patients with and without heart disease is the **weighted average of the leaf node impurities**.

# Building a Decision Tree

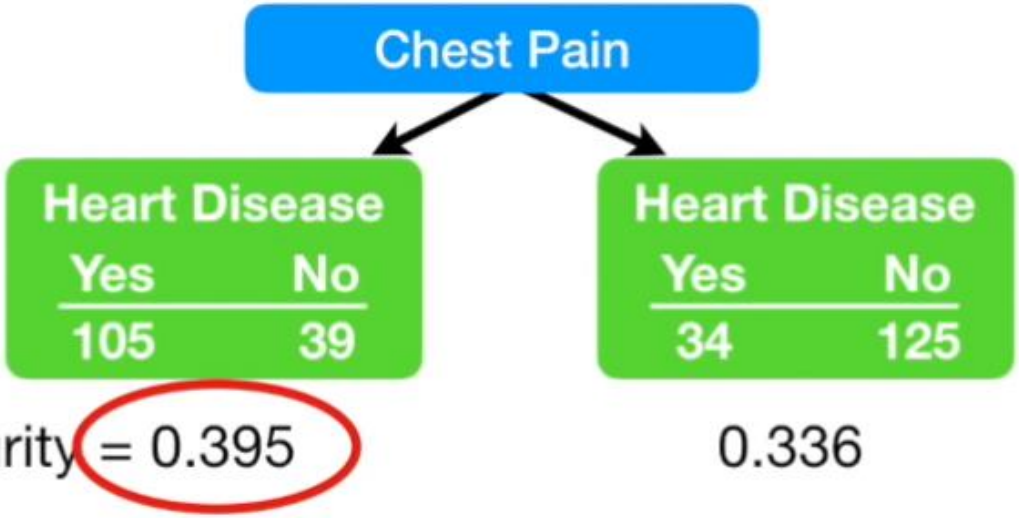



Gini impurity = 0.395 0.336

Gini impurity for Chest Pain = weighted average of Gini impurities for the leaf nodes

$$= \left(\frac{144}{144 + 159}\right) 0.395$$

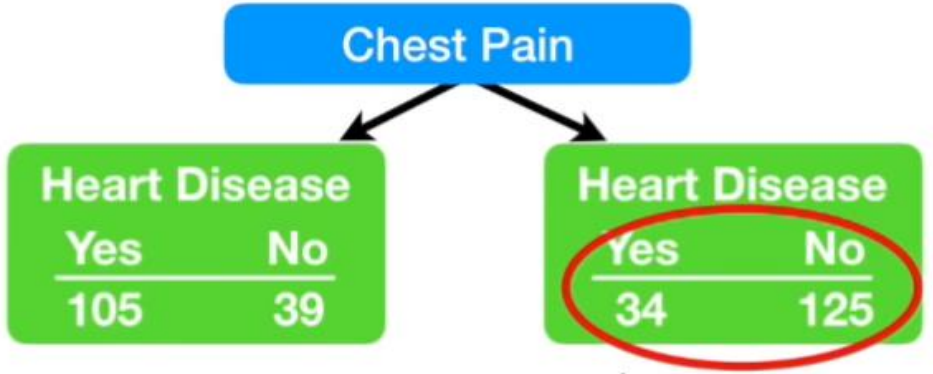
# Building a Decision Tree




Gini impurity for Chest Pain = weighted average of Gini impurities for the leaf nodes

$$= \left(\frac{144}{144 + 159}\right) 0.395$$

# Building a Decision Tree



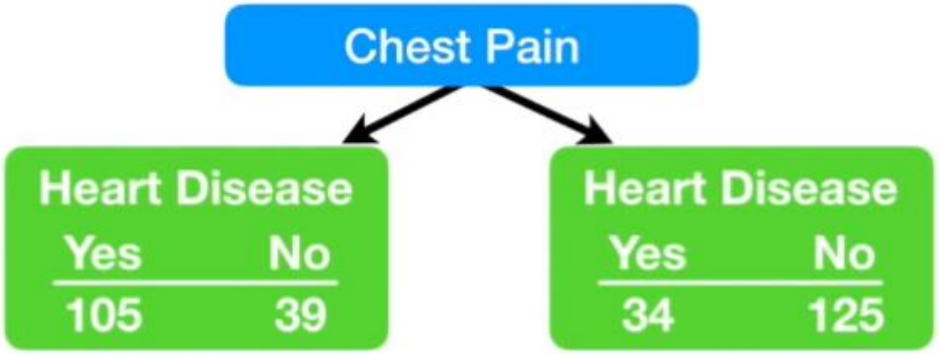

Gini impurity = 0.395

0.336

Gini impurity for Chest Pain = weighted average of Gini impurities for the leaf nodes

$$= \left(\frac{144}{144 + 159}\right) 0.395 + \left(\frac{159}{144 + 159}\right) 0.336$$

# Building a Decision Tree




Gini impurity = 0.395 0.336

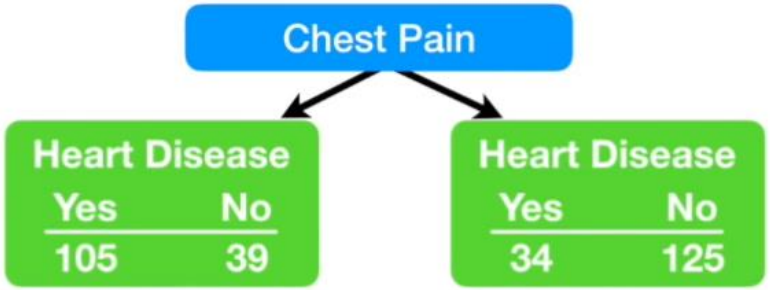
Gini impurity for Chest Pain = weighted average of Gini impurities for the leaf nodes

$$= \left(\frac{144}{144 + 159}\right) 0.395 + \left(\frac{159}{144 + 159}\right) 0.336$$

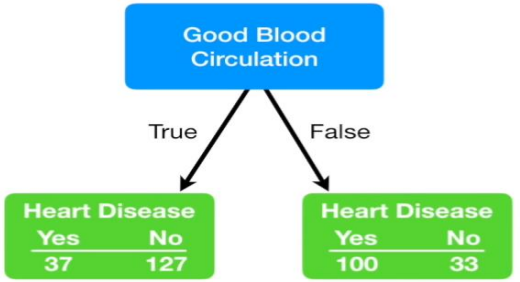
$$= 0.364$$

# Building a Decision Tree

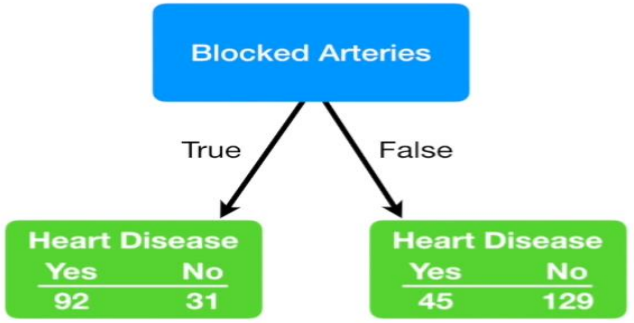
Gini impurity for Chest Pain = 0.364




Gini impurity for Good Blood Circulation = 0.360




Gini impurity for Blocked Arteries = 0.381



# Building a Decision Tree

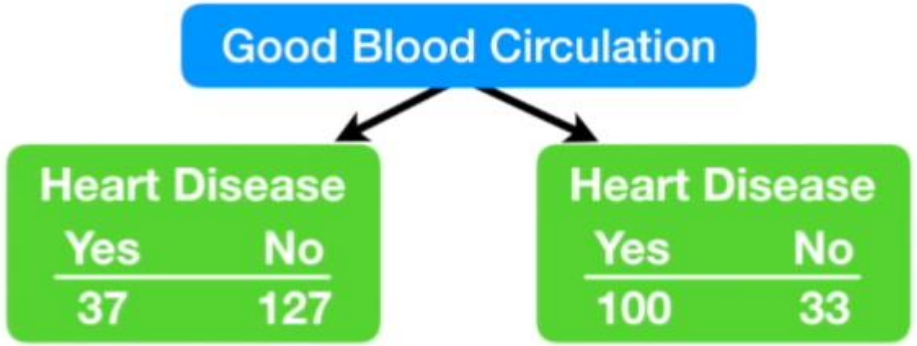
Gini impurity for Chest Pain = 0.364

Gini impurity for Good Blood Circulation = 0.360

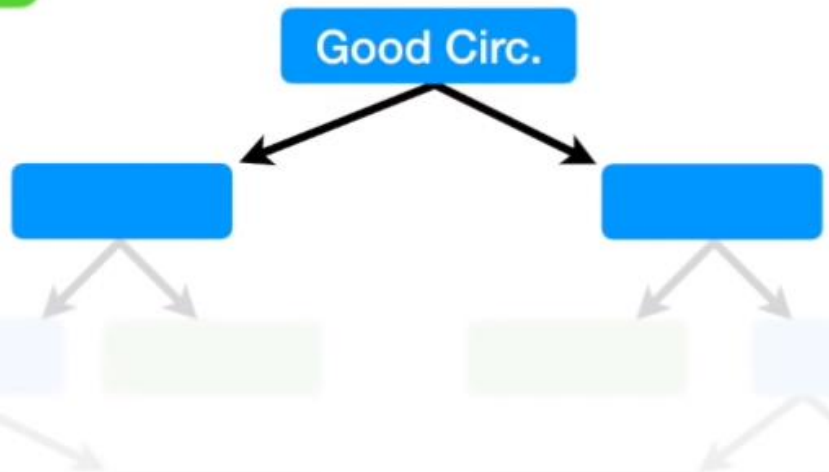
**Good Blood Circulation** has the lowest impurity (it separates patients with and without heart disease the best)…

Gini impurity for Blocked Arteries = 0.381

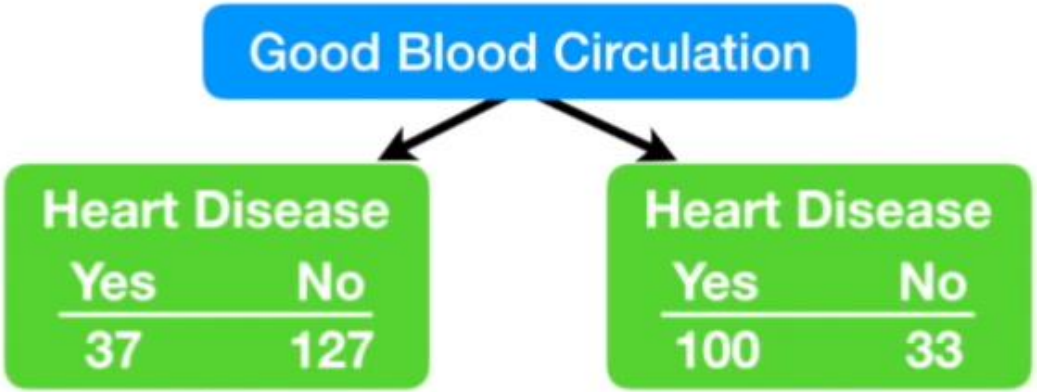
# Building a Decision Tree



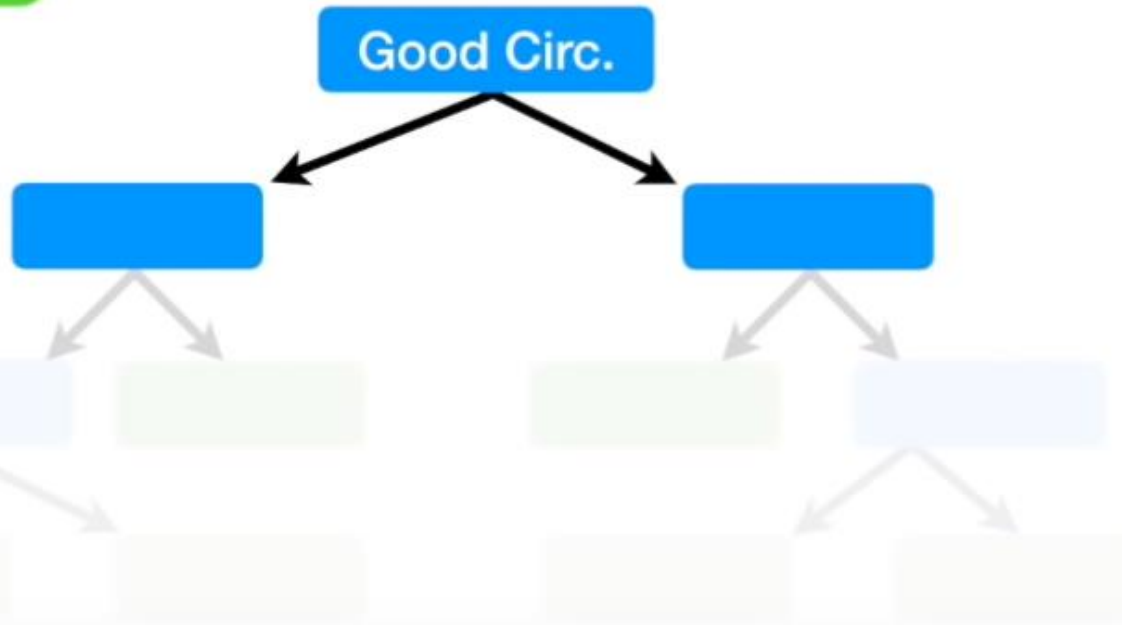

When we divided all of the patients using **Good Blood Circulation**, we ended up with "impure" leaf nodes.



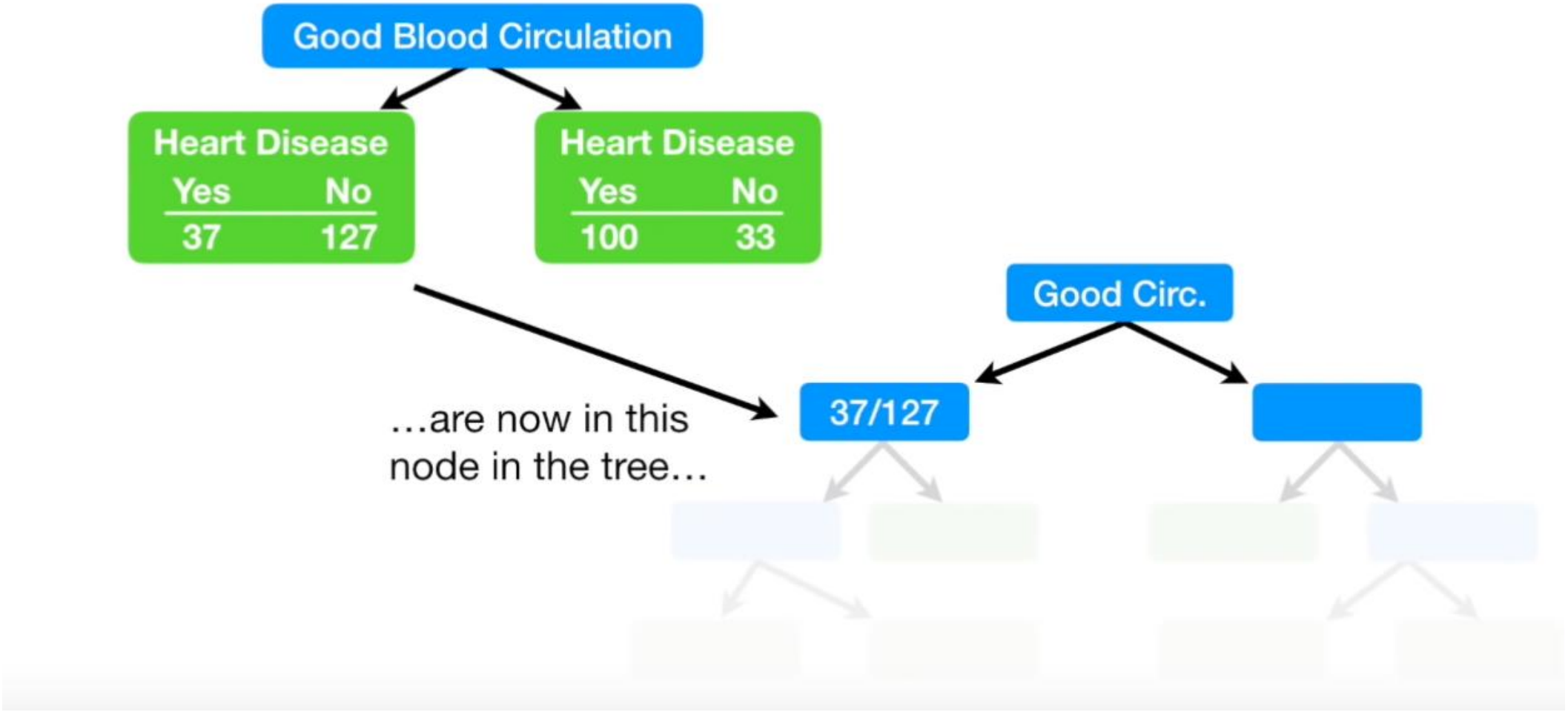
# Building a Decision Tree



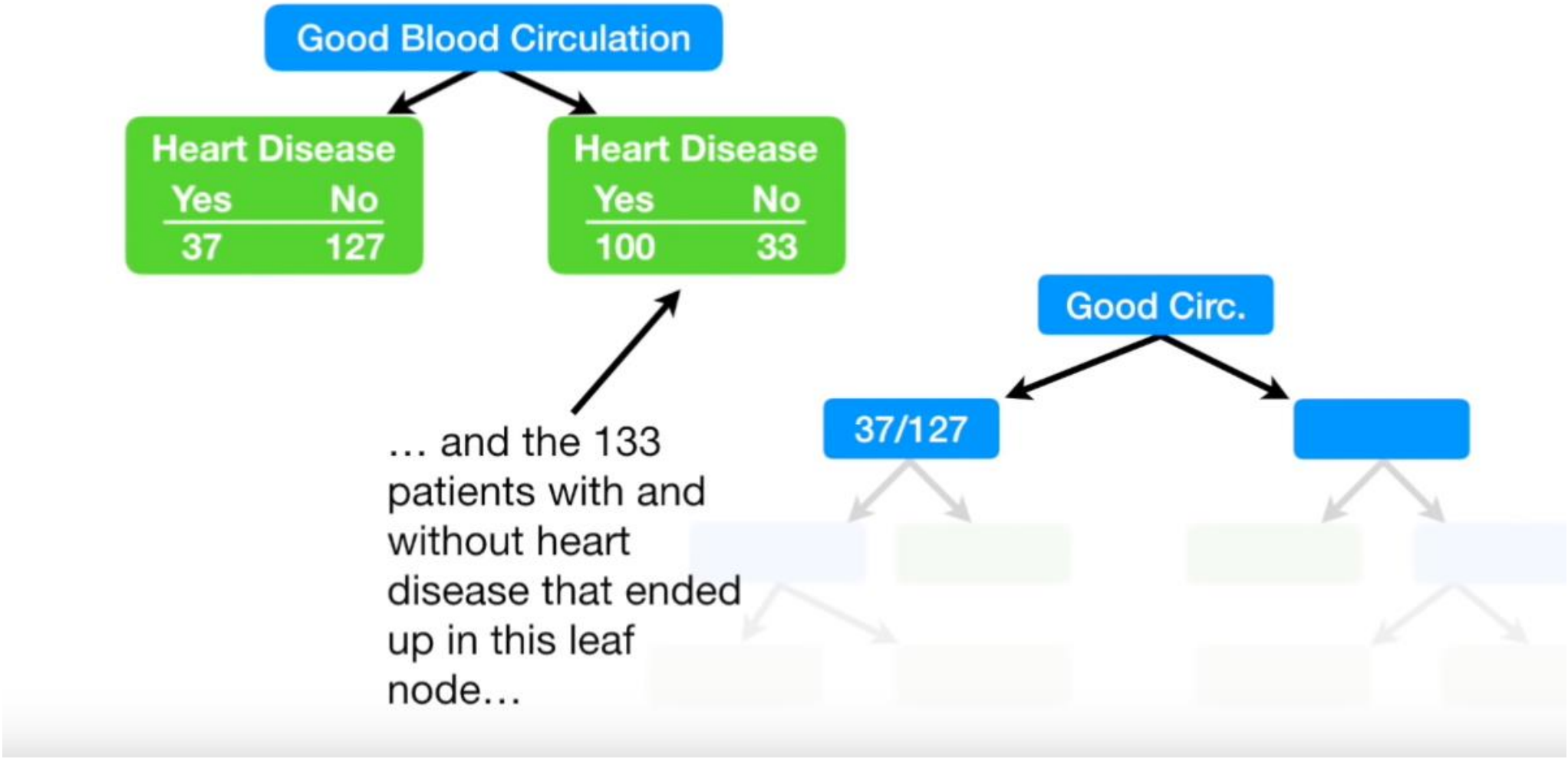

That means the 164 patients with and without heart disease that ended up in this leaf node…



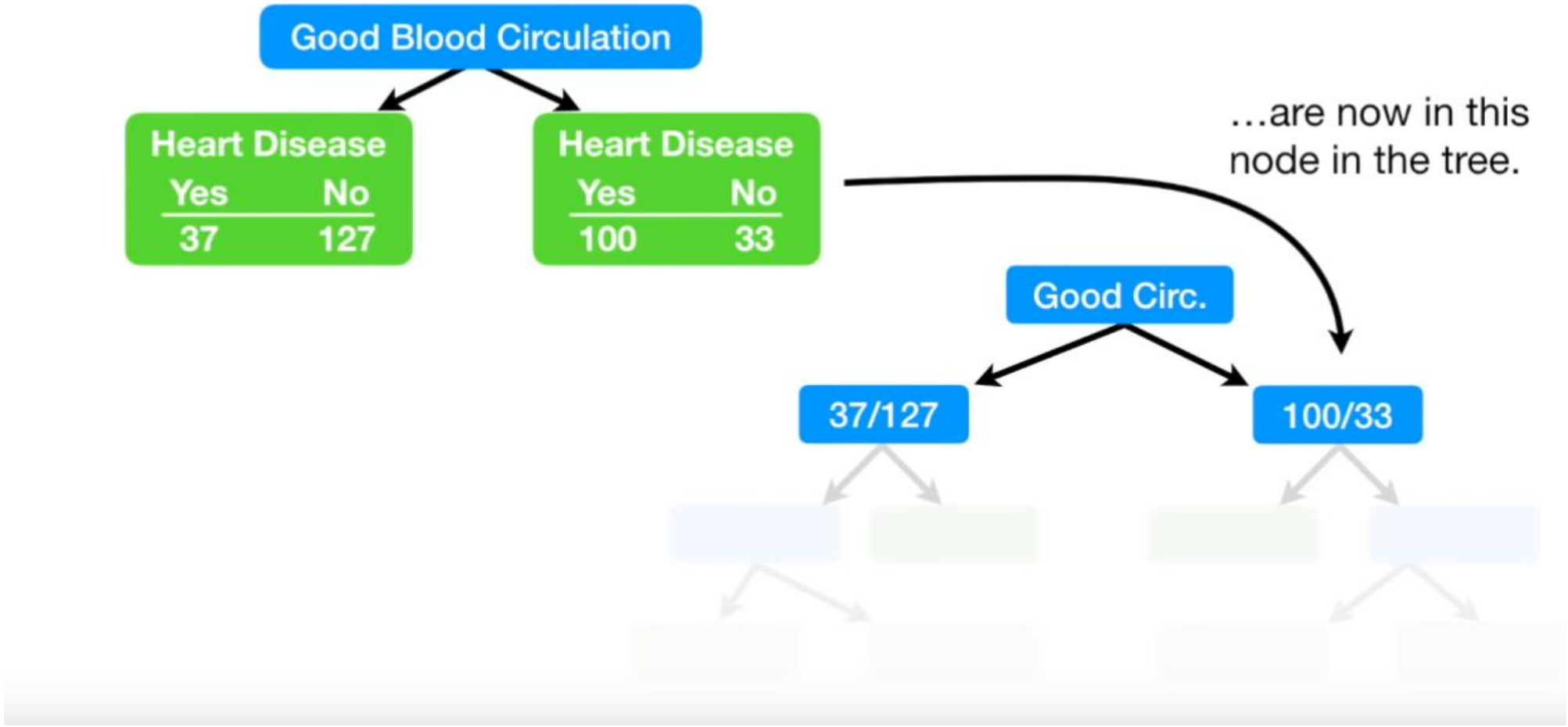
# Building a Decision Tree



# Building a Decision Tree

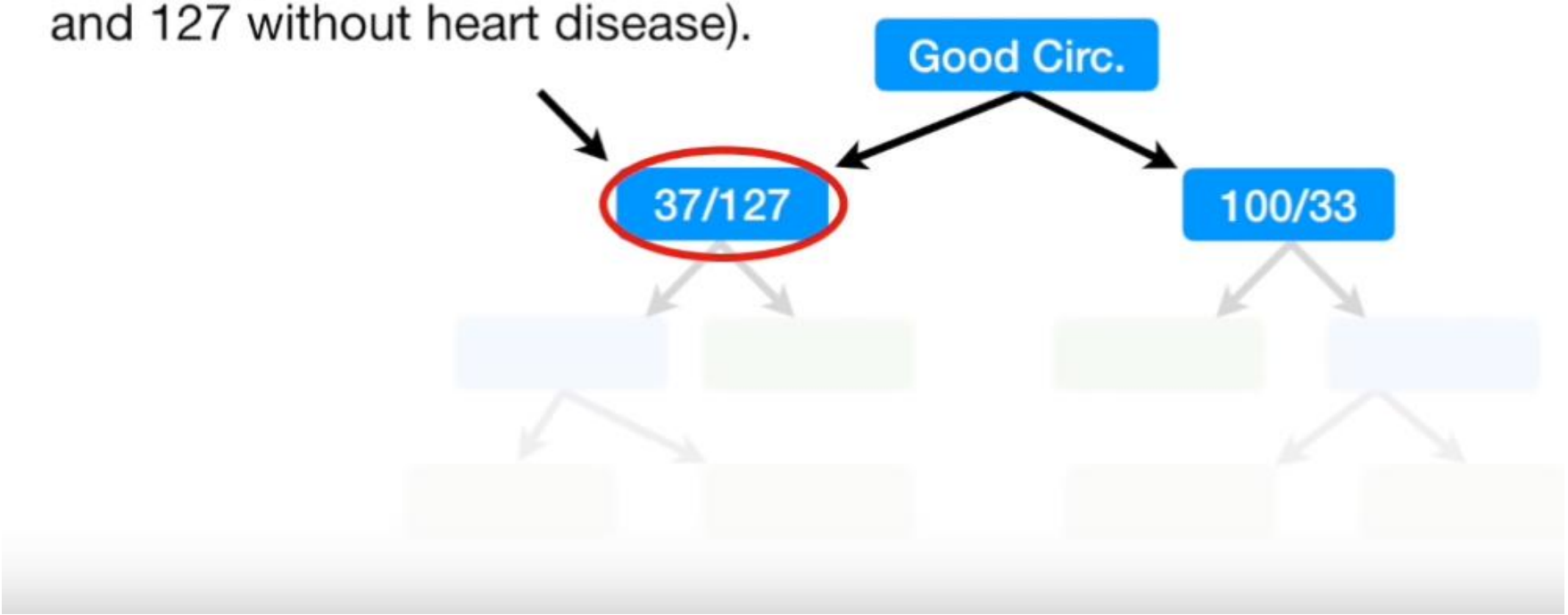


# Building a Decision Tree

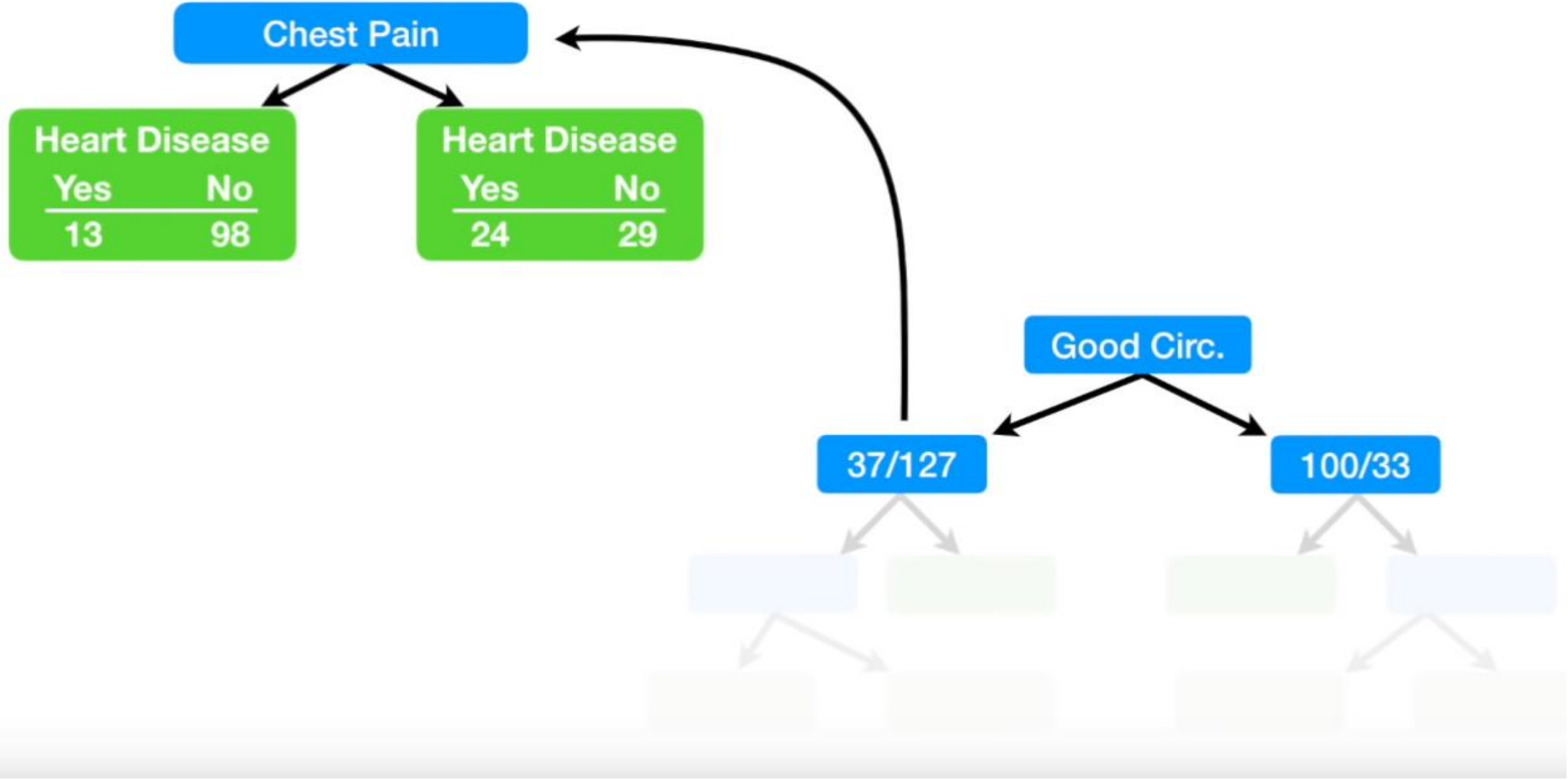


# Building a Decision Tree

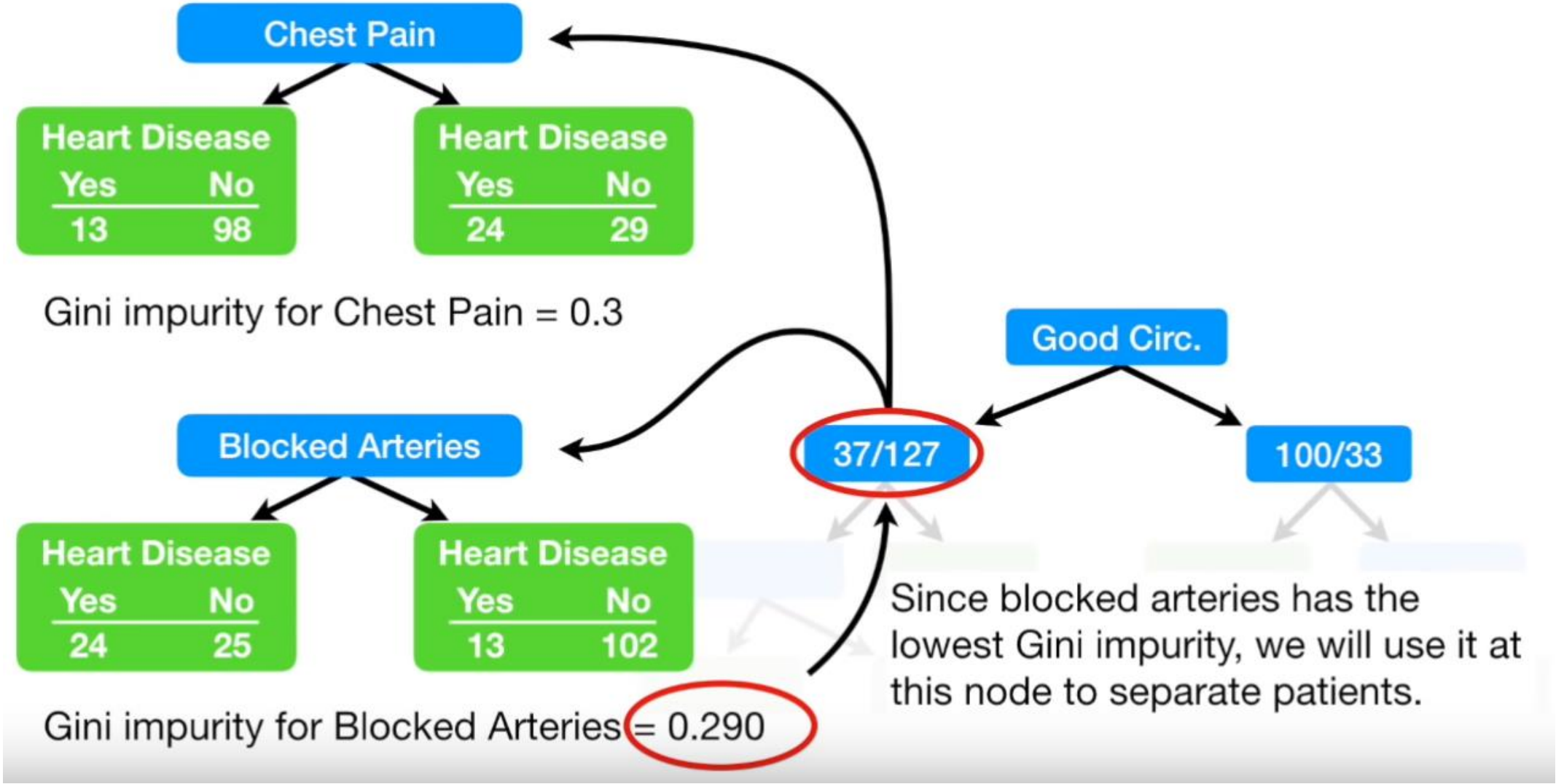
Now we need to figure how well **chest pain** and **blocked arteries** separate these 164 patients (37 with heart disease and 127 without heart disease).



# Building a Decision Tree

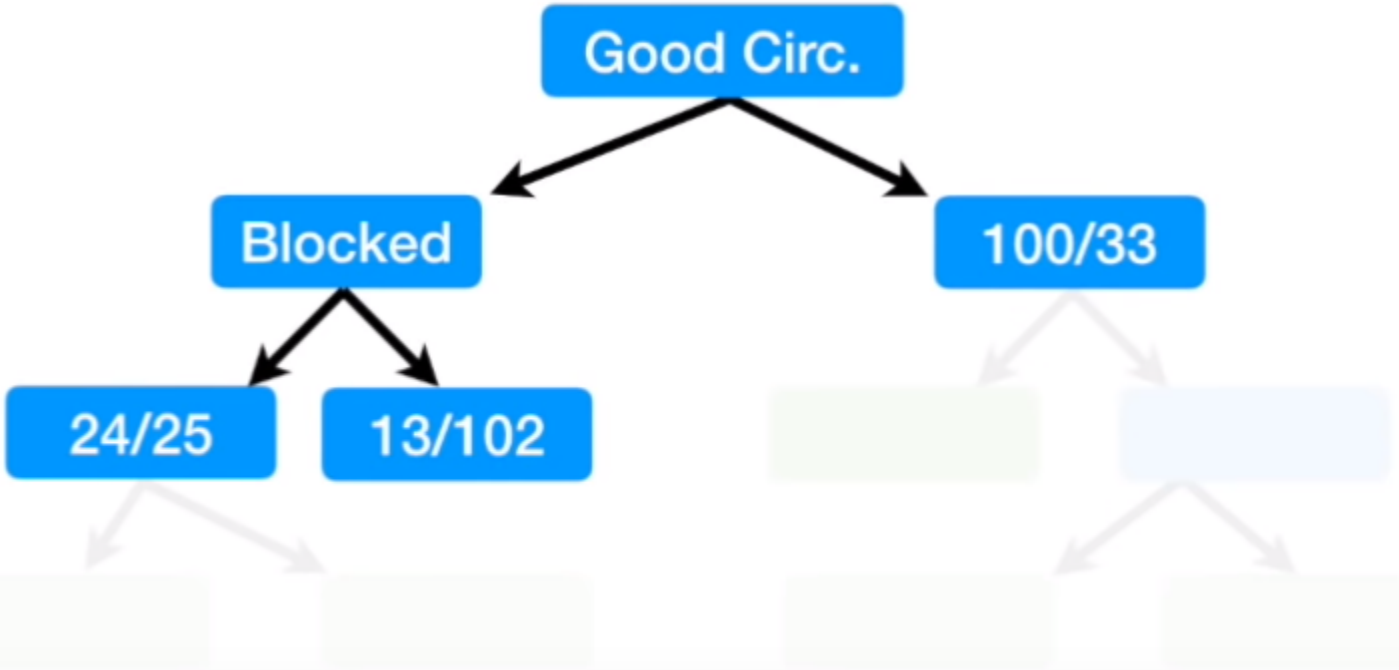


# Building a Decision Tree



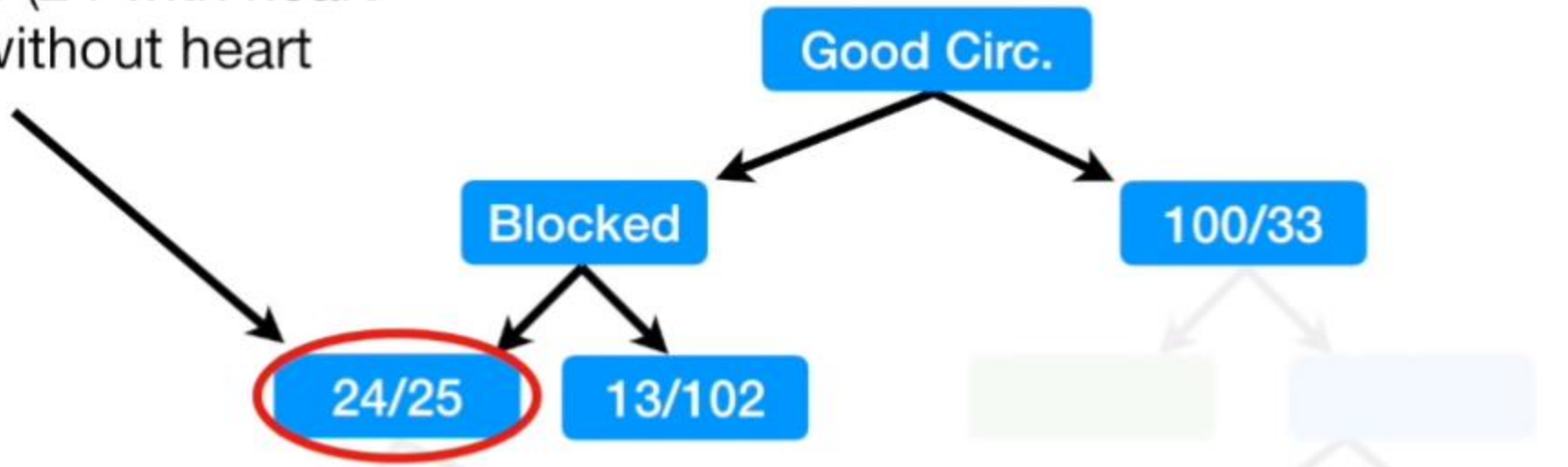
# Building a Decision Tree

Here's the tree that we've worked out so far.

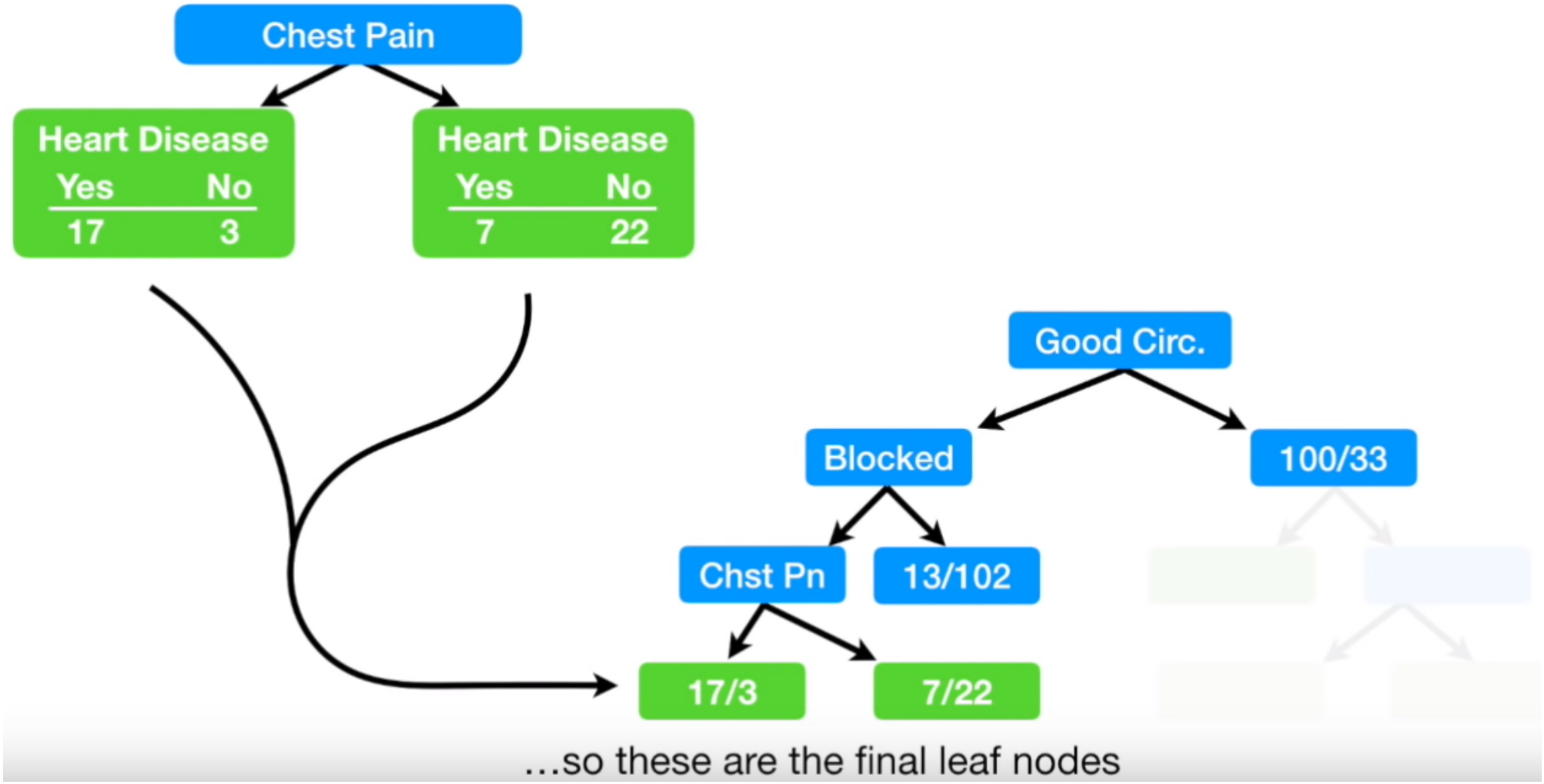


# Building a Decision Tree

All we have left is Chest Pain, so first we'll see how well it separates these 49 patients (24 with heart disease and 25 without heart disease).

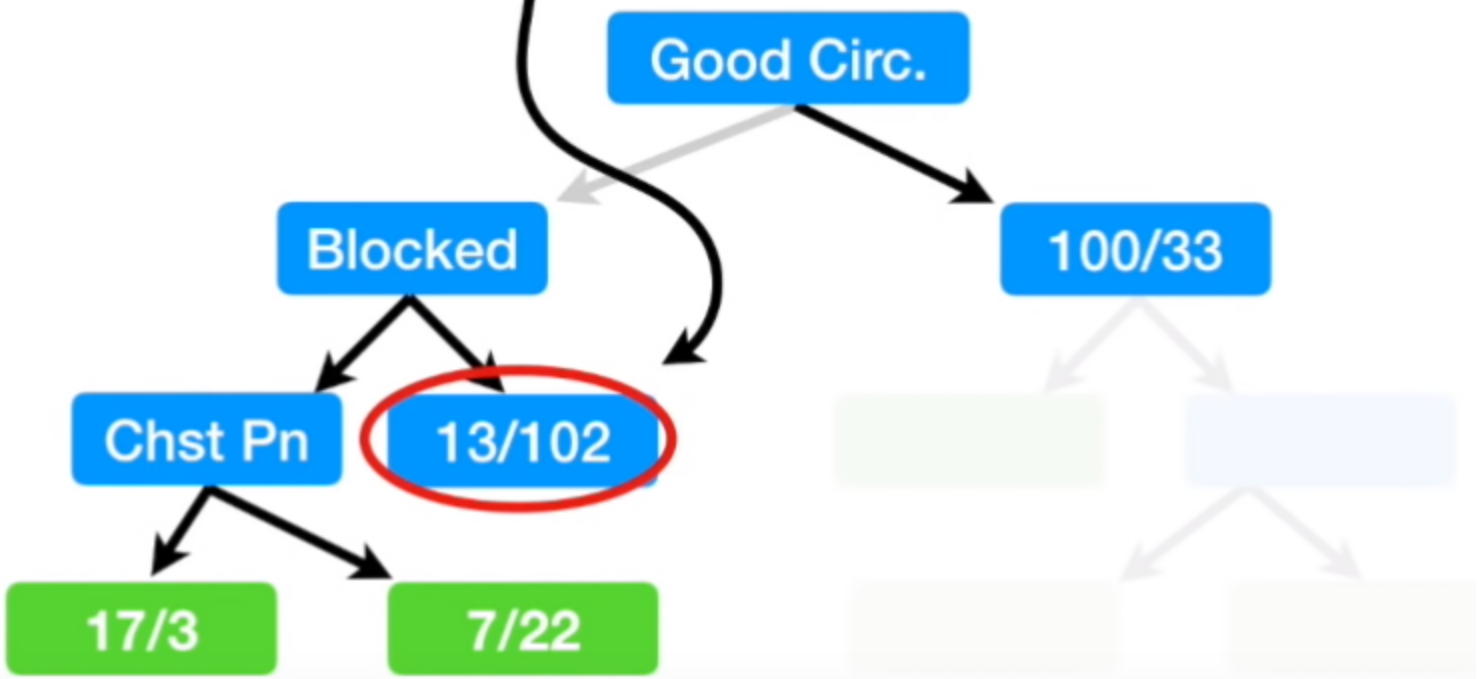


# Building a Decision Tree

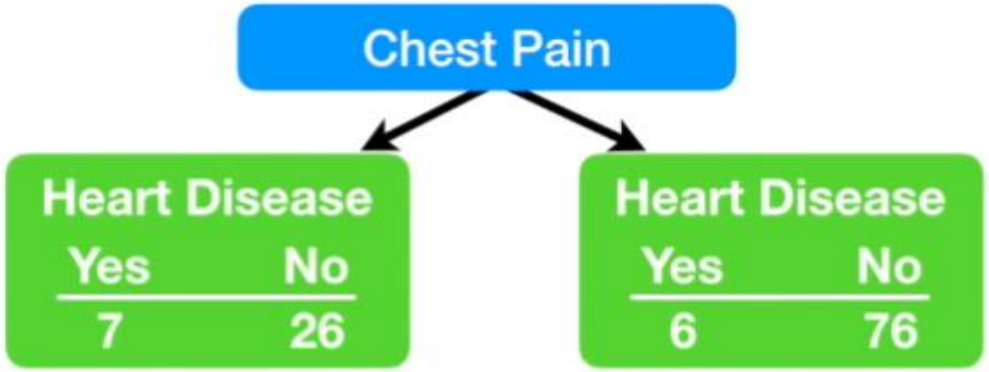


# Building a Decision Tree

Now let's see what happens when we use chest pain to divide these 115 patients (13 with heart disease and 102 without).



# Building a Decision Tree




Gini impurity for Chest Pain = 0.29

The Gini impurity for this node, before using chest pain to separate patients is…

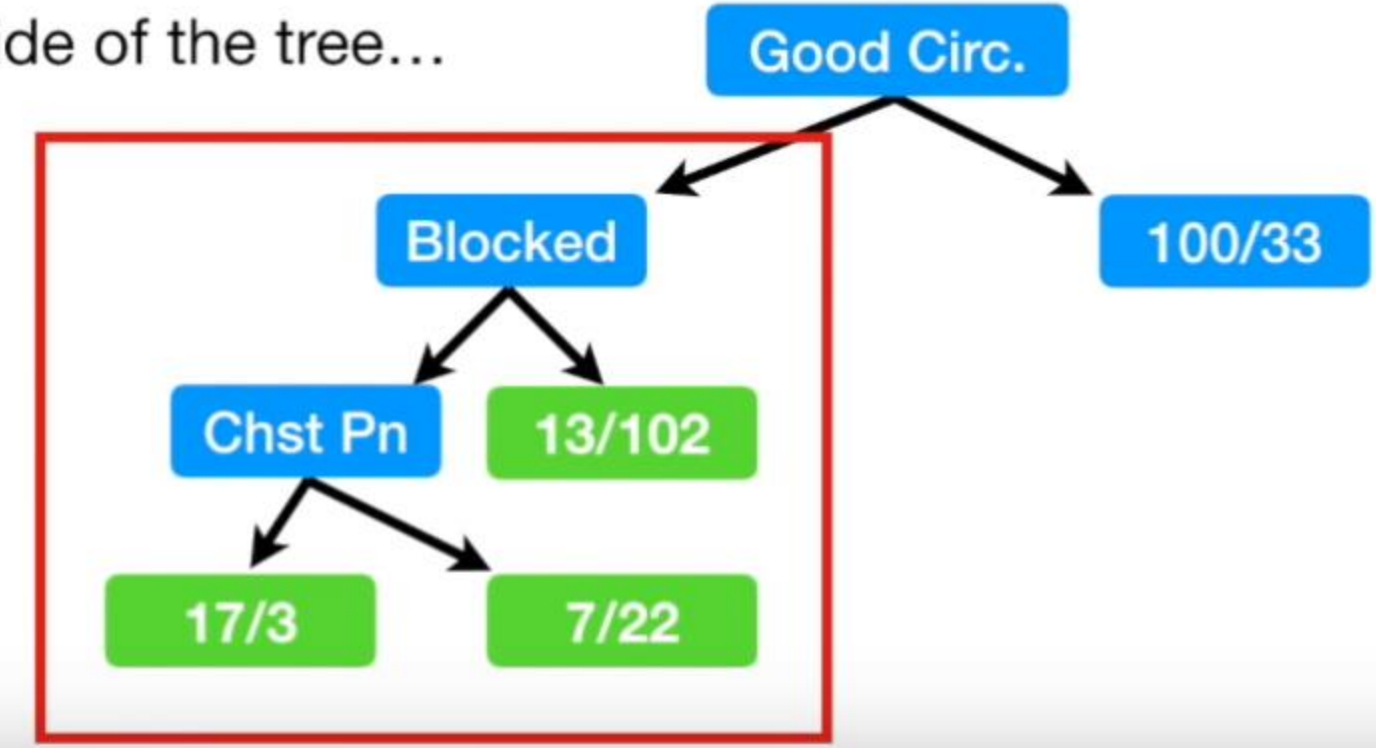
= 1 - (the probability of "yes")$^2$
- (the probability of "no")$^2$

$$= 1 - \left(\frac{13}{13 + 102}\right)^2 - \left(\frac{102}{13 + 102}\right)^2$$

0.2

# Building a Decision Tree

OK, at this point we've worked out the entire left side of the tree…

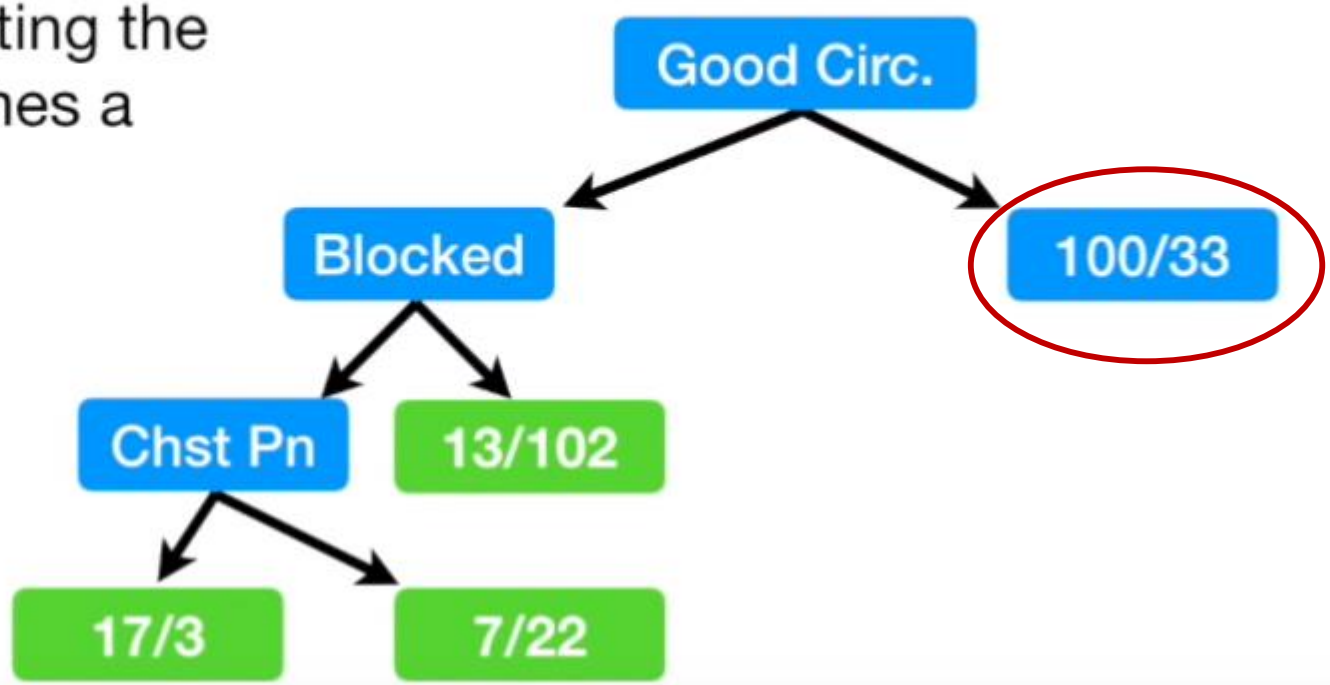


# Building a Decision Tree

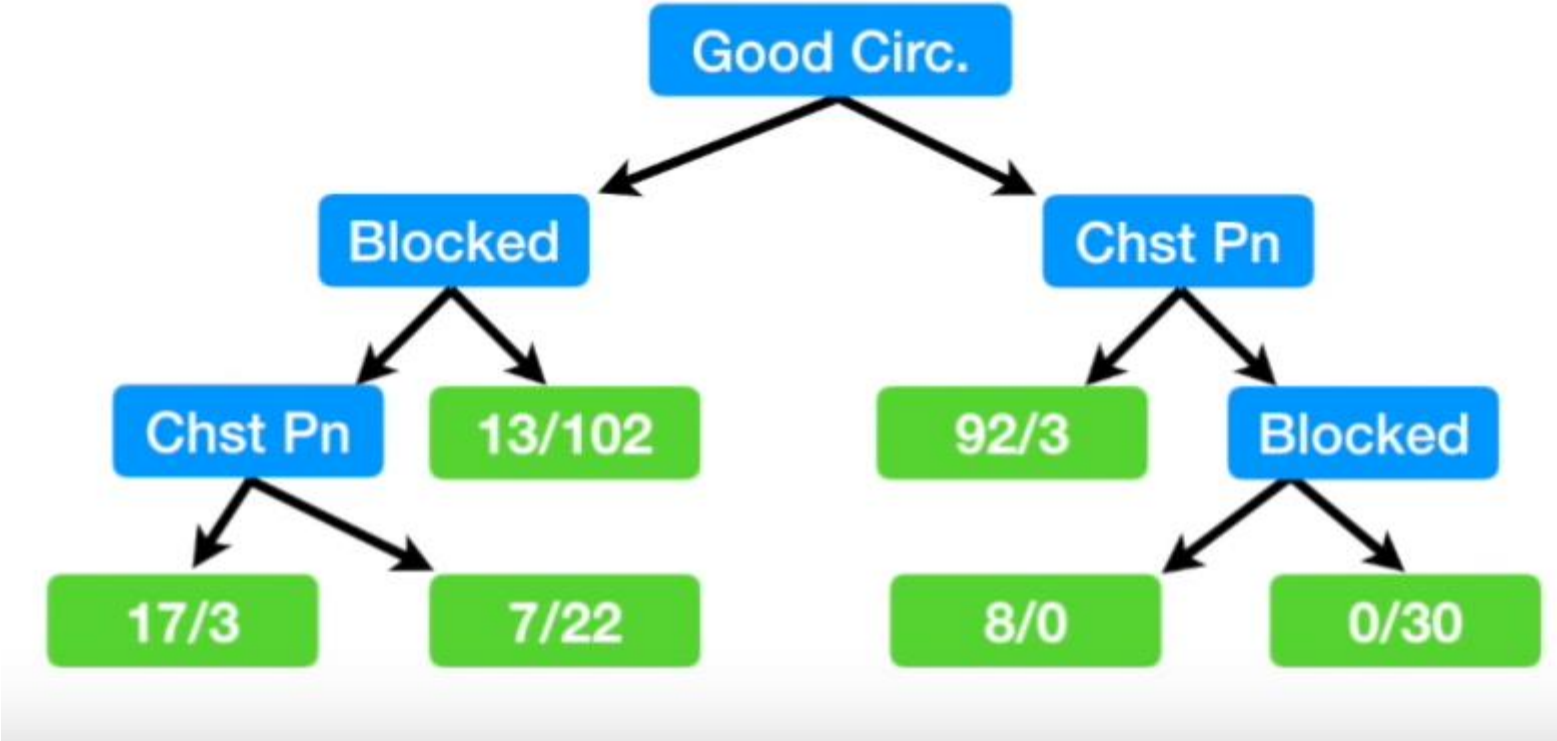
1) Calculate all of the Gini impurity scores.

2) If the node itself has the lowest score, than there is no point in separating the patients any more and it becomes a leaf node.

3) If separating the data results in an improvement, than pick the separation with the lowest impurity value.

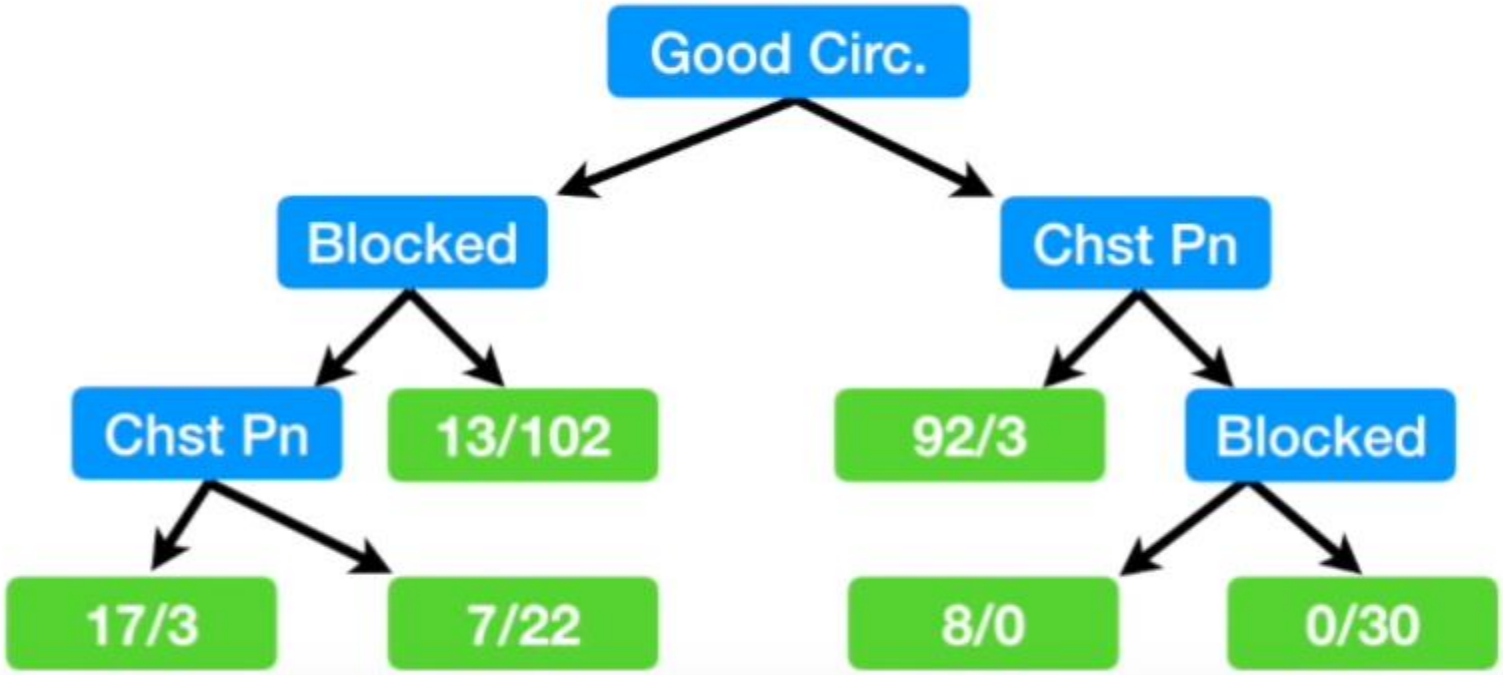


# Building a Decision Tree



# Building a Decision Tree

…but what if we have numeric data, like patient weight?



# Building a Decision Tree

| Weight | Heart Disease |
|---|---|
| 220 | Yes |
| 180 | Yes |
| 225 | Yes |
| 190 | No |
| 155 | No |

How do we determine what's the best weight to use to divide the patients?

# Building a Decision Tree

| | Weight | Heart Disease |
|---|---|---|
| Lowest | 155 | No |
| | 180 | Yes |
| | 190 | No |
| | 220 | Yes |
| Highest | 225 | Yes |

Step 1) Sort the patients by weight, lowest to highest.

# Building a Decision Tree

| Weight | Heart Disease |
|---|---|
| 155 | No |
| **167.5** | |
| 180 | Yes |
| **185** | |
| 190 | No |
| **205** | |
| 220 | Yes |
| **222.5** | |
| 225 | Yes |

Step 2) Calculate the average weight for all adjacent patients.

# Building a Decision Tree

| Weight | Heart Disease |
|---|---|
| 155 | No |
| **167.5** | |
| 180 | Yes |
| **185** | |
| 190 | No |
| **205** | |
| 220 | Yes |
| **222.5** | |
| 225 | Yes |

Step 3) Calculate the impurity values for each average weight.

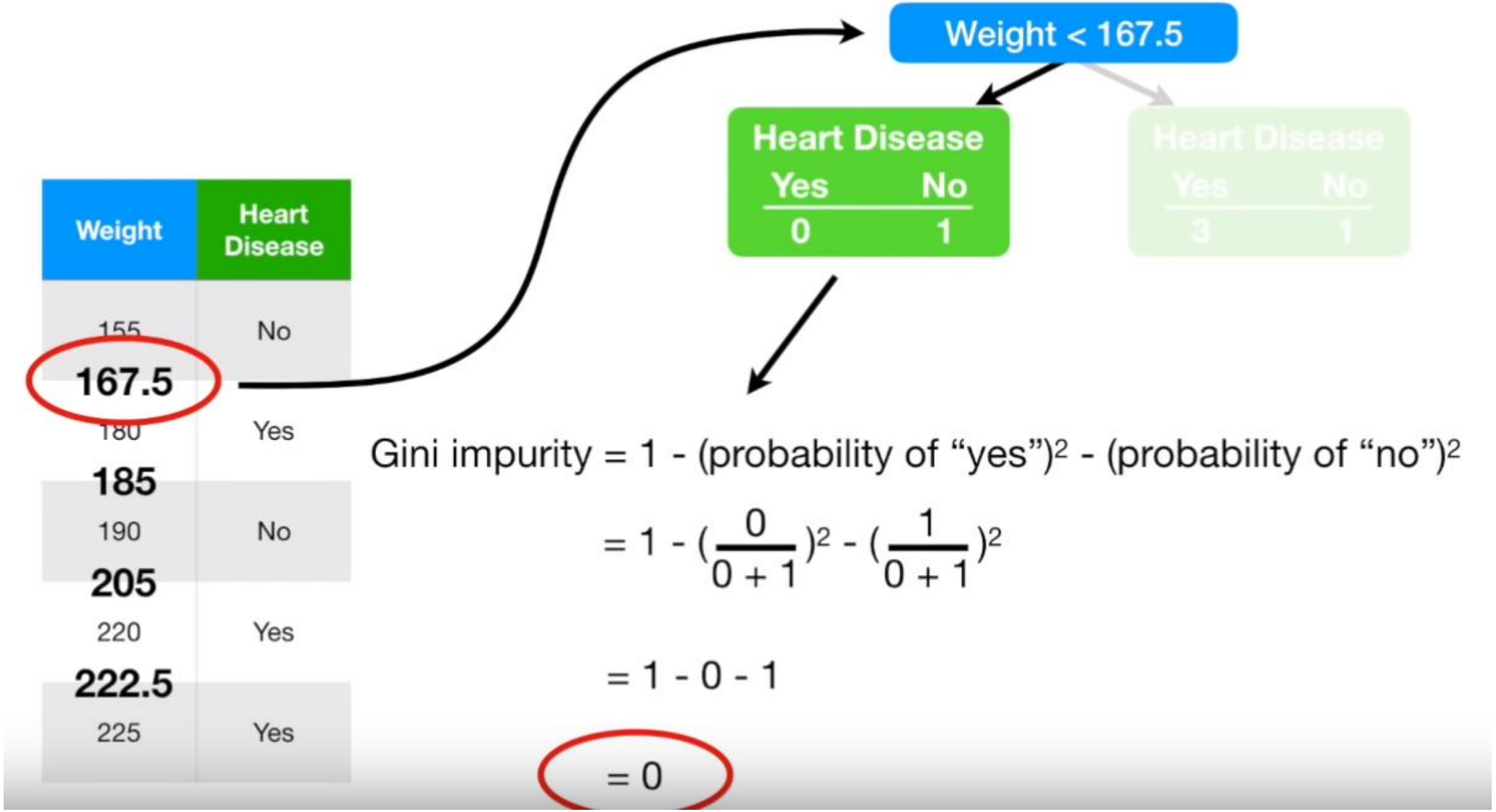
**167.5** → Gini impurity = ?

**185** → Gini impurity = ?

**205** → Gini impurity = ?

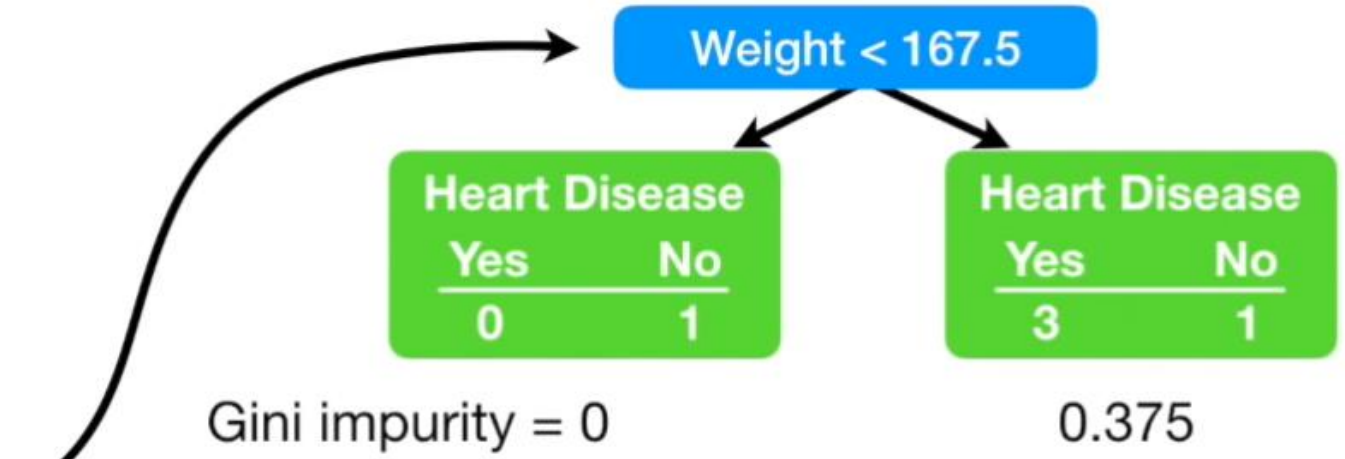
**222.5** → Gini impurity = ?

# Building a Decision Tree



# Building a Decision Tree

| Weight | Heart Disease |
|---|---|
| 155 | No |
| **167.5** | |
| 180 | Yes |
| **185** | |
| 190 | No |
| **205** | |
| 220 | Yes |
| **222.5** | |
| 225 | Yes |



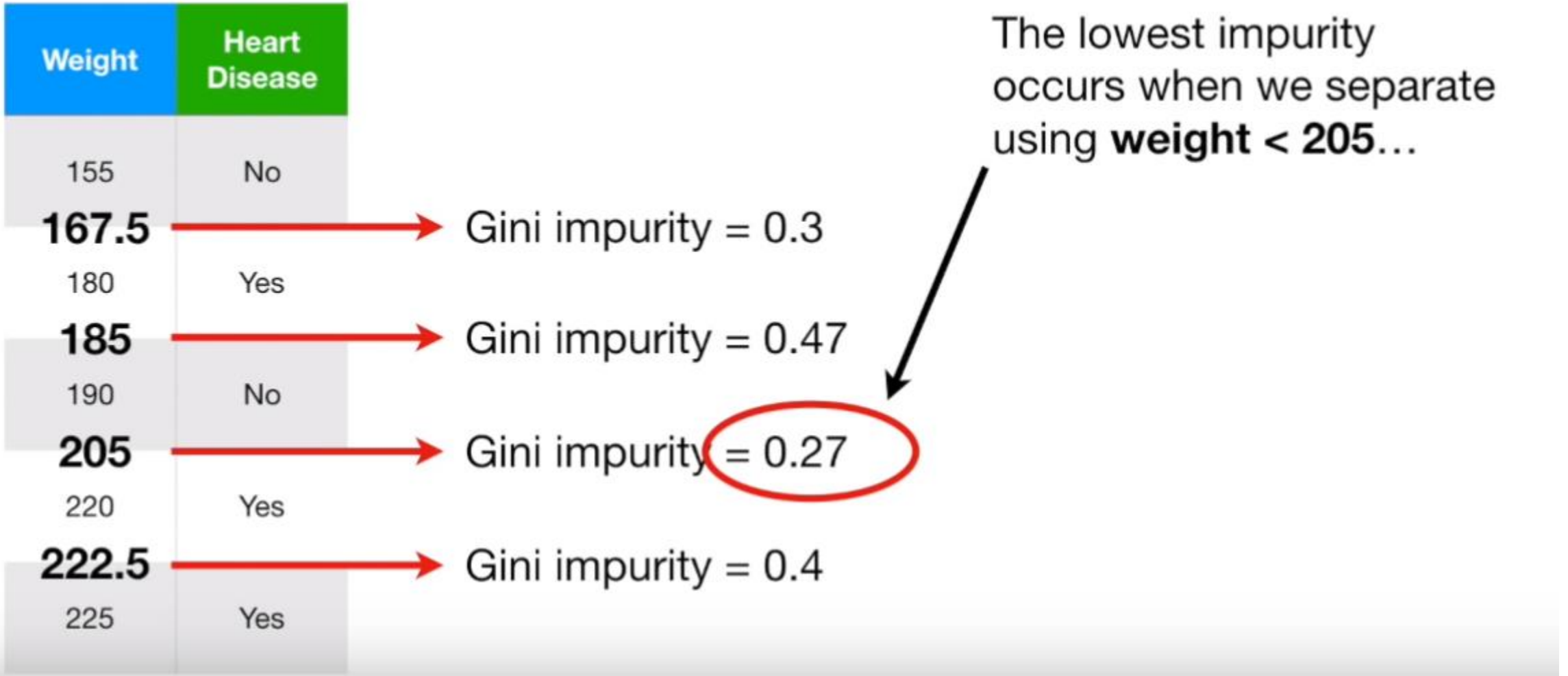

Gini impurity for Weight < 167.5 is the weighted average of the impurities for the two leaves.

$$= \left(\frac{1}{1+4}\right) 0 + \left(\frac{4}{1+4}\right) 0.336 = 0.3$$

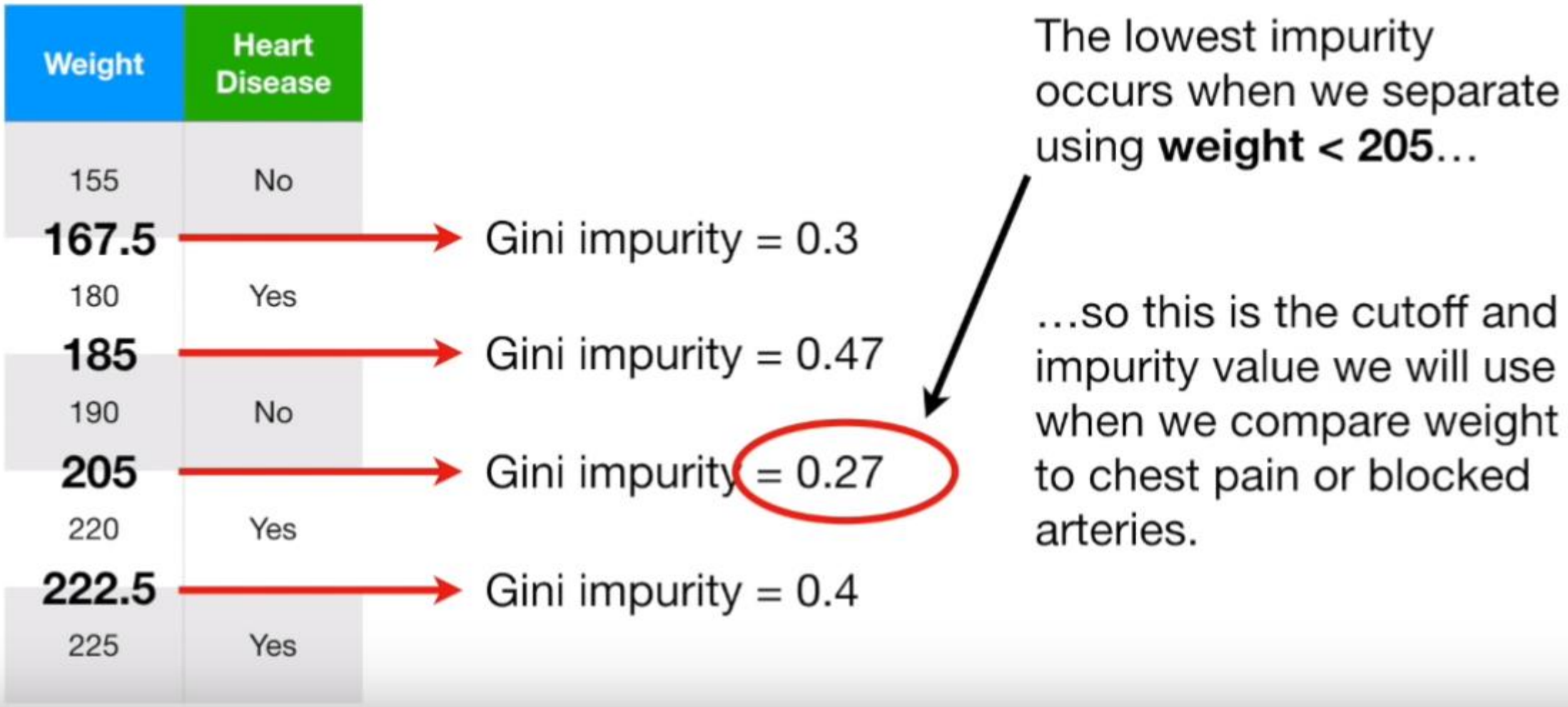
# Building a Decision Tree

| Weight | Heart Disease |
|---|---|
| 155 | No |
| **167.5** | |
| 180 | Yes |
| **185** | |
| 190 | No |
| **205** | |
| 220 | Yes |
| **222.5** | |
| 225 | Yes |

# Building a Decision Tree



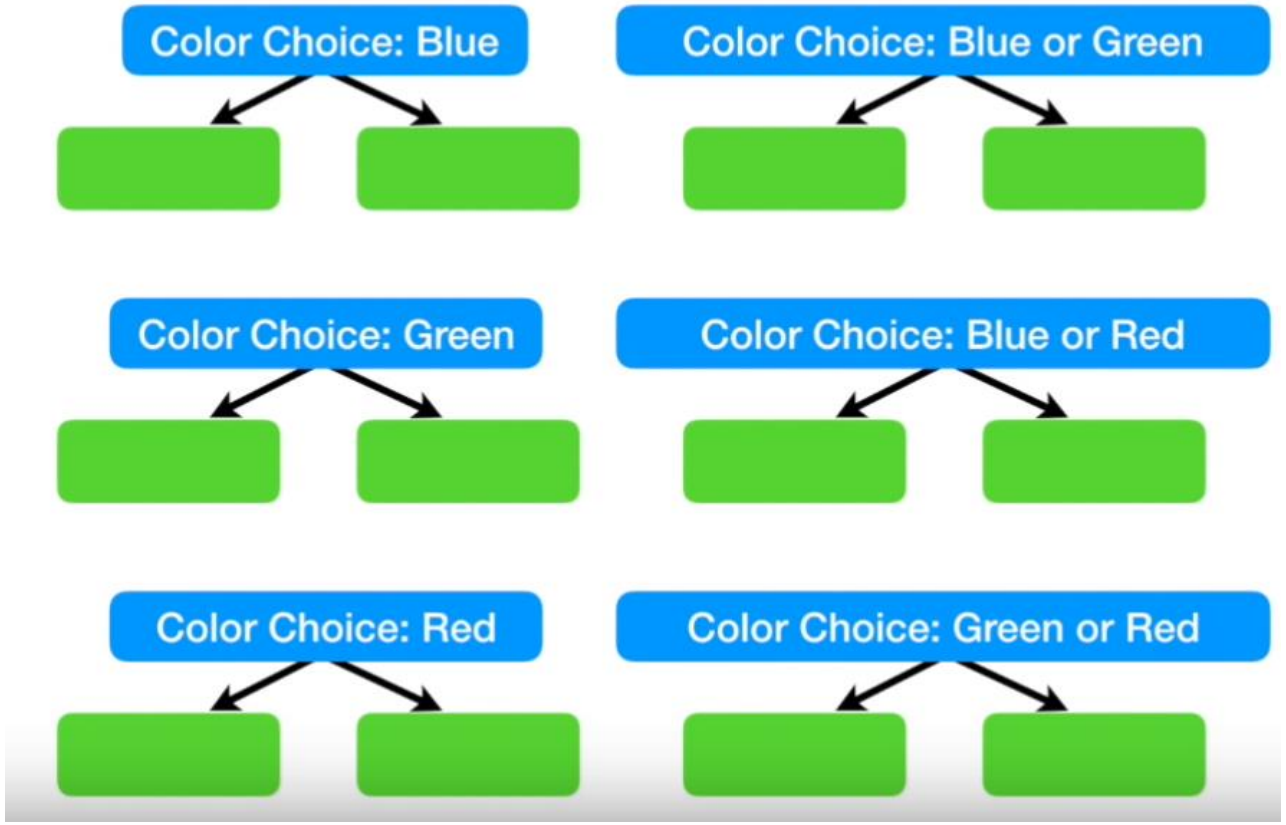
# Building a Decision Tree



# Building a Decision Tree

When there are **multiple choices**, like "**color choice can be blue, green or red**", you calculate an impurity score for each one as well as each possible combination.

For this example, with three colors (blue, green and red) we get the following options…



# Building a Decision Tree