

# Abdul Qayyum

Lecturer at University of Burgundy, France

- Postdoc in Electrical and Informatics Engineering
- PhD in Electrical & Electronics Engineering
- Masters in Electronics Engineering
- Bachelor in Computer Engineering

## Collaborations & Expertise:



# Topic: Decision Tree and Random Forest Algorithm

Instructor: Abdul Qayyum, PhD

Class: MSCV

University of Burgundy, France

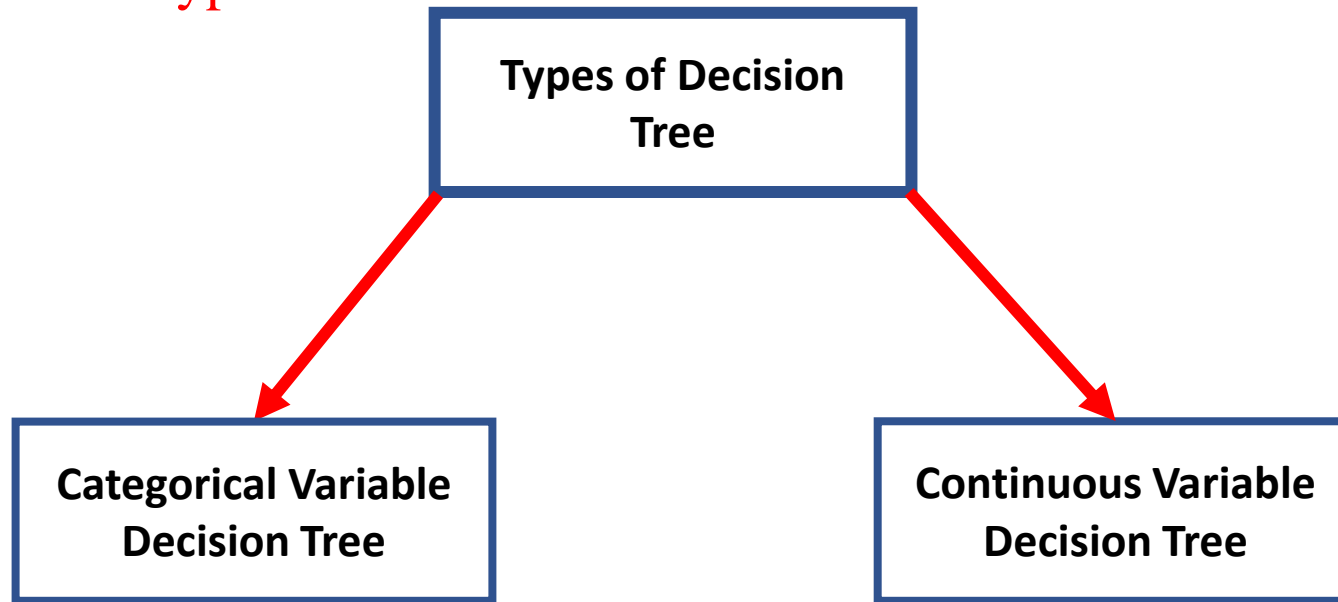
# Outlines of Lecture:

In this lecture we will consider

- What is Decision Tree?
- How to construct Decision Tree
- Use case examples for decision Tree
- Practical implementation
- What is Random Forest
- How to design random forest based on decision tree
- Working Examples

# Decision Trees

- **Types of Decision Tree Algorithms**
- Types of Decision Trees
- Types of decision trees are based on the type of target variable we have.
- **It can be of two types:**



Categorical Variable Decision Tree: Decision Tree which has a categorical target variable then it called a Categorical variable decision tree.

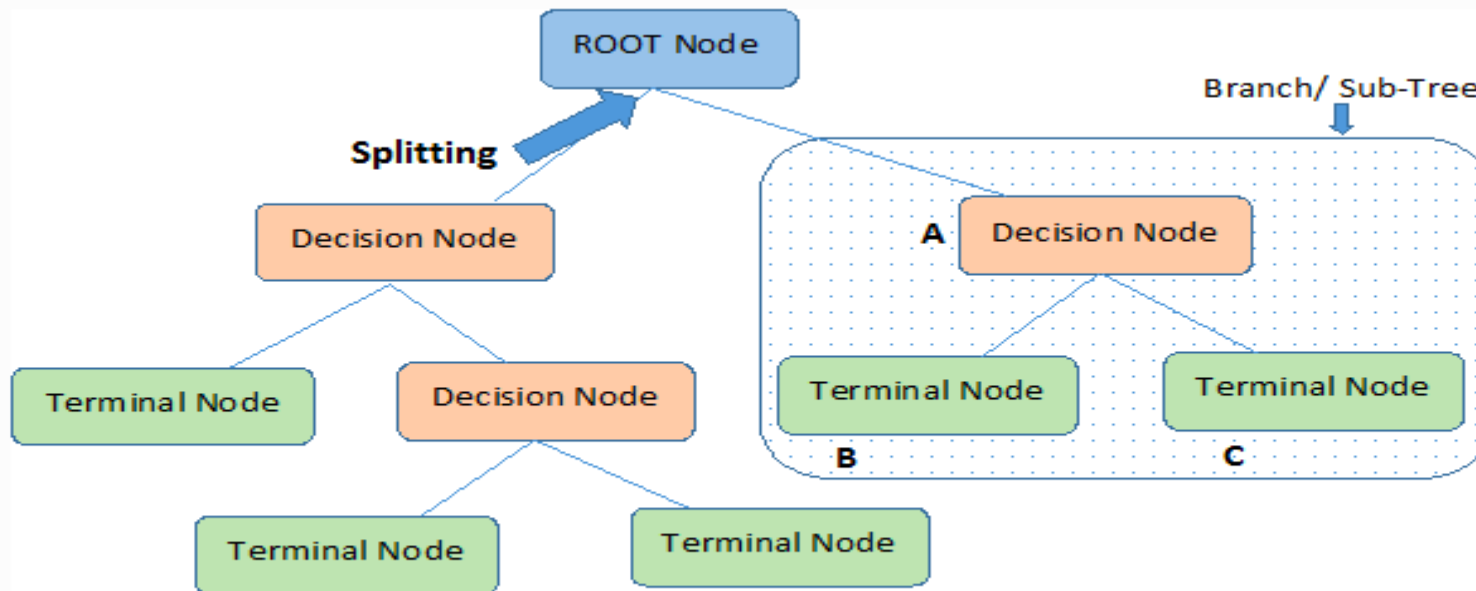
Continuous Variable Decision Tree: Decision Tree has a continuous target variable then it is called Continuous Variable Decision Tree.

# Decision Trees

- **Decision Tree terminology**
- Important Terminology related to Decision Trees
- **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
- **Splitting:** It is a process of dividing a node into two or more sub-nodes.
- **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
- **Leaf / Terminal Node:** Nodes do not split is called Leaf or Terminal node.
- **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
- **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
- **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.

# Decision Trees

## ▪ Decision Tree terminology



**Note:-** A is parent node of B and C.

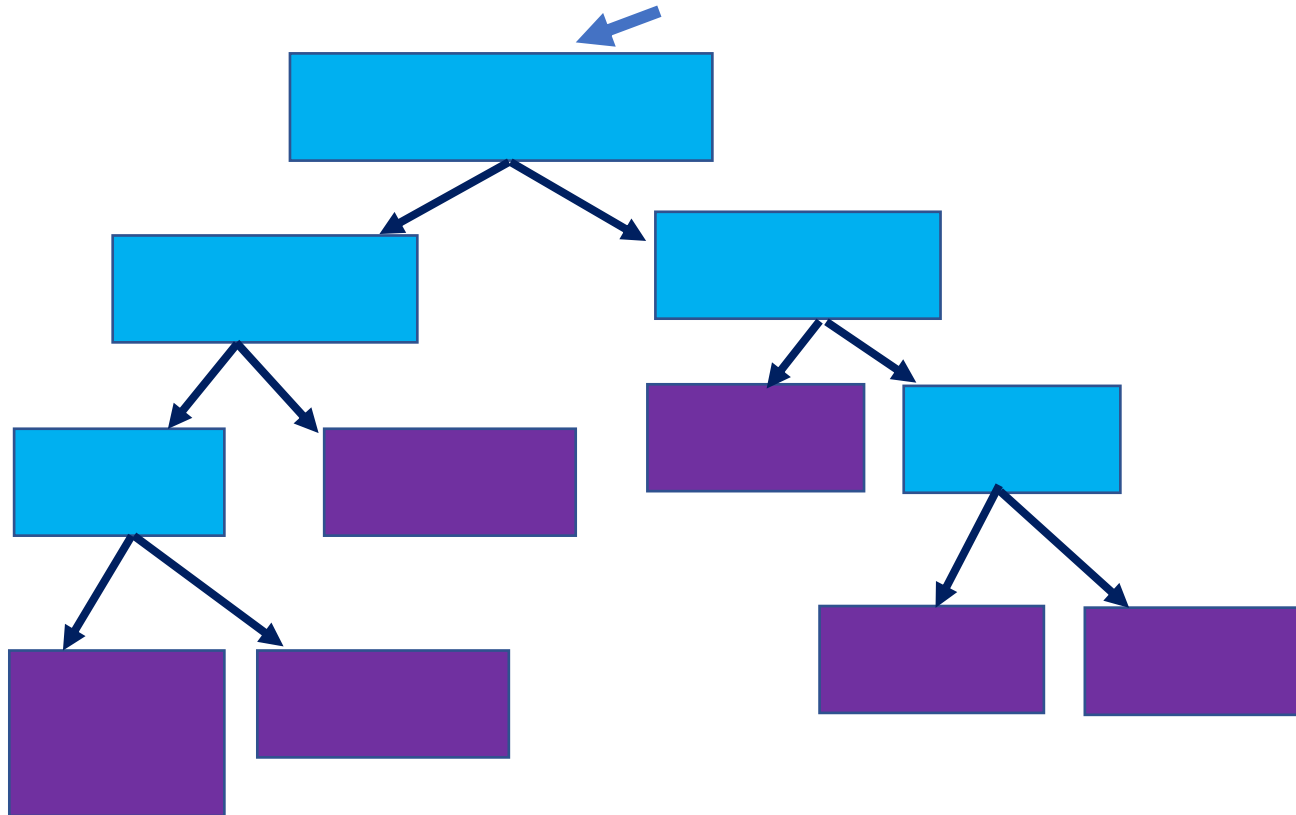
Decision trees classify the examples by sorting them down the tree from the root to some leaf/terminal node, with the leaf/terminal node providing the classification of the example.

Each node in the tree acts as a test case for some attribute, and each edge descending from the node corresponds to the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new node.

# Decision Trees

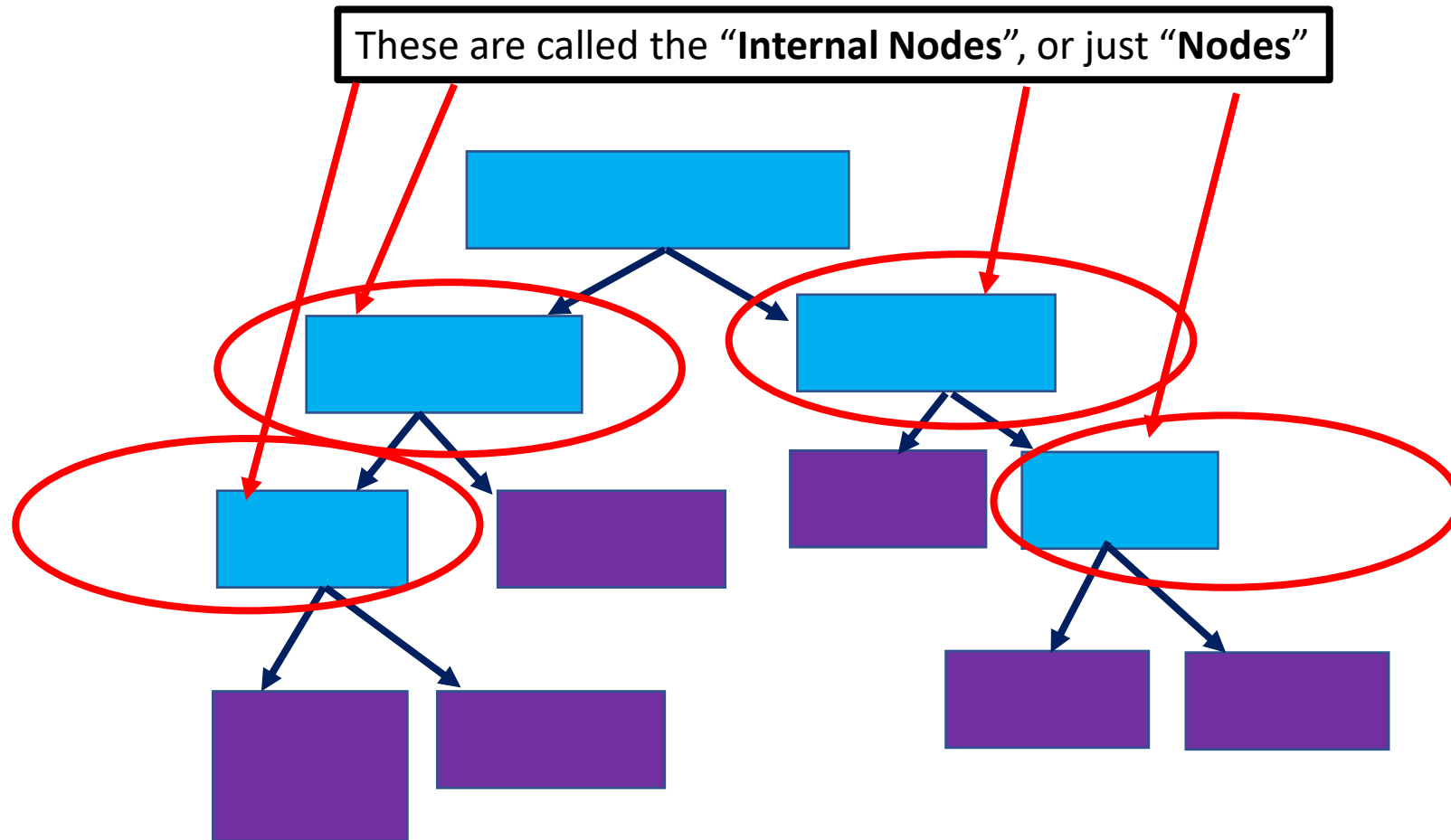
- Decision Tree terminology

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



# Decision Trees

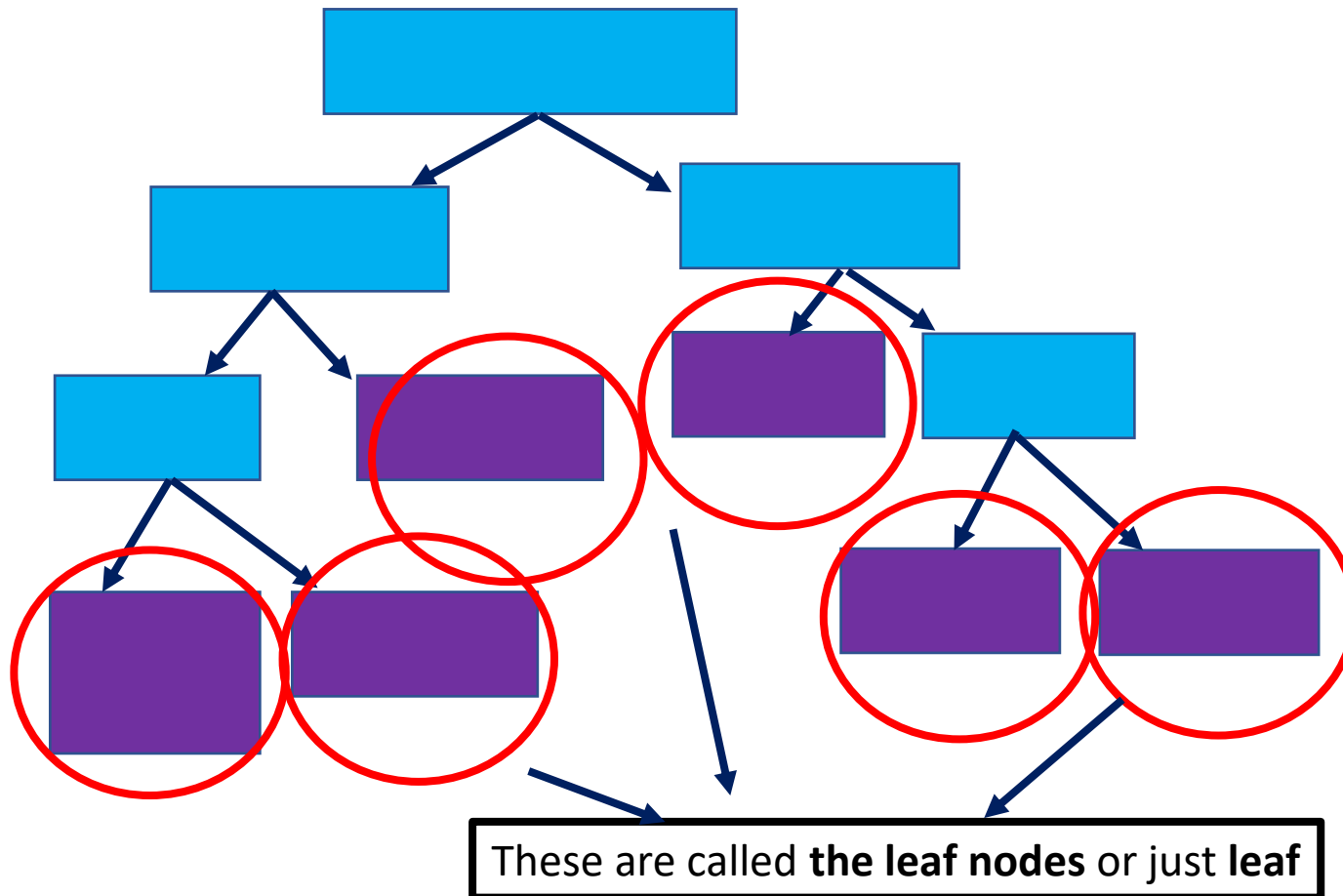
- Decision Tree terminology





# Decision Trees

- Decision Tree terminology



# Decision Trees

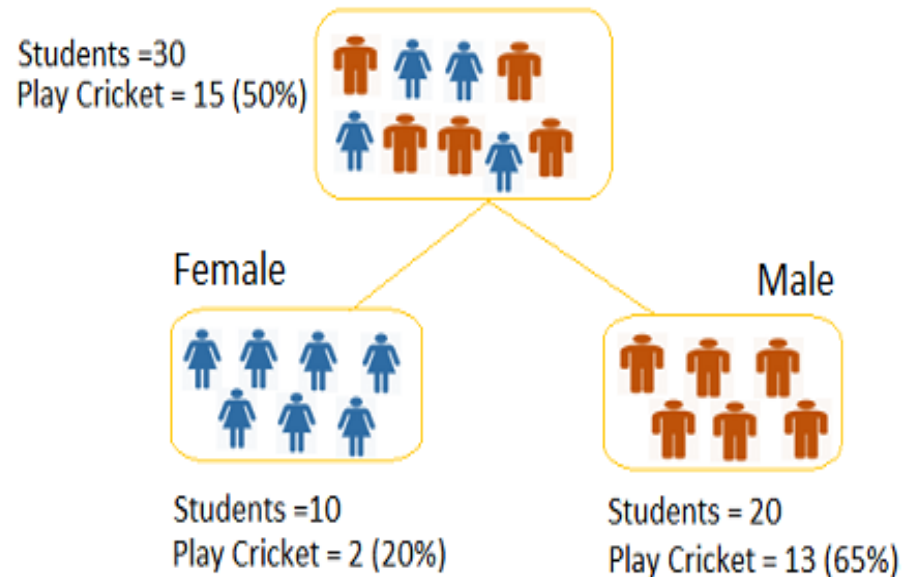
- **Building Methods for decision Trees**
- **Attribute Selection Measures**
- If the dataset consists of N attributes then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step. By just randomly selecting any node to be the root can't solve the issue. If we follow a random approach, it may give us bad results with low accuracy.
- For solving this attribute selection problem, researchers worked and devised some solutions. They suggested using some criteria like :

**Entropy,**  
**Information Gain,**  
**Gini index,**  
**Gain Ratio,**  
**Reduction in Variance**  
**Chi-Square**

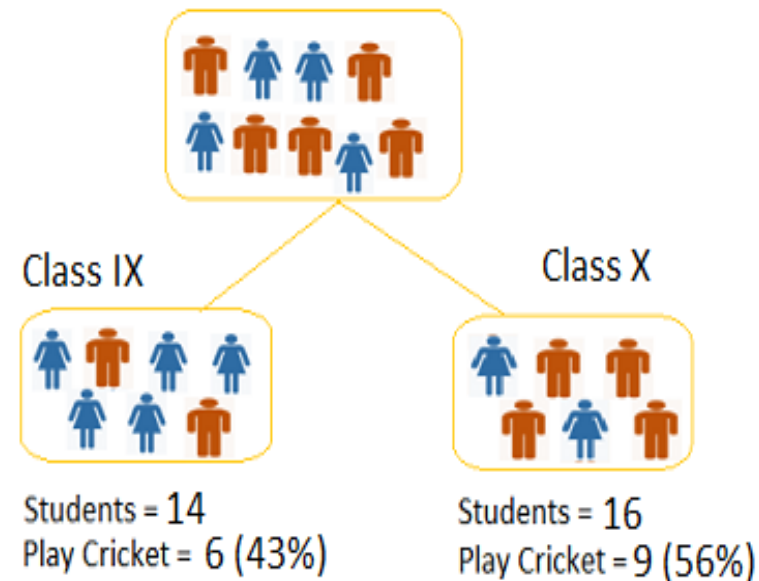
# Decision Trees

- **Compute Entropy and Gini Impurity using simple example**
- Example: –we want to segregate the students based on target variable ( playing cricket or not ).
- In the snapshot below, we split the population using two input variables Gender and Class.
- Now, I want to identify which split is producing more homogeneous sub-nodes using Gini

Split on Gender



Split on Class



# Decision Trees

$$Gini = 1 - \sum_{i=1}^C (p_i)^2$$

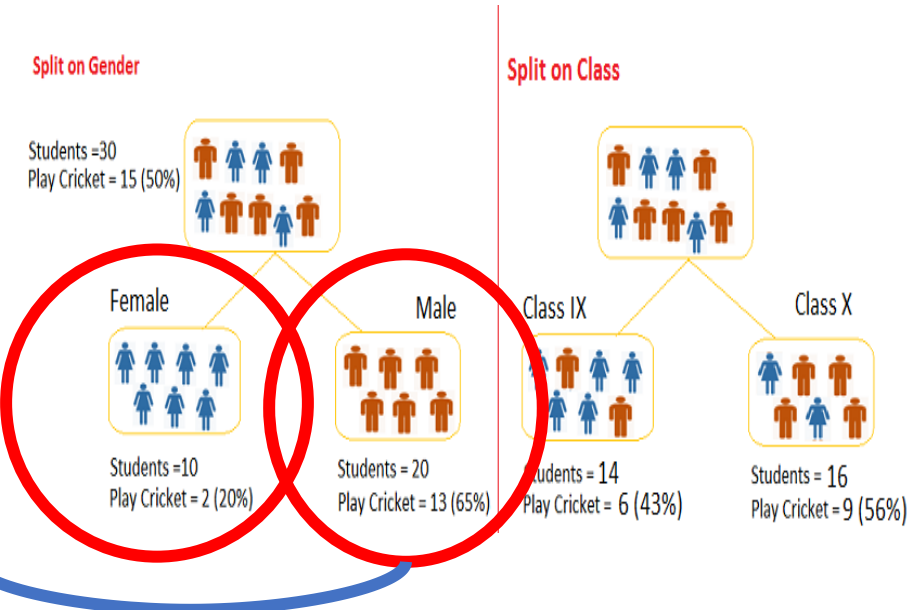
- **Compute Entropy and Gini Impurity using simple example**
- Steps to Calculate Gini for a split
- Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure ( $p^2 + q^2$ ).
- Calculate Gini for split using weighted Gini score of each node of that split

**Split on Gender:**

**Calculate, Gini for sub-node Female =**  
 $p^2 + q^2 = (0.2)^2 + (0.8)^2 = 0.68$

**Gini for sub-node Male =**  $(0.65)^2 + (0.35)^2 = 0.55$

**Calculate weighted Gini for Split Gender =**  $(10/30) * 0.68 + (20/30) * 0.55 = 0.59$



# Decision Trees

$$Gini = 1 - \sum_{i=1}^C (p_i)^2$$

- **Compute Entropy and Gini Impurity using simple example**
- Steps to Calculate Gini for a split
- Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure ( $p^2 + q^2$ ).
- Calculate Gini for split using weighted Gini score of each node of that split

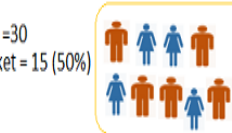
Similar for Split on Class:

$$\text{Gini for sub-node Class IX} = (0.43) * (0.43) + (0.57) * (0.57) = 0.51$$

$$\text{Gini for sub-node Class X} = (0.56) * (0.56) + (0.44) * (0.44) = 0.51$$

Split on Gender

Students = 30  
Play Cricket = 15 (50%)



Female



Students = 10  
Play Cricket = 2 (20%)

Male



Students = 20  
Play Cricket = 13 (65%)

Split on Class



Class IX



Students = 14  
Play Cricket = 6 (43%)

Class X



Students = 16  
Play Cricket = 9 (56%)

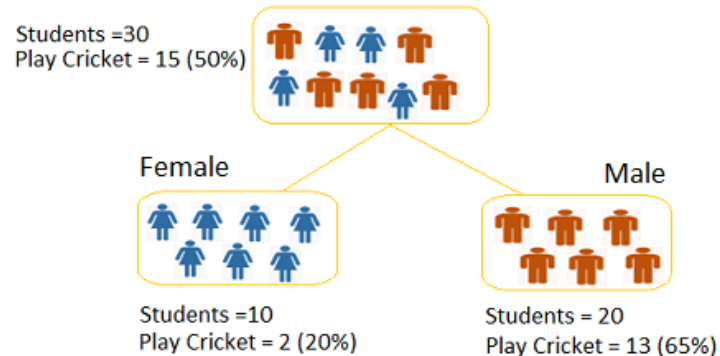
$$\text{Calculate weighted Gini for Split Class} = (14/30) * 0.51 + (16/30) * 0.51 = 0.51$$

# Decision Trees

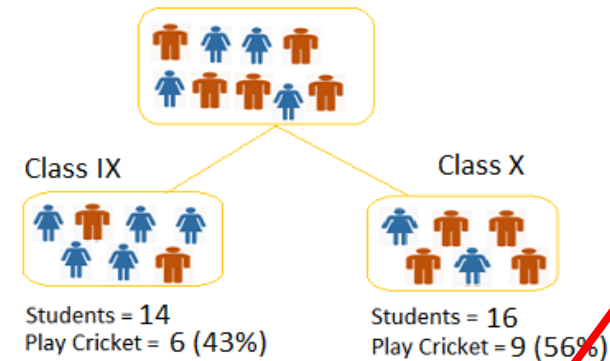
- **Compute Entropy and Gini Impurity using simple example**
- Steps to Calculate Gini for a split

$$Gini = 1 - \sum_{i=1}^C (p_i)^2$$

Split on Gender



Split on Class



Calculate weighted Gini for Split Gender =  $(10/30) * 0.68 + (20/30) * 0.55 = 0.59$

Calculate weighted Gini for Split Class =  $(14/30) * 0.51 + (16/30) * 0.51 = 0.51$

Above, you can see that Gini score for Split on Gender is higher than Split on Class, hence, the node split will take place on Gender.

You might often come across the term 'Gini Impurity' which is determined by subtracting the gini value from 1. So mathematically we can say,

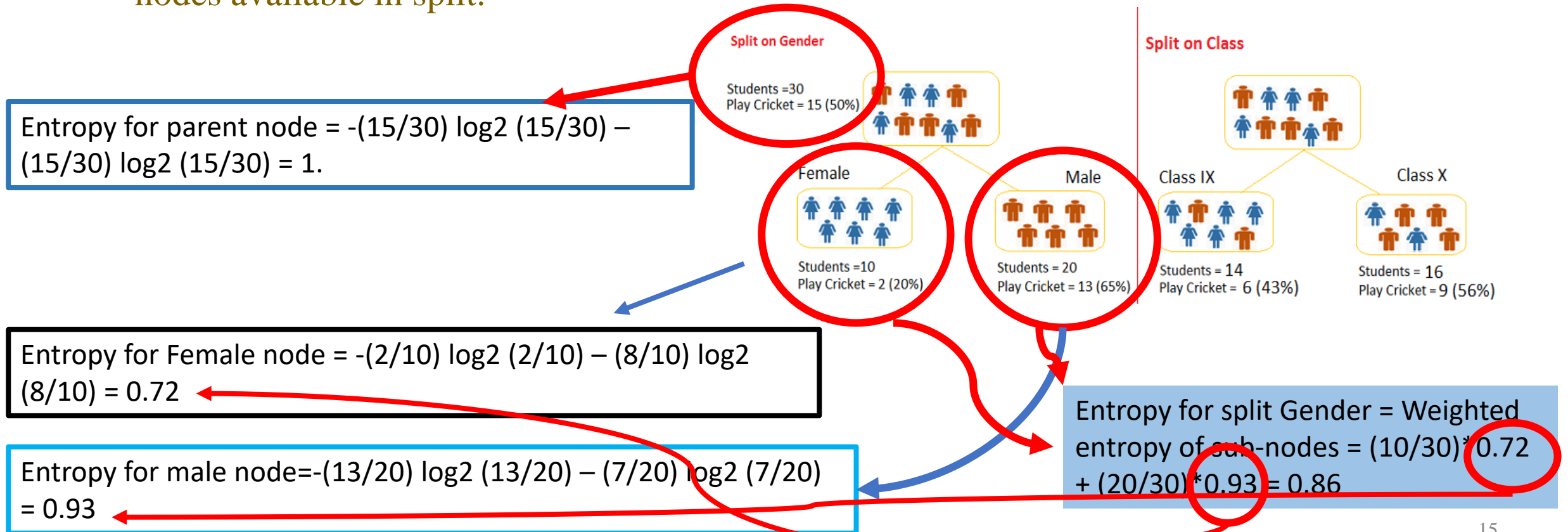
**Gini Impurity = 1-Gini**

# Decision Trees

- **Compute Entropy and Gini Impurity using simple example**
- Steps to calculate entropy for a split:
  - Calculate entropy of parent node
  - Calculate entropy of each individual node of split and calculate weighted average of all sub-nodes available in split.

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$



# Decision Trees

- **Compute Entropy and Gini Impurity using simple example**
- Steps to calculate entropy for a split:
  - Calculate entropy of parent node
  - Calculate entropy of each individual node of split and calculate weighted average of all sub-nodes available in split.

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

Entropy for split Class =  $(14/30) * 0.99 + (16/30) * 0.99 = 0.99$

Split on Gender

Students = 30  
Play Cricket = 15 (50%)

Female



Students = 10  
Play Cricket = 2 (20%)

Male



Students = 20  
Play Cricket = 13 (65%)

Split on Class

Class IX



Students = 14  
Play Cricket = 6 (43%)

Class X



Students = 16  
Play Cricket = 9 (56%)

Entropy for Class IX node,  $-(6/14) \log_2 (6/14) - (8/14) \log_2 (8/14) = 0.99$

Entropy for Class X node,  $-(9/16) \log_2 (9/16) - (7/16) \log_2 (7/16) = 0.99$



# Decision Trees

- **Compute Entropy and Gini Impurity using simple example**
- Steps to calculate entropy for a split:
- Calculate entropy of parent node
- Calculate entropy of each individual node of split and calculate weighted average of all sub-nodes available in split.

Entropy for split Gender = Weighted entropy of sub-nodes  
 $= (10/30) * 0.72 + (20/30) * 0.93 = \mathbf{0.86}$

Entropy for split Class =  $(14/30) * 0.99 + (16/30) * 0.99 = \mathbf{0.99}$

Split on Gender

Students = 30  
Play Cricket = 15 (50%)



Female



Students = 10  
Play Cricket = 2 (20%)

Male



Students = 20  
Play Cricket = 13 (65%)

Split on Class



Class IX



Students = 14  
Play Cricket = 6 (43%)

Class X



Students = 16  
Play Cricket = 9 (56%)

Above, you can see that entropy for Split on Gender is the lowest among all, so the tree will split on Gender.

**We can derive information gain from entropy as  $1 - \text{Entropy}$ .**

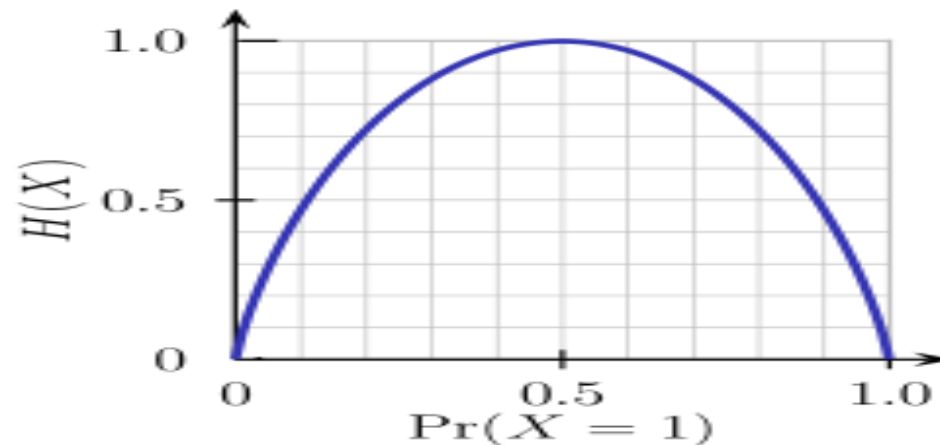
# Decision Trees

- **Attribute Selection Measures**

- These criteria will calculate values for every attribute. The values are sorted, and attributes are placed in the tree by following the order i.e, the attribute with a high value (in case of information gain) is placed at the root.
- While using Information Gain as a criterion, we assume attributes to be categorical, and for the Gini index, attributes are assumed to be continuous.

- **Entropy**

- Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. Flipping a coin is an example of an action that provides information that is random.

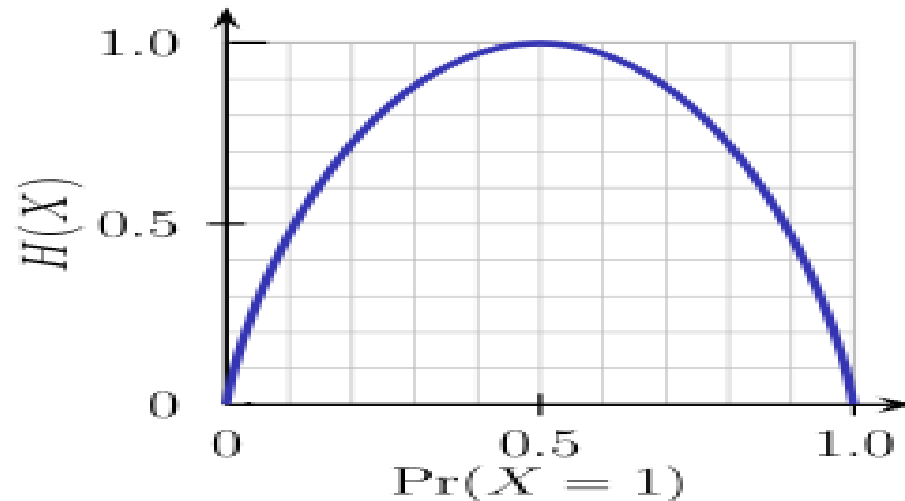


# Decision Trees

- **Attribute Selection Measures**

- **Entropy**

- From the below graph, it is quite evident that the entropy  $H(X)$  is zero when the probability is either 0 or 1. The Entropy is maximum when the probability is 0.5 because it projects perfect randomness in the data and there is no chance of perfectly determining the outcome.
- ID3 follows the rule — A branch with an entropy of zero is a leaf node and A branch with entropy more than zero needs further splitting.



# Decision Trees

- **Attribute Selection Measures**
- **Entropy**
- Mathematically Entropy for 1 attribute is represented as:

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5



Entropy(PlayGolf) = Entropy (5,9)  
= Entropy (0.36, 0.64)  
= - (0.36 log<sub>2</sub> 0.36) - (0.64 log<sub>2</sub> 0.64)  
= 0.94

# Decision Trees

- **Attribute Selection Measures**

- **Entropy**

- Where  $S \rightarrow$  Current state, and  $P_i \rightarrow$  Probability of an event  $i$  of state  $S$  or Percentage of class  $i$  in a node of state  $S$ .
- where  $T \rightarrow$  Current state and  $X \rightarrow$  Selected attribute
- Mathematically Entropy for multiple attributes is represented as:

$$E(T, X) = \sum_{c \in X} P(c)E(c)$$

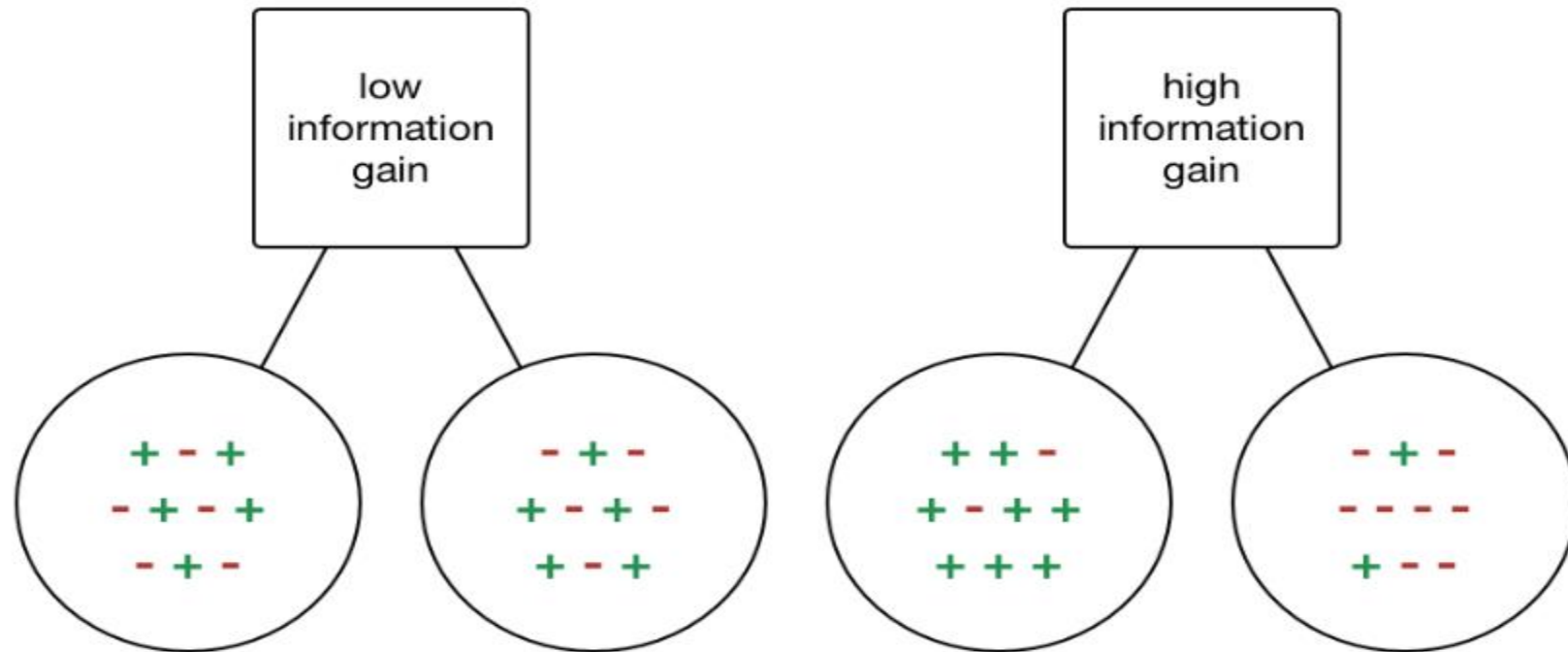
		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14



$$\begin{aligned} E(\text{PlayGolf}, \text{Outlook}) &= P(\text{Sunny}) * E(3,2) + P(\text{Overcast}) * E(4,0) + P(\text{Rainy}) * E(2,3) \\ &= (5/14) * 0.971 + (4/14) * 0.0 + (5/14) * 0.971 \\ &= 0.693 \end{aligned}$$

# Decision Trees

- **Attribute Selection Measures**
- **Information Gain**
- Information gain or IG is a statistical property that measures how well a given attribute separates the training examples according to their target classification.
- Constructing a decision tree is all about finding an attribute that returns **the highest information gain and the smallest entropy.**



# Decision Trees

- **Attribute Selection Measures**
- **Information Gain**
- Information gain is a decrease in entropy. It computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values.
- **ID3 (Iterative Dichotomiser) decision tree algorithm uses information gain.**
- **Mathematically, IG is represented as:**

$$\text{Information Gain}(T,X) = \text{Entropy}(T) - \text{Entropy}(T, X)$$

$$\begin{aligned}\text{IG}(\text{PlayGolf}, \text{Outlook}) &= E(\text{PlayGolf}) - E(\text{PlayGolf}, \text{Outlook}) \\ &= 0.940 - 0.693 \\ &= 0.247\end{aligned}$$

# Decision Trees

- **Attribute Selection Measures**
- **Information Gain**
- In a much simpler way, we can conclude that:

$$\text{Information Gain} = \text{Entropy}(\text{before}) - \sum_{j=1}^K \text{Entropy}(j, \text{after})$$

## Information Gain

Where “before” is the dataset before the split, K is the number of subsets generated by the split, and (j, after) is subset j after the split.



# Decision Trees

- **Attribute Selection Measures**

- **Gini Index**

- You can understand the Gini index as a cost function used to evaluate splits in the dataset. It is calculated by subtracting the sum of the squared probabilities of each class from one. It favors larger partitions and easy to implement whereas information gain favors smaller partitions with distinct values.
- **Gini Index works with the categorical target variable “Success” or “Failure”. It performs only Binary splits.**
- Higher the value of Gini index higher the homogeneity.
- Steps to Calculate Gini index for a split
- Calculate Gini for sub-nodes, using the formula for success(p) and failure(q) ( $p^2+q^2$ ).
- Calculate the Gini index for split using the weighted Gini score of each node of that split.
- **CART (Classification and Regression Tree) uses the Gini index method to create split points.**

$$Gini = 1 - \sum_{i=1}^C (p_i)^2$$

# Decision Trees

- **Attribute Selection Measures**
- **Reduction in Variance**
- Reduction in variance is an algorithm used for continuous target variables (regression problems). This algorithm uses the standard formula of variance to choose the best split. The split with lower variance is selected as the criteria to split the population:
- Above  $\bar{X}$  is the mean of the values,  $X$  is actual and  $n$  is the number of values.
- **Steps to calculate Variance:**
- Calculate variance for each node.
- Calculate variance for each split as the weighted average of each node variance.

$$\text{Variance} = \frac{\Sigma(X - \bar{X})^2}{n}$$

# Decision Trees

- **How to avoid/counter Overfitting in Decision Trees?**
- The common problem with Decision trees, especially having a table full of columns, they fit a lot. Sometimes it looks like the tree memorized the training data set. If there is no limit set on a decision tree, it will give you 100% accuracy on the training data set because in the worse case it will end up making 1 leaf for each observation. Thus this affects the accuracy when predicting samples that are not part of the training set.
- **Here are two ways to remove overfitting:**

- **Pruning Decision Trees.**
- **Random Forest**

## **Pruning Decision Trees**

The splitting process results in fully grown trees until the stopping criteria are reached. But, the fully grown tree is likely to overfit the data, leading to poor accuracy on unseen data.

# Decision Trees

## ■ Which is better Linear or tree-based models?

- Well, it depends on the kind of problem you are solving.
- If the relationship between dependent & independent variables is well approximated by a linear model, linear regression will outperform the tree-based model.
- If there is a high non-linearity & complex relationship between dependent & independent variables, a tree model will outperform a classical regression method.
- If you need to build a model that is easy to explain to people, a decision tree model will always do better than a linear model. Decision tree models are even simpler to interpret than linear regression!

- **Mixed Decision Tree**

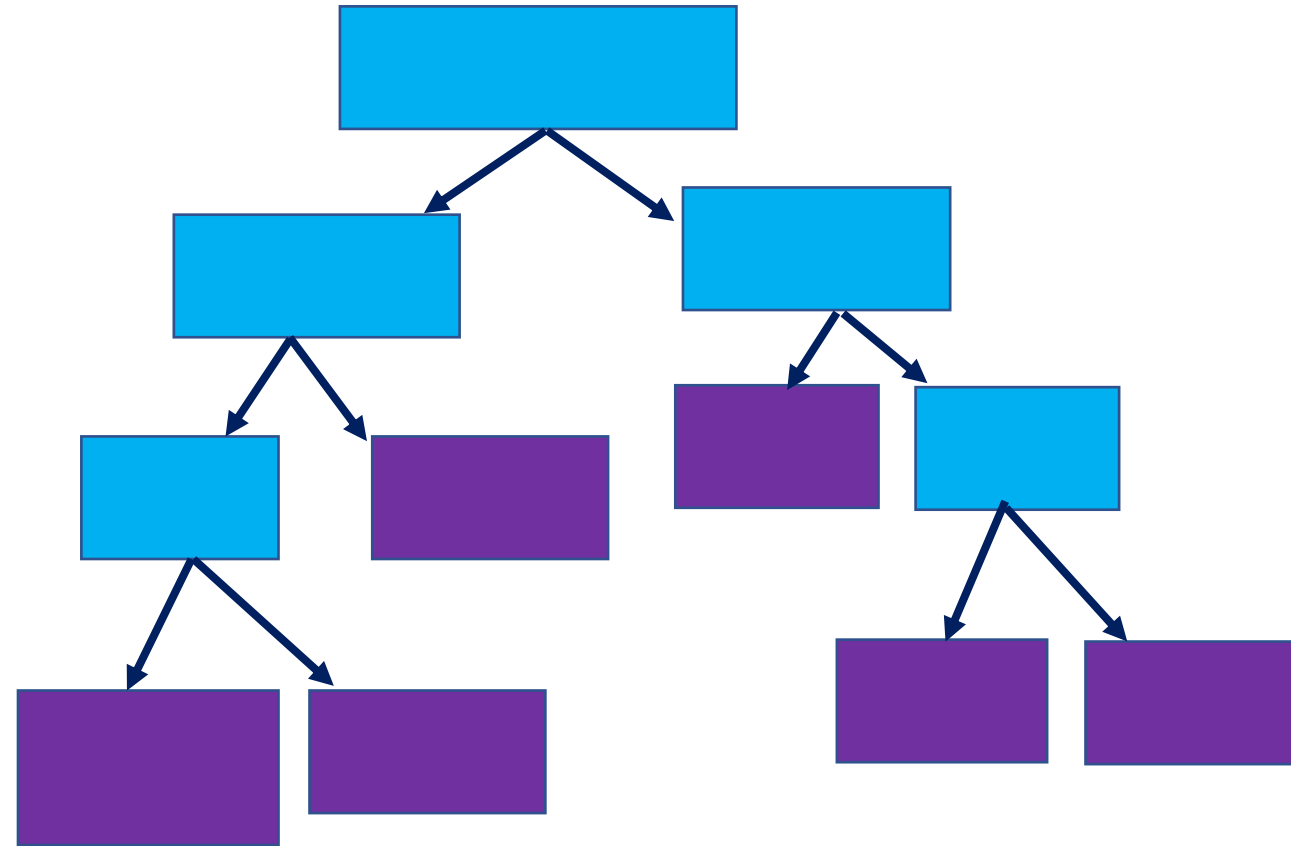


# Decision Tree Algorithm (Working example using Gini Impurity Method)

# Decision Trees

In this example, we want to create a tree that uses **chest pain, good blood circulation and blocked artery status** to predict heart disease

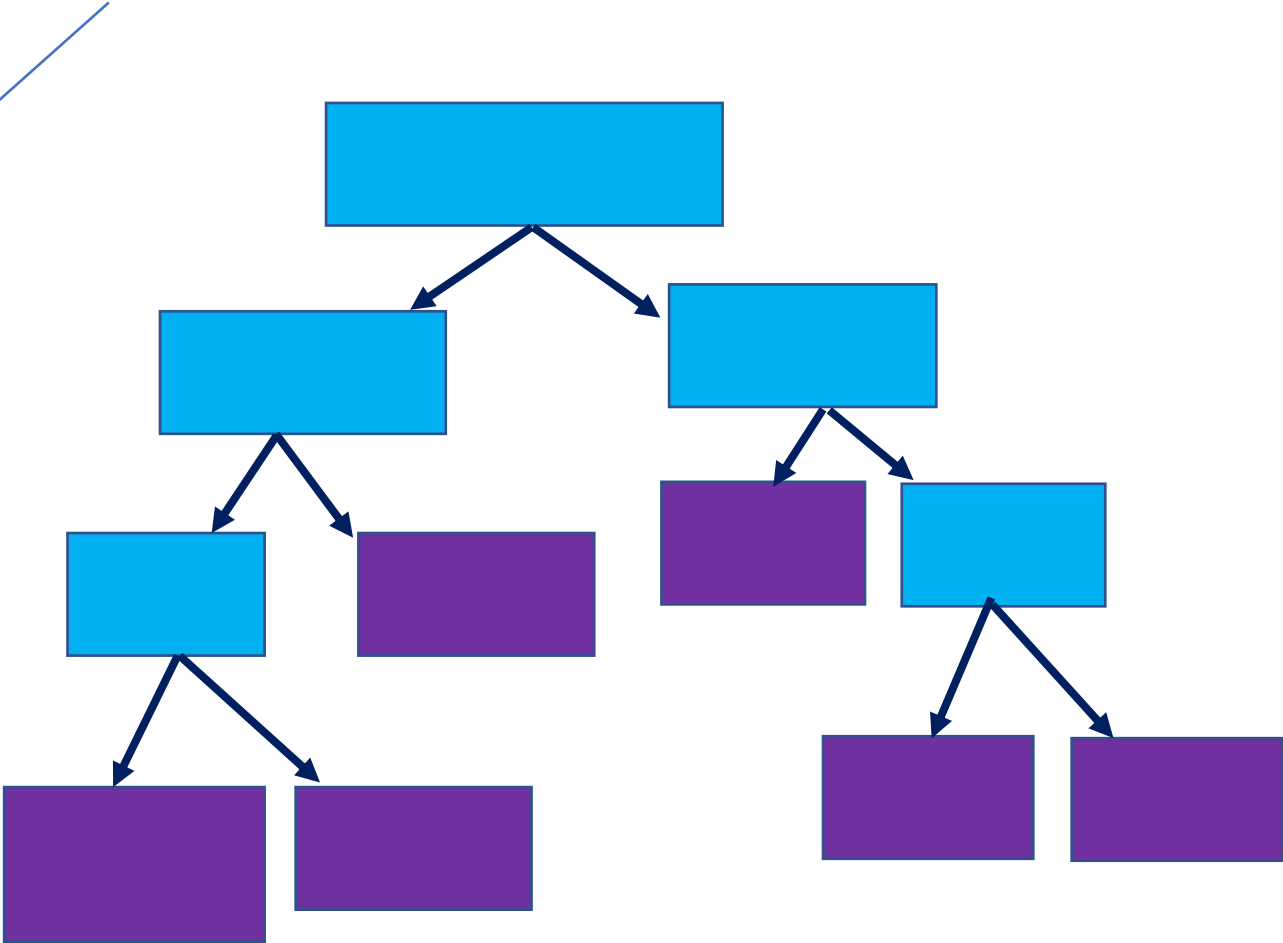
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



# Decision Trees

Whether or not a patient has 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..

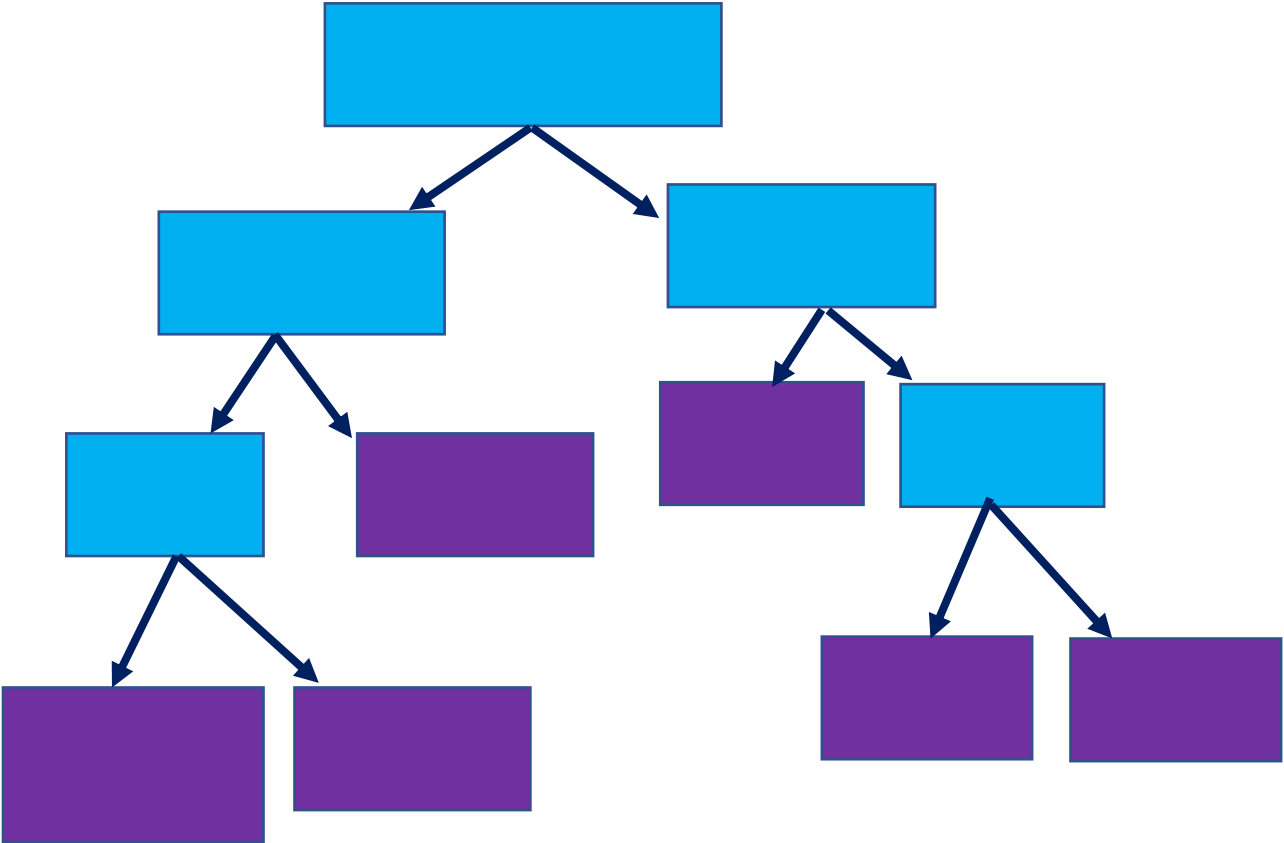




# Decision Trees

The first thing we want to know is whether Chest Pain, good Blood Circulation or Blocked arteries should be at the very top level

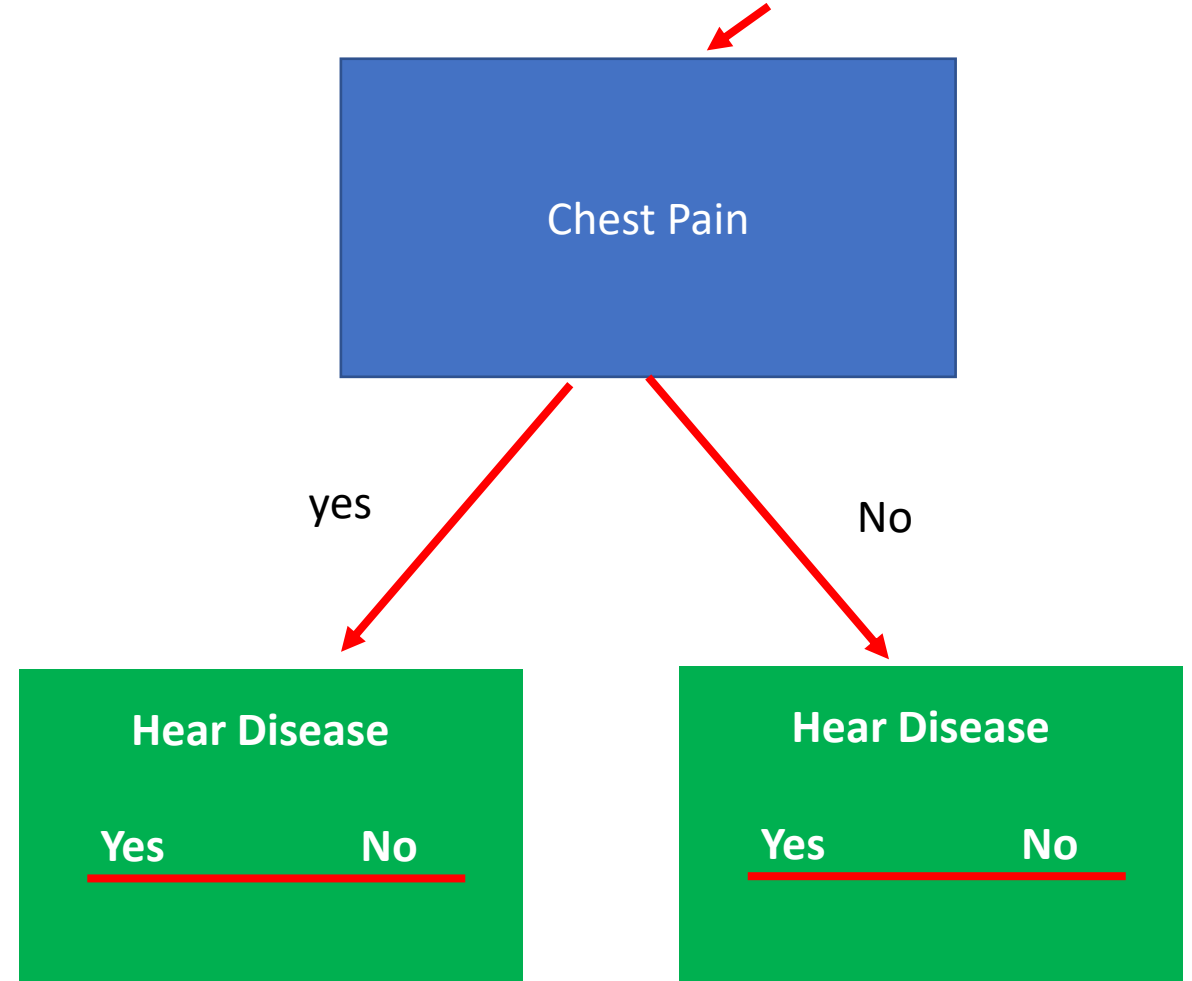
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



We start by looking at how well Chest pain alone predicts heart disease

Here is tree that only contains chest pain into account

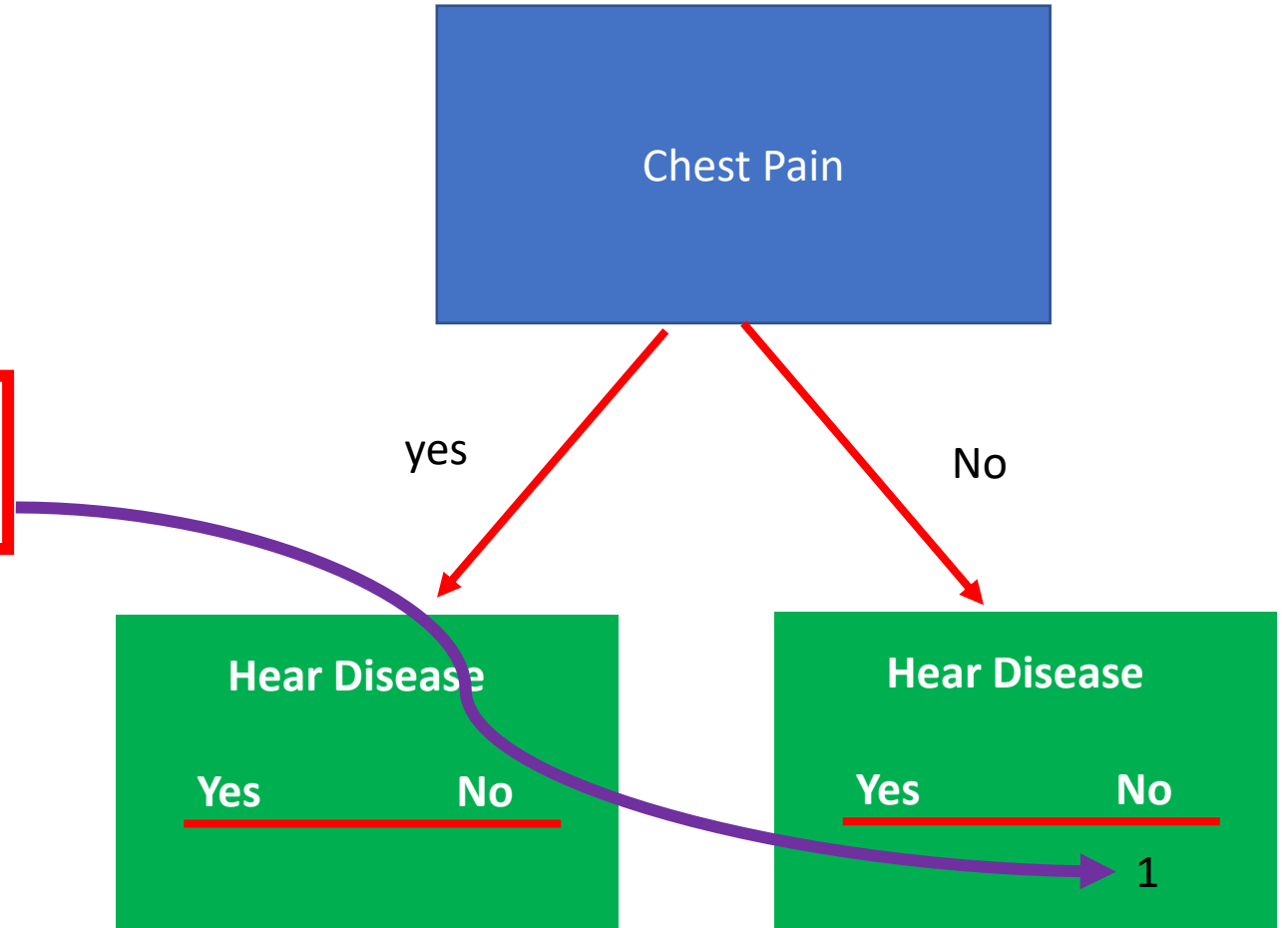
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



# Decision Trees

first patient does not have chest pain and does not have 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..

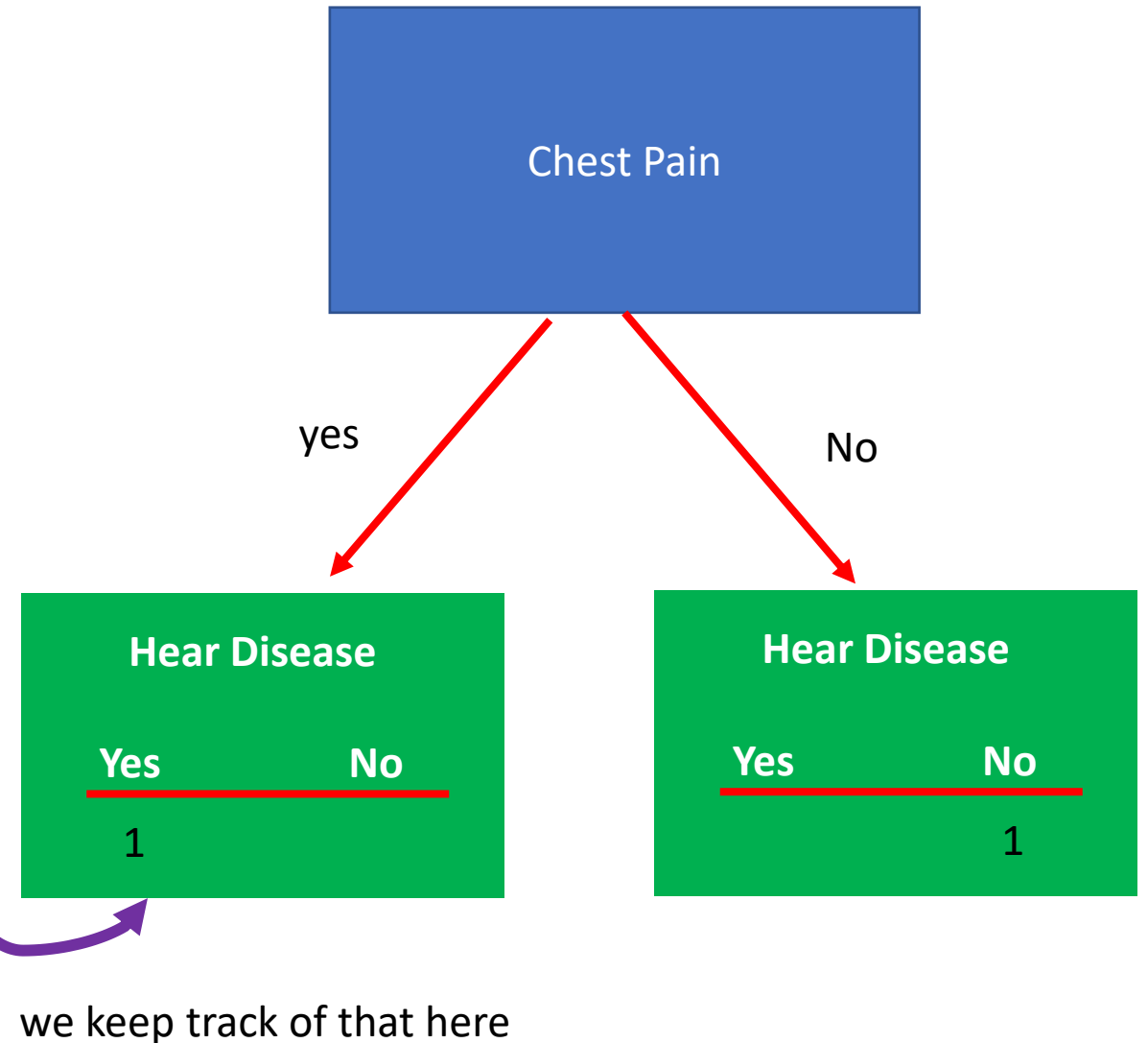


we keep track of that here

# Decision Trees

The second patient has chest pain and has heart disease

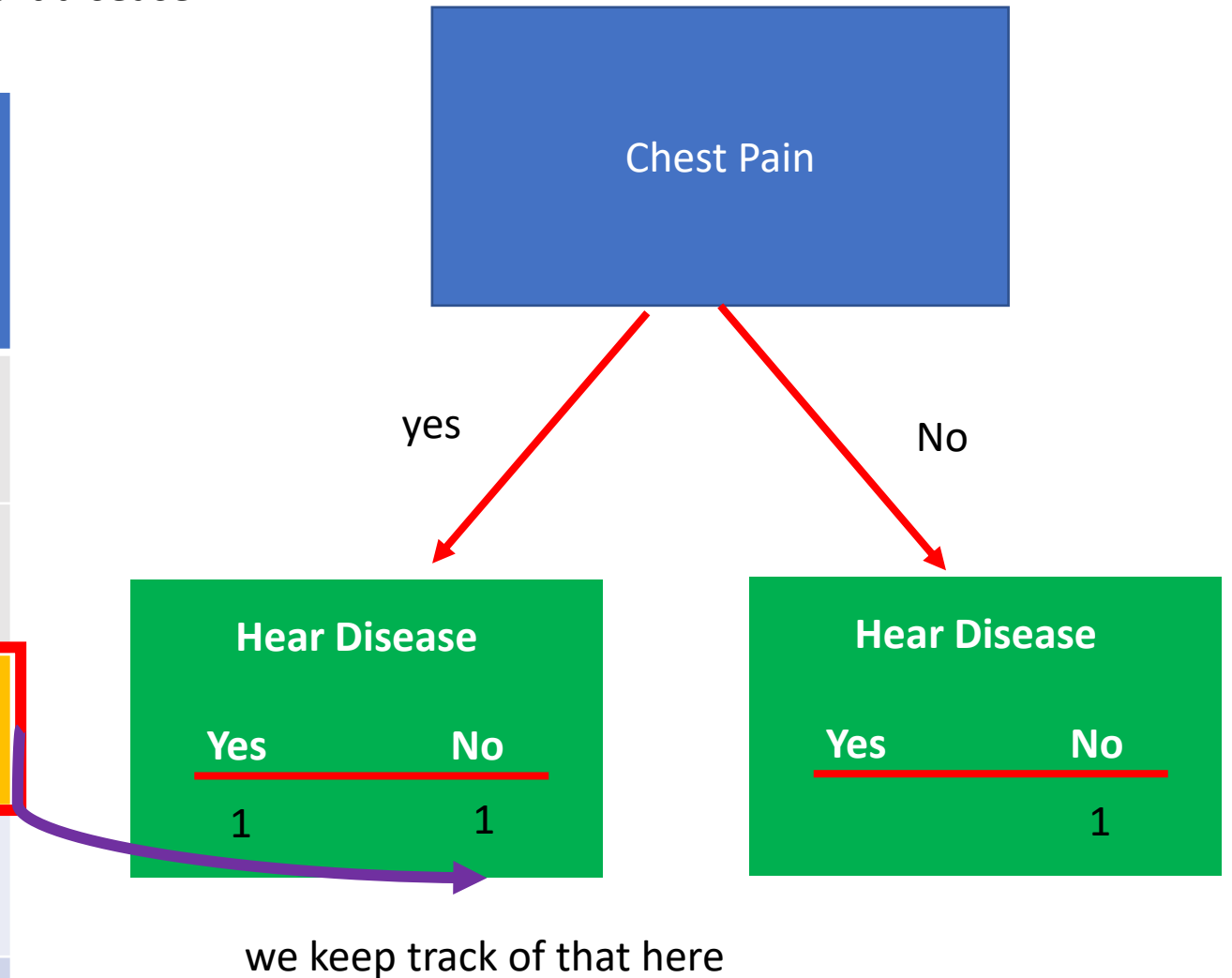
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



# Decision Trees

The 3<sup>rd</sup> patient has chest pain but does not have heart disease

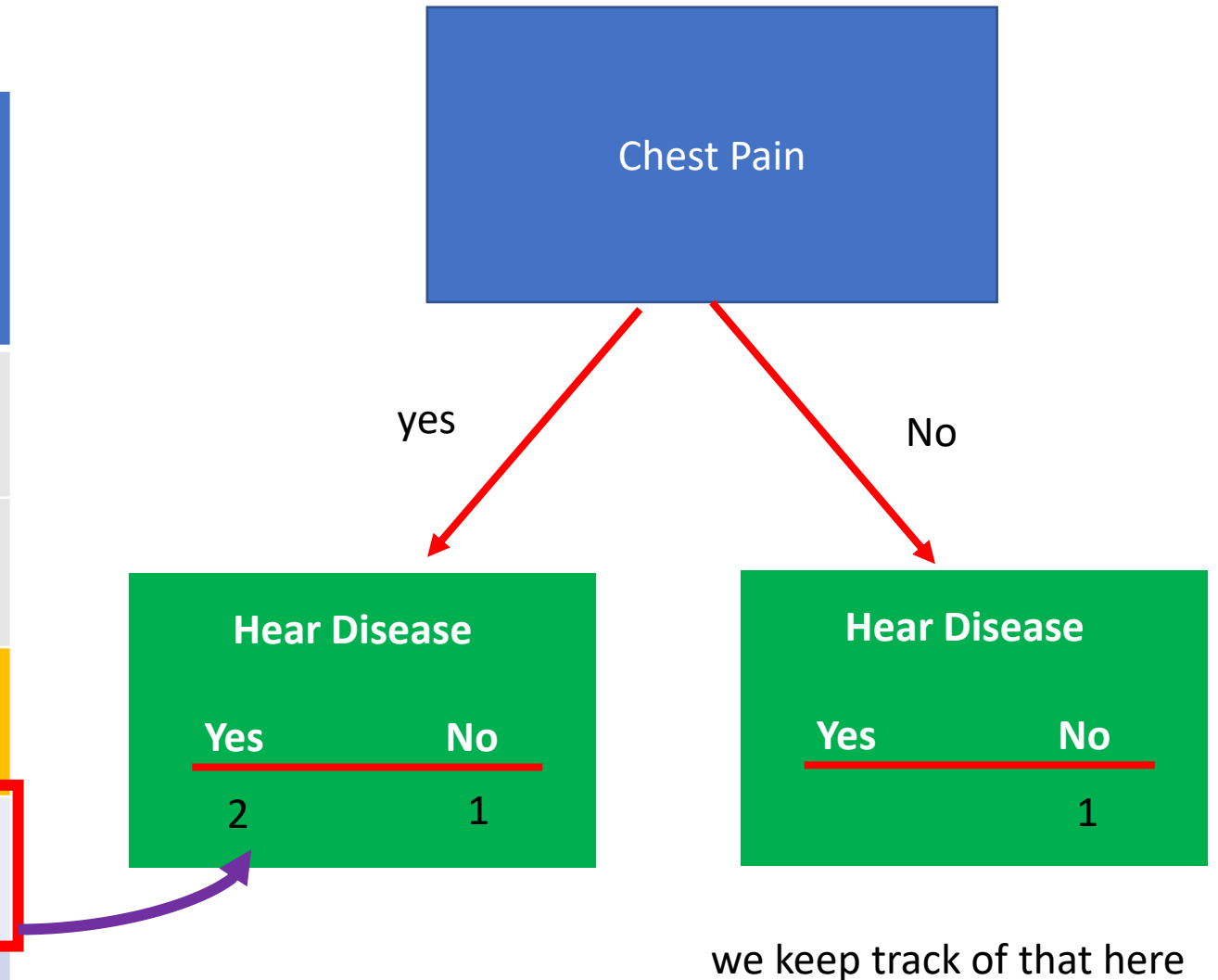
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



# Decision Trees

The 4<sup>th</sup> patient has chest pain and has heart disease

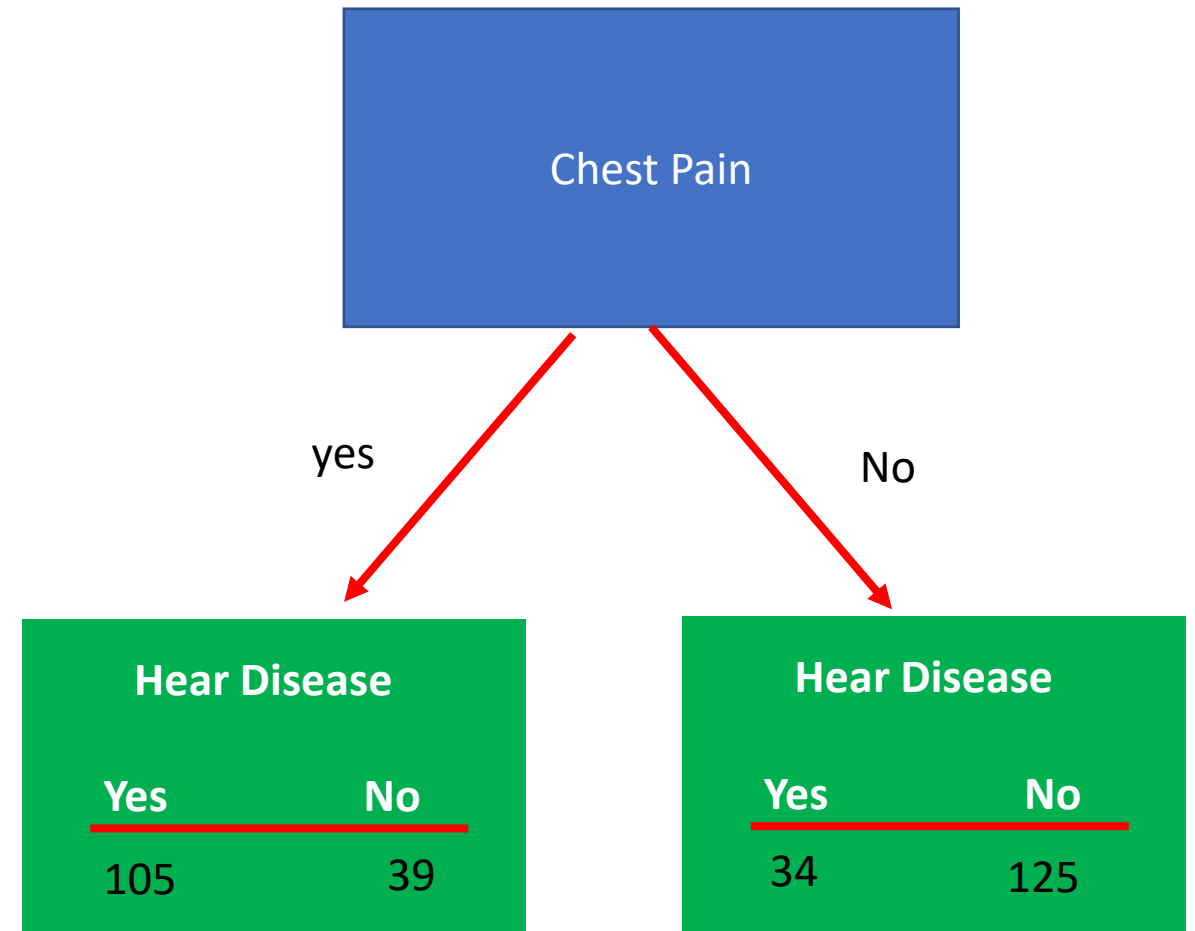
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



# Decision Trees

The 4<sup>th</sup> patient has chest pain and has 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..

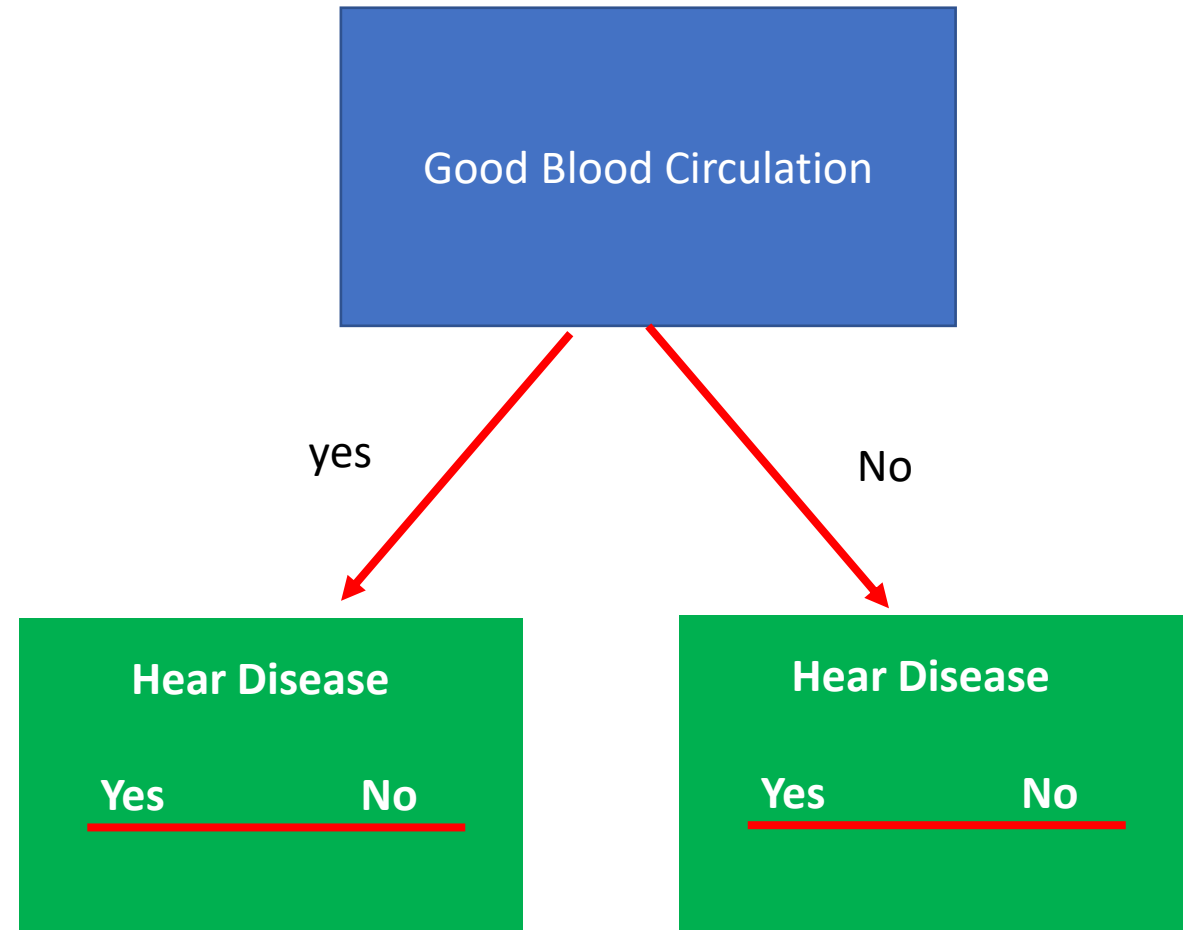


We looked at chest pain and heart disease for all 303 patients in this study

# Decision Trees

Now we do the exact same thing  
with for Good Blood Circulation

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..

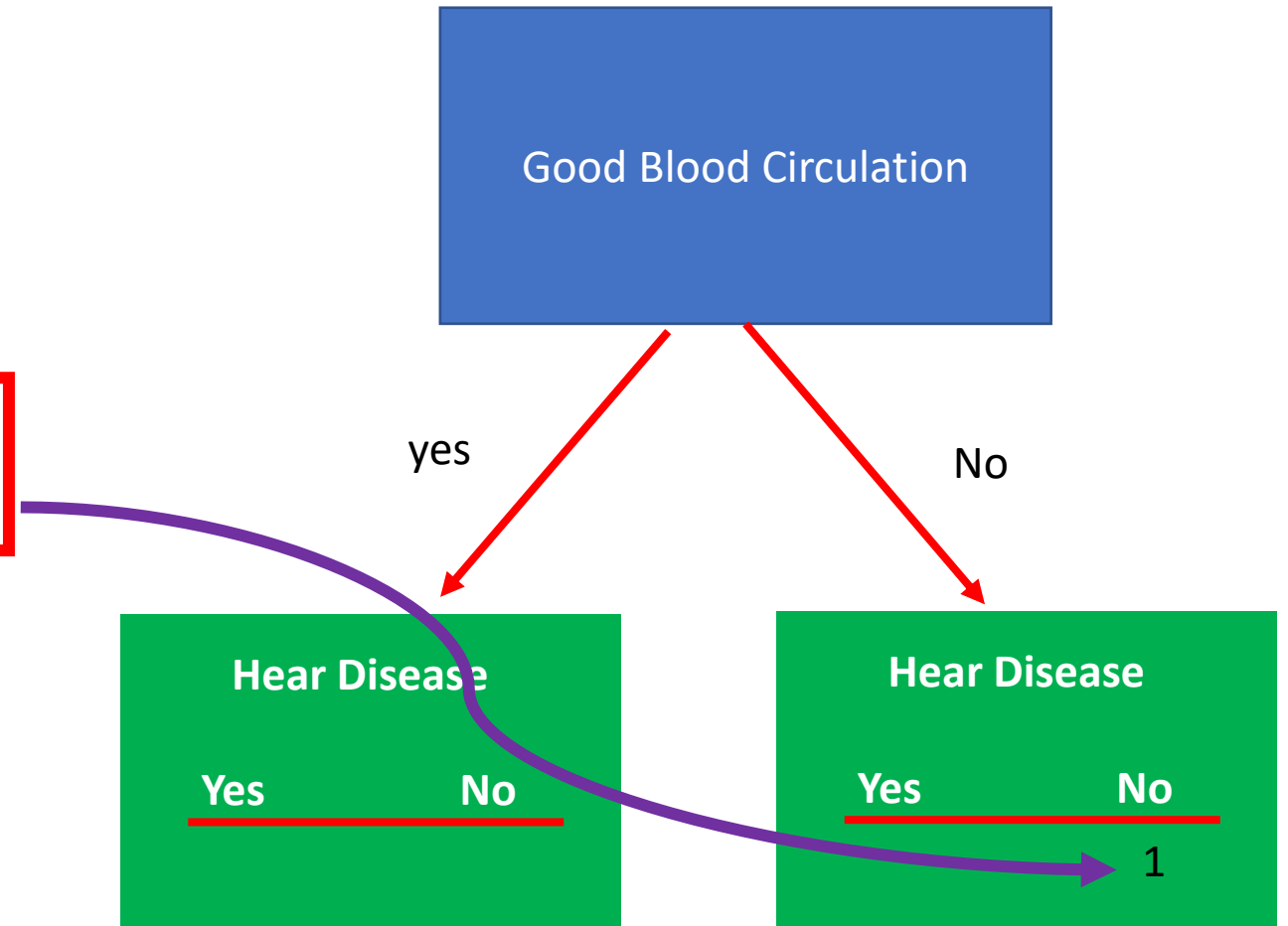




# Decision Trees

Now we do the exact same thing  
with for Good Blood Circulation

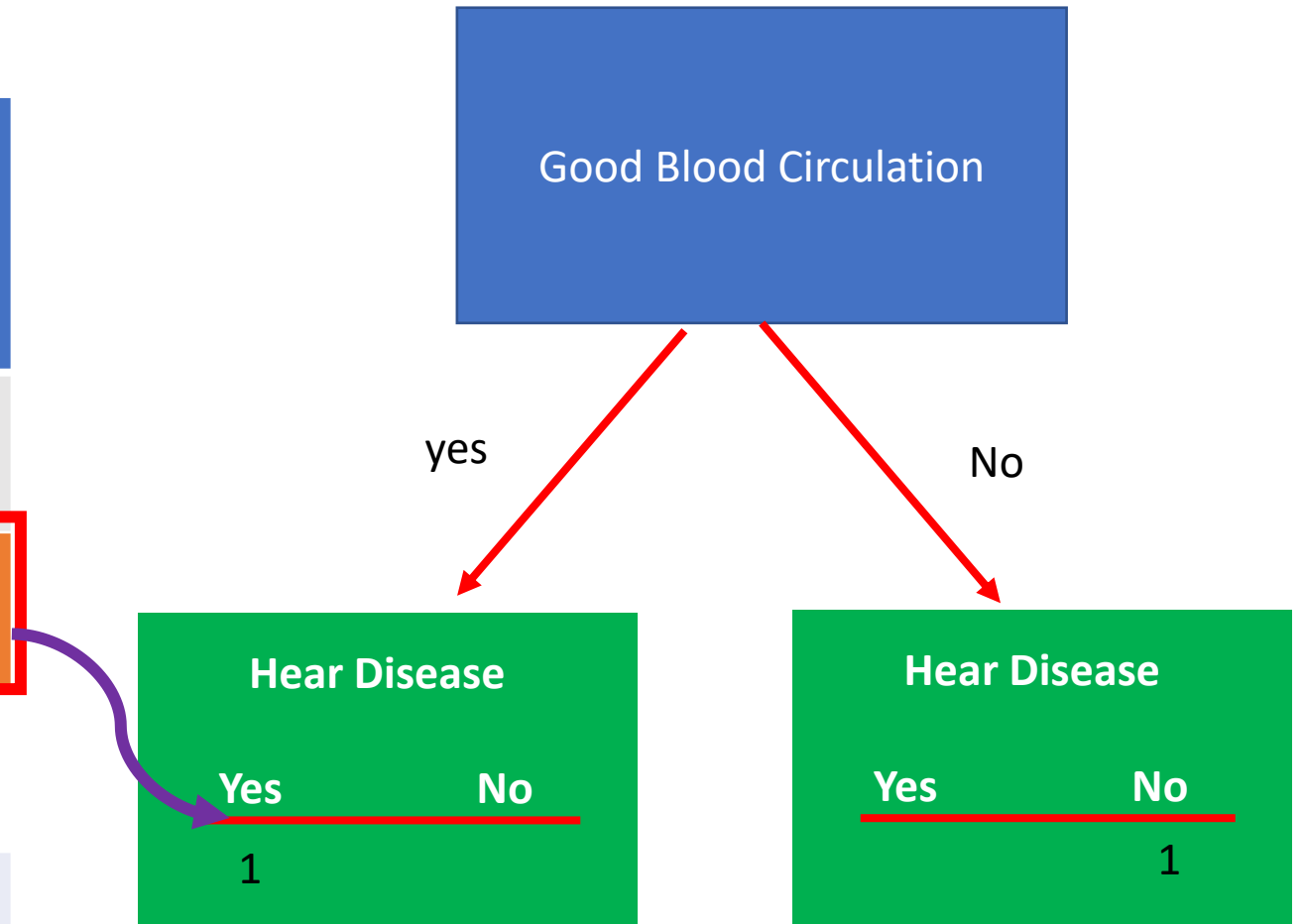
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



# Decision Trees

Now we do the exact same thing  
with for Good Blood Circulation

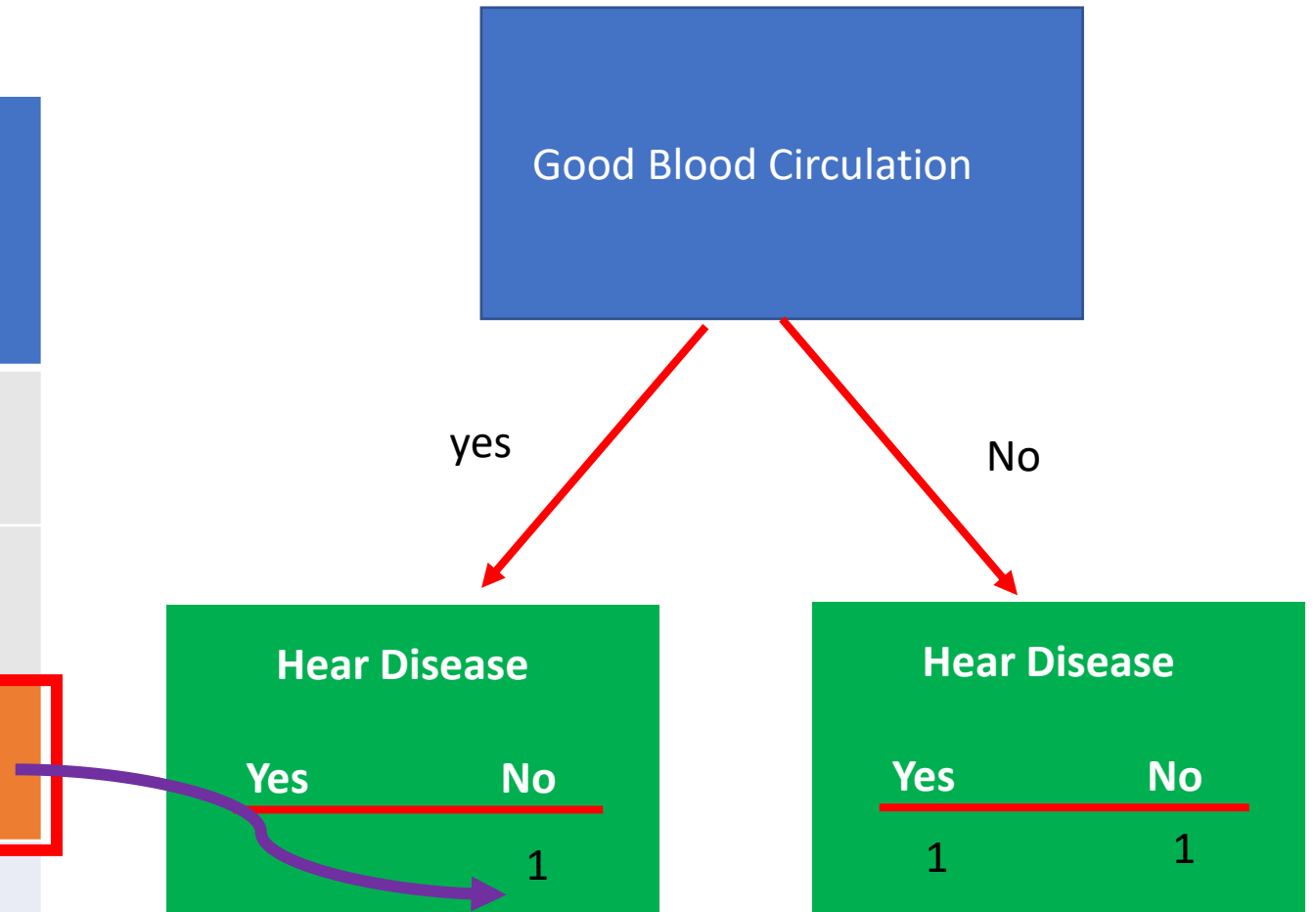
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



# Decision Trees

Now we do the exact same thing  
with for Good Blood Circulation

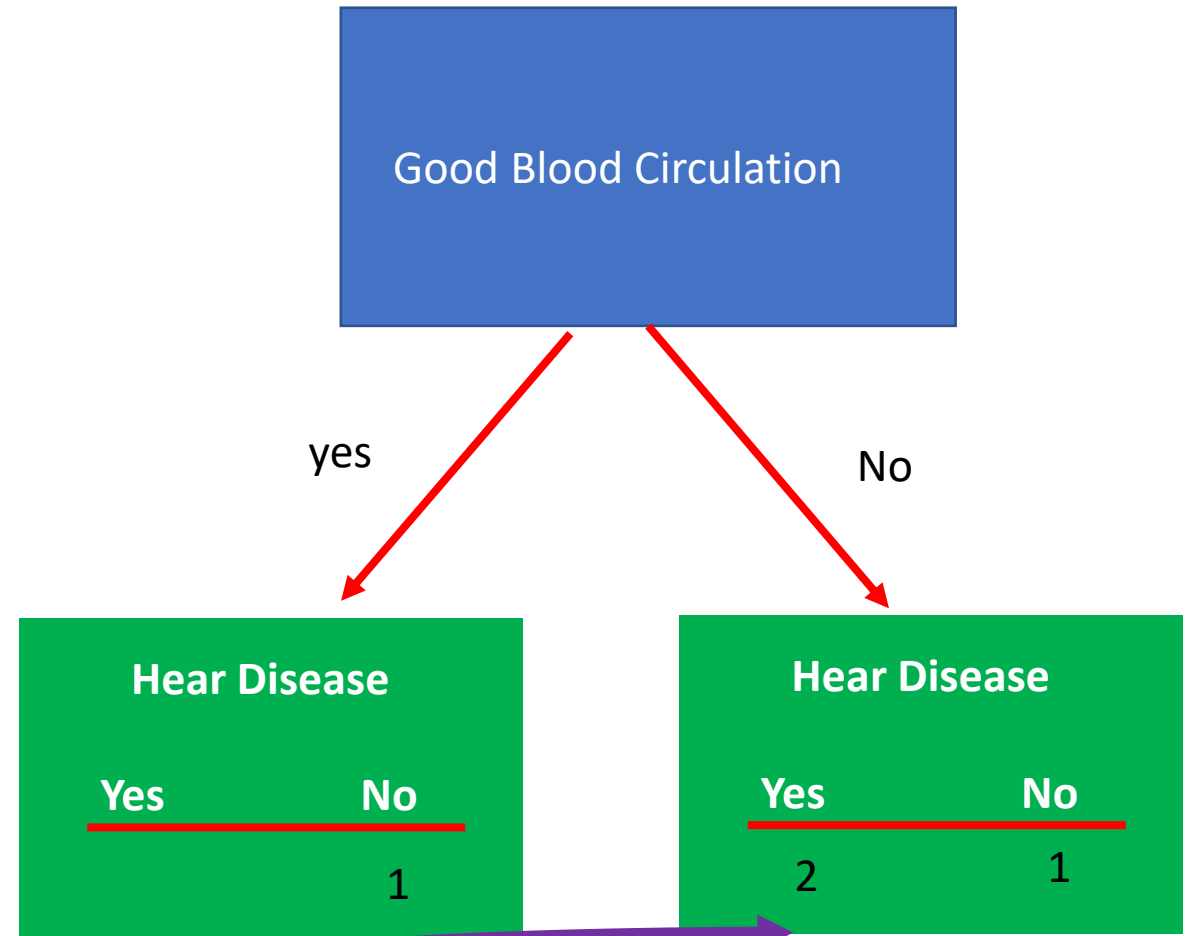
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



# Decision Trees

Now we do the exact same thing  
with for Good Blood Circulation

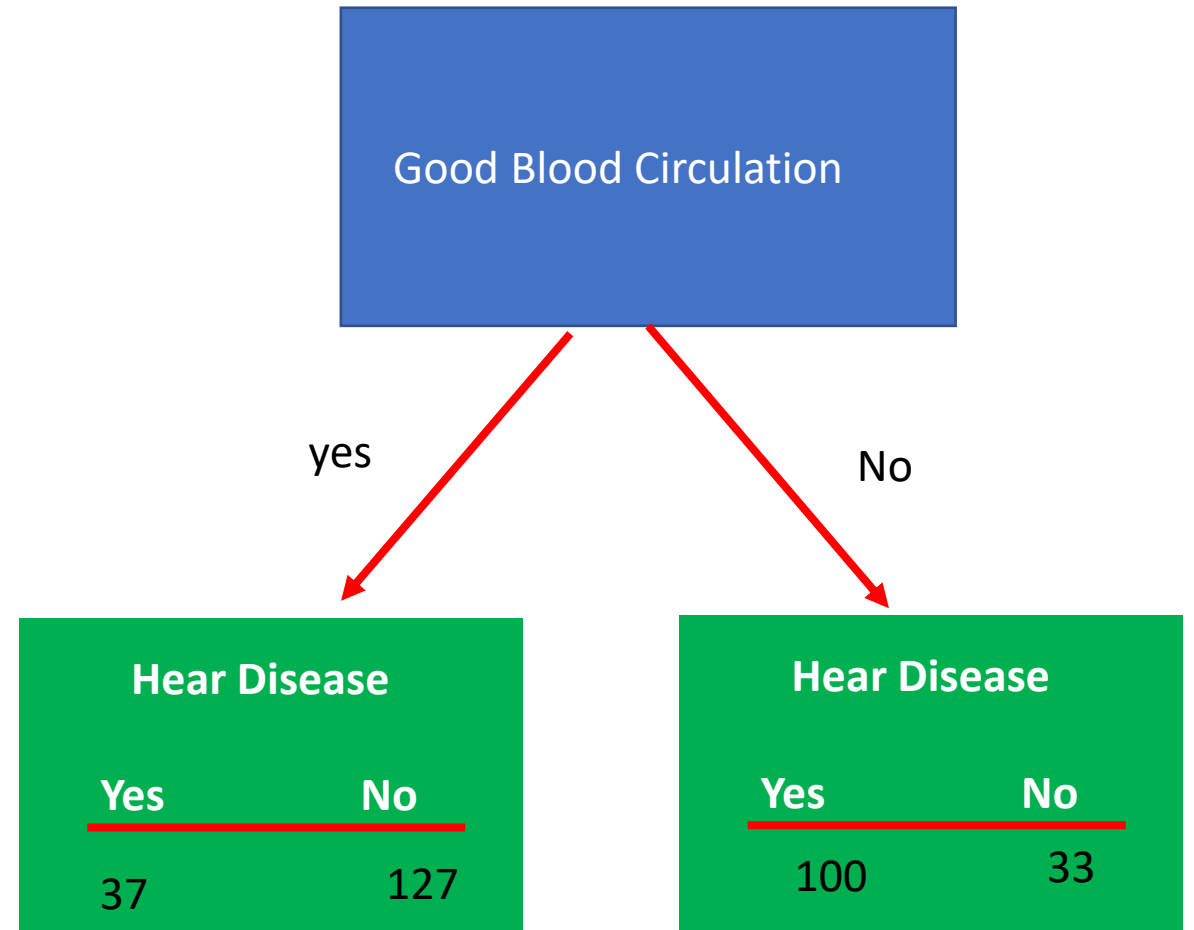
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



# Decision Trees

Now we do the exact same thing  
with for Good Blood Circulation

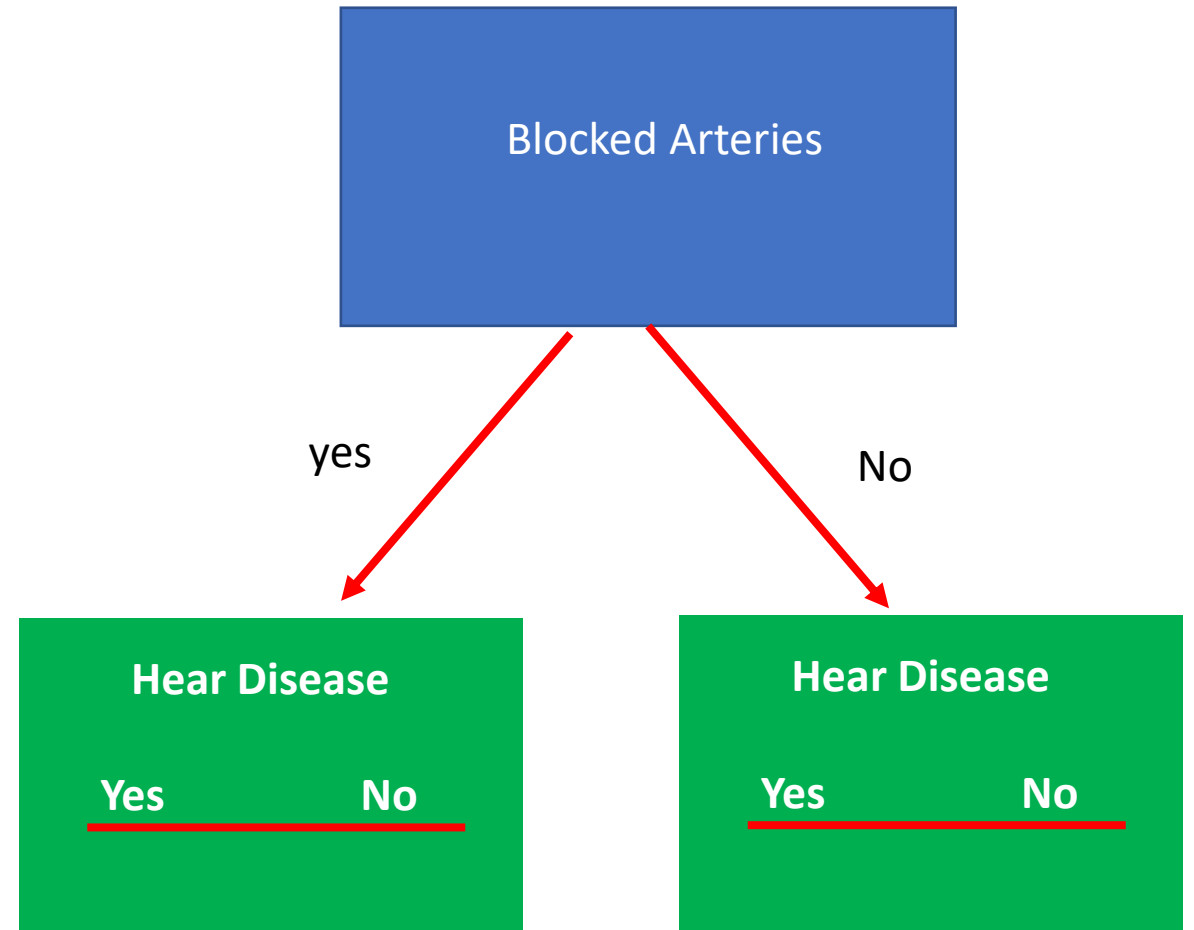
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



# Decision Trees

Lastly, we look at how **Blocked Arteries** separates the patients with and without heart disease

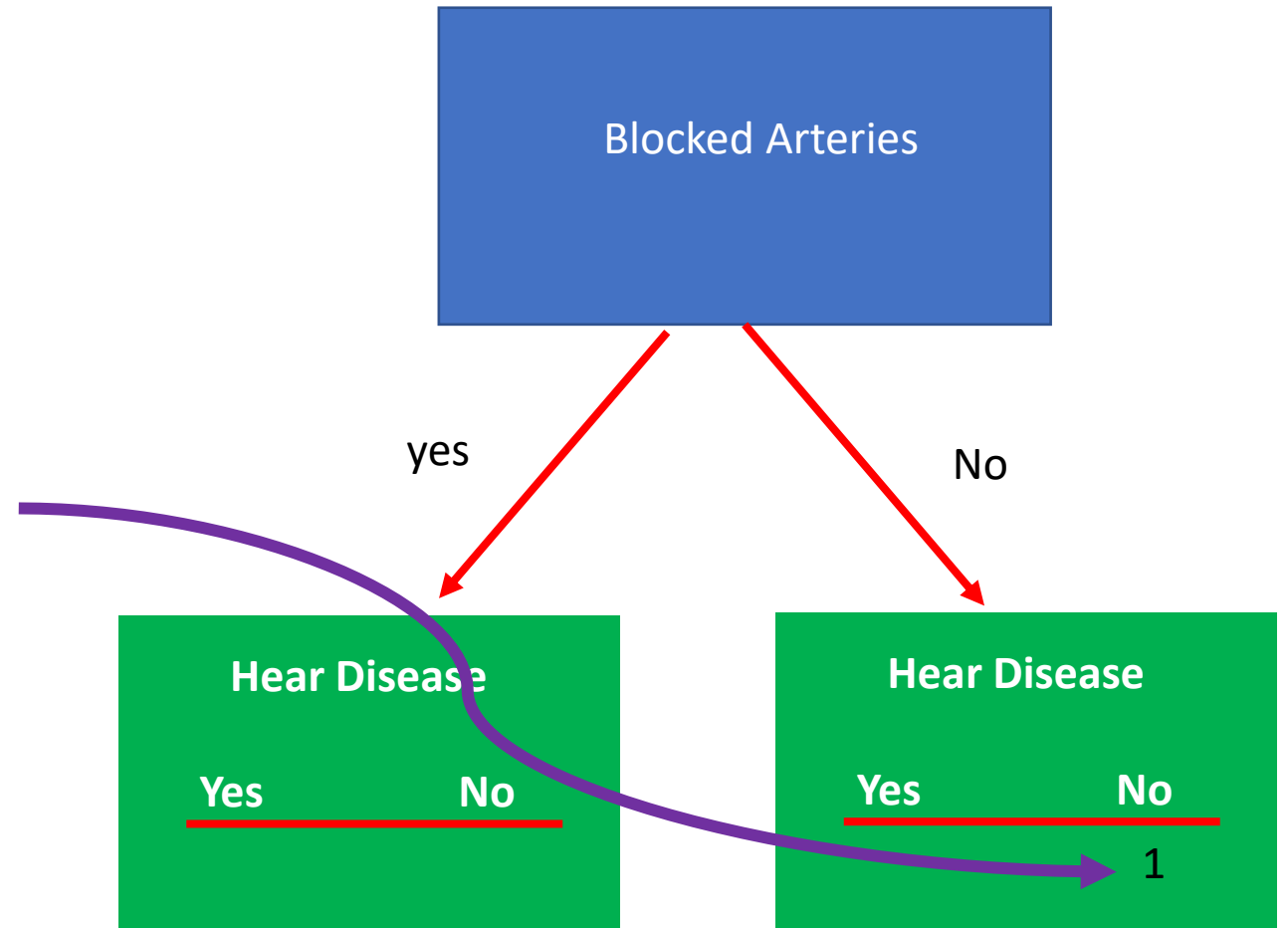
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



# Decision Trees

Lastly, we look at how **Blocked Arteries** separates the patients with and without heart disease

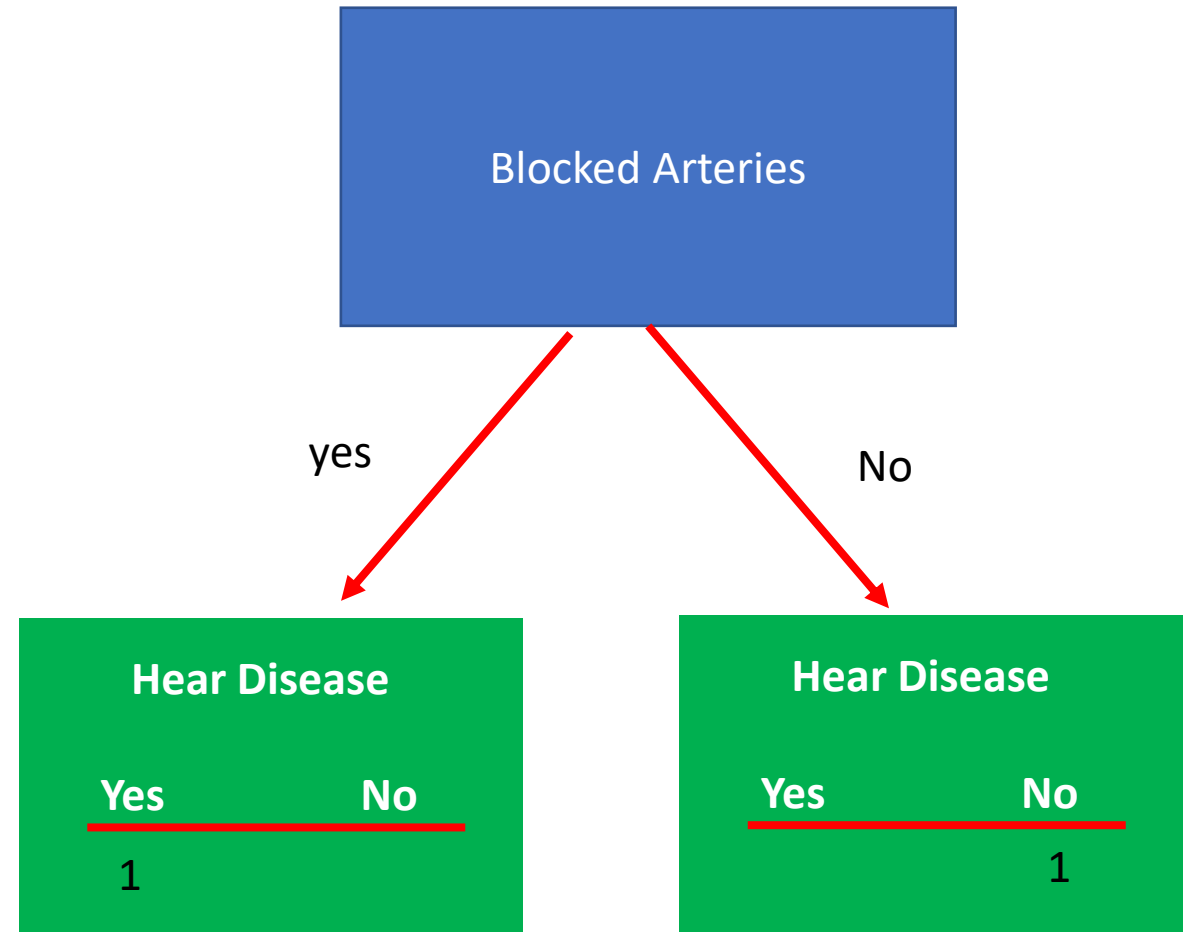
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



# Decision Trees

Lastly, we look at how **Blocked Arteries** separates the patients with and without 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..

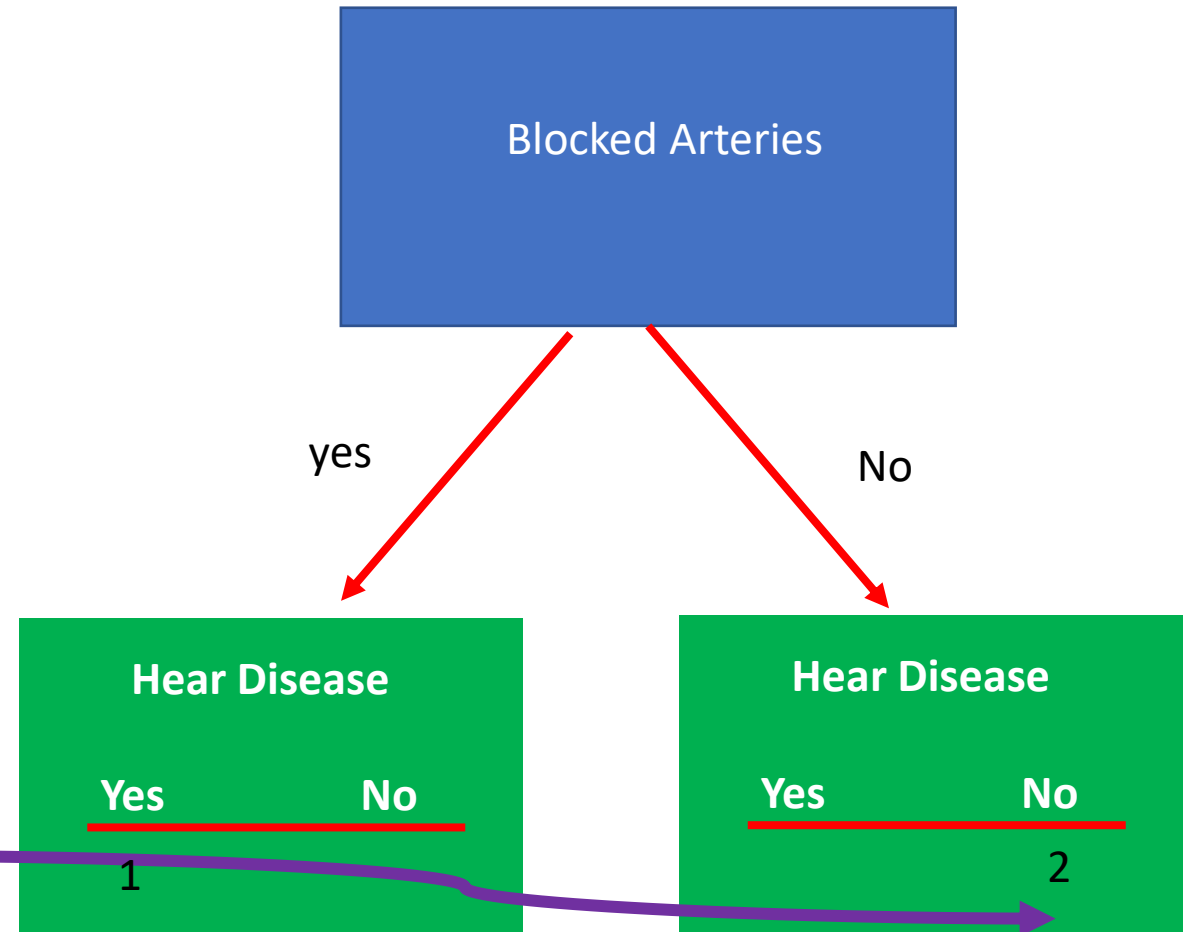




# Decision Trees

Lastly, we look at how **Blocked Arteries** separates the patients with and without heart disease

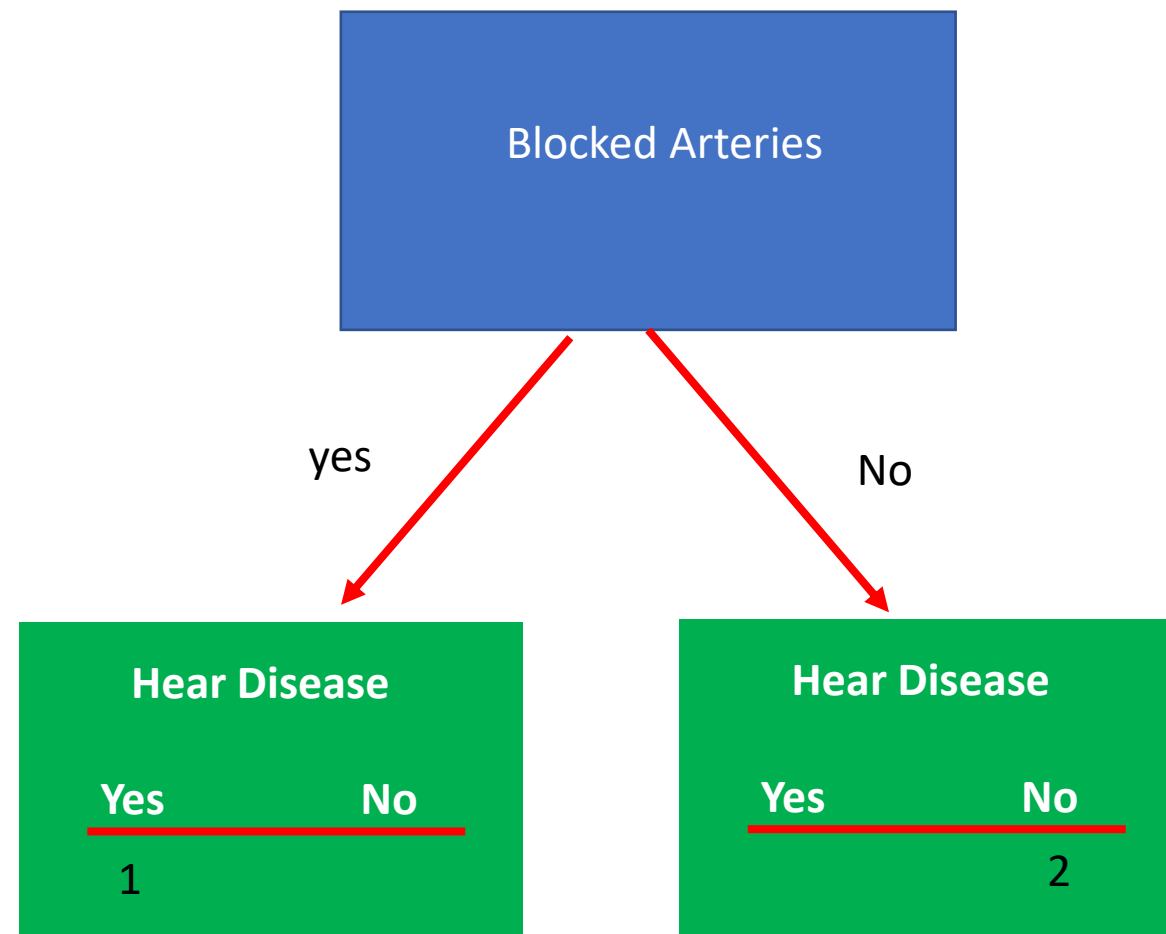
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



# Decision Trees

Lastly, we look at how **Blocked Arteries** separates the patients with and without 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..

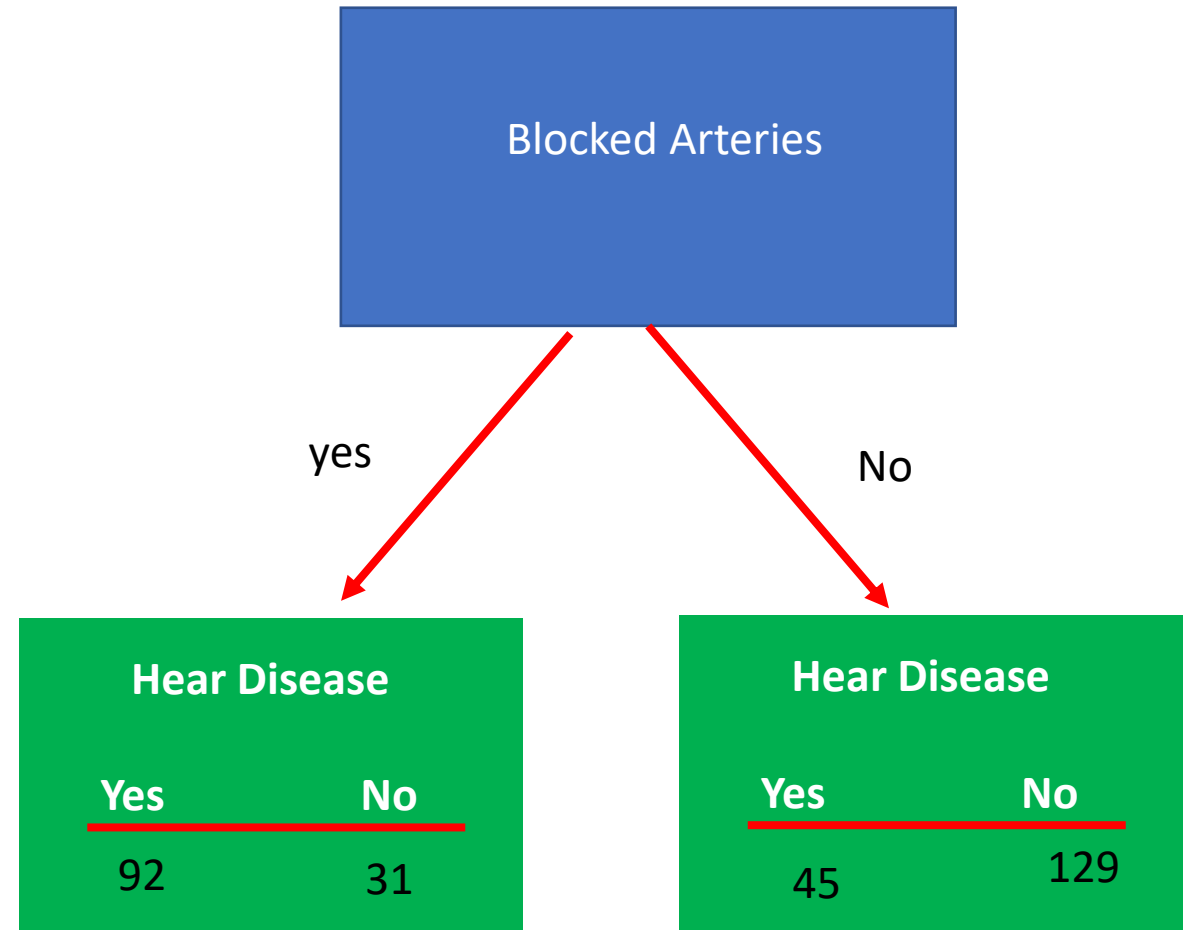


Since we don't know if those patient had blocked arteries  
We will skip it

# Decision Trees

Lastly, we look at how **Blocked Arteries** separates the patients with and without heart disease

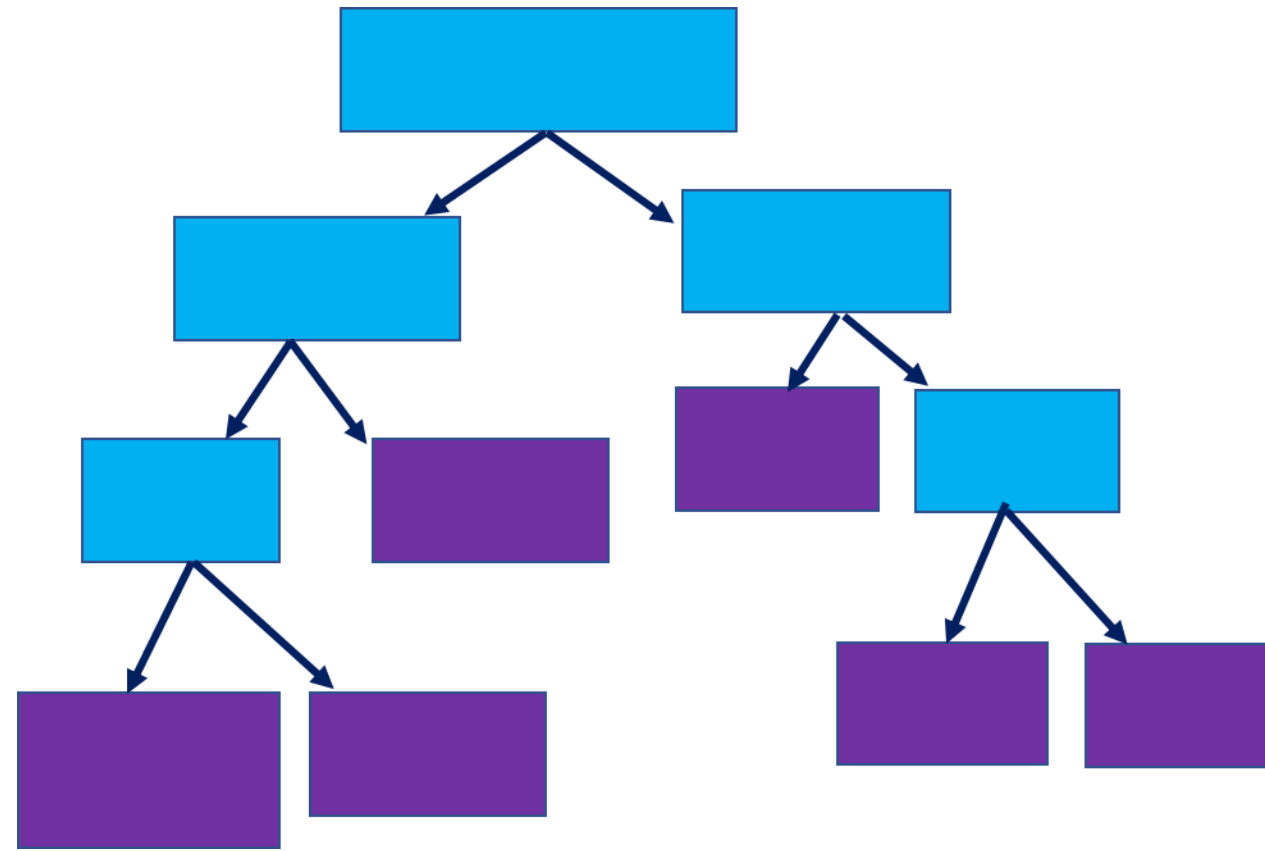
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



# Decision Trees

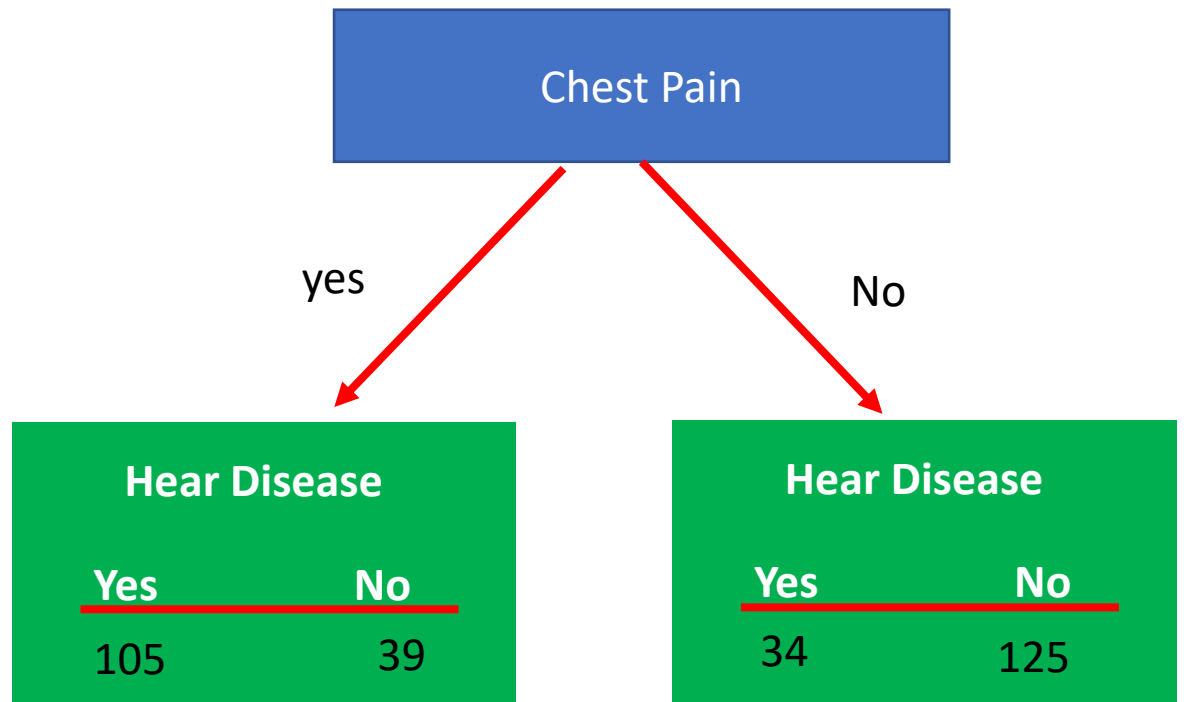
The goal is to decide whether **Chest pain**, **Good Blood Circulation** or **Blocked Arteries** should be the first thing in our Decision tree(**Root Node**)

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..



# Decision Trees

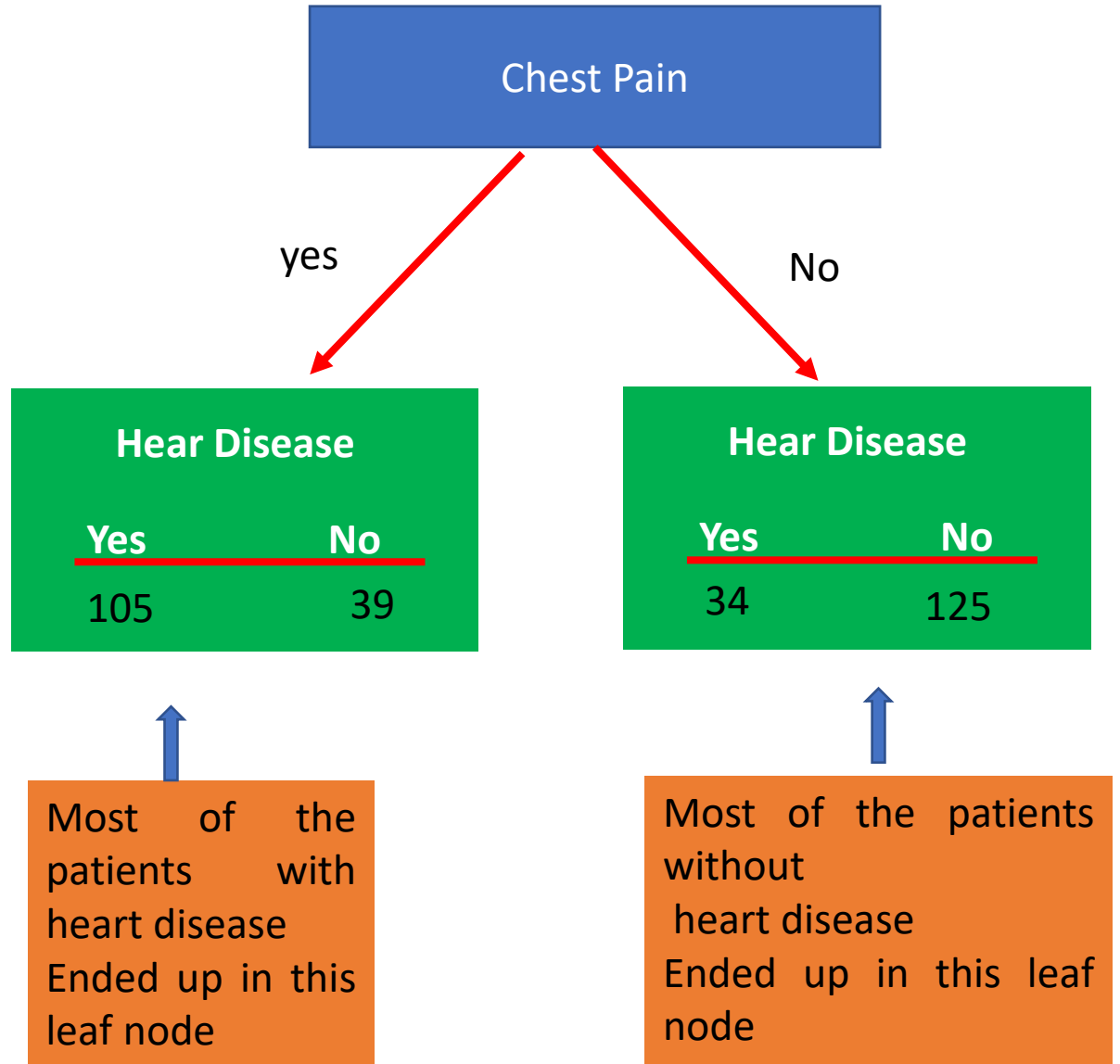
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc..	etc..	etc..	etc..



We looked at who well **chest pain**  
 Separated patients with and without heart disease

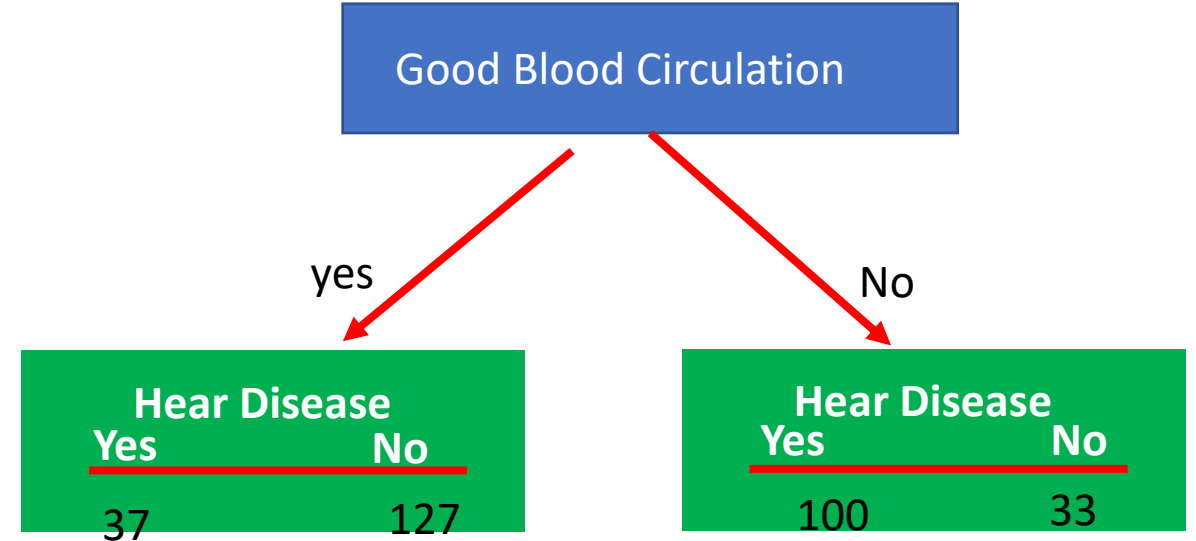
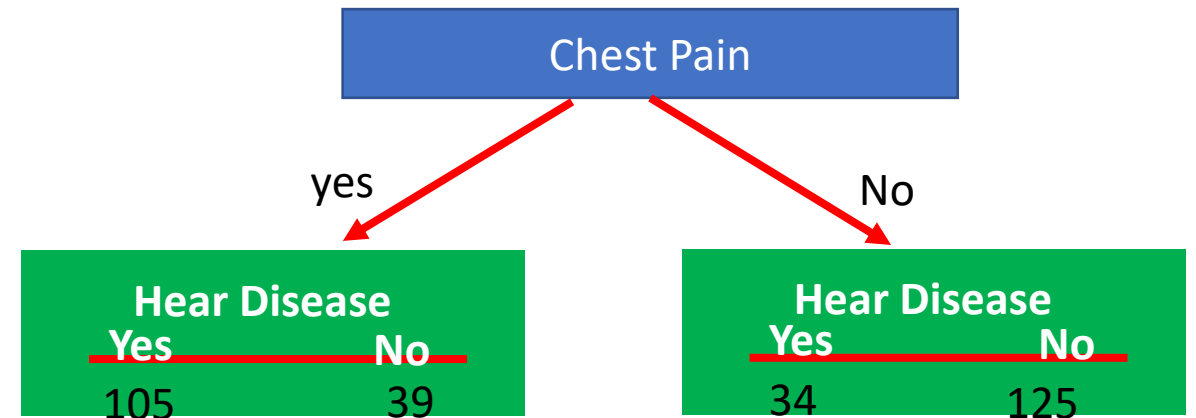
# Decision Trees

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..



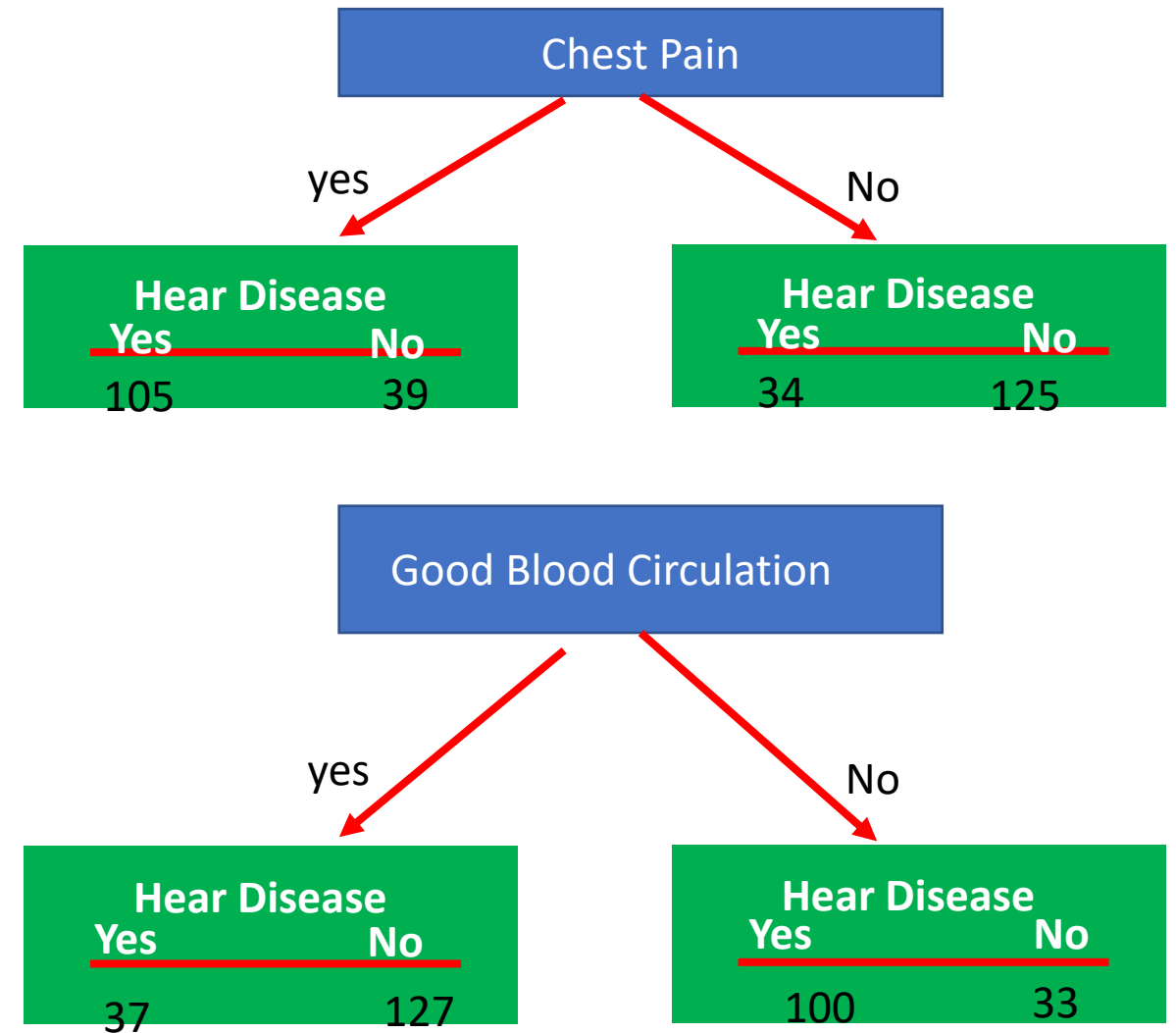
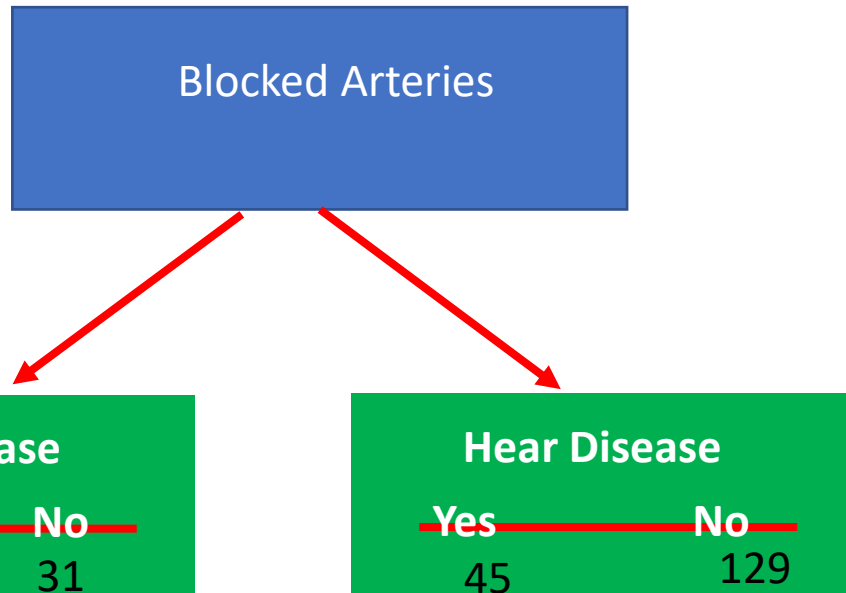
# Decision Trees

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..



Then we looked at how well the **Blood circulation** separated patients with and without heart disease

# Decision Trees

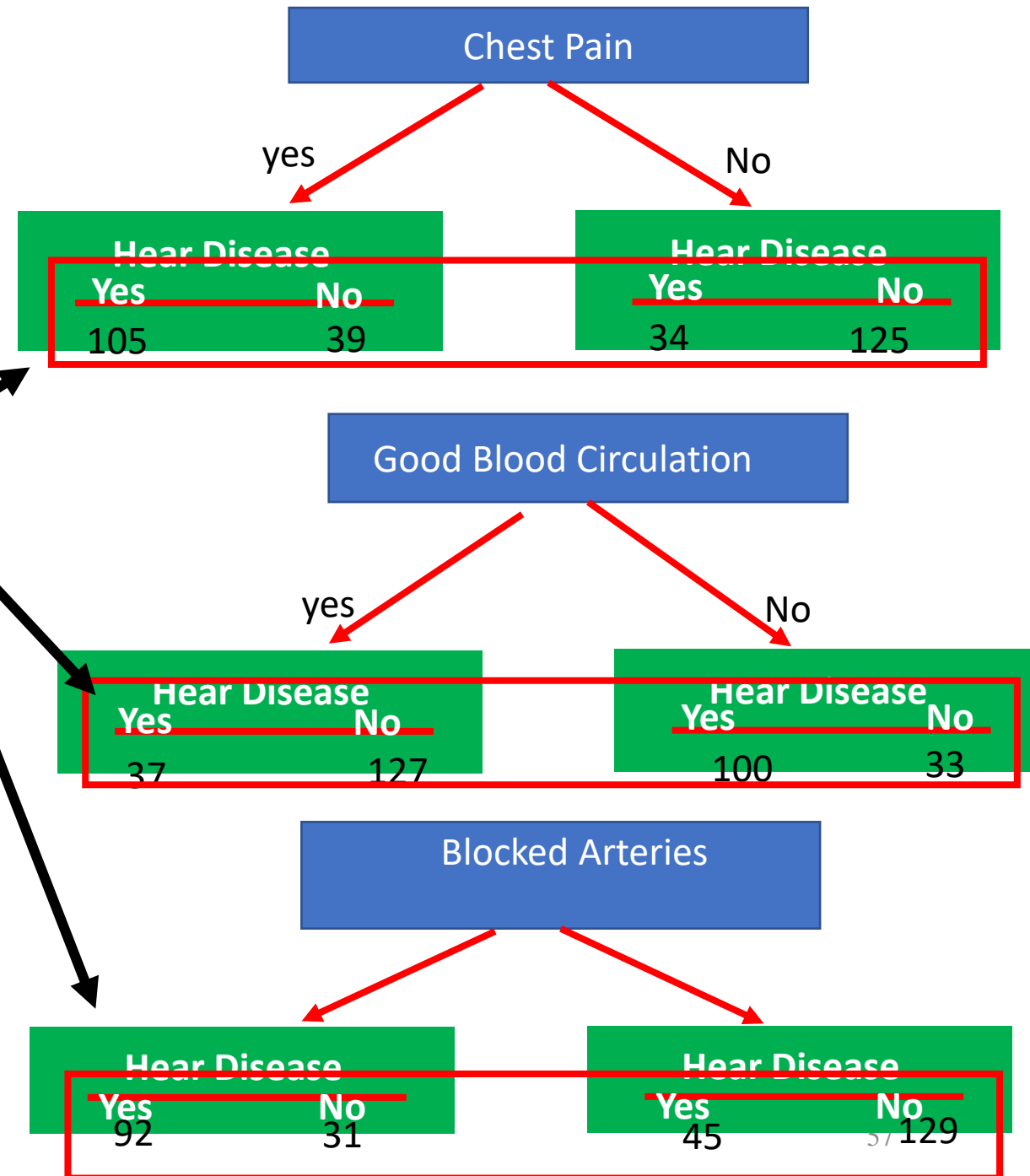


Then we looked at how well the **Block Arteries** separated patients with and without heart disease



# Decision Trees

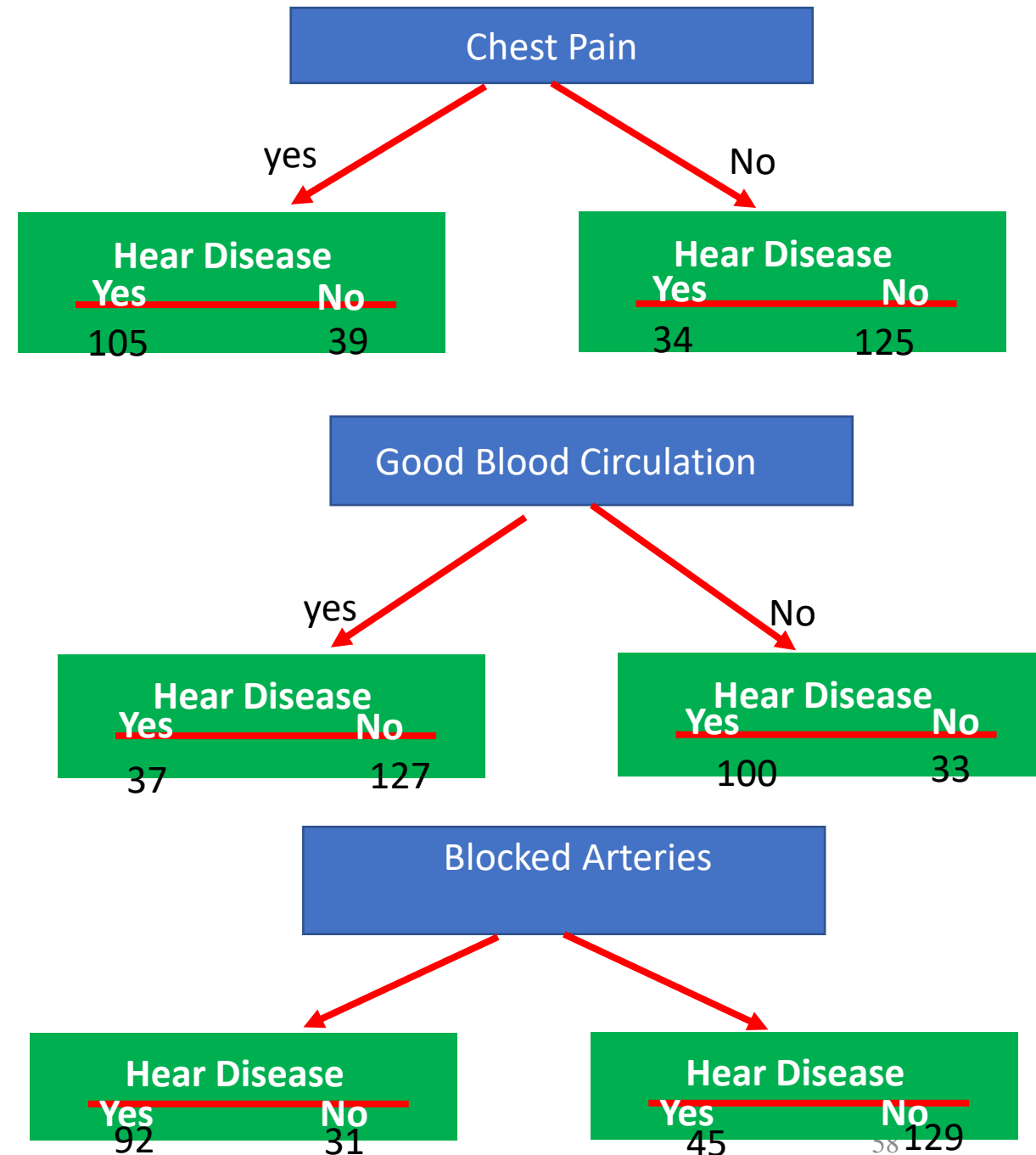
The total number of patients with heart disease is different for chest pain, good blood circulation and blocked arteries because some patients had measurements for chest pain but not for blocked arteries etc.



# Decision Trees

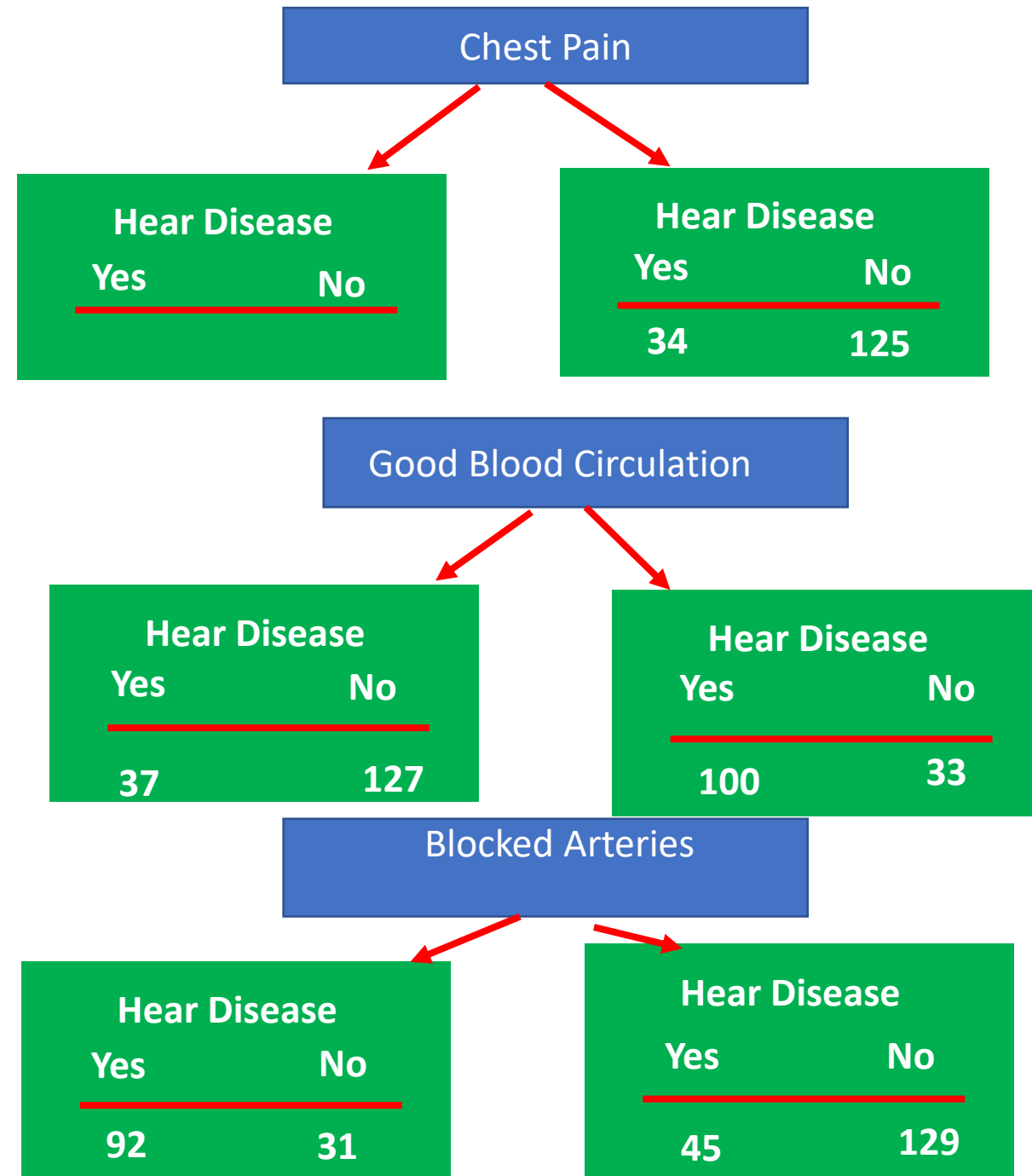
None of the leaf nodes are **100 % Yes heart disease** or **100 % No heart disease**, they are all considered “**impure**”

To determine which separation is best, we need a way to measure and compare “**impurity**”



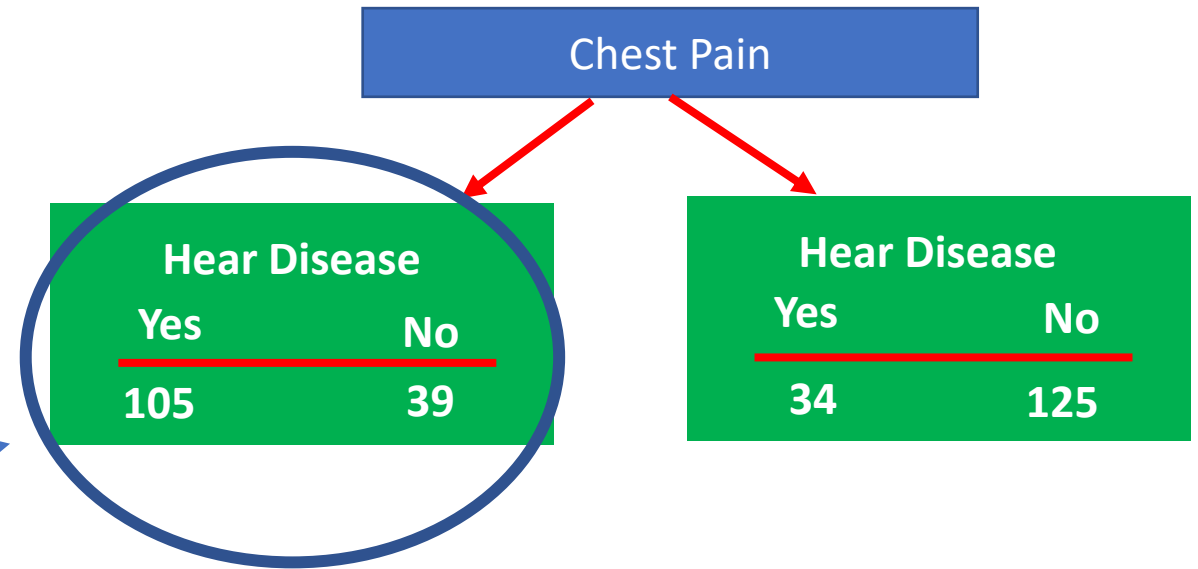
# Decision Trees

There are a bunch of ways to measure impurity, we are going to discuss or focus on a very popular one called “**Gini**”



# Decision Trees

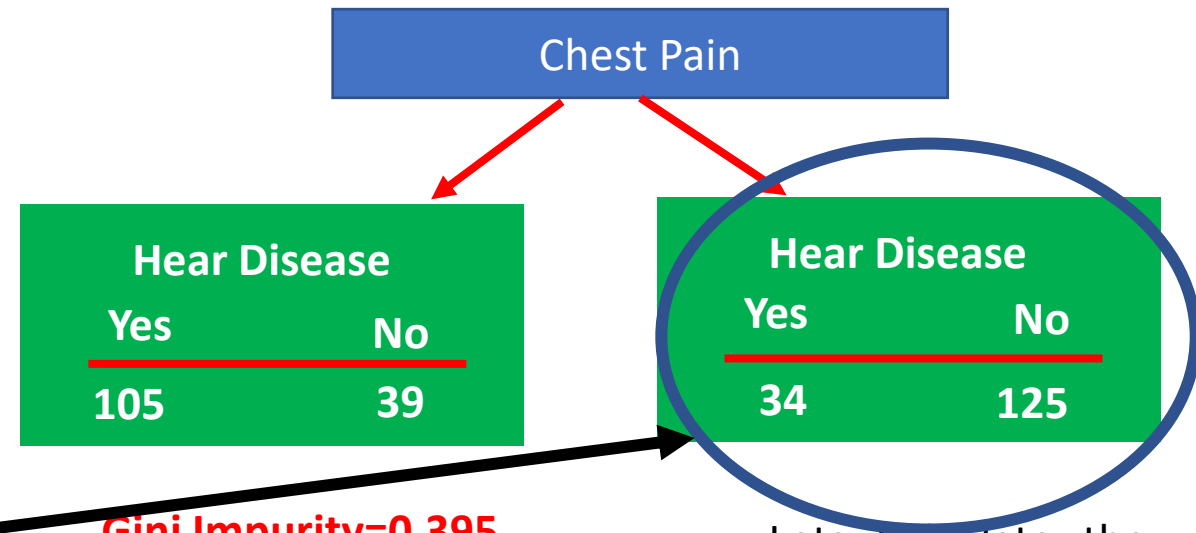
Lets start by calculating Gini impurity for Chest Pain



For this leaf, the Gini impurity= $1-(\text{probability of "yes"})^2-(\text{probability of "no"})^2$   
 $=1-\left(\frac{105}{105+39}\right)^2-\left(\frac{39}{105+39}\right)^2=0.395$

# Decision Trees

Lets start by calculating Gini impurity for Chest Pain



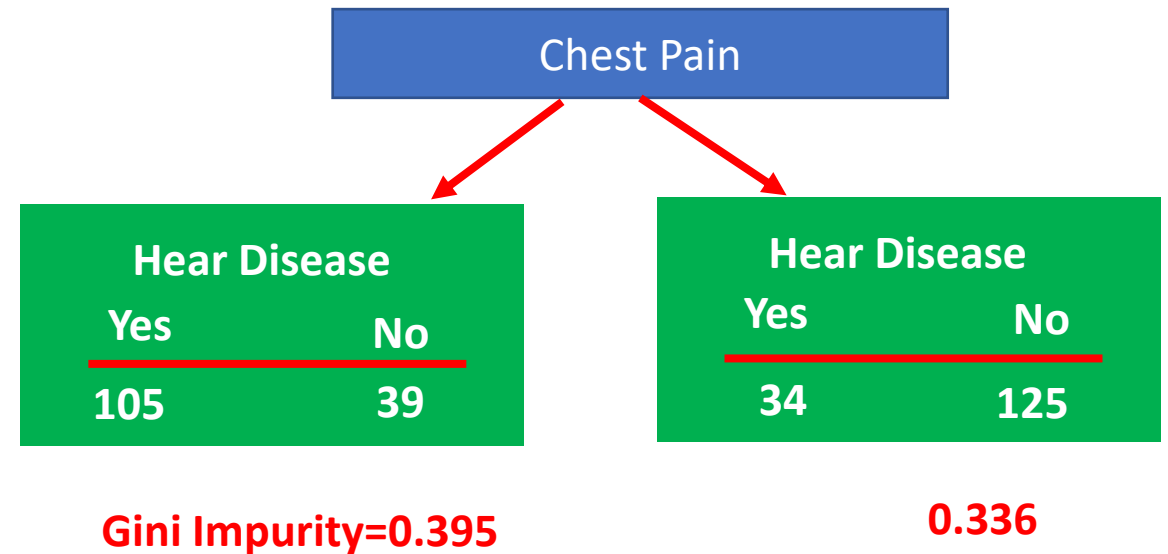
Gini Impurity=0.395

Lets Calculate the Gini impurity For this leaf node

For this leaf, the Gini impurity= $1-(\text{probability of "yes"})^2-(\text{probability of "no"})^2$   
 $=1-\left(\frac{34}{34+125}\right)^2-\left(\frac{125}{34+125}\right)^2=0.336$

# Decision Trees

Lets start by calculating Gini impurity for Chest Pain

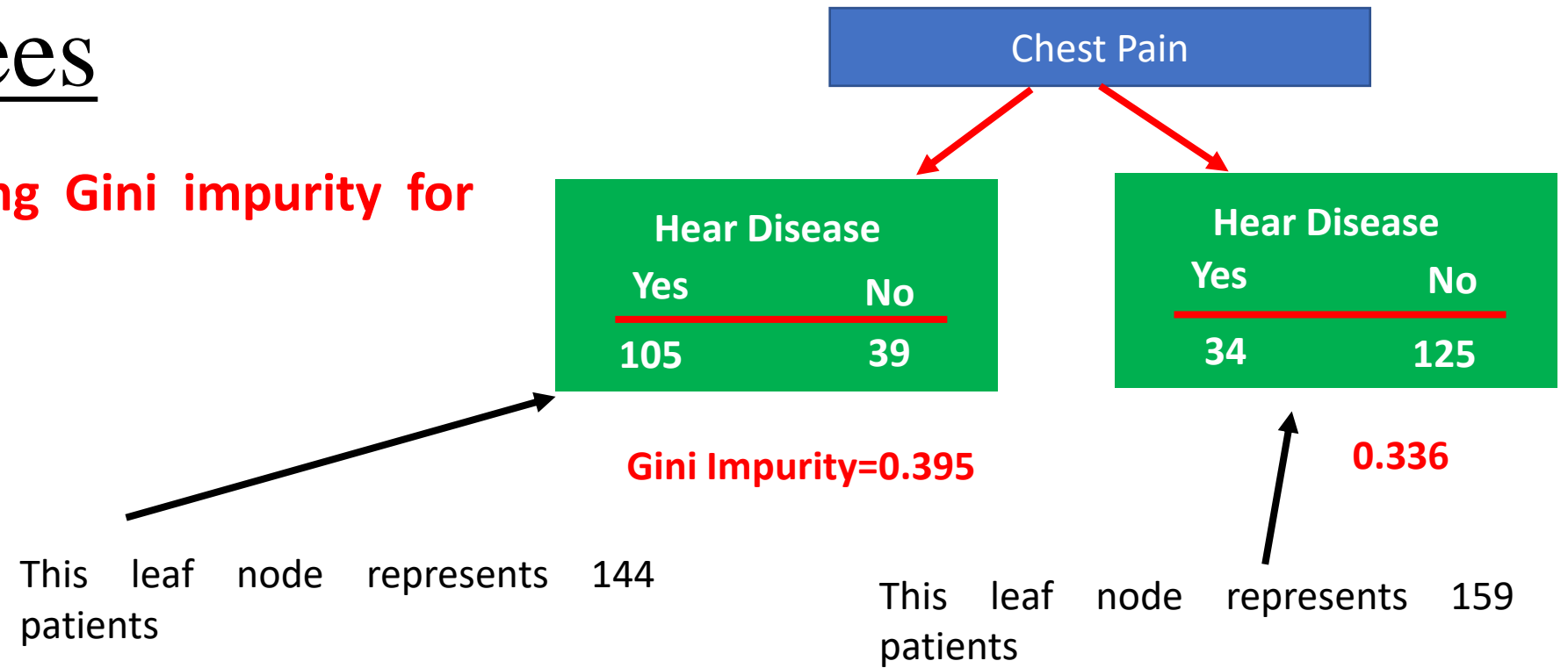


Lets we have measured Gini impurity for both leaf nodes , we can calculate the total Gini impurity for using chest pain to separate the patients with and without heart disease

Lets we have measured Gini impurity for both leaf nodes , we can calculate the total Gini impurity for using chest pain to separate the patients with and without heart disease

# Decision Trees

Lets start by calculating Gini impurity for Chest Pain

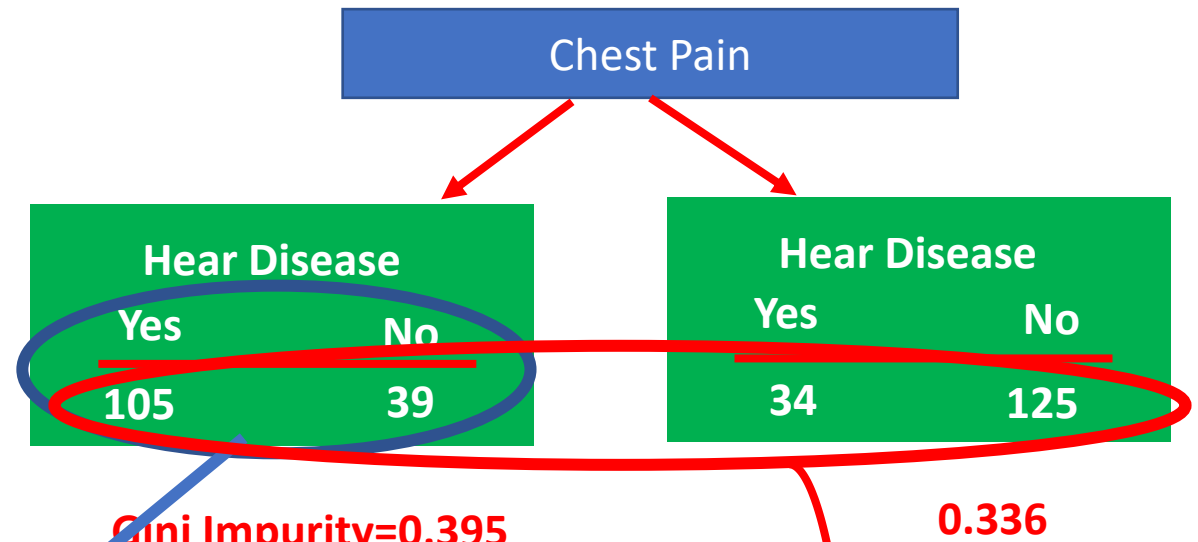


The leaf node do not represent the same number of patients, thus

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**

# Decision Trees

Lets start by calculating Gini impurity for Chest Pain



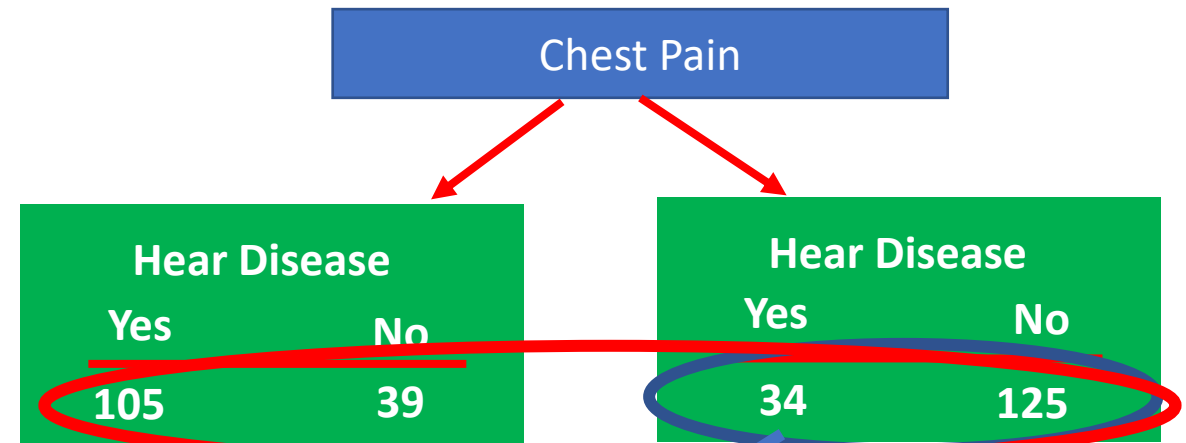
Thus , Gini impurity for chest pain=weighted average of the Gini impurities for the leaf node

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



# Decision Trees

Lets start by calculating Gini impurity for Chest Pain



Gini Impurity=0.395

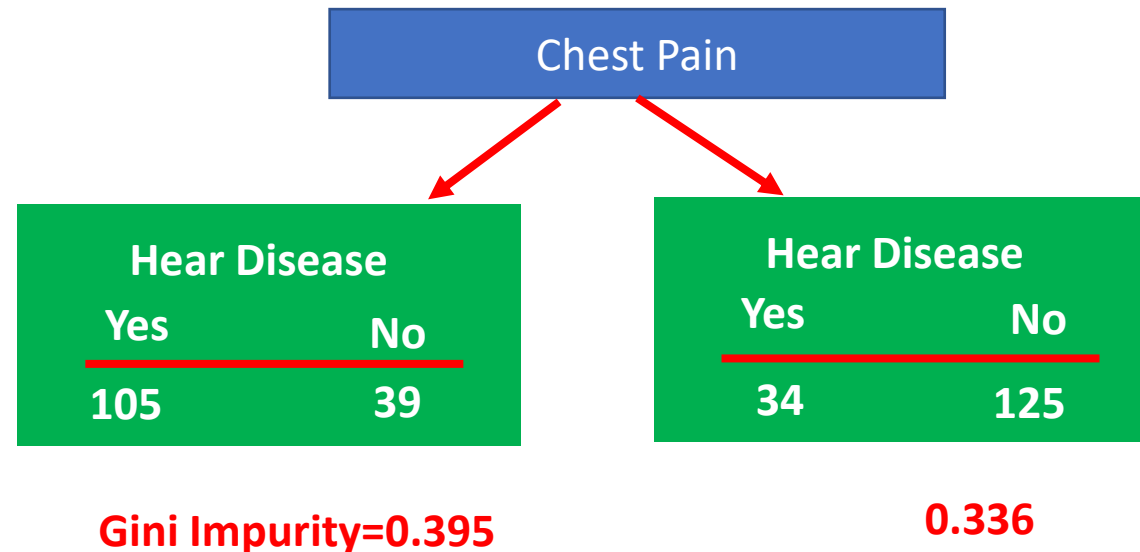
0.336

Thus , Gini impurity for chest pain=weighted average of the Gini impurities for the leaf node

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

# Decision Trees

Lets start by calculating Gini impurity for Chest Pain



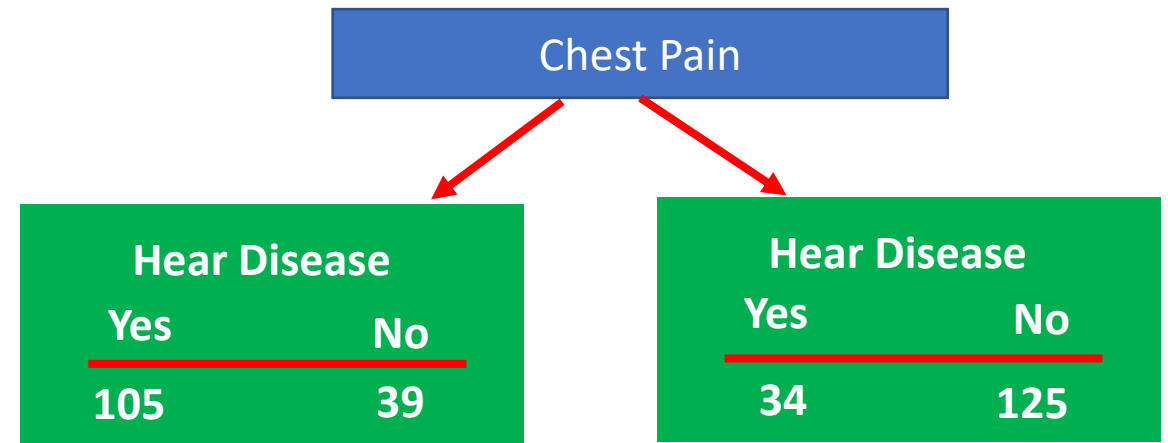
Thus , Gini impurity for chest pain=weighted average of the Gini impurities for the leaf node

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

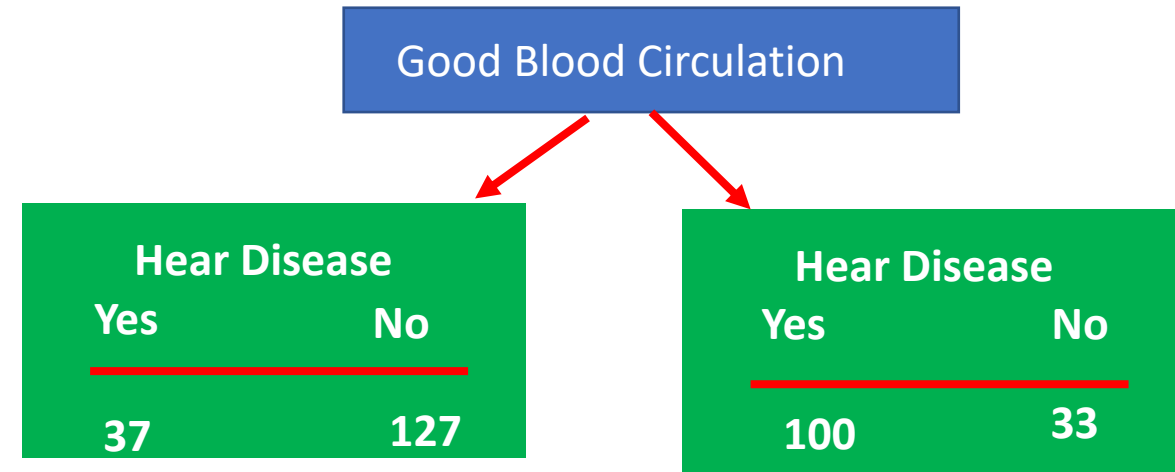
$$=0.364$$

# Decision Trees

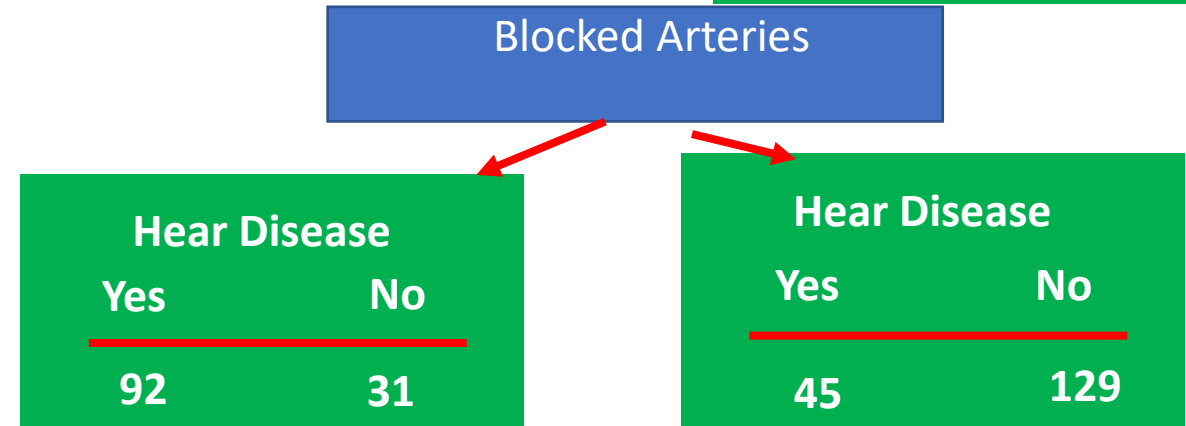
Gini impurity for chest pain=0.364



Gini impurity for Good Blood Circulation=0.360



Gini impurity for Blocked Arteries=0.381

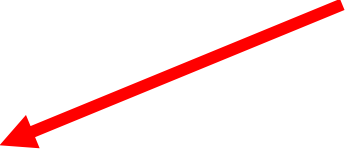


# Decision Trees

Gini impurity for chest pain=0.364

Gini impurity for Good Blood Circulation=0.360

**Good Blood Circulation** has the lowest Impurity (it separate the patients with and Without heart disease best)



Gini impurity for Blocked Arteries=0.381

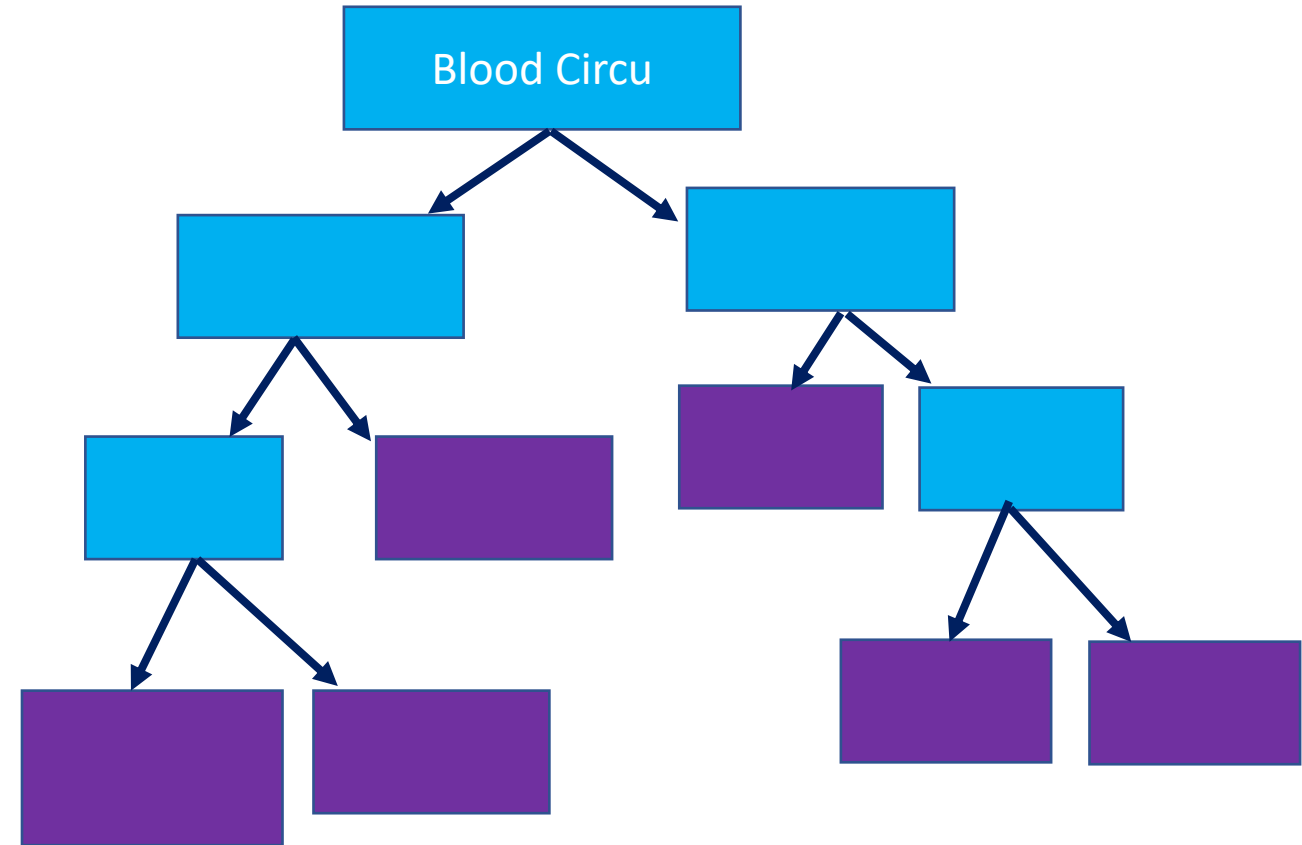
# Decision Trees

Good Blood Circulation

Hear Disease		Hear Disease	
Yes	No	Yes	No
37	127	100	33

when we divided all the patient using  
Good Blood Circulation We ended up  
with impure leaf node

Each leaf contained a mixture of patients  
with and without heart  
disease



# Decision Trees

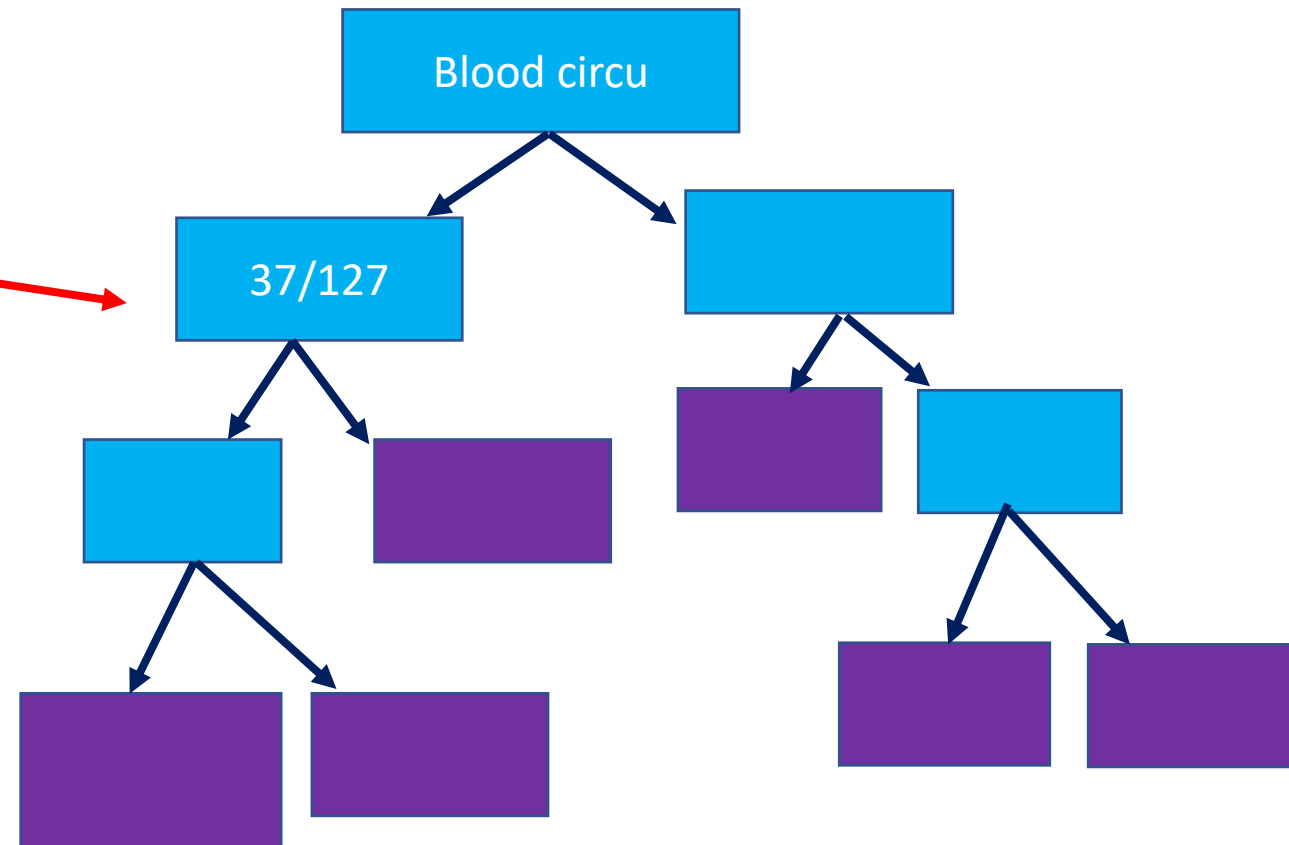
Good Blood Circulation

Gini impurity for Good Blood Circulation=0.360

Hear Disease	
Yes	No
37	127

Hear Disease	
Yes	No
100	33

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



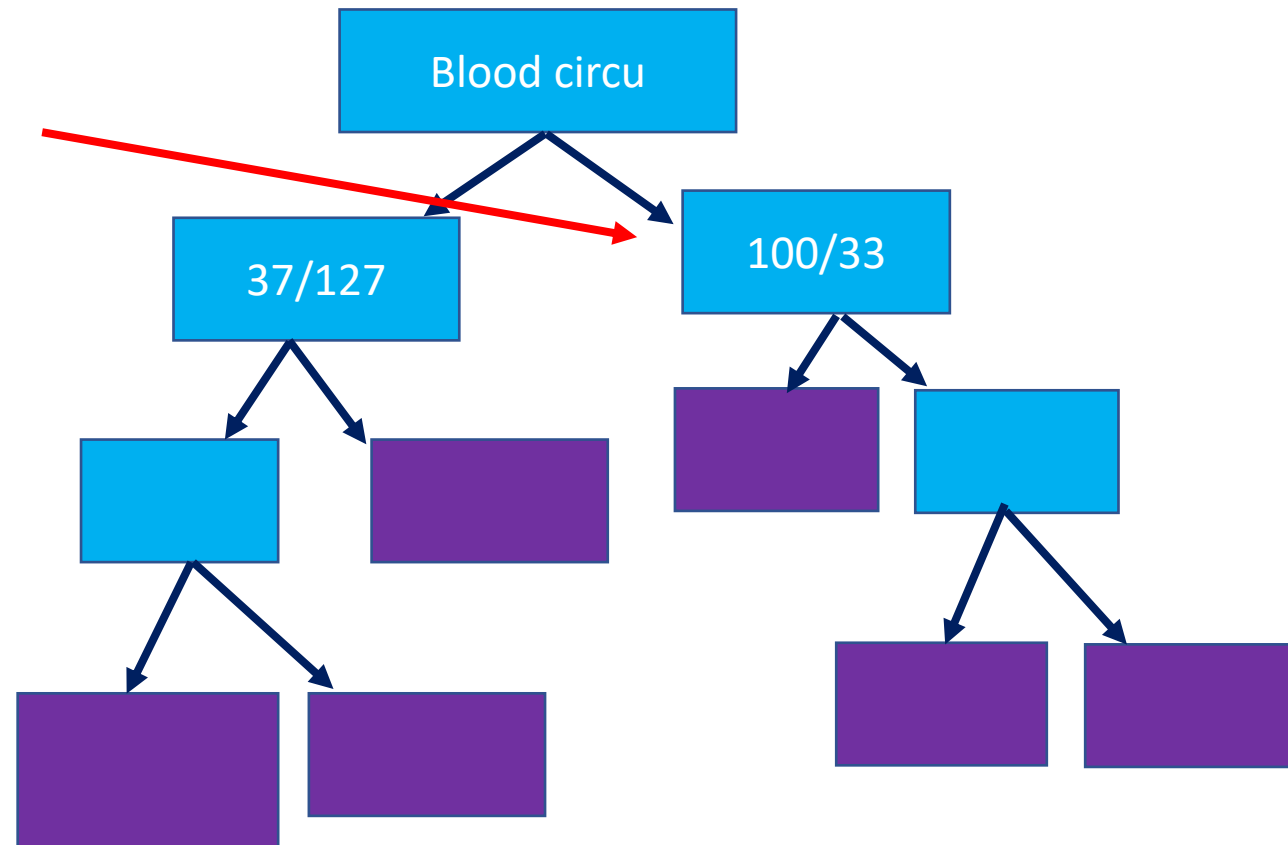
# Decision Trees

Good Blood Circulation

Hear Disease	
Yes	No
37	127

Hear Disease	
Yes	No
100	33

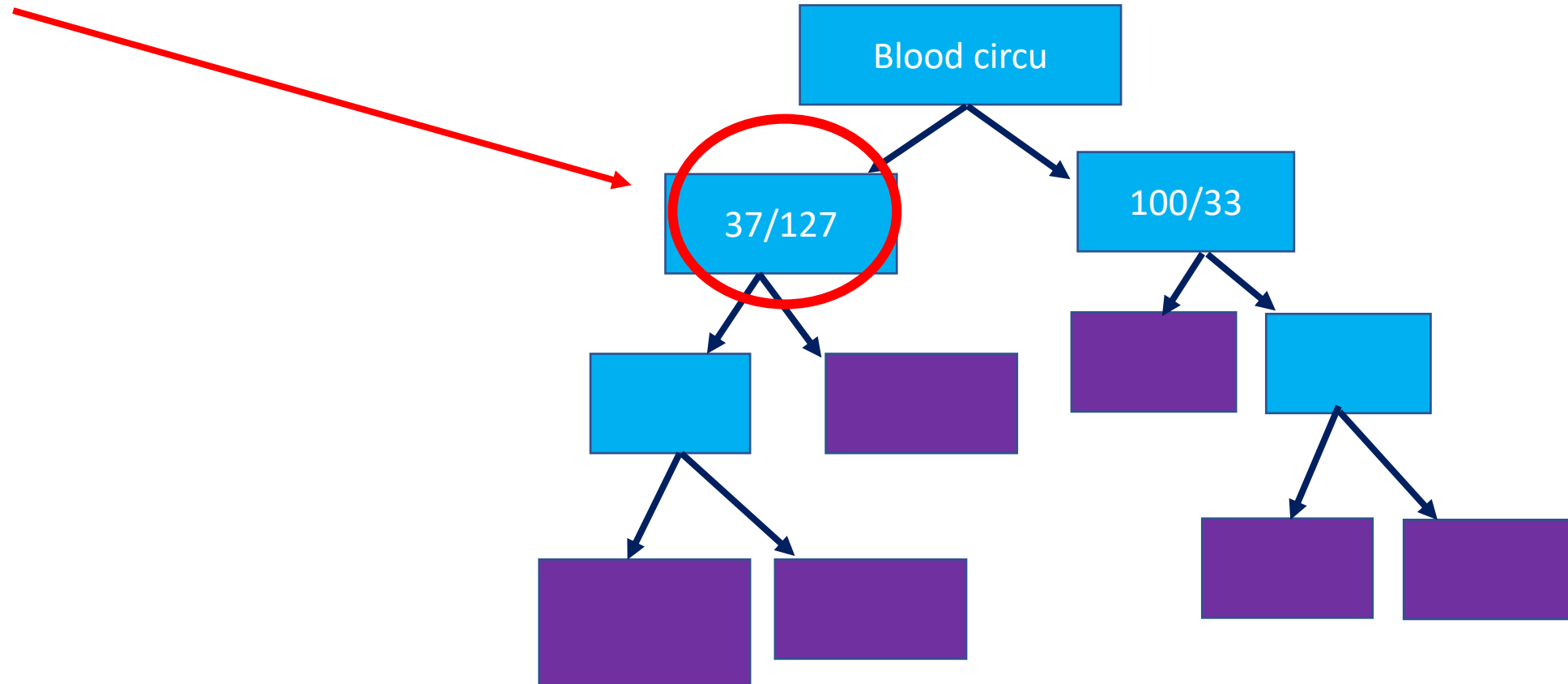
and the 133 patients with and without heart disease  
That ended up in this leaf node



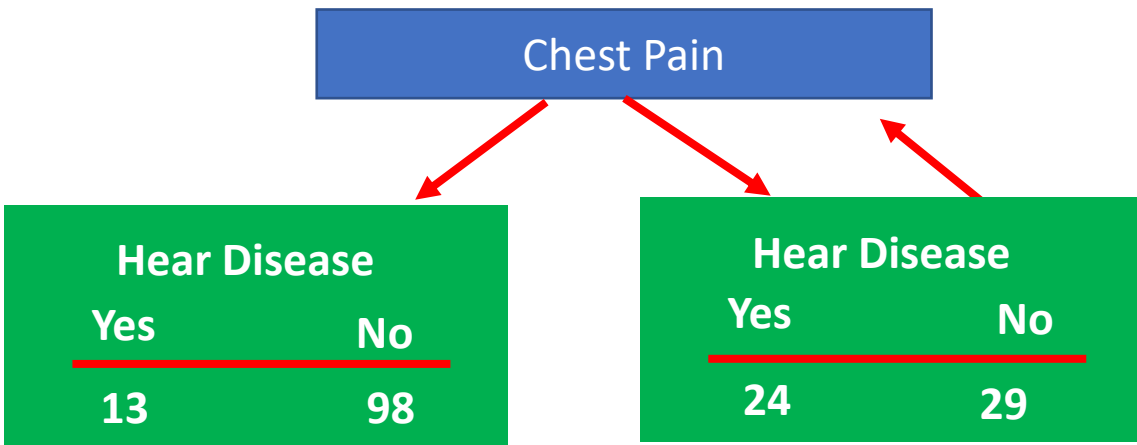
# Decision Trees

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

Gini impurity for Good Blood Circulation=0.360

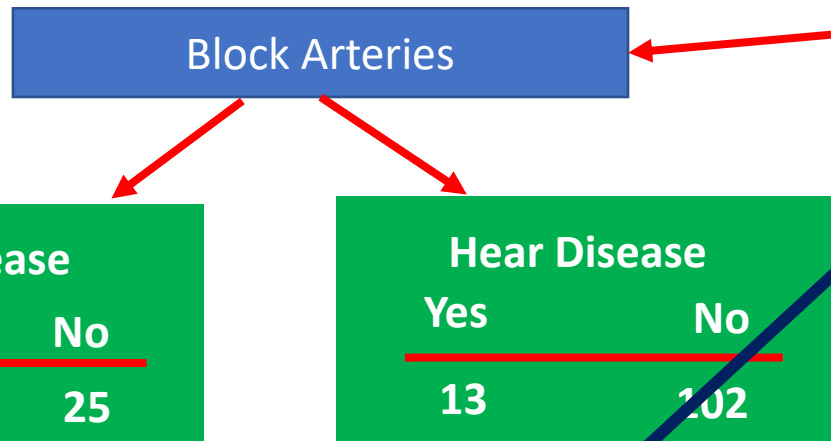




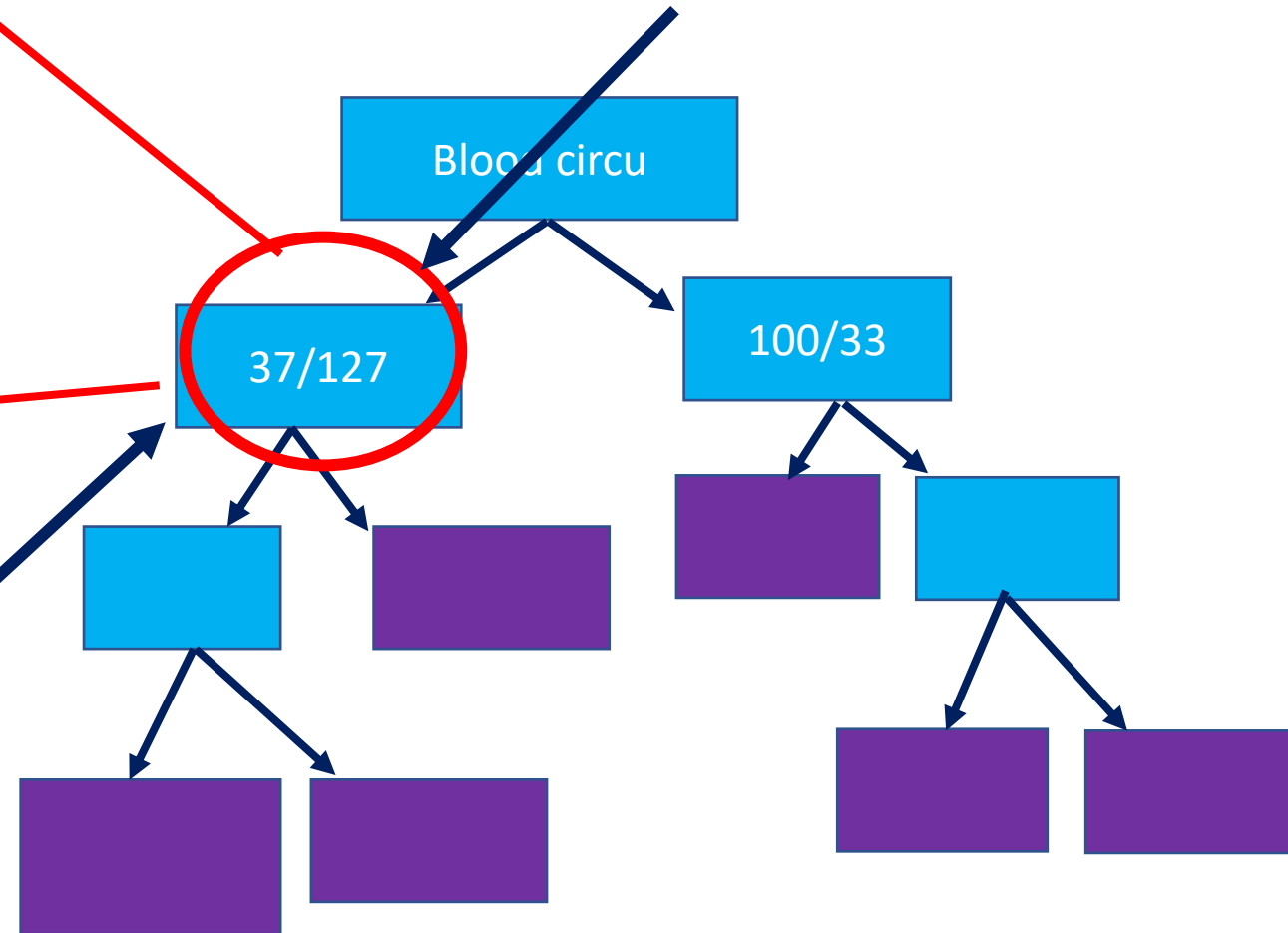


Gini impurity for chest pain=0.3

Block arteries has lowest Gini impurity  
We will use it at this node to separate the patients

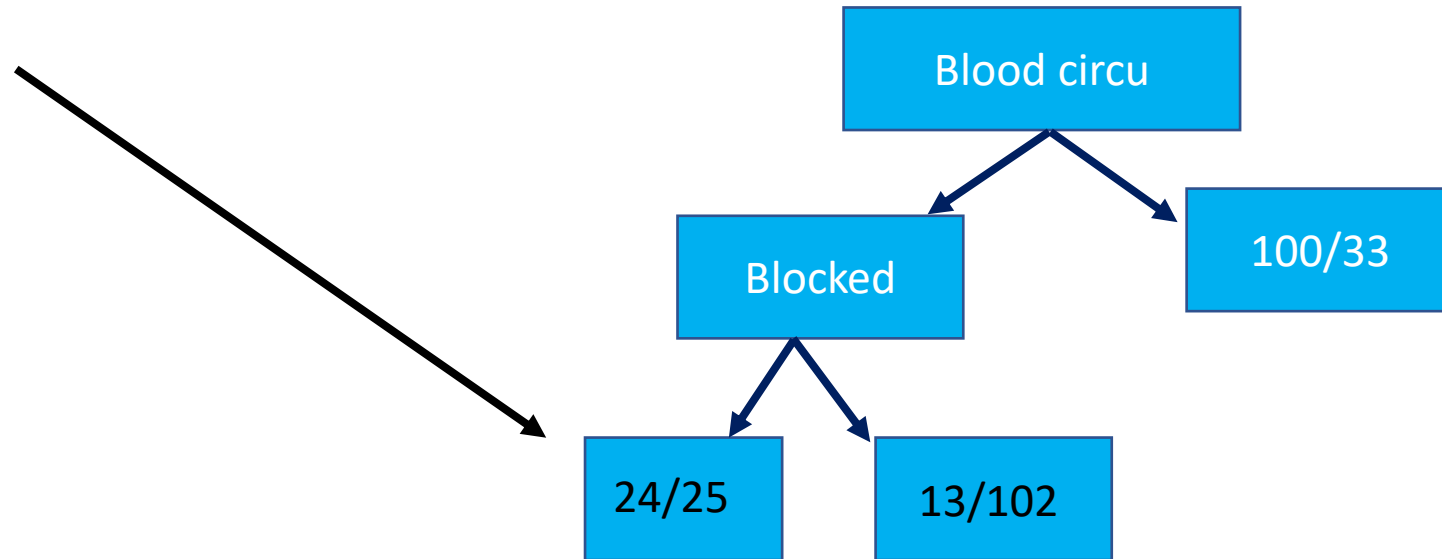


Gini impurity for chest pain=0.290

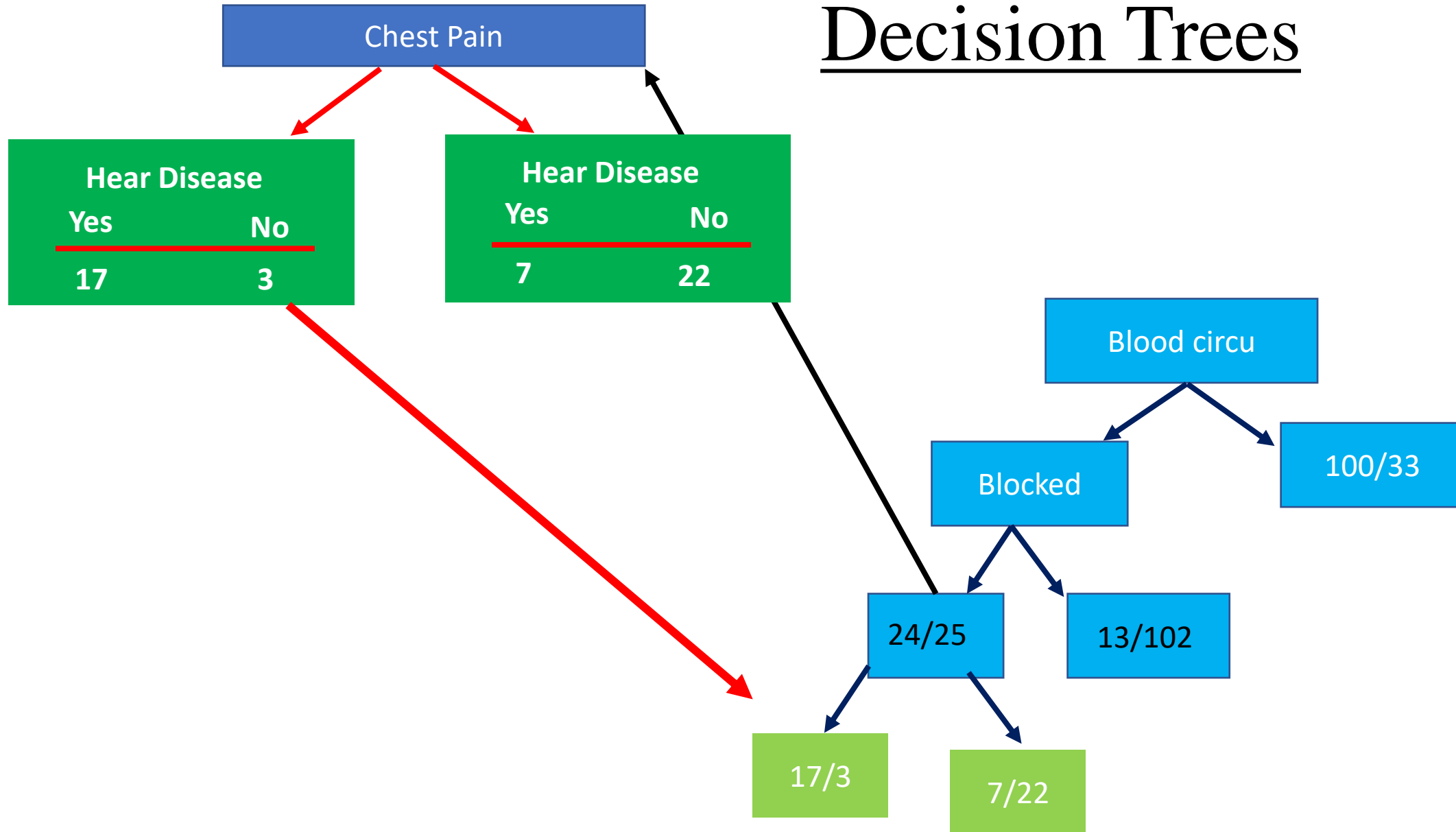


# Decision Trees

Now we need All we have left **chest pain**, so  
First we will see how well separate these 49 patients  
(24 with heart diseases and 25 without heart disease)

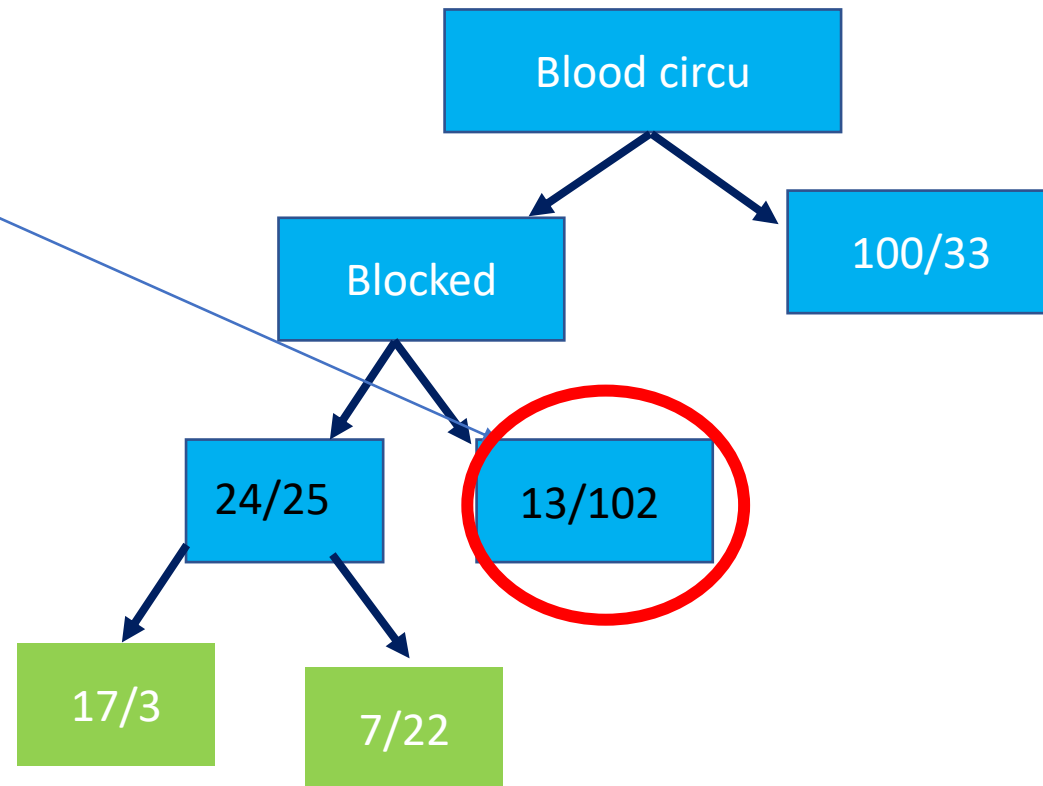


# Decision Trees

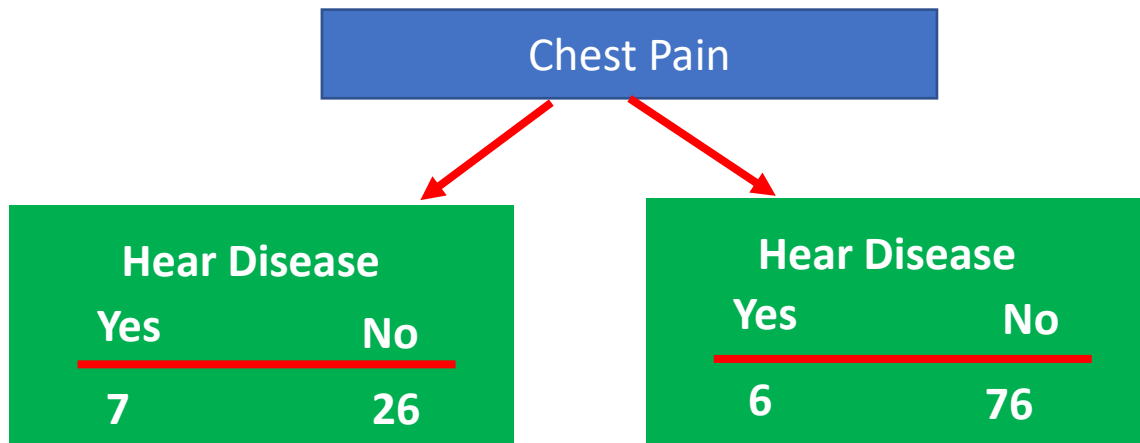


# Decision Trees

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



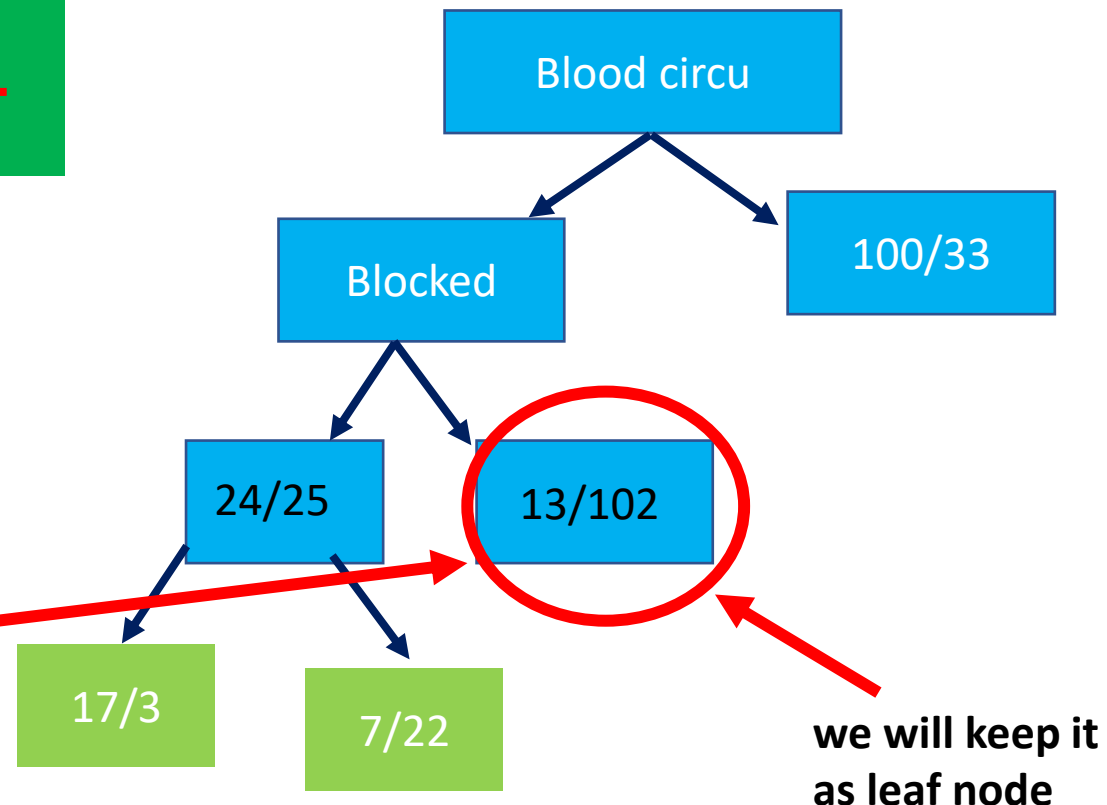
# Decision Trees



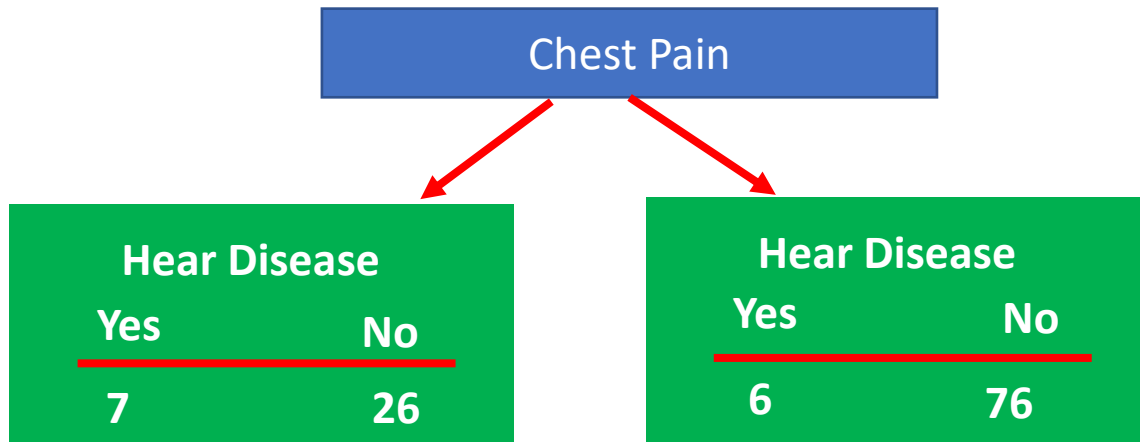
Gini impurity for chest pain=0.29

The Gini impurity for this node  
before using pain to separate patience is

$$= 1 - \left( \frac{13}{13+102} \right)^2 - \left( \frac{102}{13+102} \right)^2$$
$$= 0.2$$



# Decision Trees

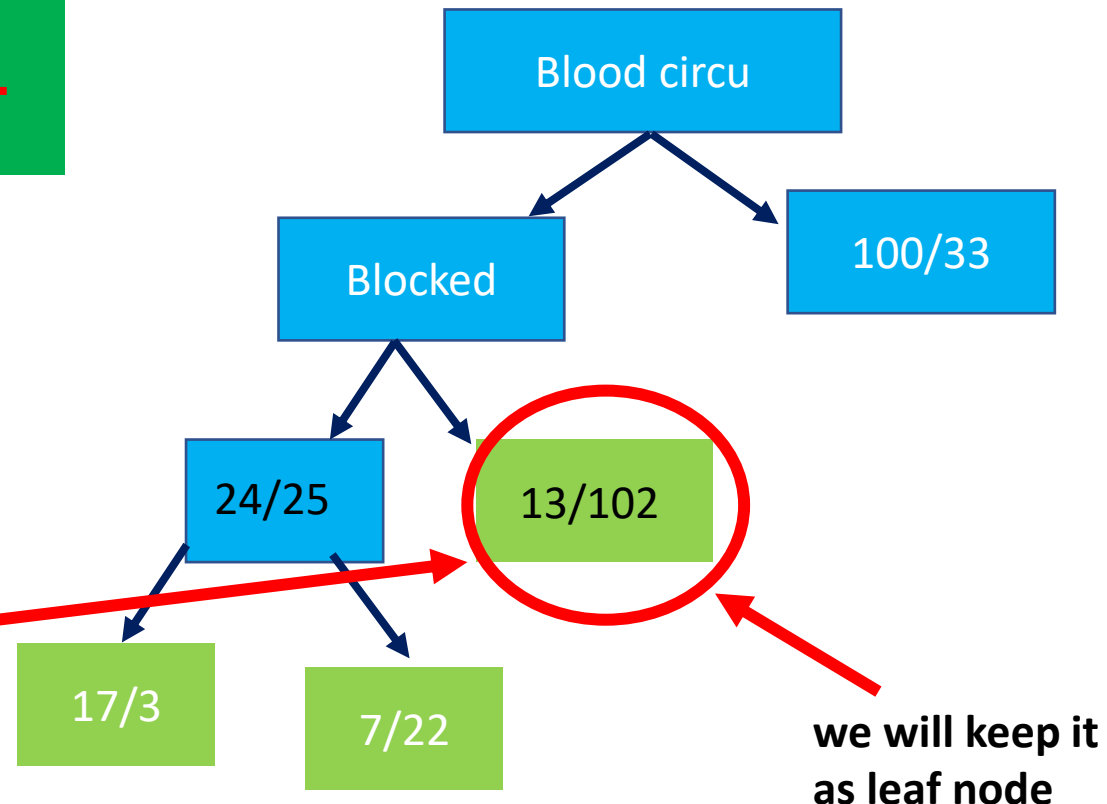


Gini impurity for chest pain=0.29

The Gini impurity for this node before using pain to separate patience is

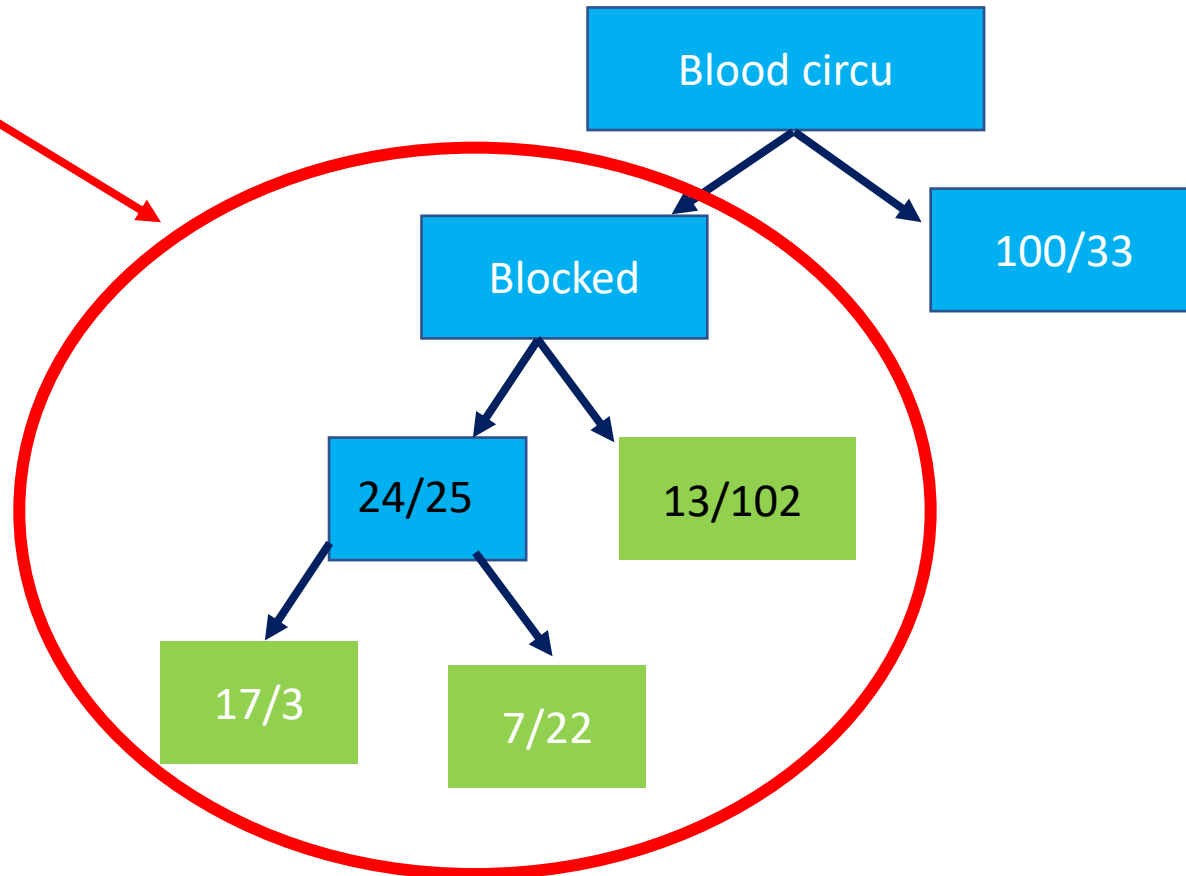
$$= 1 - \left( \frac{13}{13+102} \right)^2 - \left( \frac{102}{13+102} \right)^2$$

$$= 0.2$$



# Decision Trees

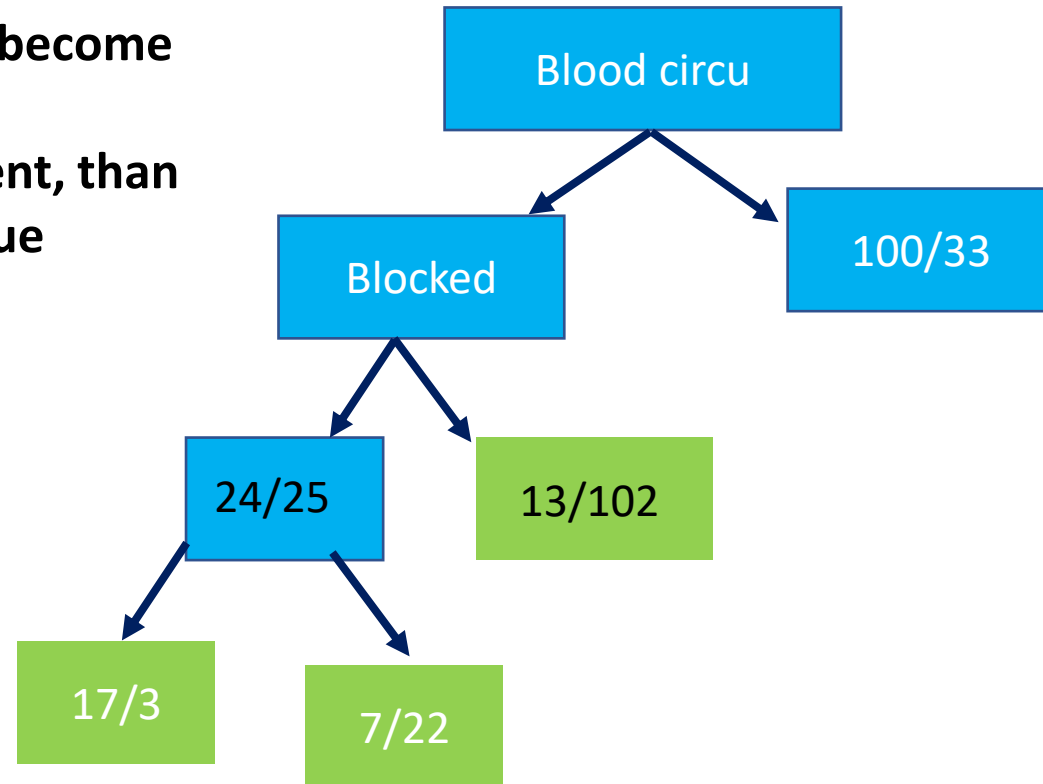
We worked out the entire left subtree



# Decision Trees

We follow the exact same steps as we did on the left side:

- 1- Calculate all 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 become A leaf node
- 3- if separating the data results in an improvement, than Pick the separation with the lowest impurity value





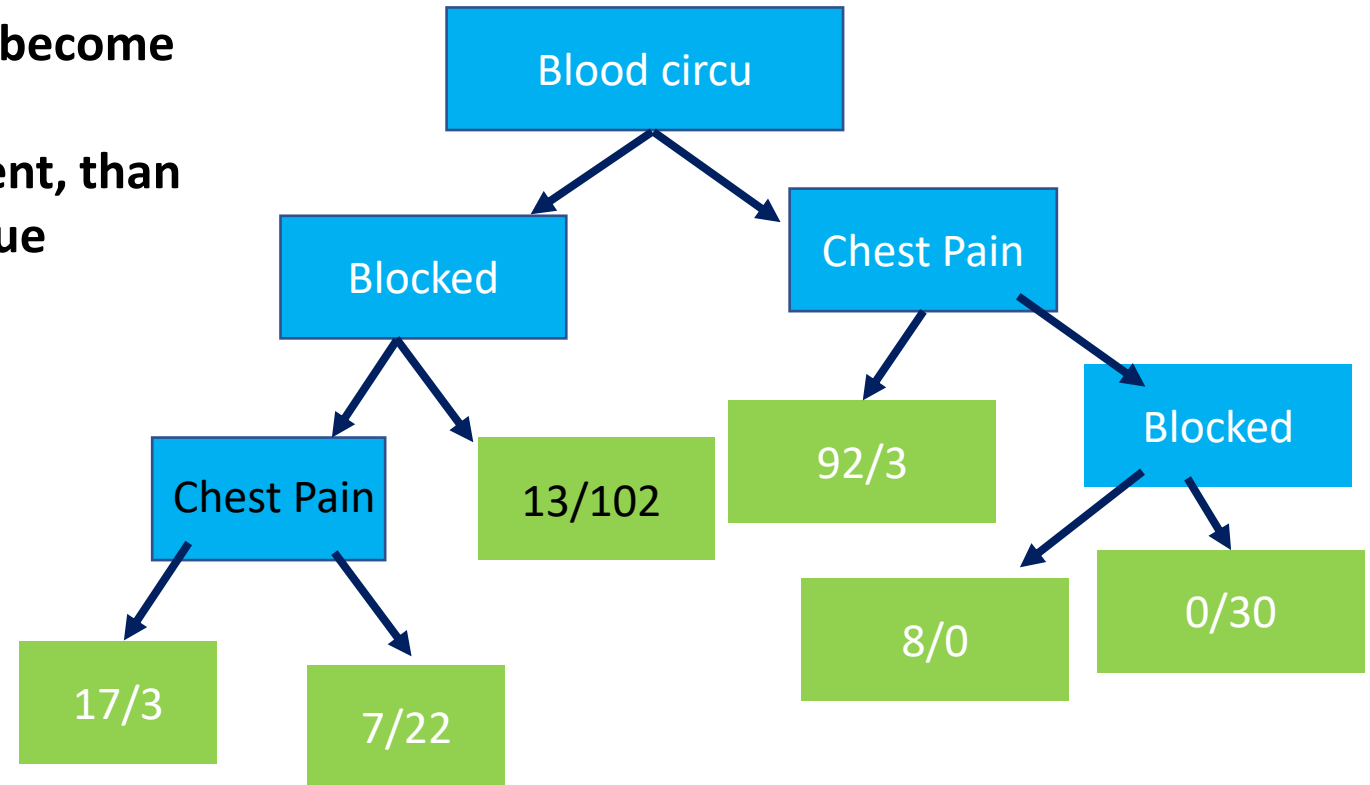
# Decision Trees

We follow the exact same steps as we did on the left side:

1- Calculate all 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 become A leaf node

3- if separating the data results in an improvement, than Pick the separation with the lowest impurity value



# Decision Trees

- **So what do we know until now?**
- In principal decision trees can be used to predict the target feature of an unknown query instance by building a model based on existing data for which the target feature values are known (supervised learning). Additionally, we know that this model can make predictions for unknown query instances because it models the relationship between the known descriptive features and the know target feature. In our following example, the tree model learns “how a specific animal species look” respectively the combination of descriptive feature values distinctive for animal species.
- Additionally, we know that to train a decision tree model we need a dataset consisting of several training examples characterized by several descriptive features and a target feature.

# Decision Tree Algorithm (Working example using Information gain and Entropy method)

# Decision Trees

- **Working Example of Decision Tree Algorithm**
- We want, given a dataset, train a model which kind of learns the relationship between the descriptive features and a target feature such that we can present the model a new, unseen set of query instances and predict the target feature values for these query instances.
- Let's further recapitulate the general shape of a decision tree.
- We know that we have at the bottom of the tree leaf nodes which contain (in the optimal case) target feature values.
- To make this more illustrative we use as a practical example a simplified version of the UCI machine learning Zoo Animal Classification dataset which includes properties of animals as descriptive features and the animal species as target feature.
- In our example, the animals are classified as being Mammals or Reptiles based on whether they are toothed, have legs and do breathe. The dataset looks like:

# Decision Trees

- **Working Example of Decision Tree Algorithm**
- In our example, the animals are classified as being Mammals or Reptiles based on whether they are toothed, have legs and do breathe. The dataset looks like:

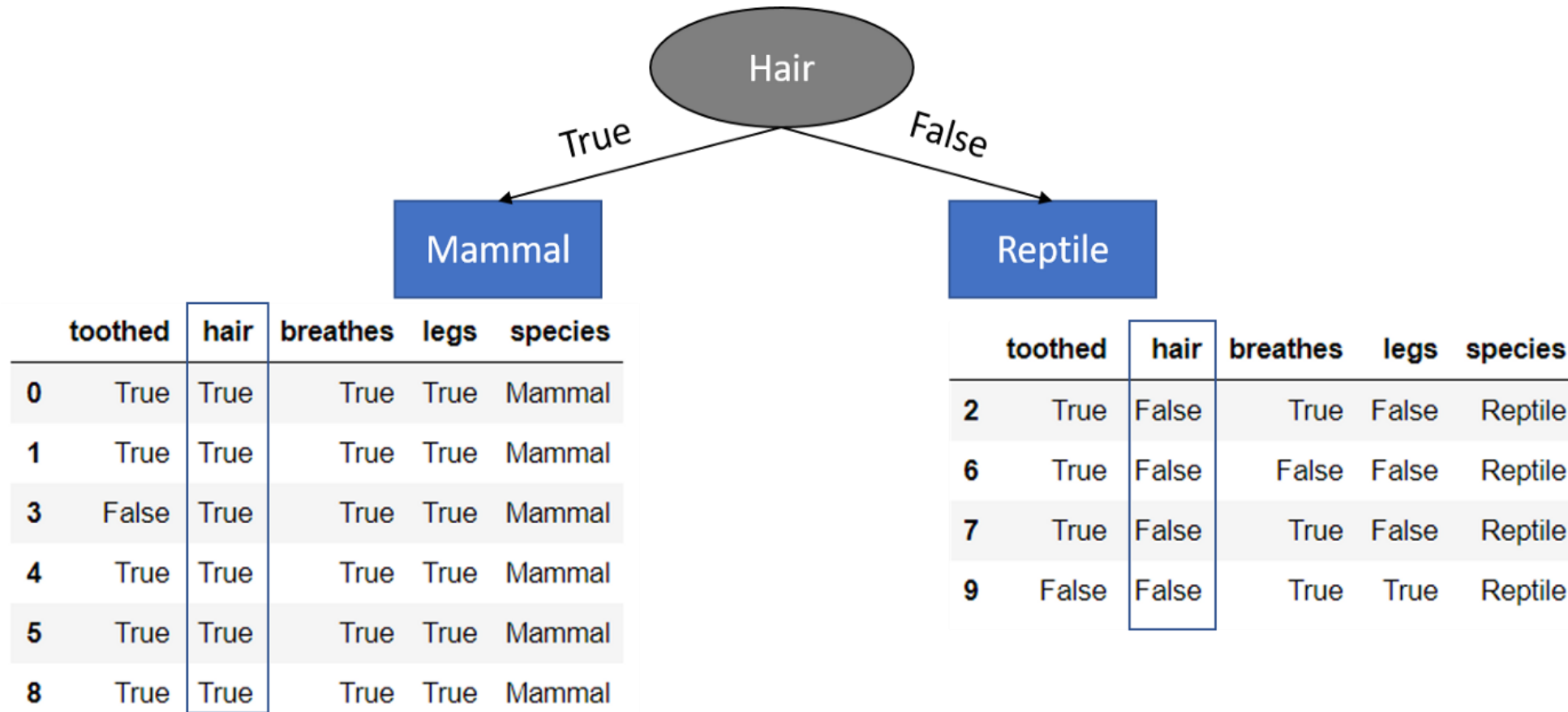
	toothed	hair	breathes	legs	species
0	True	True	True	True	Mammal
1	True	True	True	True	Mammal
2	True	False	True	False	Reptile
3	False	True	True	True	Mammal
4	True	True	True	True	Mammal
5	True	True	True	True	Mammal
6	True	False	False	False	Reptile
7	True	False	True	False	Reptile
8	True	True	True	True	Mammal
9	False	False	True	True	Reptile

# Decision Trees

- **Working Example of Decision Tree Algorithm**
- Hence, to come back to our initial question, each leaf node should (in the best case) only contain “Mammals” or “Reptiles”.
- The task for us is now to find the best “way” to split the dataset such that this can be achieved. What do I mean when I say split?
- Well consider the dataset above and think about what must be done to split the dataset into a Dataset 1 containing as target feature values (species) only Mammals and a Dataset 2, containing only Reptiles.
- To achieve that, in this simplified example, we only need the descriptive feature hair since if the hair is TRUE, the associated species is always a Mammal

# Decision Trees

- **Working Example of Decision Tree Algorithm**
- Hence, in this case, our tree model would look like:



# Decision Trees

- **Working Example of Decision Tree Algorithm**
- That is, we have split our dataset by asking the question if the animal has hair or not. Now, in that case, the splitting has been very easy for us, because we could easily identify the classification relationship between having hair and being mammal and vice versa.
- Some may differentiate this table dataset easily because they knew mammal have hair while reptiles don't. But a machine can detect it so easily. Moreover, most of the time datasets are not that easily separable and we must split the dataset more than one time (“ask more than one question”).
- **So leaving hair no other attribute makes it easy to obtain leaf nodes directly. We'll omit the hair attribute to obtain a real-world like situation.**



# Decision Trees

## Working Example of Decision Tree Algorithm

	toothed	breathes	legs	species
0	True	True	True	Mammal
1	True	True	True	Mammal
2	True	True	False	Reptile
3	False	True	True	Mammal
4	True	True	True	Mammal
5	True	True	True	Mammal
6	True	False	False	Reptile
7	True	True	False	Reptile
8	True	True	True	Mammal
9	False	True	True	Reptile

toothed == True

toothed == False

After computing the IG of feature *toothed*  
do this for features *breathes* and *legs*

	toothed	breathes	legs	species
0	True	True	True	Mammal
1	True	True	True	Mammal
2	True	True	False	Reptile
4	True	True	True	Mammal
5	True	True	True	Mammal
6	True	False	False	Reptile
7	True	True	False	Reptile
8	True	True	True	Mammal

	toothed	breathes	legs	species
3	False	True	True	Mammal
9	False	True	True	Reptile

1. Calculate the entropy  
for *toothed* == True



2. Calculate the entropy  
for *toothed* == False



3. Sum up the entropies  
of 1. and 2.



4. Subtract this sum  
from the whole datasets  
entropy → InfoGain

# Decision Trees

## ■ Working Example of Decision Tree Algorithm

Hence the entropy of our dataset regarding the target feature is calculated with:  
Entropy of parent node =  $-(6/10) \cdot \log_2(6/10) - (4/10) \cdot \log_2(4/10) = 0.971$

	toothed	breathes	legs	species
0	True	True	True	Mammal
1	True	True	True	Mammal
2	True	True	False	Reptile
3	False	True	True	Mammal
4	True	True	True	Mammal
5	True	True	True	Mammal
6	True	False	False	Reptile
7	True	True	False	Reptile
8	True	True	True	Mammal
9	False	True	True	Reptile

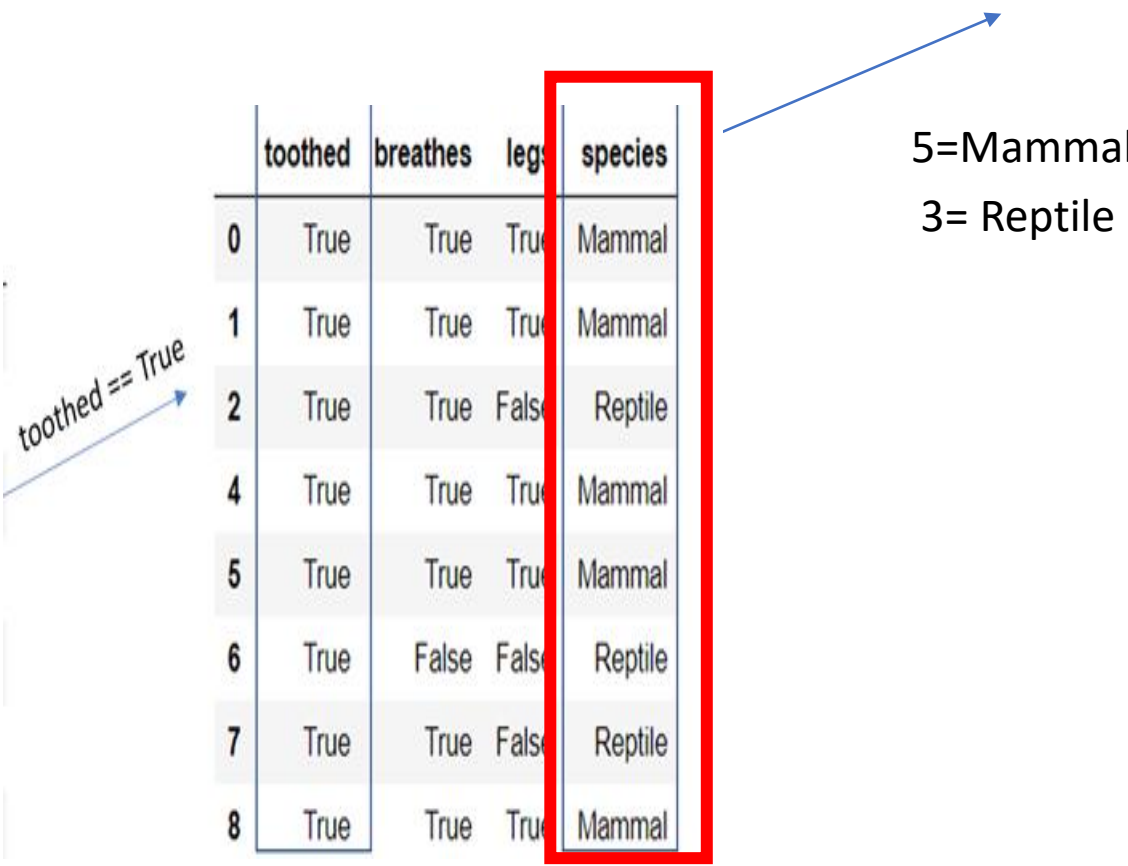
6=Mammal  
4= Reptile

After computing the IG of feature *toothed*  
do this for features *breathes* and *legs*

# Decision Trees

## ▪ Working Example of Decision Tree Algorithm

Entropy of (toothed == True) =  $-(5/8) \log_2 (5/8) - (3/8) \log_2 (3/8) = 0.95$



	toothed	breathes	legs	species
0	True	True	True	Mammal
1	True	True	True	Mammal
2	True	True	False	Reptile
4	True	True	True	Mammal
5	True	True	True	Mammal
6	True	False	False	Reptile
7	True	True	False	Reptile
8	True	True	True	Mammal

5=Mammal  
3= Reptile

# Decision Trees

## ▪ Working Example of Decision Tree Algorithm

Entropy of (toothed == False) =  $-(1/2) \log_2 (1/2) - (1/2) \log_2 (1/2) = 1$

toothed == False

	toothed	breathes	legs	species
3	False	True	True	Mammal
9	False	True	True	Reptile

1=Mammal  
1= Reptile

# Decision Trees

## Working Example of Decision Tree Algorithm

	toothed	breathes	legs	species
0	True	True	True	Mammal
1	True	True	True	Mammal
2	True	True	False	Reptile
3	False	True	True	Mammal
4	True	True	True	Mammal
5	True	True	True	Mammal
6	True	False	False	Reptile
7	True	True	False	Reptile
8	True	True	True	Mammal
9	False	True	True	Reptile

	toothed	breathes	legs	species
0	True	True	True	Mammal
1	True	True	True	Mammal
2	True	True	False	Reptile
4	True	True	True	Mammal
5	True	True	True	Mammal
6	True	False	False	Reptile
7	True	True	False	Reptile
8	True	True	True	Mammal

	toothed	breathes	legs	species
3	False	True	True	Mammal
9	False	True	True	Reptile

After computing the IG of feature *toothed*  
do this for features *breathes* and *legs*

1. Calculate the entropy  
for *toothed* == True

2. Calculate the entropy  
for *toothed* == False

3. Sum up the entropies  
of 1. and 2.

4. Subtract this sum  
from the whole datasets  
entropy → InfoGain

$$\text{Entropy for split on toothed} = 8/10(0.95) + 2/10(1) = 0.96$$

$$\text{Information Gain} = 0.971 - 0.96 = 0.011$$

# Decision Trees

## ▪ Working Example of Decision Tree Algorithm

### Toothed:

Entropy of (toothed == True) =  $-(5/8) \log_2 (5/8) - (3/8) \log_2 (3/8) = \mathbf{0.95}$

Entropy of (toothed == False) =  $-(1/2) \log_2 (1/2) - (1/2) \log_2 (1/2) = \mathbf{1}$

Entropy for split on toothed =  $8/10(0.95) + 2/10(1) = \mathbf{0.96}$

Information Gain =  $0.971 - 0.96 = \mathbf{0.011}$

### breathes:


$H(\text{breathes}) = (9/10 * -((6/9 * \log_2(6/9)) + (3/9 * \log_2(3/9)))) + 1/10 * -((0) + (1 * \log_2(1))) = 0.82647$

InfoGain(breathes) =  $0.971 - 0.82647 = \mathbf{0.1445}$

### legs:

$H(\text{legs}) = 7/10 * -((6/7 * \log_2(6/7)) + (1/7 * \log_2(1/7))) + 3/10 * -((0) + (1 * \log_2(1))) = 0.41417$

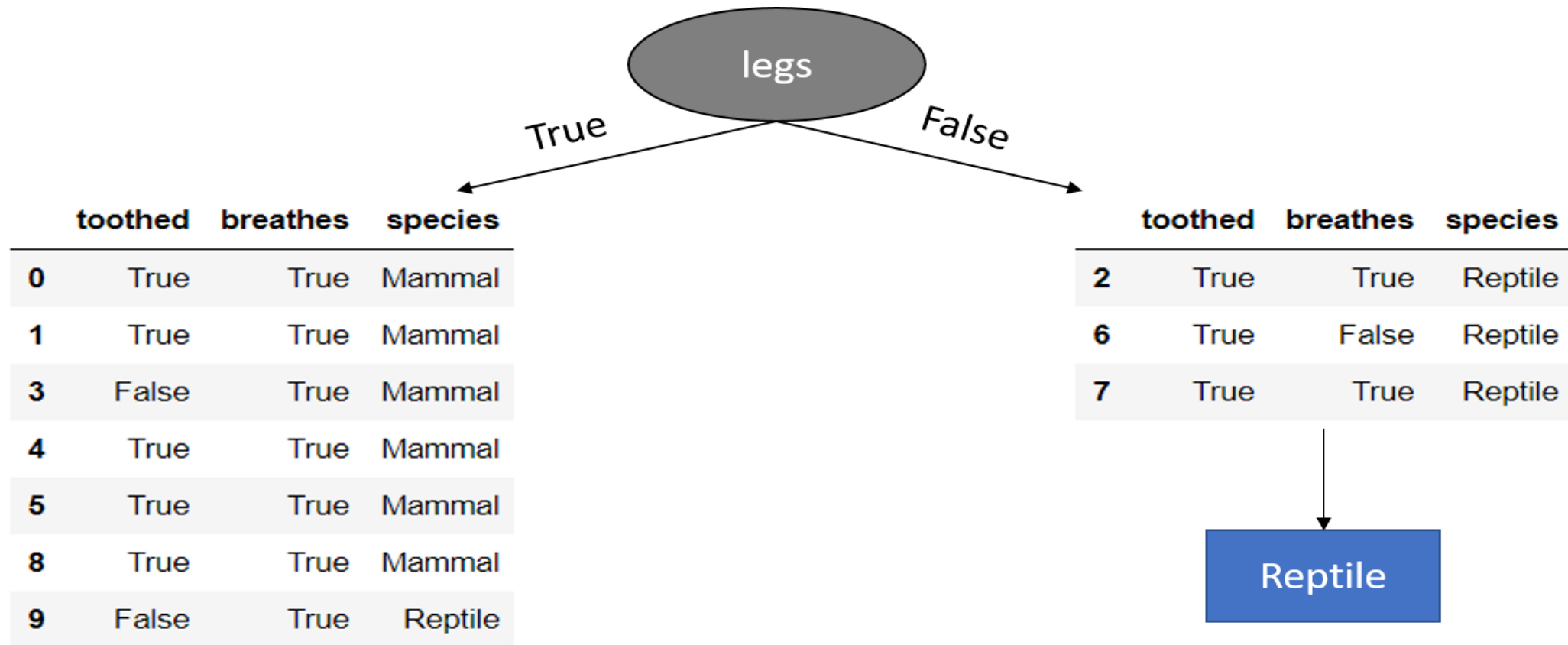
InfoGain(legs) =  $0.971 - 0.41417 = \mathbf{0.5568}$



Hence the splitting the dataset along the feature legs results in the largest information gain and we should use this feature for our root node.

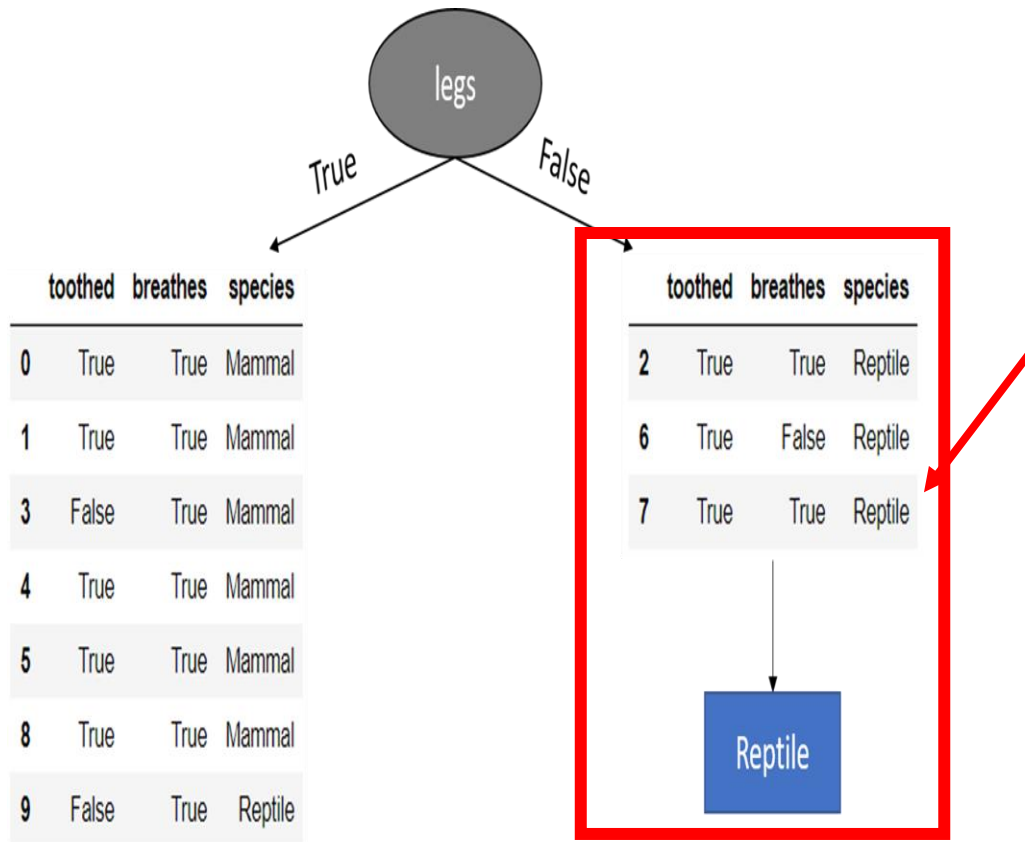
# Decision Trees

- **Working Example of Decision Tree Algorithm**
- Hence for the time being the decision tree model looks like:



# Decision Trees

## ■ Working Example of Decision Tree Algorithm

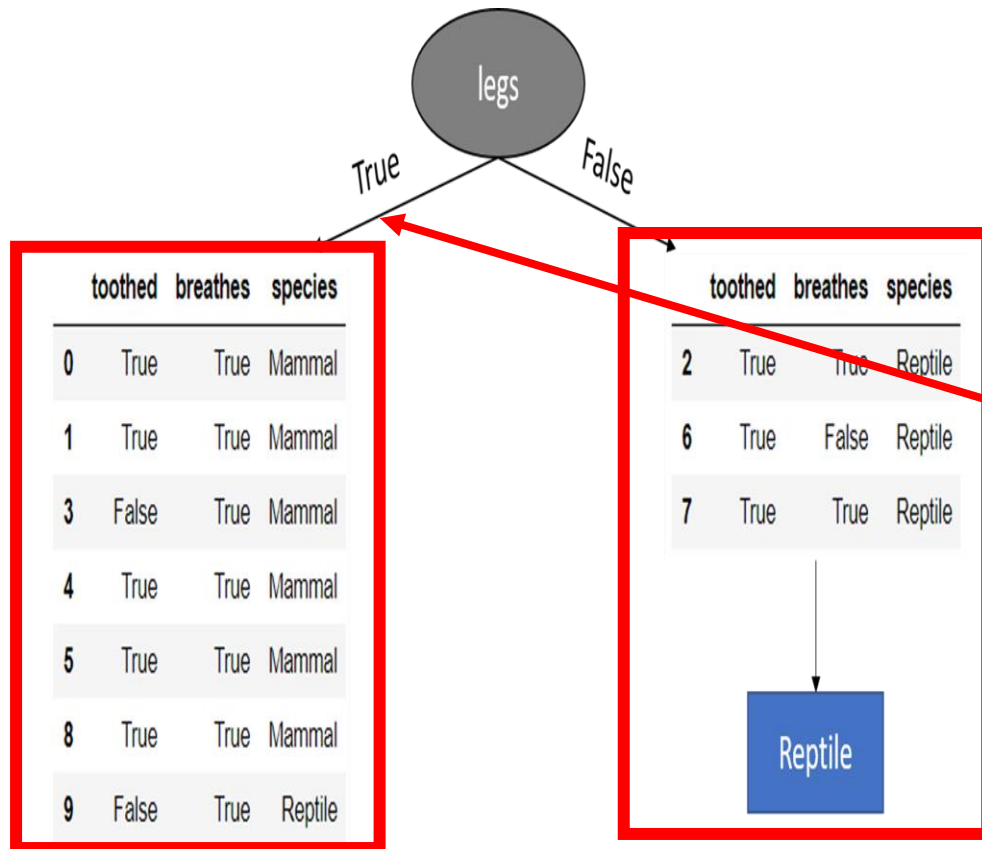


We see that for legs == False, the target feature values of the remaining dataset are all Reptile and hence we set this as leaf node because we have a pure dataset (Further splitting the dataset on any of the remaining two features would not lead to a different or more accurate result since whatever we do after this point, the prediction will remain Reptile). Additionally, you see that the feature legs are no longer included in the remaining datasets. Because we already have used this (categorical) feature to split the dataset on it must not be further used.



# Decision Trees

## ▪ Working Example of Decision Tree Algorithm

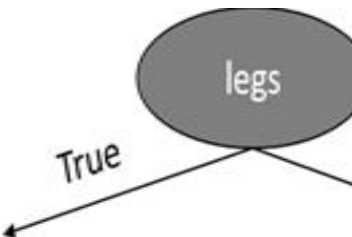


Until now we have found the feature for the root node as well as a leaf node for legs == False.

The same steps for information gain calculation must now be accomplished also for the remaining dataset for legs == True since here we still have a mixture of different target feature values. Hence:

# Decision Trees

## Working Example of Decision Tree Algorithm



	toothed	breathes	species
0	True	True	Mammal
1	True	True	Mammal
3	False	True	Mammal
4	True	True	Mammal
5	True	True	Mammal
8	True	True	Mammal
9	False	True	Reptile

Information gain calculation for the features toothed and breathes for the remaining dataset legs == True:

The Entropy of the (new) sub data set after the first split:

$$H(D) = -((6/7 * \log_2(6/7)) + (1/7 * \log_2(1/7))) = 0.5917$$

toothed:

$$H(\text{toothed}) = 5/7 * -((1 * \log_2(1)) + (0)) + 2/7 * -((1/2 * \log_2(1/2)) + (1/2 * \log_2(1/2))) = 0.285$$

$$\text{InfoGain}(\text{toothed}) = 0.5917 - 0.285 = 0.3067$$

breathes:

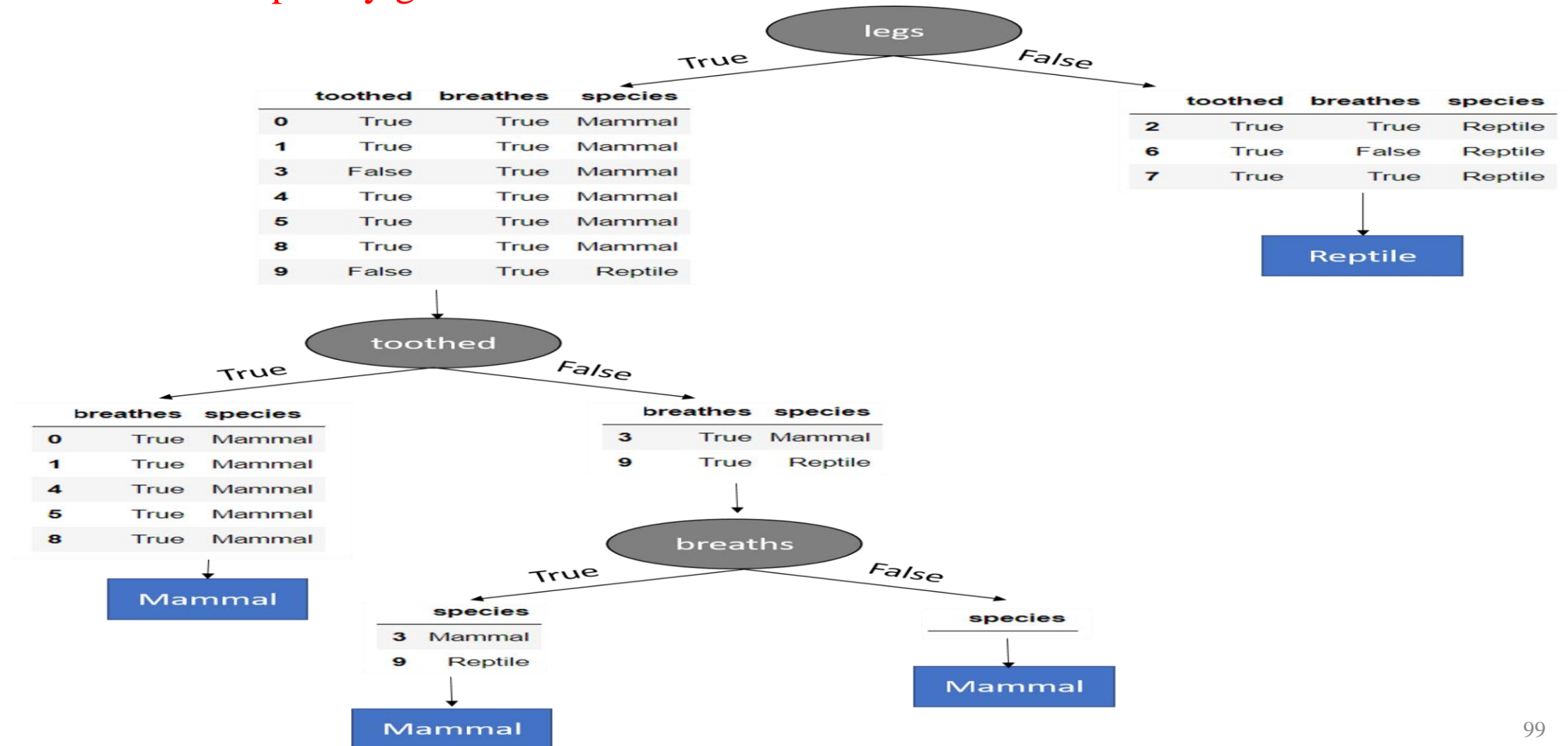
$$H(\text{breathes}) = 7/7 * -((6/7 * \log_2(6/7)) + (1/7 * \log_2(1/7))) + 0 = 0.5917$$

$$\text{InfoGain}(\text{breathes}) = 0.5917 - 0.5917 = 0$$

The dataset for toothed == False still contains a mixture of different target feature values why we proceed partitioning on the last left feature (== breathes)

# Decision Trees

- **Working Example of Decision Tree Algorithm**
- Hence the completely grown tree looks like:



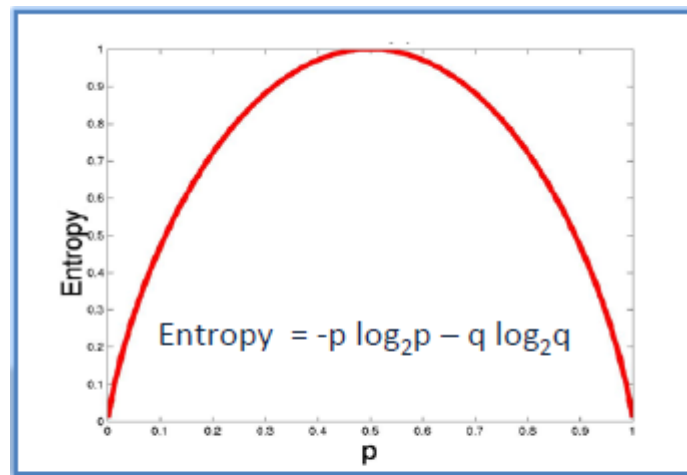
# Decision Trees

- **Working Example of Decision Tree Algorithm**
- Hence the completely grown tree looks like:
- Mind the last split (node) where the dataset got split on the breathes feature. Here the breathes feature solely contains data where breaths == True. Hence for breathes == False there are no instances in the dataset and therewith there is no sub-Dataset which can be built.
- In that case, we return the most frequently occurring target feature value in the original dataset which is Mammal.
- This is an example of how our tree model generalizes behind the training data.
- If we consider the other branch, that is breathes == True we know, that after splitting the Dataset on the values of a specific feature (breathes{True, False}) in our case, the feature must be removed. Well, that leads to a dataset where no more features are available to further split the dataset on. Hence we stop growing the tree and return the mode value of the direct parent node which is “Mammal”.

# Decision Trees

- **Tree Selection**

- The process of finding the smallest tree that fits the data. Usually this is the tree that yields the lowest cross-validated error.
- Key Factors :
- 1. Entropy
- A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogeneous).
- ID 3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

# Decision Trees

- **Tree Selection**
- The process of finding the smallest tree that fits the data. Usually this is the tree that yields the lowest cross-validated error.
- **2. Information Gain**
- The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

# Decision Trees

- **Tree Selection**
- **2. Information Gain**

Steps Involved

Step 1:

**Calculate entropy of the target.**

Step 2:

The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

Step 3:

Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

Pros:

Computationally cheap to use, easy for humans to understand results and it can deal with irrelevant features also

Cons:

Prone to Overfitting.(It refers to the process when models is trained on training data too well that any noise in testing data can bring negative impacts to performance of model.)

# Decision Tree Algorithm in Scikit-Learn



# Decision Trees

- **Decision Tree Classifier Building in Scikit-learn**
- Importing Required Libraries
- Let's first load the required libraries.

```
# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
# Import Decision Tree Classifier
from sklearn.model_selection import train_test_split
# Import train_test_split function
from sklearn import metrics
# Import scikit-learn metrics module for accuracy
calculation
```

# Decision Trees

- **Decision Tree Classifier Building in Scikit-learn**
- Loading Data
- Let's first load the required Pima Indian Diabetes dataset using pandas' read CSV function. You can download the data [here](#).

```
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi',  
'pedigree', 'age', 'label']  
# load dataset  
pima = pd.read_csv("pima-indians-diabetes.csv",  
header=None, names=col_names)  
pima.head()
```

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

# Decision Trees

- **Decision Tree Classifier Building in Scikit-learn**
- Feature Selection
- Here, you need to divide given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

```
#split dataset in features and target variable  
feature_cols = ['pregnant', 'insulin', 'bmi',  
                'age','glucose','bp','pedigree']  
X = pima[feature_cols] # Features  
y = pima.label # Target variable
```

# Decision Trees

- **Decision Tree Classifier Building in Scikit-learn**
- Splitting Data
- To understand model performance, dividing the dataset into a training set and a test set is a good strategy.
- Let's split the dataset by using function `train_test_split()`. You need to pass 3 parameters features, target, and test\_set size.

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.3,
random_state=1) # 70% training and 30% test
```

# Decision Trees

- **Decision Tree Classifier Building in Scikit-learn**
- Building Decision Tree Model
- Let's create a Decision Tree Model using Scikit-learn. Know about `sklearn.tree.DecisionTreeClassifier`

```
# Create Decision Tree classifier object
clf = DecisionTreeClassifier()
# Train Decision Tree Classifier
clf = clf.fit(X_train,y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

# Decision Trees

- **Decision Tree Classifier Building in Scikit-learn**
- Evaluating Model
- Let's estimate, how accurately the classifier or model can predict the type of cultivars.
- Accuracy can be computed by comparing actual test set values and predicted values.

```
# Model Accuracy, how often is the classifier correct?  
print("Accuracy:",metrics.accuracy_score(y_test,  
y_pred))  
Accuracy: 0.6753246753246753
```



Well, you got a classification rate of 67.53%, considered as good accuracy. You can improve this accuracy by tuning the parameters in the Decision Tree Algorithm.

# Decision Trees

- **Decision Tree Classifier Building in Scikit-learn**
- **Optimizing Decision Tree Performance**
- As you can see above the tree is too vast and not that accurate. This tree also takes time, so for better results, we need to optimize the tree by tuning the parameter. Here I'll be explaining only the important parameter of Decision Tree Algorithm, its tuning will require a whole new blog post. If you take a look at the parameters the DecisionTreeClassifier can take, you might be surprised so, let's look at some of them.

**criterion:** This parameter determines how the impurity of a split will be measured. The default value is “gini” but you can also use “entropy” as a metric for impurity.

**splitter:** This is how the decision tree searches the features for a split. The default value is set to “best”. That is, for each node, the algorithm considers all the features and chooses the best split. If you decide to set the splitter parameter to “random,” then a random subset of features will be considered. The split will then be made by the best feature within the random subset. The size of the random subset is determined by the max\_features parameter. This is partly where a Random Forest gets its name.

# Decision Trees

- **Decision Tree Classifier Building in Scikit-learn**
- **Optimizing Decision Tree Performance**

**max\_depth:** This determines the maximum depth of the tree. In our case, we use a depth of two to make our decision tree. The default value is set to none. This will often result in over-fitted decision trees. The depth parameter is one of the ways in which we can regularize the tree, or limit the way it grows to prevent over-fitting.

**min\_samples\_split:** The minimum number of samples a node must contain to consider splitting. The default value is two. You can use this parameter to regularize your tree.

**min\_samples\_leaf:** The minimum number of samples needed to be considered a leaf node. The default value is set to one. Use this parameter to limit the growth of the tree.

**max\_features:** The number of features to consider when looking for the best split. If this value is not set, the decision tree will consider all features available to make the best split. Depending on your application, it's often a good idea to tune this parameter.



# Decision Trees

- **Decision Tree Classifier Building in Scikit-learn**
- For syntax purposes, lets set some of these parameters:

```
Setting parameterstree = DecisionTreeClassifier(criterion  
= "entropy", splitter = "random", max_depth = 2,  
min_samples_split = 5,min_samples_leaf = 2,  
max_features = 2).fit(X,y)
```

# Random forest Algorithm

# Random Forest


- **Random Forest**
- **Decision Tree problem:** They work great with the data used to create them, but they are not flexible when it comes to classifying new sample
- The good thing is that Random Forests combine the simplicity of decision trees with flexibility resulting in a vast improvement in accuracy
- **Step1: Create a Bootstrap Dataset**

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	yes	167	Yes

# Random Forest

Imagine that these 4 samples are the entire dataset  
That we are going to build a tree from...

Original Dataset



Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
------------	------------------------	------------------	--------	---------------

To create a bootstrapped dataset that is the same size as the original, we just randomly select samples from The original dataset

# Random Forest

Original Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes

so it's the first sample in our bootstrapped dataset

# Random Forest

Original Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No

This is the second randomly selected samples from The original dataset...

# Random Forest

Bootstrapped Dataset

Original Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	yes	167	Yes

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	yes	167	Yes

Here is the third randomly selected samples from The original dataset...

# Random Forest

Original Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	yes	167	Yes
Yes	No	yes	167	Yes

And again selected the 4<sup>th</sup> randomly selected samples from The original dataset...



# Random Forest

**Step2: Create a decision tree using the bootstrapped dataset, But only use a random subset of variable (or columns) at each step**

In this example, we will only consider 2 variables(columns)  
At each step

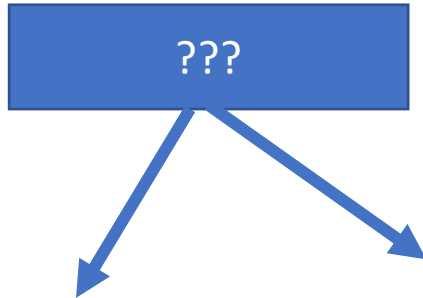
Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	yes	167	Yes
Yes	No	yes	167	Yes

# Random Forest

**Step2: Create a decision tree using the bootstrapped dataset, But only use a random subset of variable (or columns) at each step**

**Instead of considering all 4 variables to figure out how to split  
The root node**



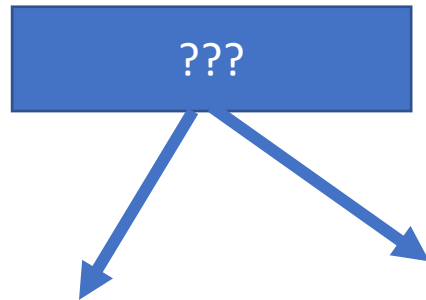
Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	yes	167	Yes
Yes	No	yes	167	Yes

# Random Forest

**Step2: Create a decision tree using the bootstrapped dataset, But only use a random subset of variable (or columns) at each step**

Instead of considering all 4 variables to figure out how to split  
The root node, **we randomly select 2**



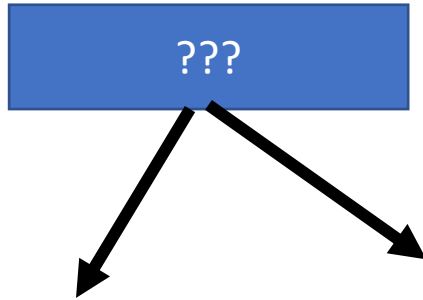
Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	yes	167	Yes
Yes	No	yes	167	Yes

# Random Forest

**Step2: Create a decision tree using the bootstrapped dataset, But only use a random subset of variable (or columns) at each step**

Instead of considering all 4 variables to figure out how to split  
The root node, **we randomly select 2**



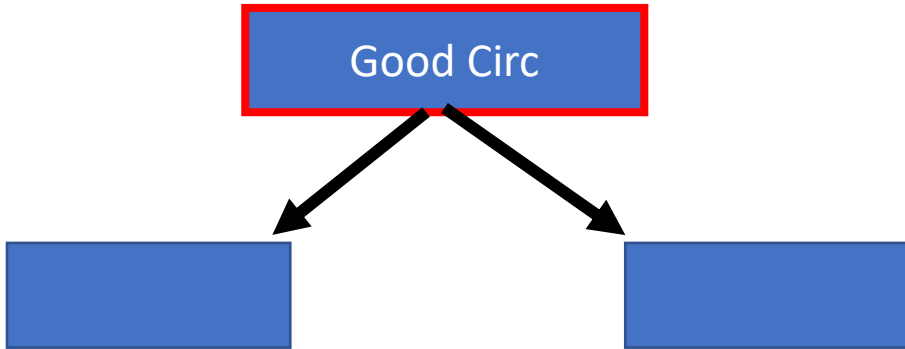
In this case, we randomly selected **Good Blood Circulation**  
**And Blocked Arteries** as candidates for root node

Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	yes	167	Yes
Yes	No	yes	167	Yes

# Random Forest

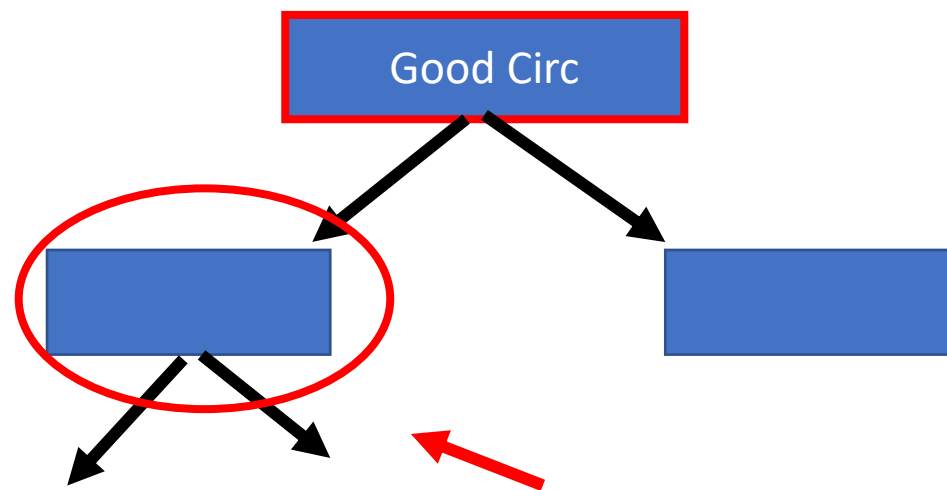
Just for the sake of the example, assume that **Good Blood Circulation** did the best job separating the samples



Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	yes	167	Yes
Yes	No	yes	167	Yes

# Random Forest



Now we need to figure out how to split samples at this node

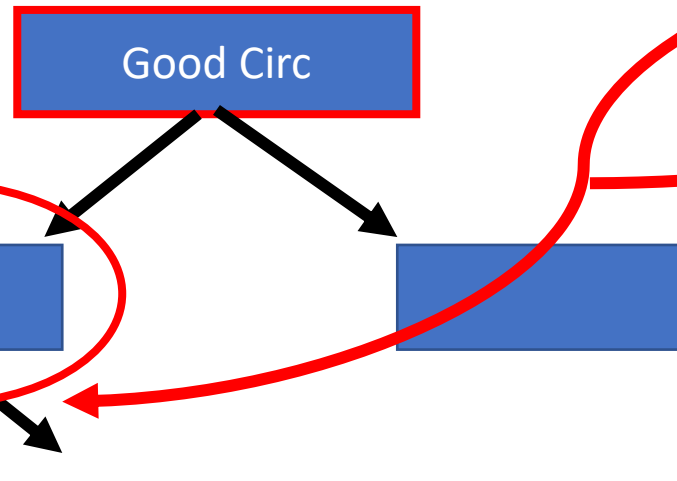
Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	yes	167	Yes
Yes	No	yes	167	Yes

# Random Forest

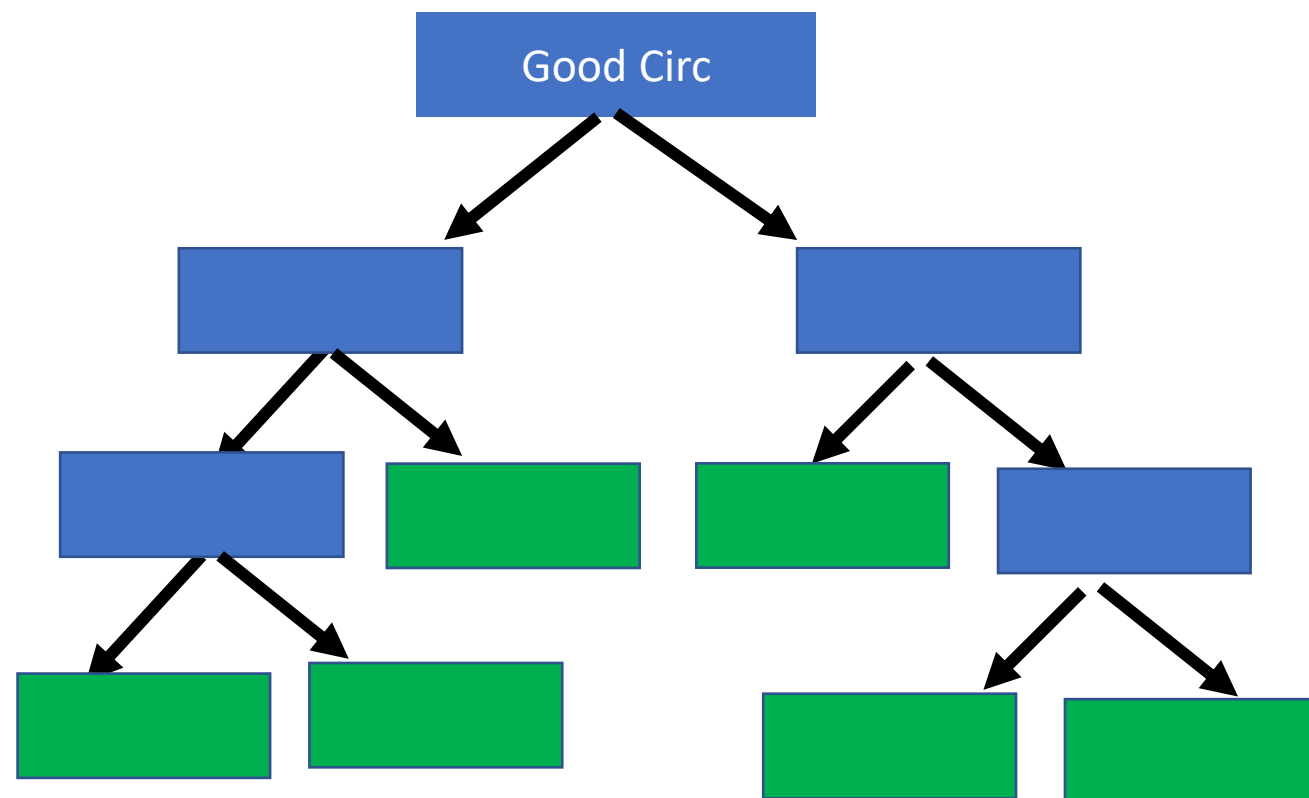
Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	yes	167	Yes
Yes	No	yes	167	Yes



Just like for the root, we randomly select 2 variables as candidate  
Instead of all three remaining columns

# Random Forest



Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes

And we just build a tree as usual, but only considering the random subset of variables at each step

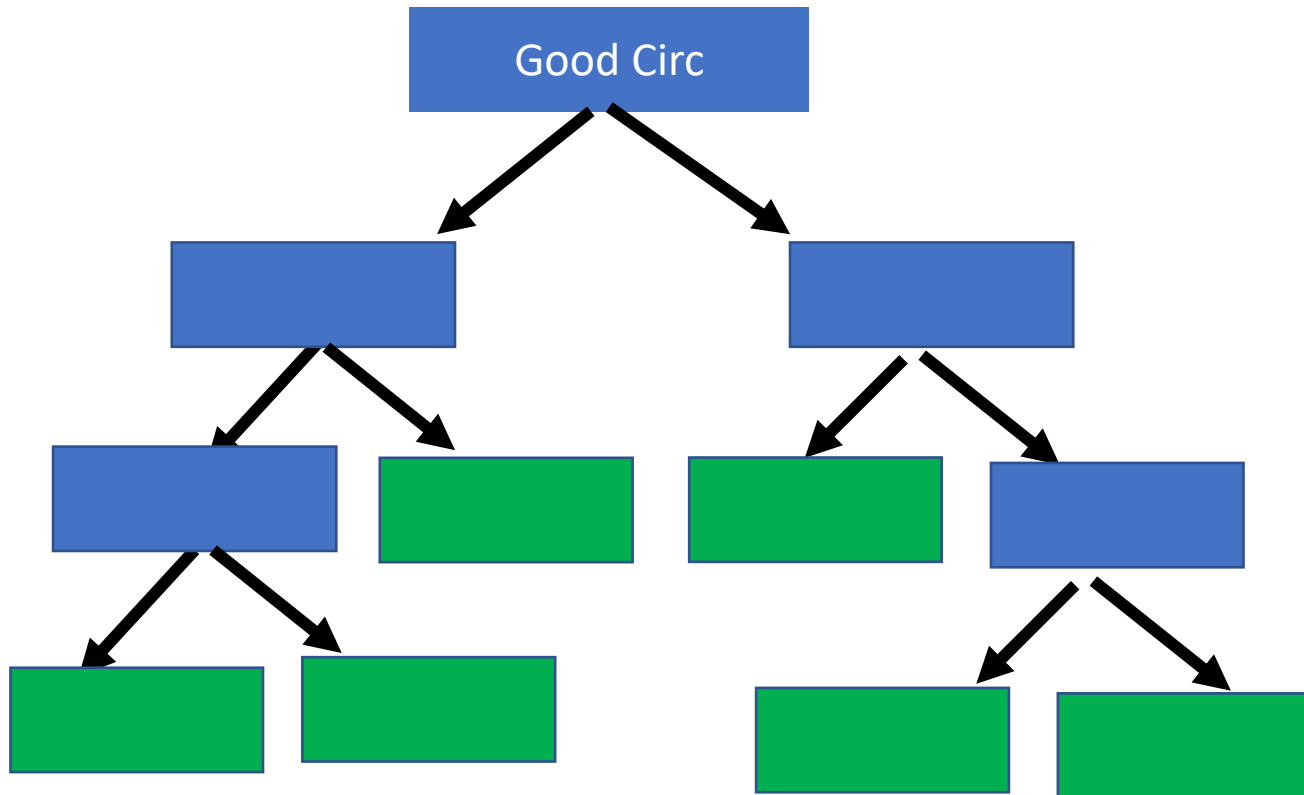
Yes	No	yes	167	Yes
-----	----	-----	-----	-----



# Random Forest

## We built a tree...

- 1) Using a bootstrapped dataset
- 2) Only considering a random subset of variables at each step

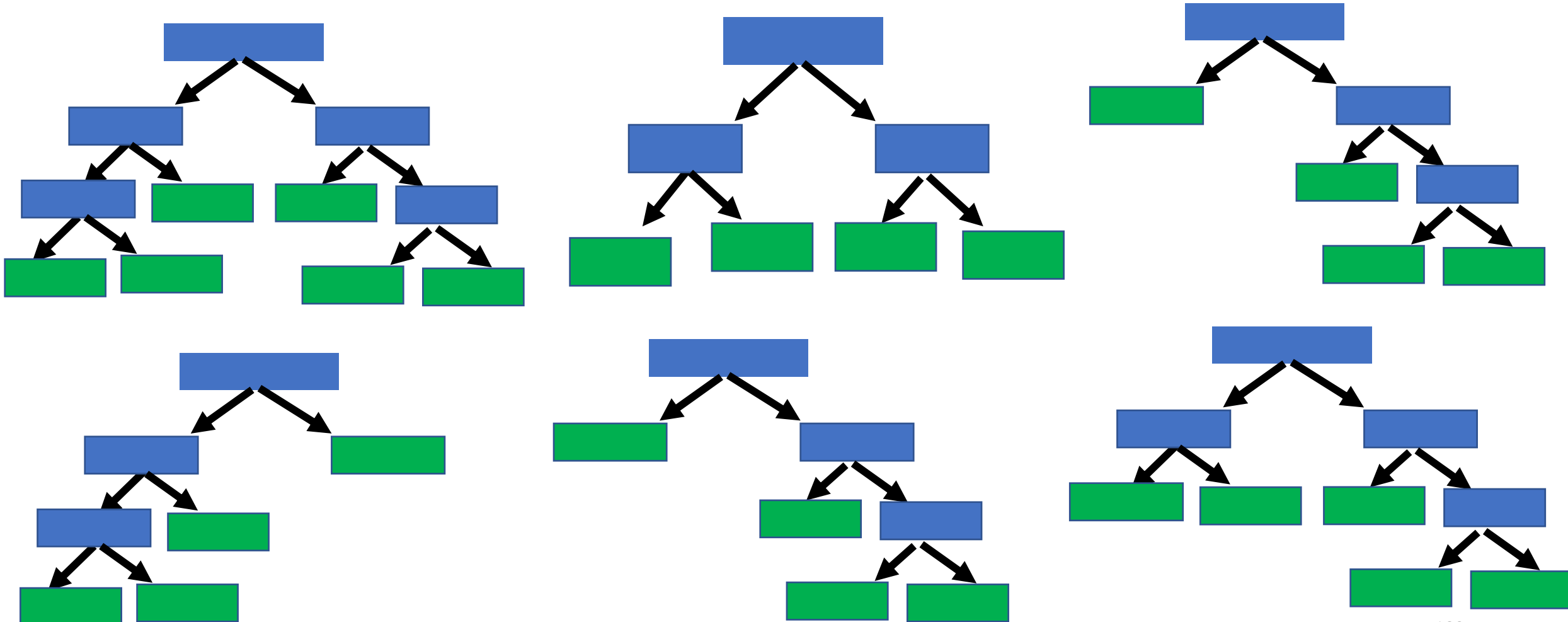


## Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	yes	167	Yes
Yes	No	yes	167	Yes

# Random Forest

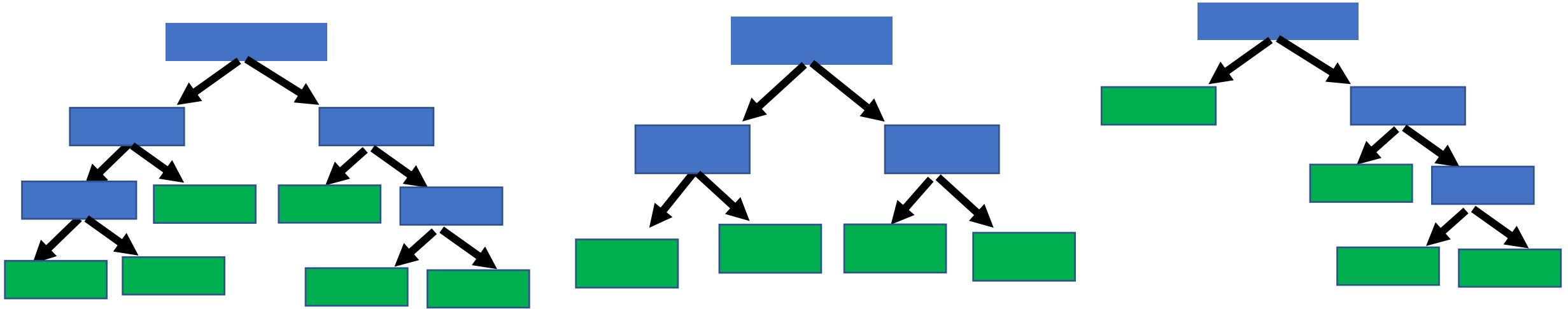
Now go back to step 1 and repeat: Make a new bootstrapped dataset and build a tree considering a subset of variables at each step



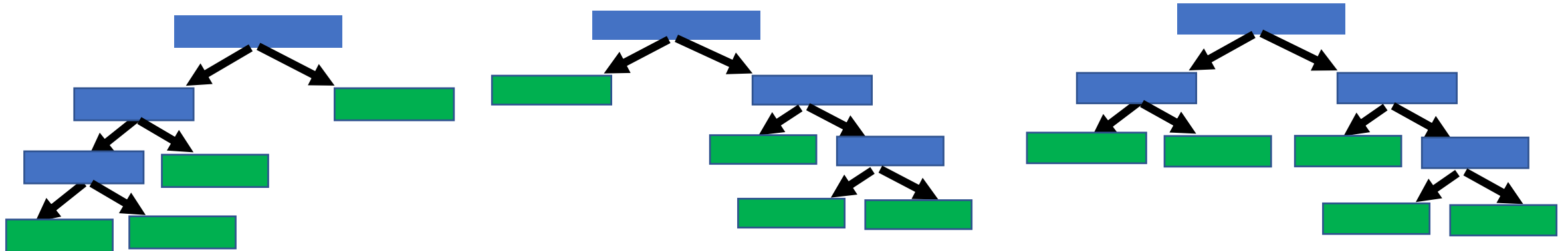
# Random Forest

Now Ideally, you do this 100 s of times, but we only have to show 6...

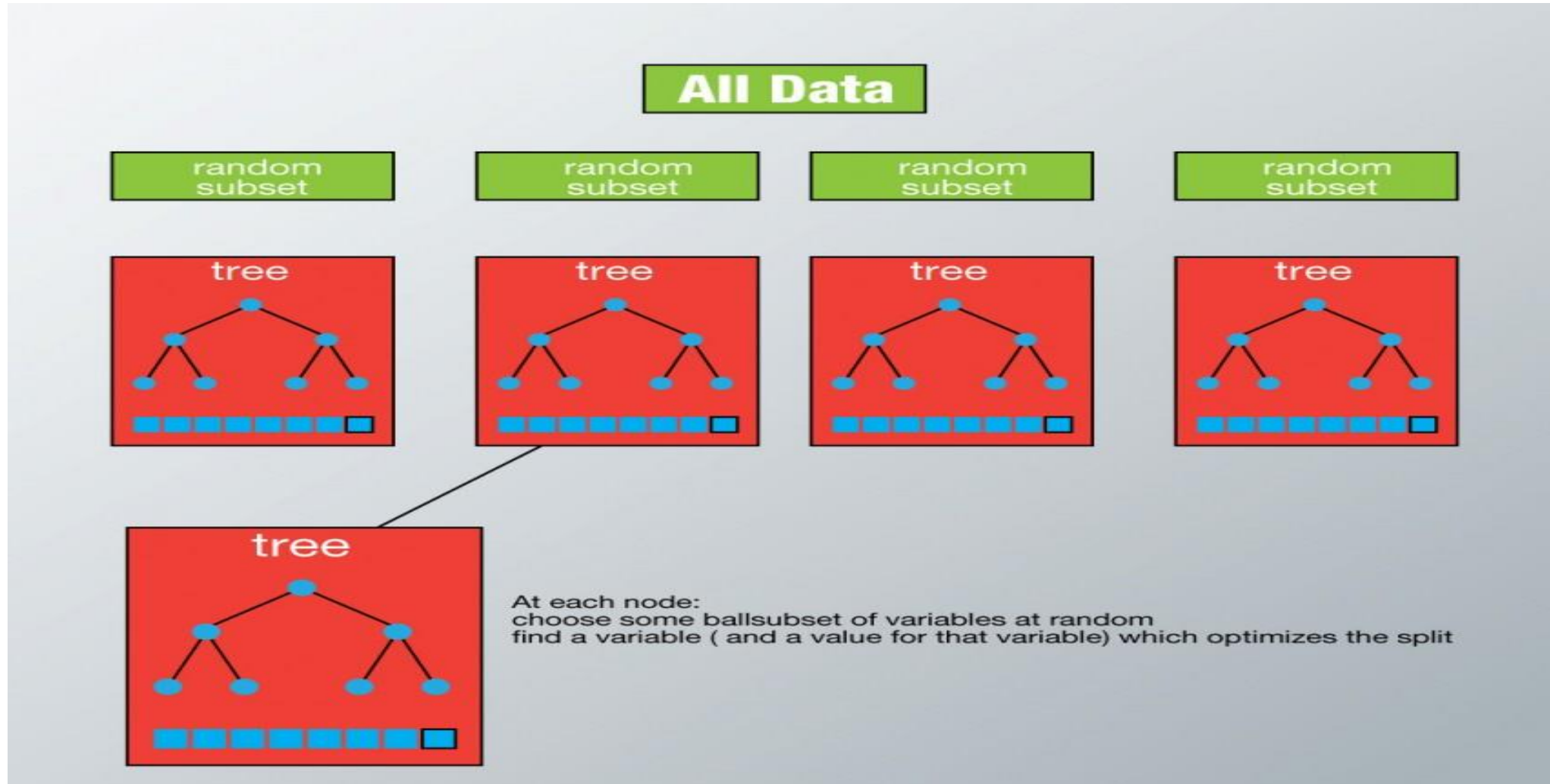
Using a bootstrapped sample and considering only a subset of the variables at each step results in a wide variety of trees



The variety is what makes random forest effective than individual decision



# Random Forest

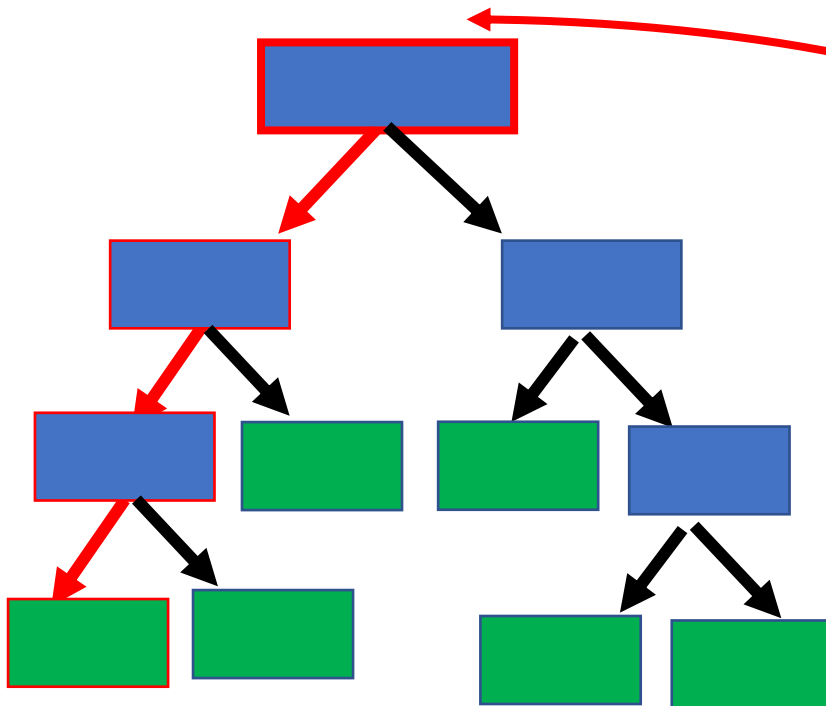


# Random Forest

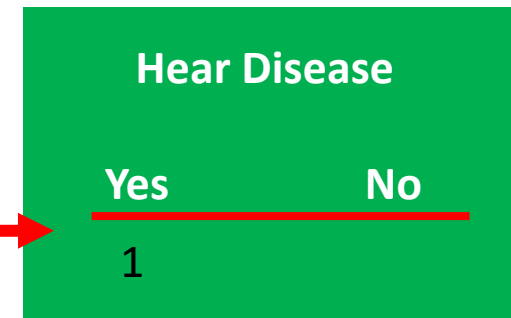
So we take the data and run it down the first tree that we made.....

We get a new patient

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
yes	No	No	168	



We get all the measurements and we want to predict if they have Heart disease or not

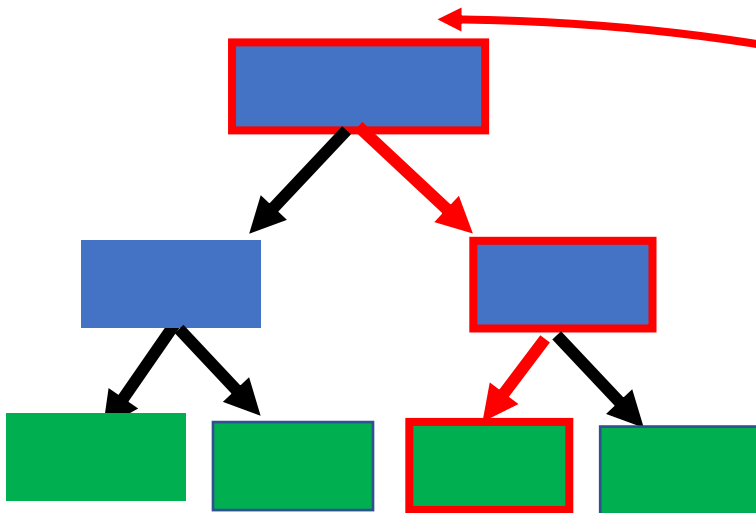


The first tree says yes.....

# Random Forest

Now we run the data down the second tree that we made

We get a new patient



Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
yes	No	No	168	

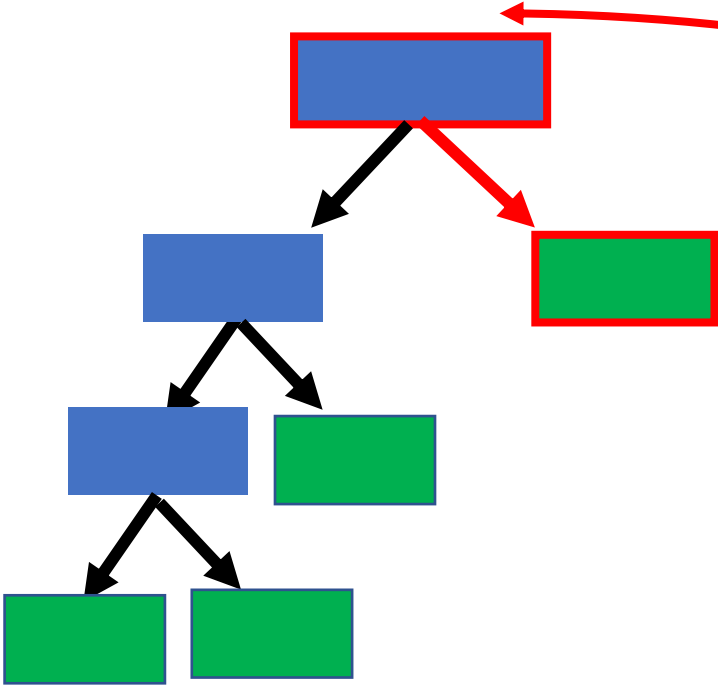
The second tree says yes.....

Hear Disease	
Yes	No
2	

# Random Forest

Then we repeat for all the trees that we made...

We get a new patient



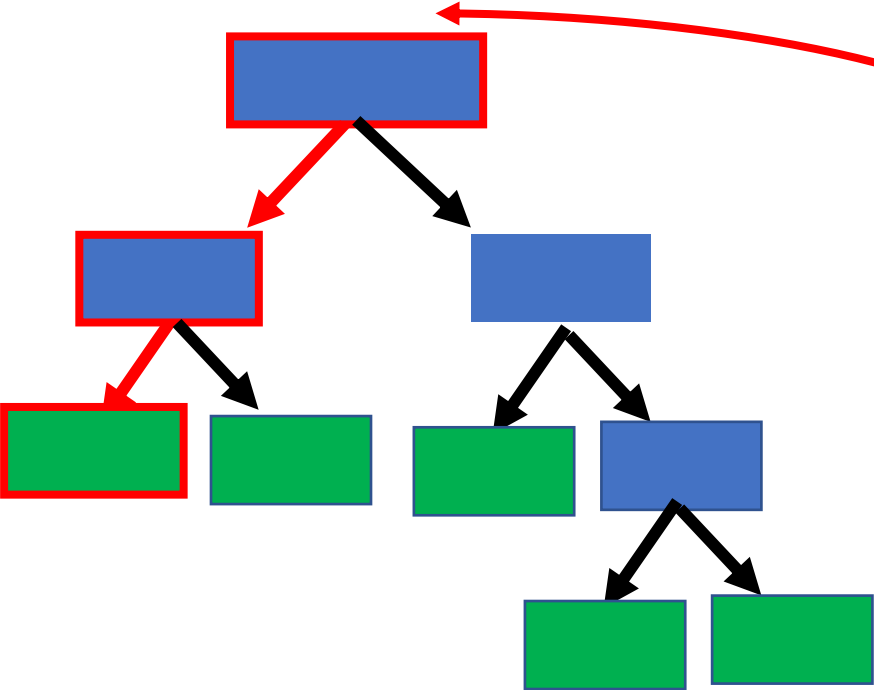
Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
yes	No	No	168	

Hear Disease	
Yes	No
2	1

# Random Forest

Then we repeat for all the trees that we made...

We get a new patient



Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
yes	No	No	168	

Hear Disease	
Yes	No
5	1



# Random Forest

We get a new patient

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	No	No	168	Yes

After running the data down all of the trees in the random forest,  
We see which option received more votes...

In this case, “yes ” received the most votes, so we will concludes that  
This patient has heart disease

Hear Disease	
Yes	No
5	1

# Random Forest

We get a new patient

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	No	No	168	Yes

**Bootstrapping** the data plus using the aggregate to make a decision is called **“Bagging”**

# Random Forest

We allowed duplicate entries in the bootstrapped dataset

Original Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	yes	167	Yes
Yes	No	yes	167	Yes

# Random Forest

As a result, This entry was not included in the bootstrapped dataset

Original Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	yes	167	Yes

Bootstrapped Dataset


Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	yes	167	Yes
Yes	No	yes	167	Yes

# Random Forest

Original Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	yes	167	Yes

Here is the entry that did not end up in the bootstrapped dataset



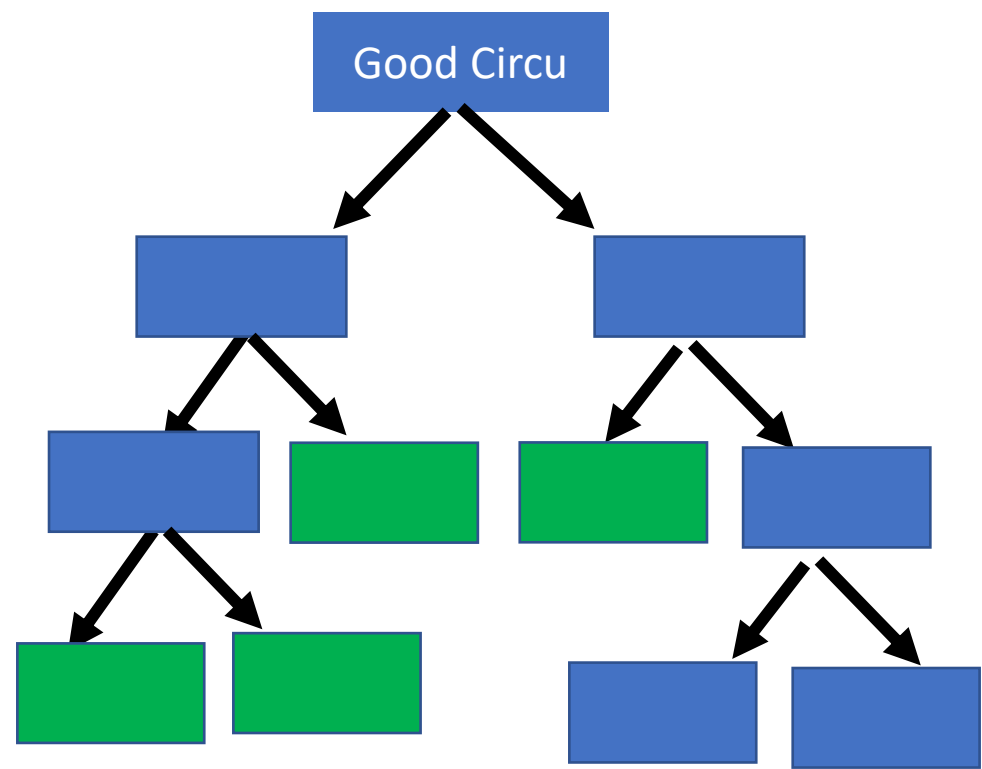
Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	No	210	No



This is called the **“Out-Of-Bag Dataset”**

# Random Forest

Since the out of bag data was not used to create this Tree..

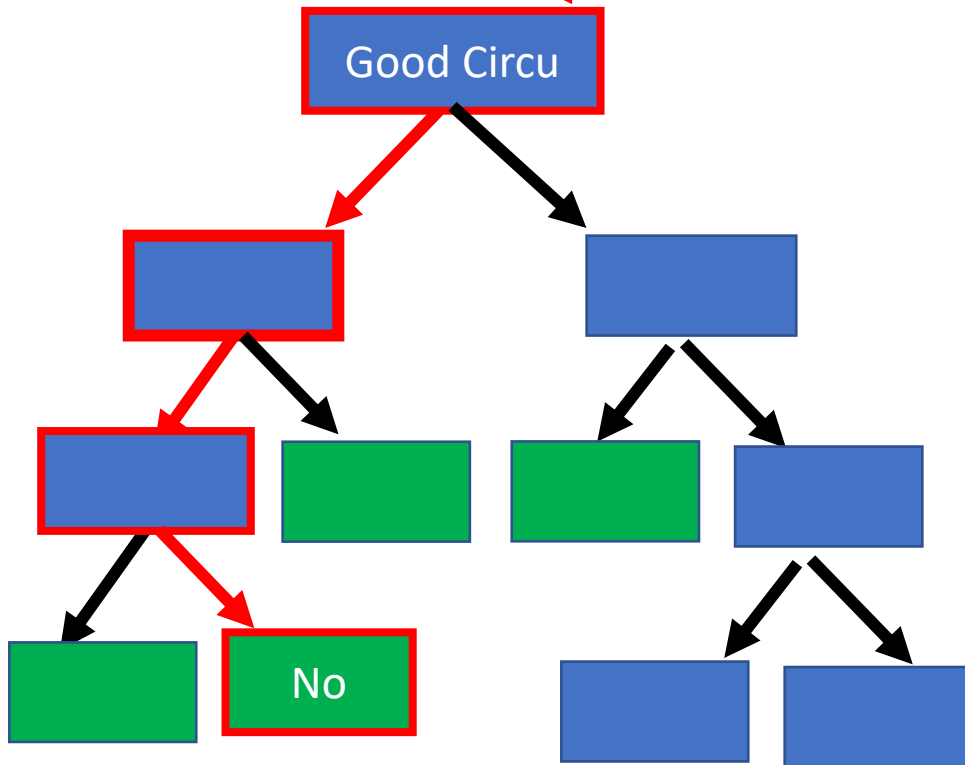


This is called the **“Out-Of-Bag Dataset”**

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	No	210	No

# Random Forest

Since the out of bag data was not used to create this Tree.. We can run it through and see if it correctly classifies The sample as **“NO Heart Disease”**

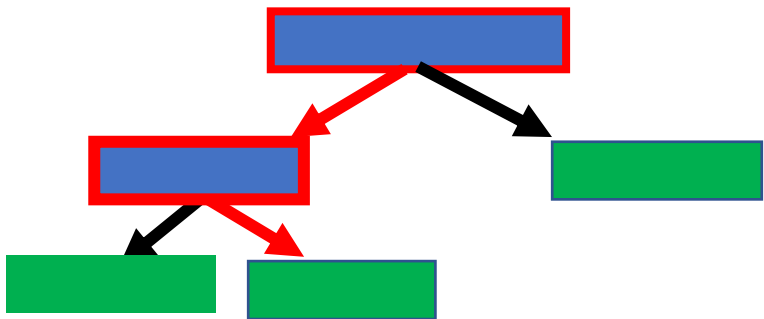
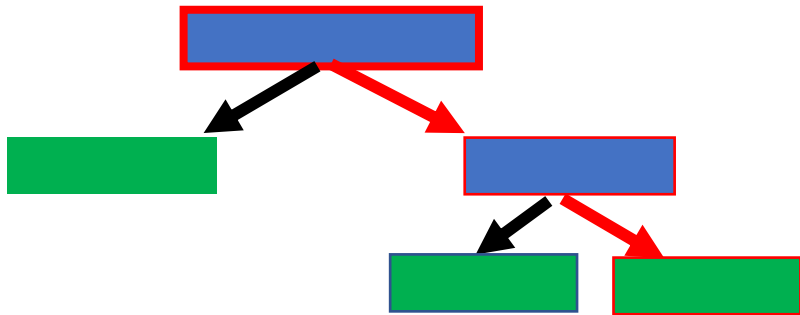
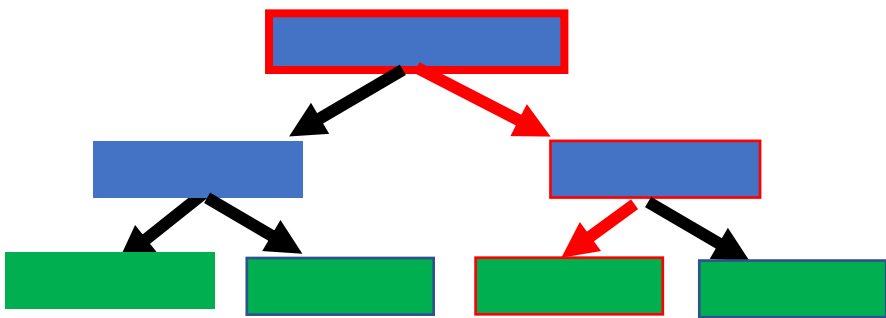


This is called the **“Out-Of-Bag Dataset”**

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	No	210	No

In this case, the tree correctly labels the Out-of-Bag sample **“NO”**

# Random Forest

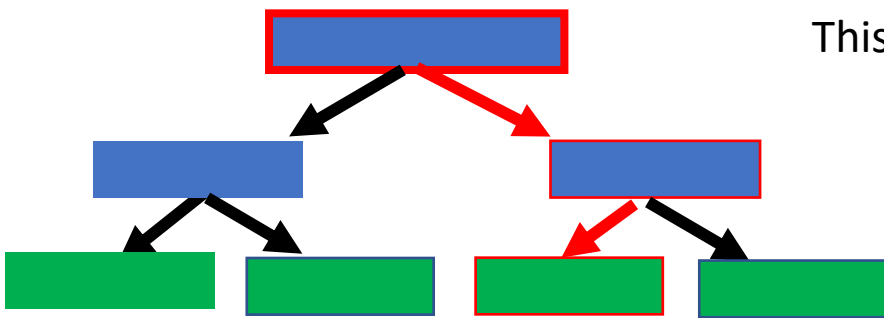


Then we run this Out-Of-Bag sample through all of The other trees that were built without it.

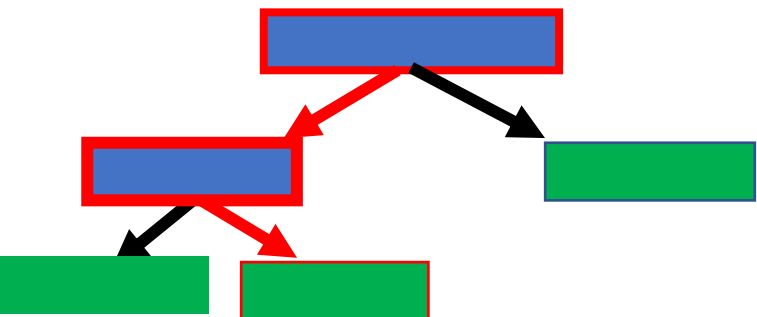
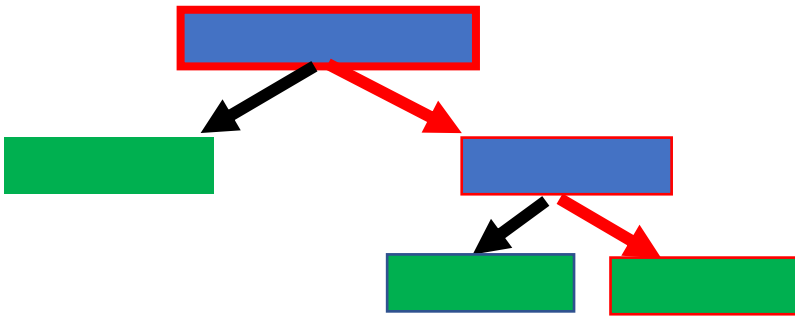
Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	No	210	No



# Random Forest



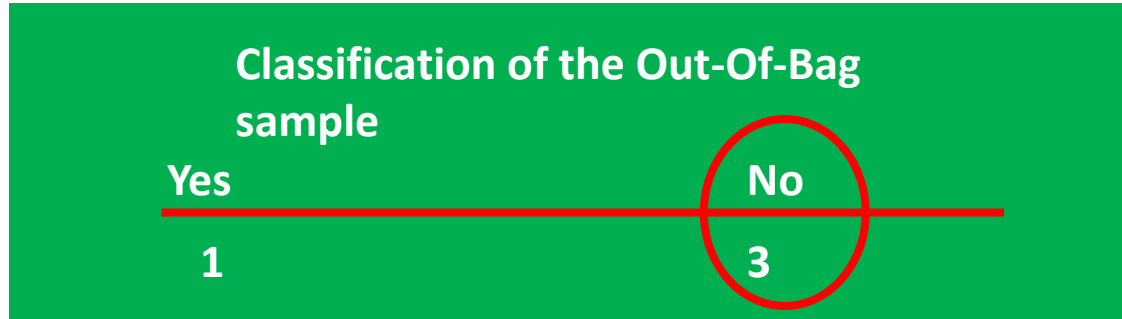
This tree incorrectly labels the Out-of-Bag sample “yes”



Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	No	210	No

These tree correctly labels the Out-of-Bag sample “No”

# Random Forest



Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	No	210	No

Since the label with the most votes wins , it is the label that  
We assign this Out-of-Bag sample

In this case, the Out-of-Bag sample is correctly  
labeled by the Random Forest

# Random Forest

Classification of the Out-Of-Bag  
sample

Yes

No

1

3

We the do the same thing for all of the other  
Out-Of-Bag samples for all of the trees

# Random Forest

## Classification of the Out-Of-Bag sample

Yes

No

1

3

## Classification of the Out-Of-Bag sample

Yes

No

4

0

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes

In this case, the Out-of-Bag sample was also correctly labeled by the Random Forest

# Random Forest

Classification of the Out-Of-Bag  
sample

Yes

1

No

3

Classification of the Out-Of-Bag  
sample

Yes

4

No

0

Classification of the Out-Of-Bag  
sample

Yes

3

No

1

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
No	No	No	125	No

In this case, the Out-of-Bag sample was incorrectly labeled by the Random Forest

# Random Forest

## Classification of the Out-Of-Bag sample

Yes

1

No

3

## Classification of the Out-Of-Bag sample

Yes

4

No

0

## Classification of the Out-Of-Bag sample

Yes

3

No

1

Ultimately, we can measure how accurate our random forest is  
By the proportion of Out-Of-Bag samples that were correctly  
Classified by the random Forest

The proportion of Out-Of-Bag samples that were incorrectly  
Classified is the “**Out-Of-Bag Error**”

# Random Forest

**We now know how to:**

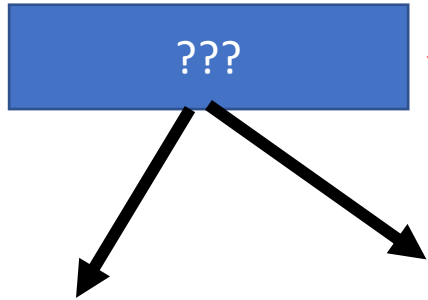
**1)Build a Random Forest**

**2)Use a Random Forest**

**3)Estimate the accuracy of a Random Forest**

# Random Forest

Remember when we built our first tree and  
We only used 2 variables(columns of data) to make  
A decision at each step?



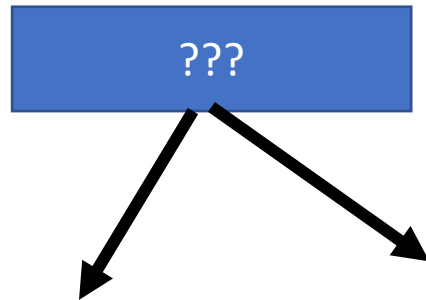
Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	yes	167	Yes
Yes	No	yes	167	Yes



# Random Forest

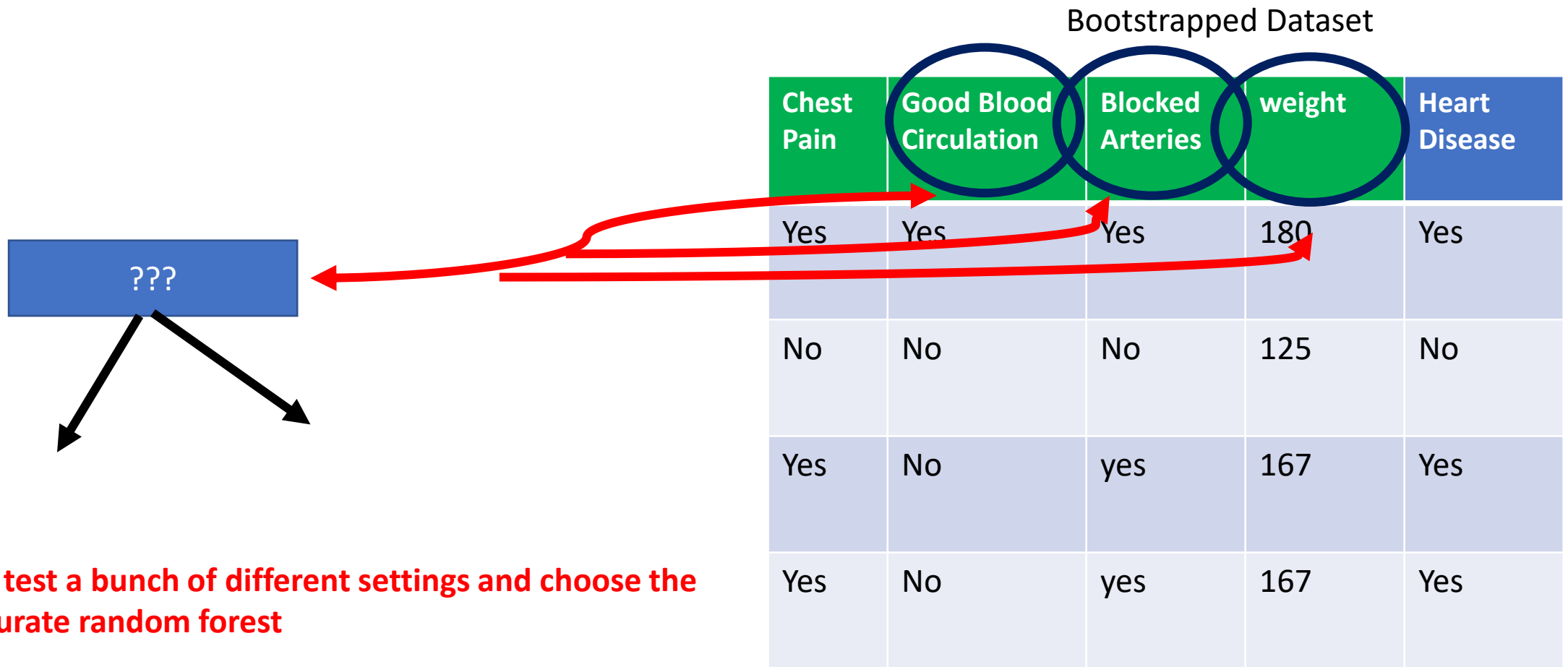
Now we can compare the Out-Of-Bag error for a random Forest built using only 2 variables per step...  
... to a random forest built using 3 variables per step



Bootstrapped Dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	yes	167	Yes
Yes	No	yes	167	Yes

# Random Forest



...and we test a bunch of different settings and choose the Most accurate random forest

# Random Forest

In other words.....

- 1) Build a Random Forest
  - 2) Estimate the accuracy of a Random Forest
- ..... change the number of variables used per step...
- 

Do this for bunch of times and then choose the one that is most accurate  
Typically, we start by using the **square of the number of variables** and try a few setting above and below that values.

# Random Forest

- **Advantages of Random Forest**
- This algorithm can solve both type of problems i.e. classification and regression and does a decent estimation at both fronts.
- One of benefits of Random forest which excites me most is, the power of handle large data set with higher dimensionality. It can handle thousands of input variables and identify most significant variables so it is considered as one of the dimensionality reduction methods. Further, the model outputs Importance of variable, which can be a very handy feature (on some random data set).
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing errors in data sets where classes are imbalanced.
- The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.

# Random Forest

- **Advantages of Random Forest**

Random Forest involves sampling of the input data with replacement called as bootstrap sampling. One third of the data is normally used for training and can be used to testing. These are called the **out of bag samples**. Error estimated on these out of bag samples is known as **out of bag error**. Study of error estimates by Out of bag, gives evidence to show that the out-of-bag estimate is as accurate as using a test set of the same size as the training set. Therefore, using the out-of-bag error estimate removes the need for a set aside test set.

# Random Forest

- **Disadvantages of Random Forest**

It surely does a good job at classification but not as good as for regression problem as it does not give precise continuous nature predictions. In case of regression, it doesn't predict beyond the range in the training data, and that they may over-fit data sets that are particularly noisy.

Random Forest can feel like a **black box approach for statistical modelers** – you have very little control on what the model does. **You can at best – try different parameters and random seeds**



Q&A