

Machine Learning Basis

Random Forests

05/01/2020

傳統統計方法最常被用來解決的問題

➤ Group difference (比較: 不同組群的組別差異)

-- diseased vs. healthy control subjects

(group factor: disease)

-- elder vs. younger subjects

(group factor: age)

... etc.

➤ Correlation (相關性, 自變數” independent variable” 與應變數” dependent variable” 之間的關係式)

-- 身高 = $a * \text{體重} + b$

-- blood T1 = $1817 - 8.2 * \text{age} - 37.4 * \text{sex}$ (sex: male=1, female=0)

... etc.

Machine Learning

- 監督式學習 (Supervised learning)

- 分類 (Classification)

Group difference

- 迴歸 (Regression)

Correlation/ Group difference

- 非監督式學習 (Unsupervised learning)

- 分群 (Clustering)

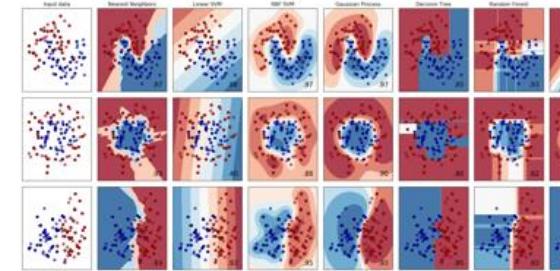
Group difference

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

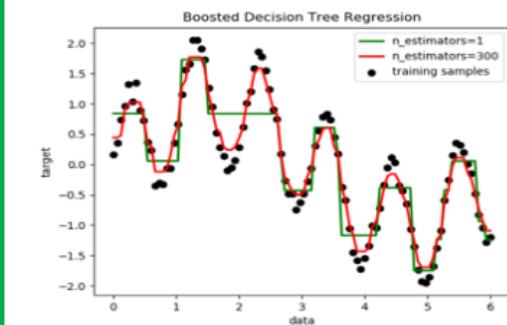
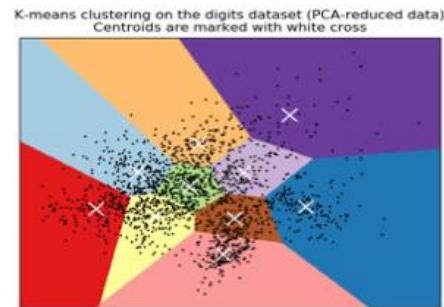
Algorithms: SVR, nearest neighbors, random forest, and more...

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



Machine Learning Algorithms

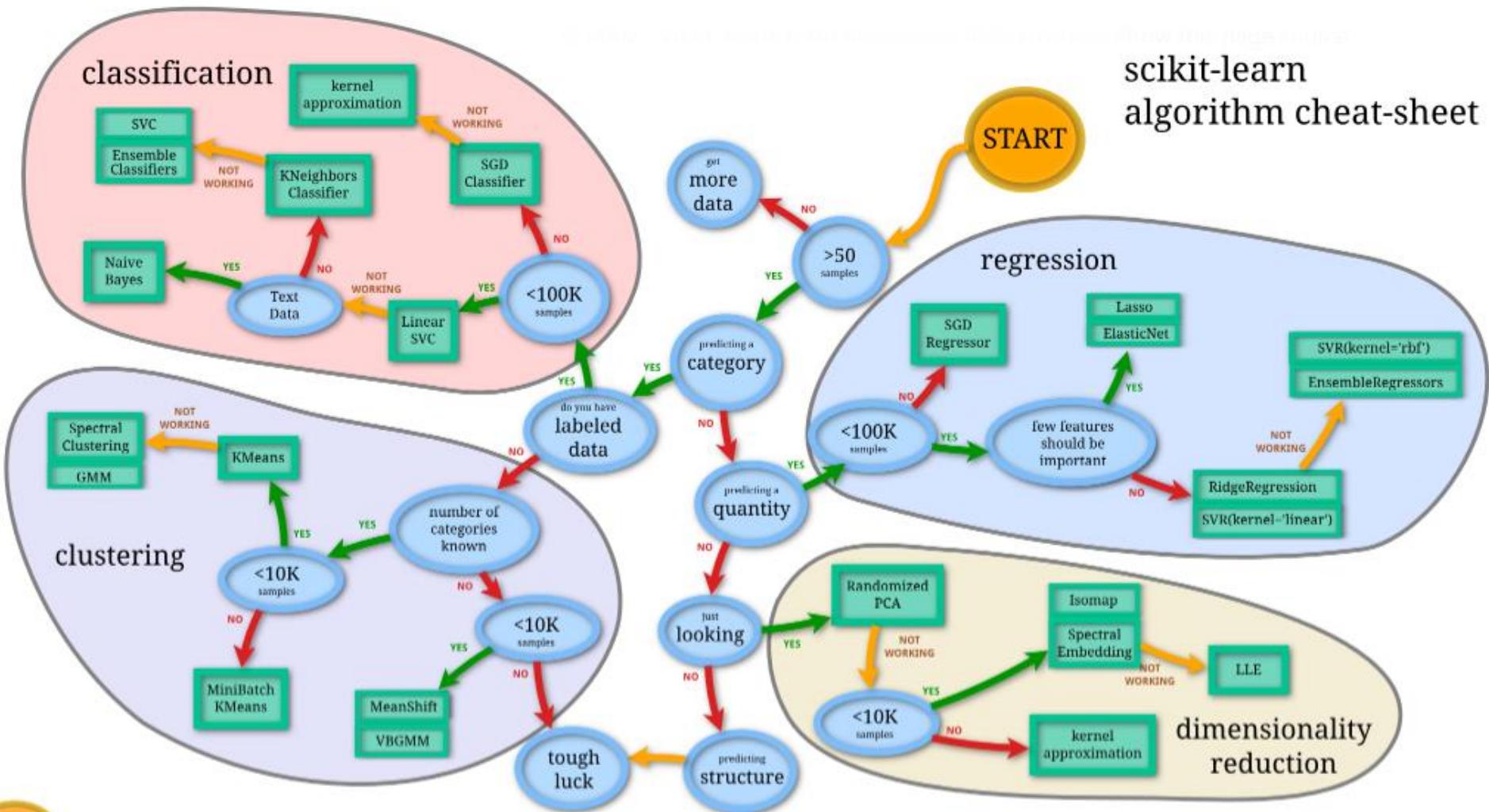
常見的演算法

- Random forest (Heather)
- SVM (support vector machine, 支撐向量機) (CK)
- k-nearest neighbors (Charles)
- k-means (Danny)
- Regression
 - linear regression, multiple regression, logistic regression (姝儀+郁慈)
 - PCA (Principal Component Analysis, 主成分分析) (Larry)
- CNN (Convolutional Neural Network)
- AdaBoost
- Gradient Boosting
- Naive Bayes
- LDA
- QDA
- RBMs
- ...

Requirements for your presentations in leading the Discussion

- PPT files
- References for everyone

Machine Learning Map



Machine Learning Algorithms in Classification

Classifiers

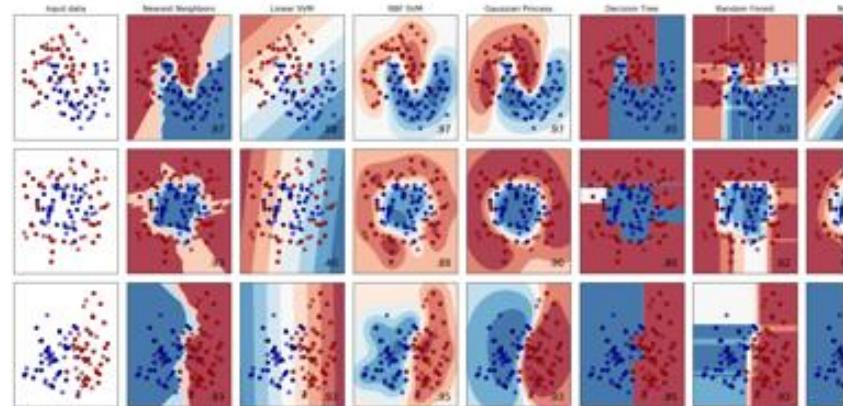
- SVM (Support Vector Machine)
- k-nearest neighbors
- Random Forest
- AdaBoost Classifier
- Gradient Boosting
- Naive Bayes
- LDA
- QDA
- RBMs
- Logistic Regression
- RBM + Logistic Regression Classifier
- CNN (Convolutional Neural Network)
- ...

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



Machine Learning Algorithms in Regression

Regression algorithms

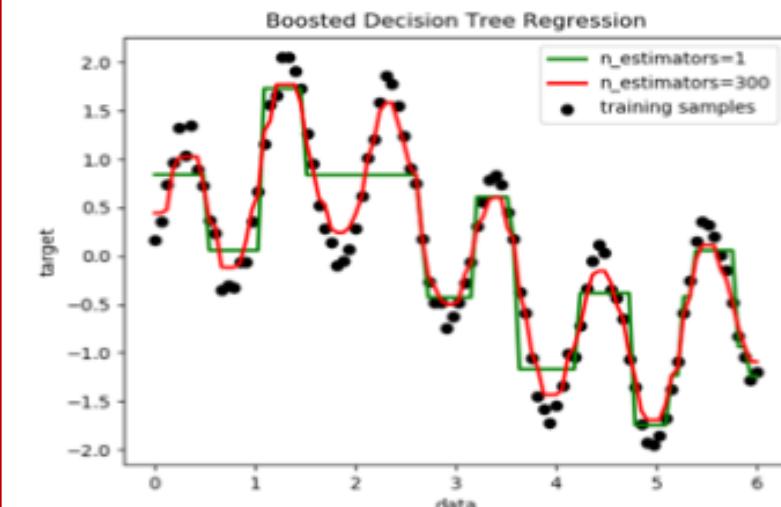
- SVM (Support Vector Machine)
- k-nearest neighbors
- Random Forest
- SVR
- LASSO
- RBMs
- Logistic Regression
- RBM + Logistic Regression Classifier
- CNN (Convolutional Neural Network)
- ...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...



Machine Learning Algorithms in Clustering

Clustering methods

- K-means (FCM)
- Affinity propagation
- Mean-shift
- Spectral clustering
- Ward hierarchical clustering
- Agglomerative clustering
- DBSCAN
- OPTICS
- Gaussian mixtures
- Birch

MRI 常用: imaging segmentation

Clustering

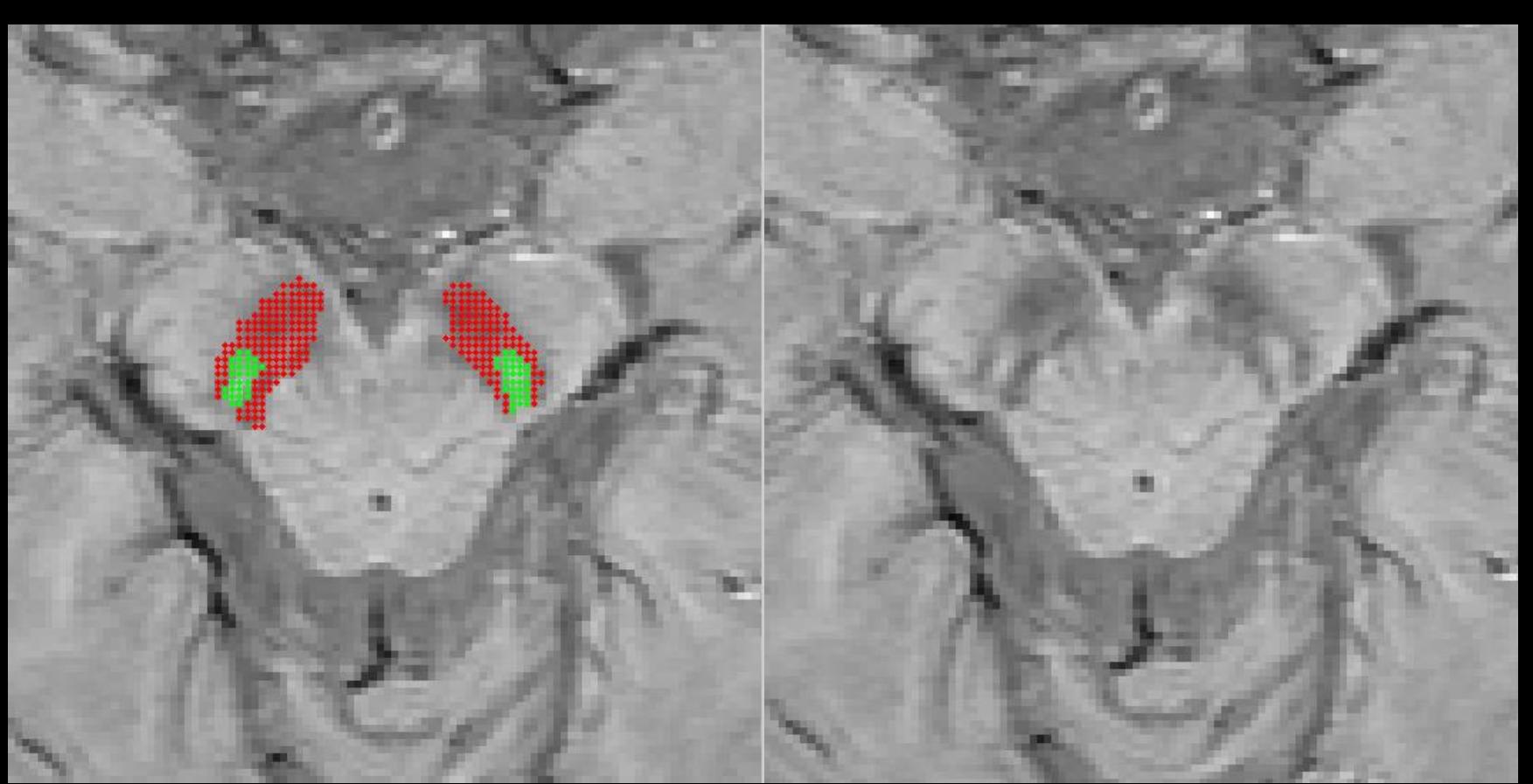
Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



FCM (fuzzy-cluster mean) in Nigrosome-1 segmentation

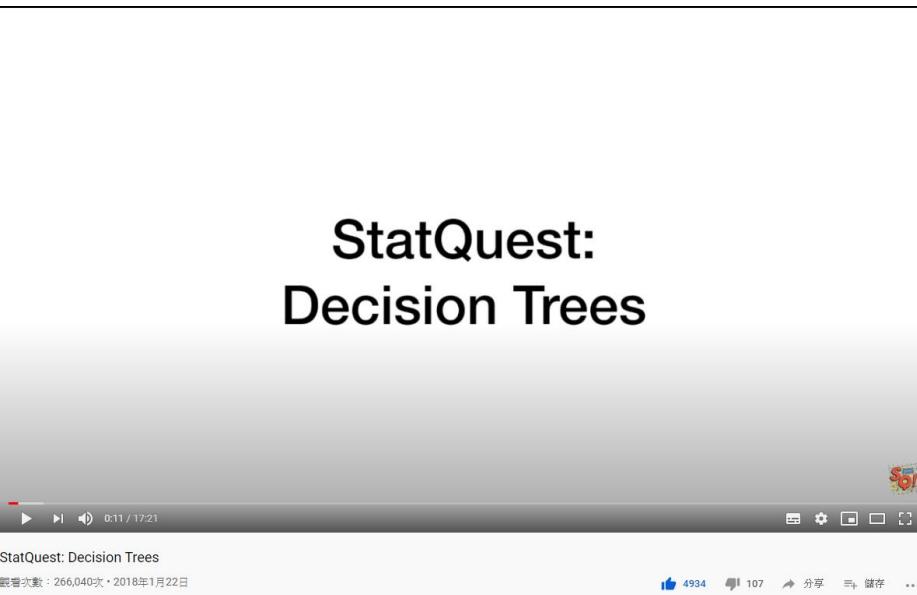


Random Forests Algorithm

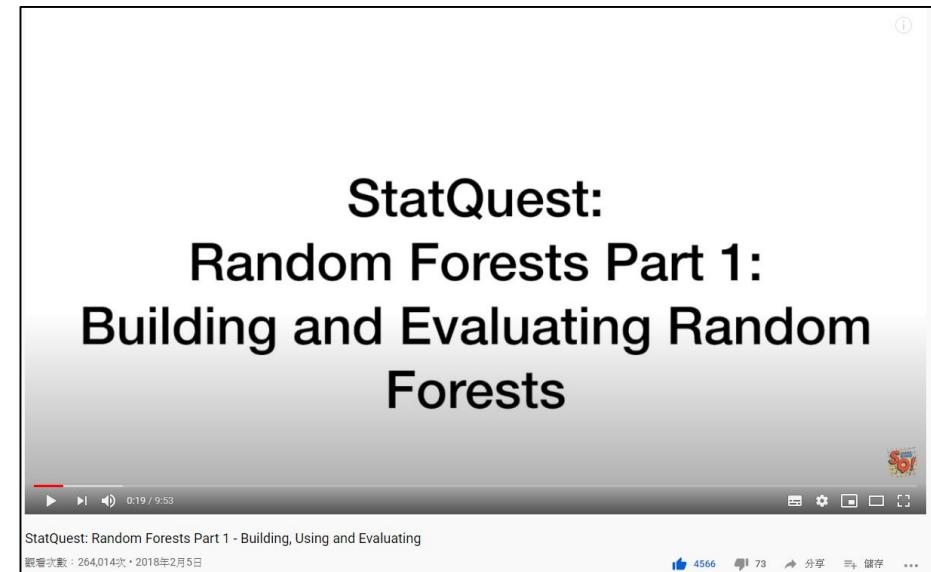
Random Forests Algorithm

Main references

StatQuest:
Decision Trees



StatQuest:
Random Forests Part 1:
Building and Evaluating Random
Forests



<https://www.youtube.com/watch?v=7VeUPuFGJHk>

<https://www.youtube.com/watch?v=J4Wdy0WcxQ>

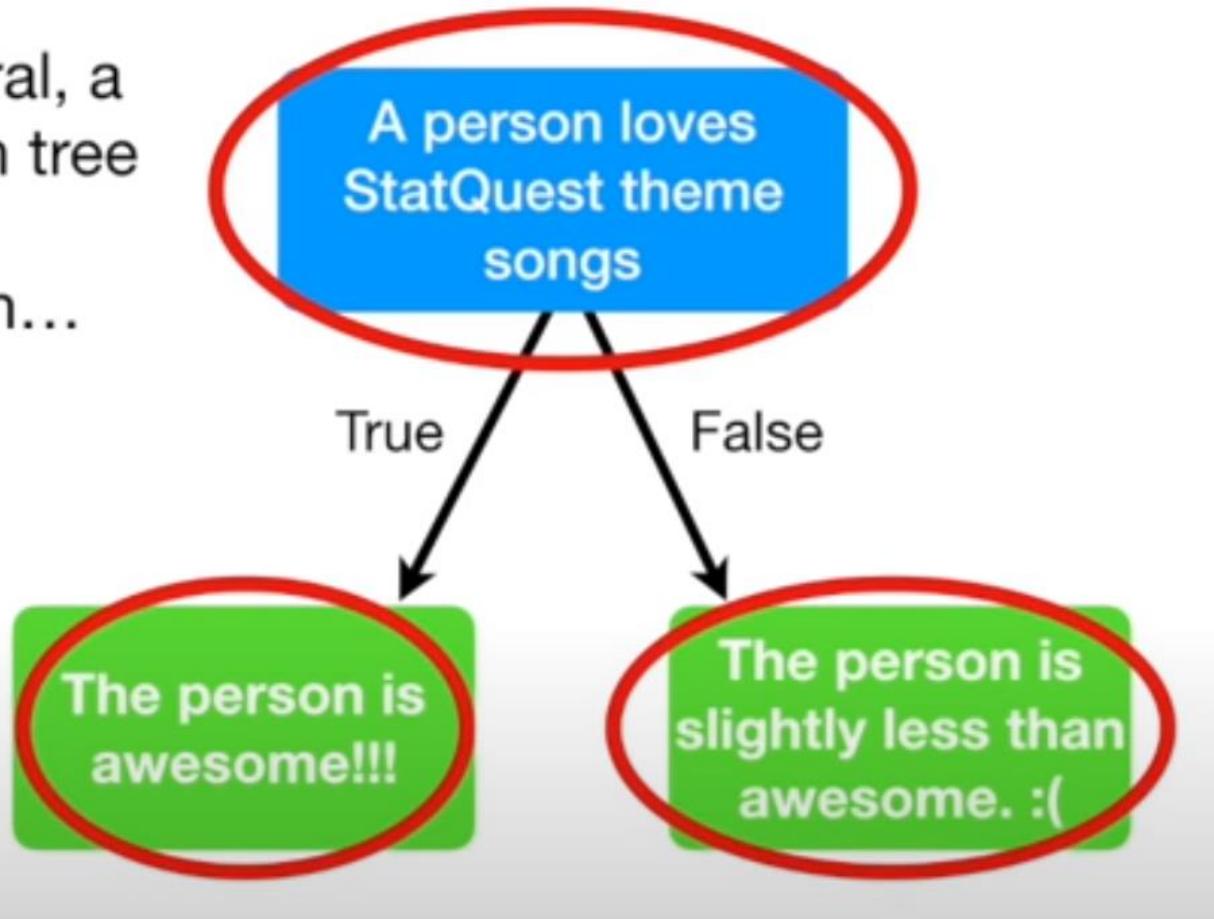
Outline

- Decision trees
- Random Forests
 - Classification
 - Regression
- Performance Scores
- Examples:
 - *Magnetic resonance perfusion image features uncover an angiogenic subgroup of glioblastoma patients with poor survival and better response to antiangiogenic treatment (Liu et al., Neuro-Oncology 2016)*
 - *Prediction of Core Signaling Pathway by Using Diffusion- and Perfusion-based MRI Radiomics and Next-generation Sequencing in Isocitrate Dehydrogenase Wild-type Glioblastoma (Park et al., Radiology 2020)*

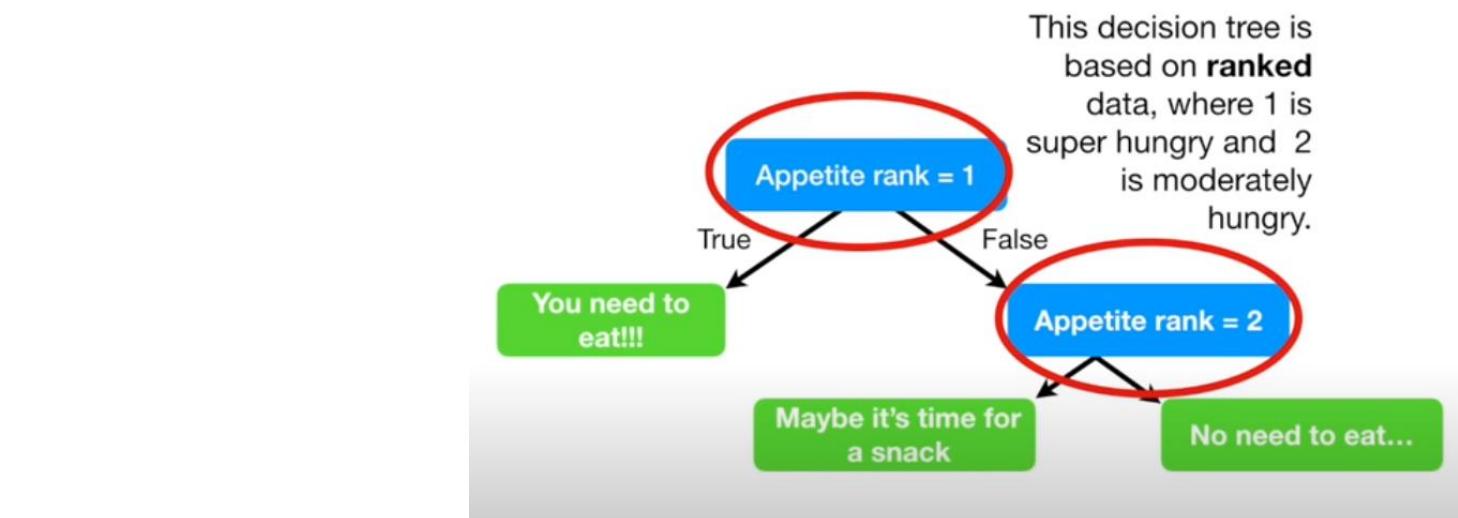
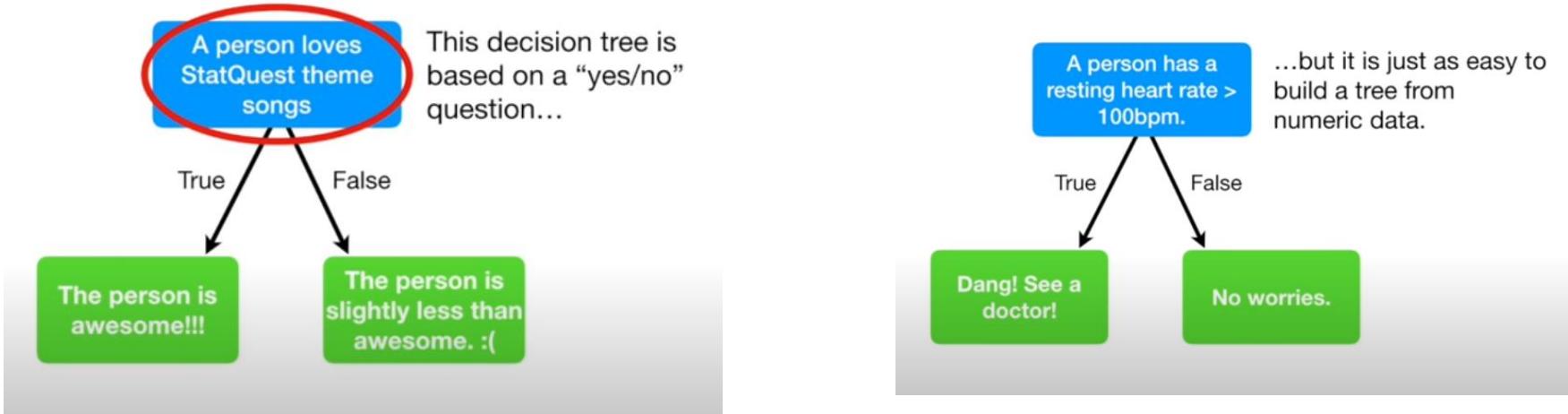
Decision trees: ask a question, then classifies the answer

In general, a decision tree asks a question...

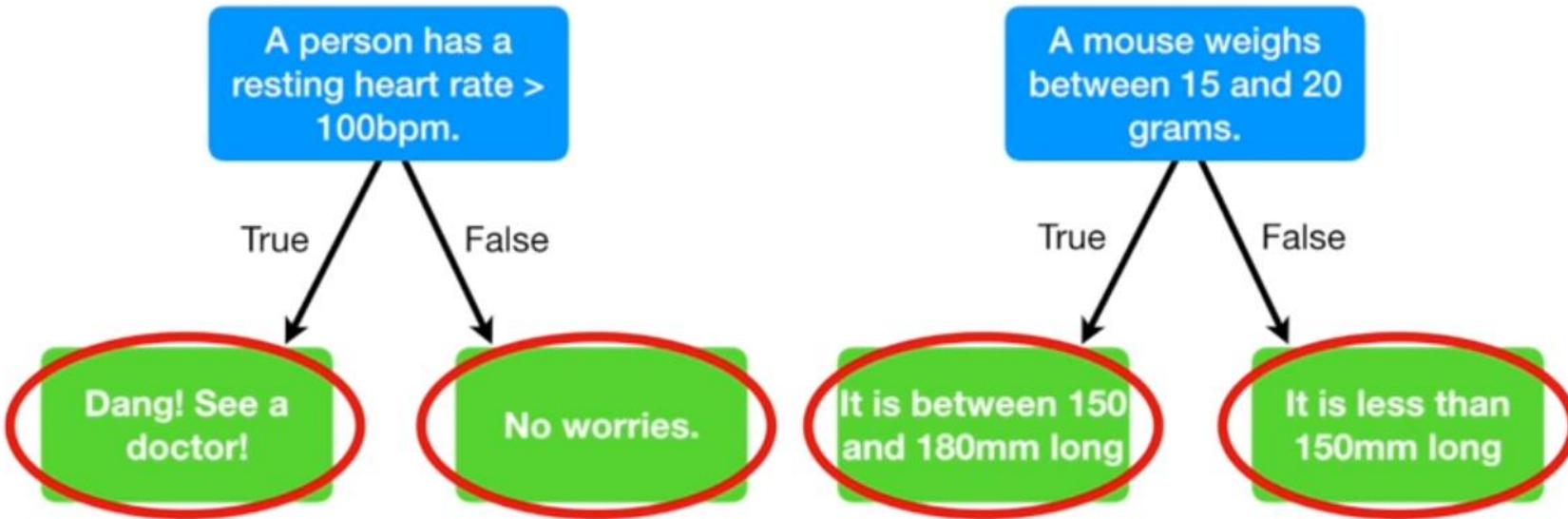
...and then classifies the person based on the answer.



Decision trees: types of questions



Decision trees: types of answers



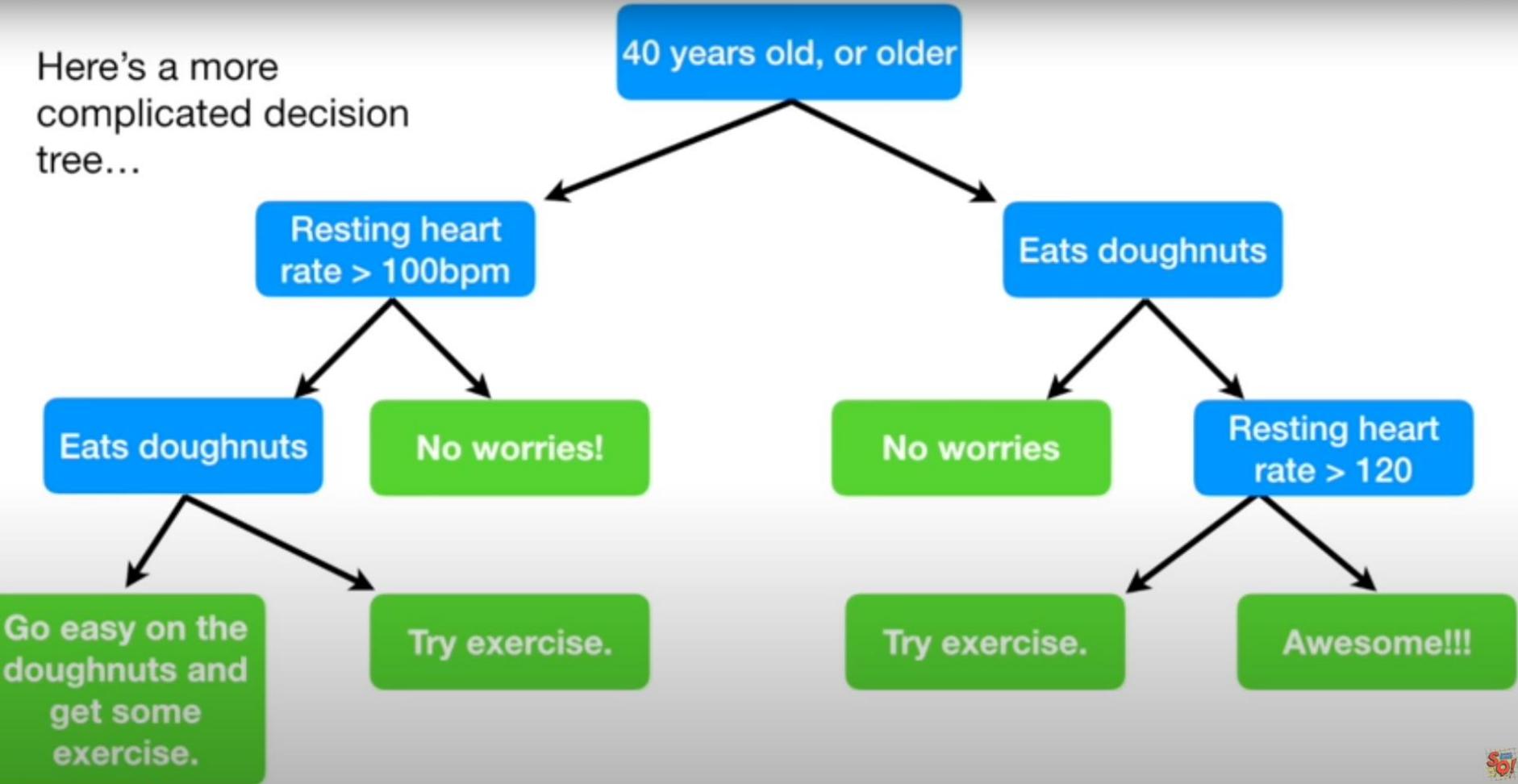
NOTE: The classification can be categories...

...or numeric.

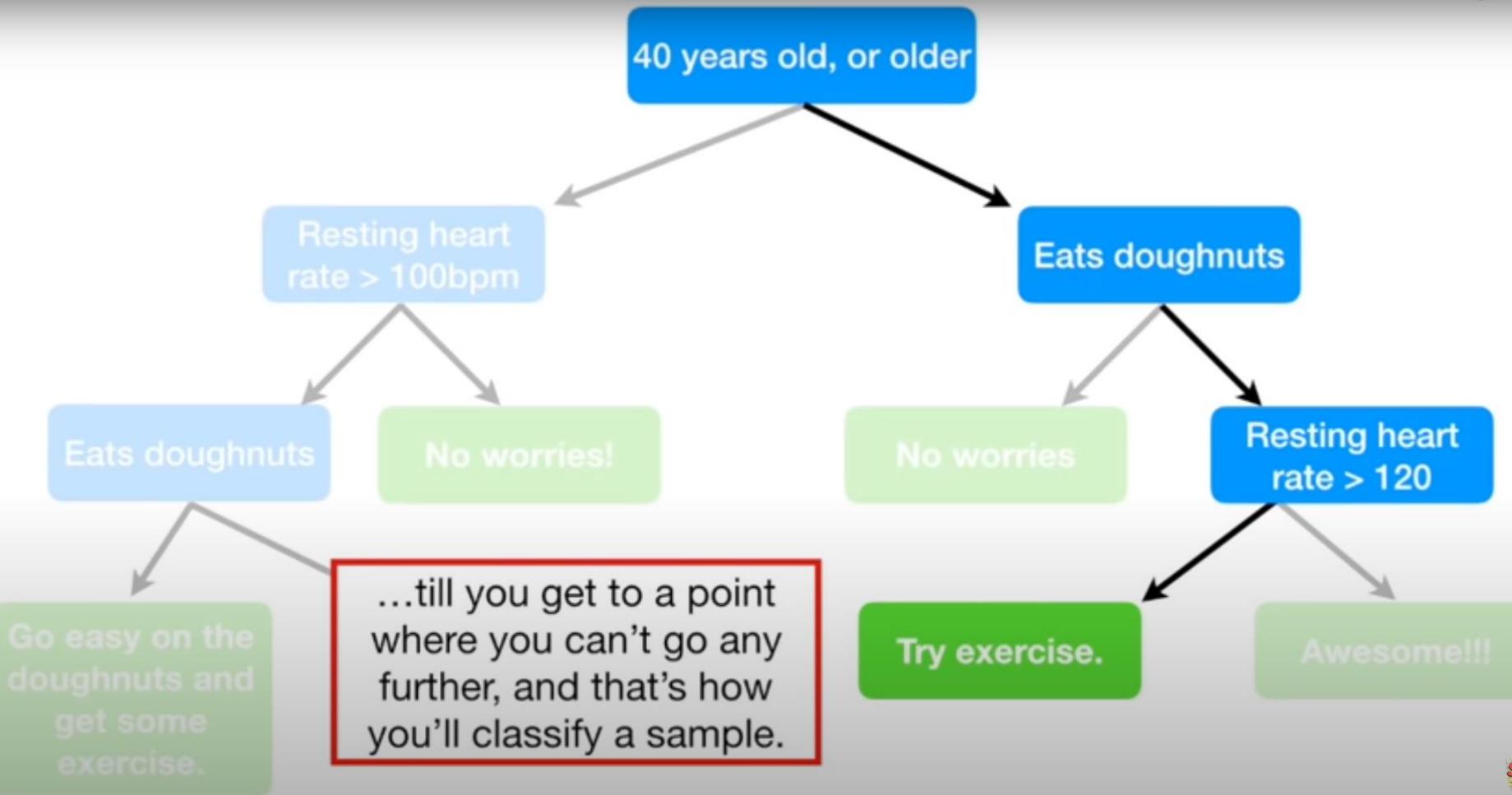
A more complicated decision tree...

StatQuest: Decision Trees

Here's a more complicated decision tree...

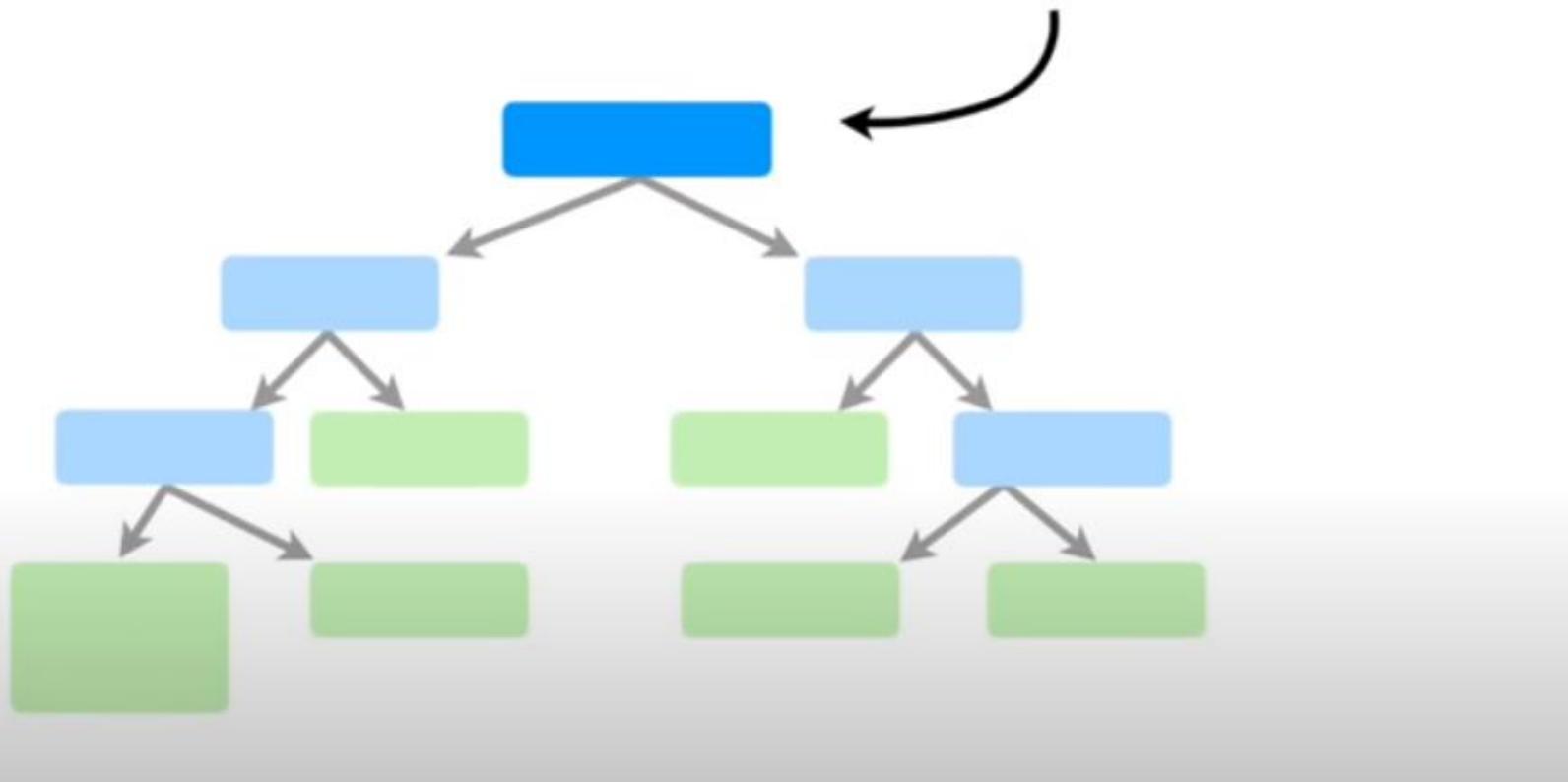


Decision trees



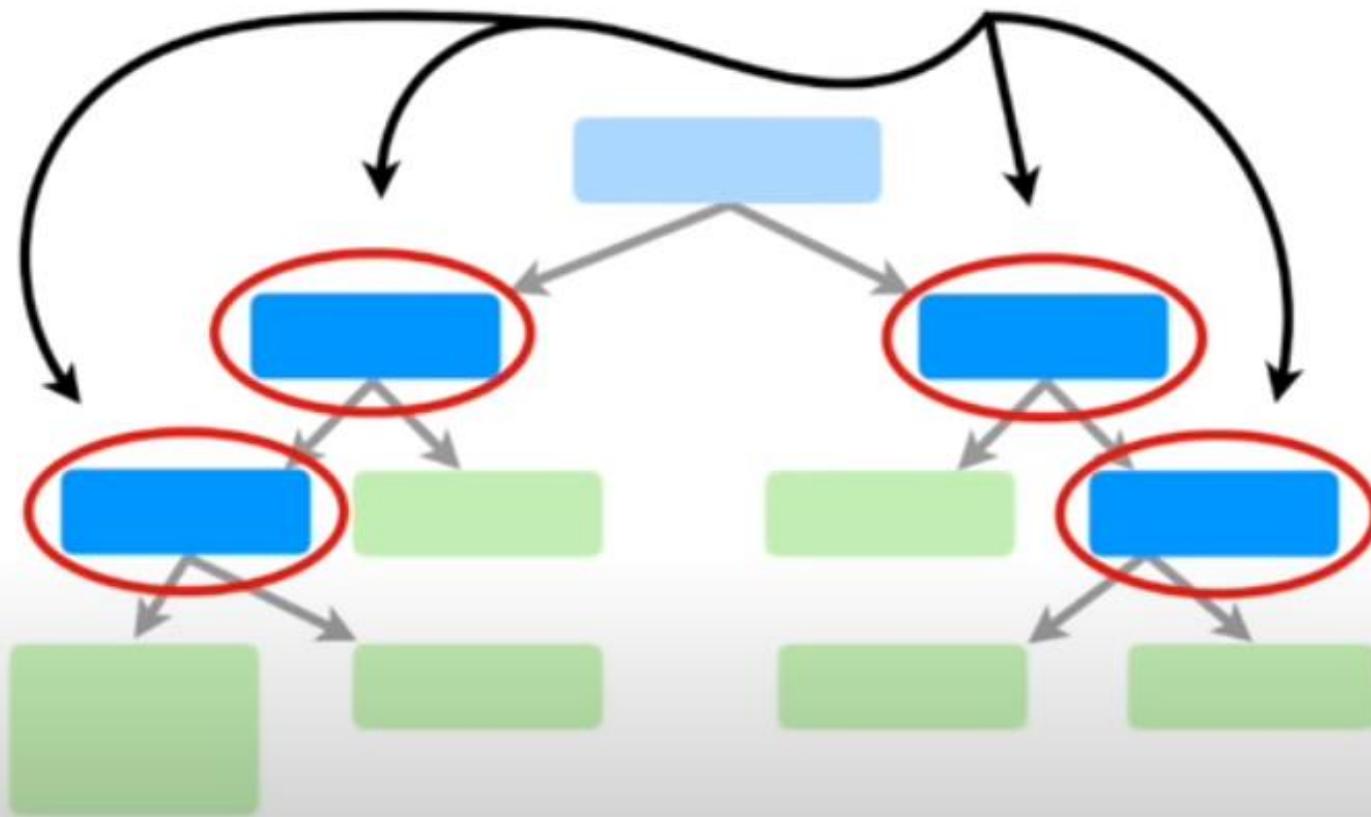
Decision trees: Root Node (The Root)

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

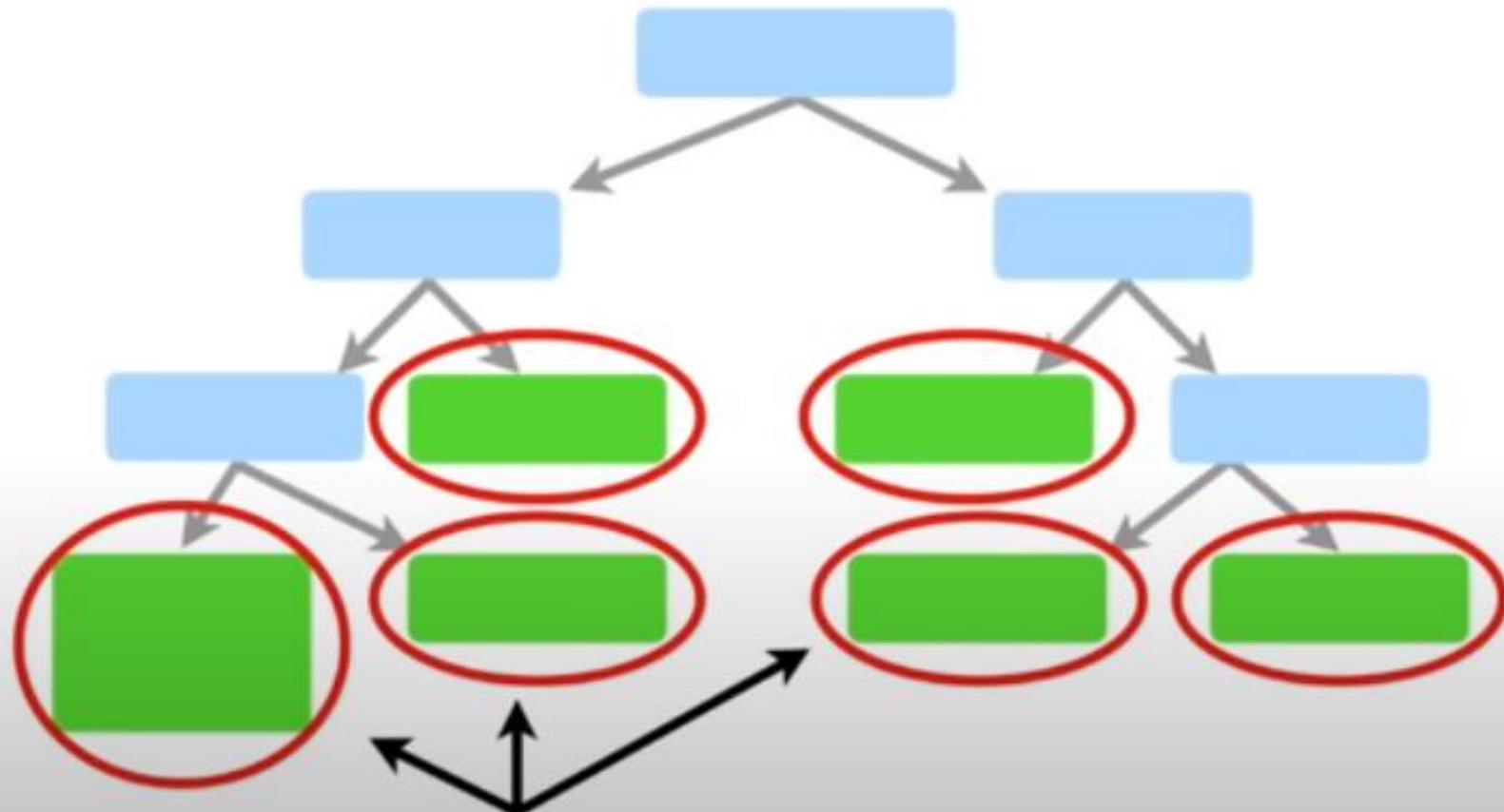


Decision trees: Internal Nodes (Nodes)

These are called “**Internal Nodes**”, or just “**Nodes**”.



Decision trees: Leaf Nodes (Leaves)



Lastly, these are called “**Leaf Nodes**”, or just “**Leaves**”

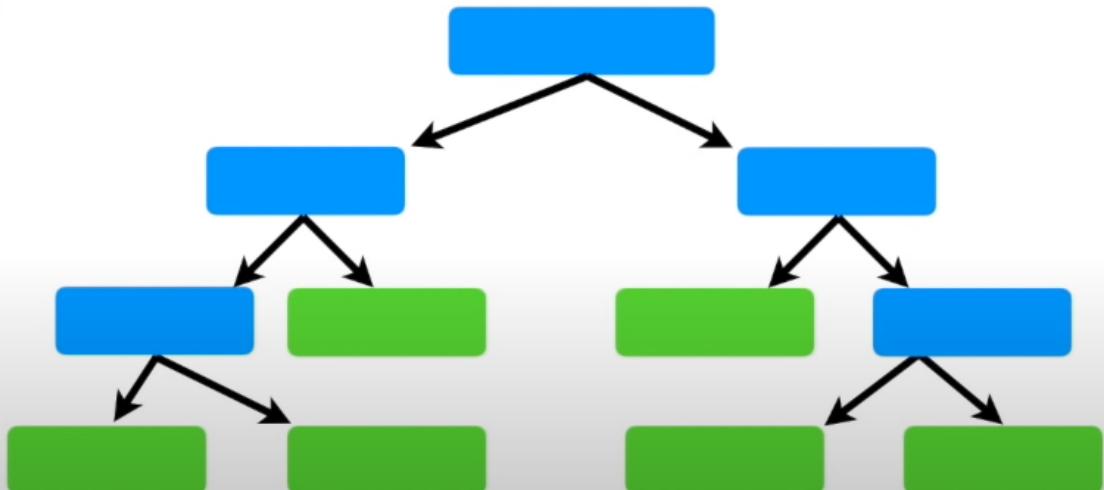
Example

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

...whether or not a patient has heart disease.

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.

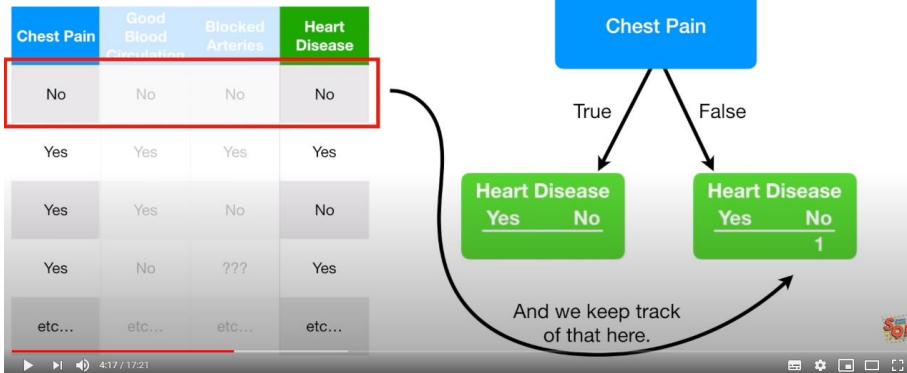
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc...	etc...	etc...	etc...



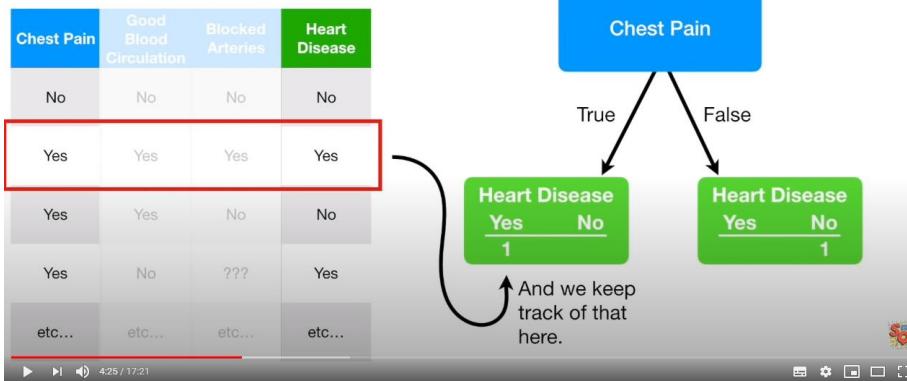
We start by looking at how well **Chest Pain** alone predicts heart disease...

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...

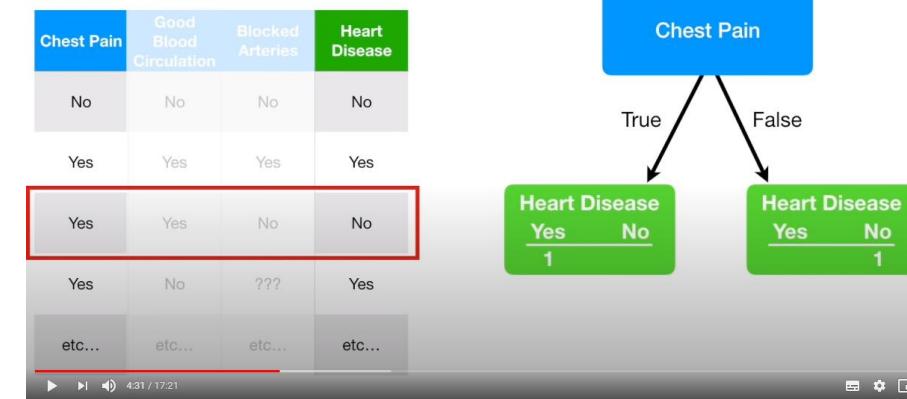
The first patient does not have chest pain and does not have heart disease.



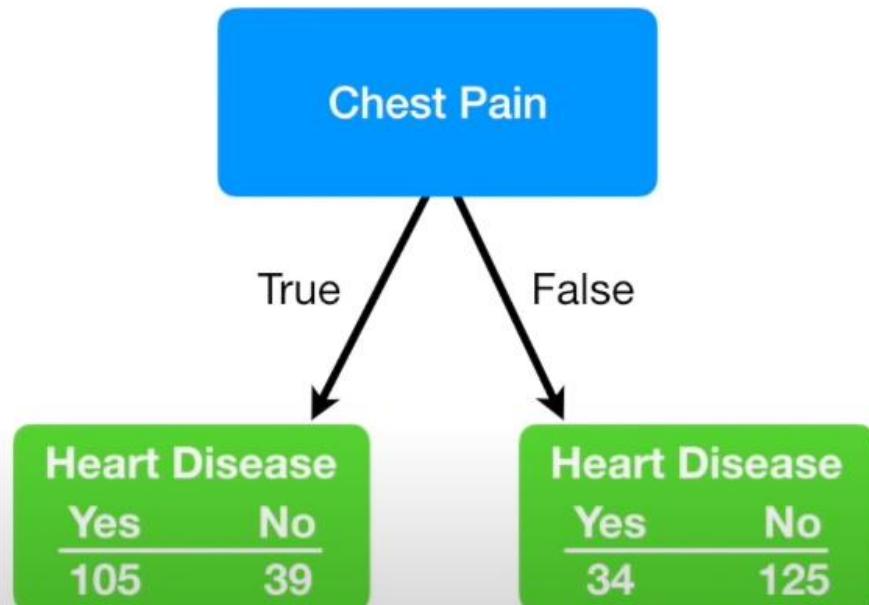
The 2nd patient has chest pain and heart disease.



The 3rd 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...



Ultimately, we look at chest pain and heart disease for all 303 patients in this study.

Now we do the exact same thing 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...

Good Blood Circulation

True

False

Heart Disease

Yes No

37 127

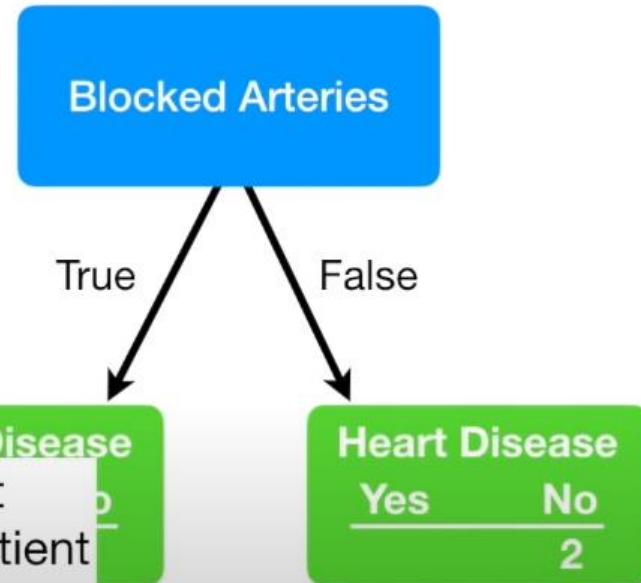
Heart Disease

Yes No

100 33

Decision trees: Missing data

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...

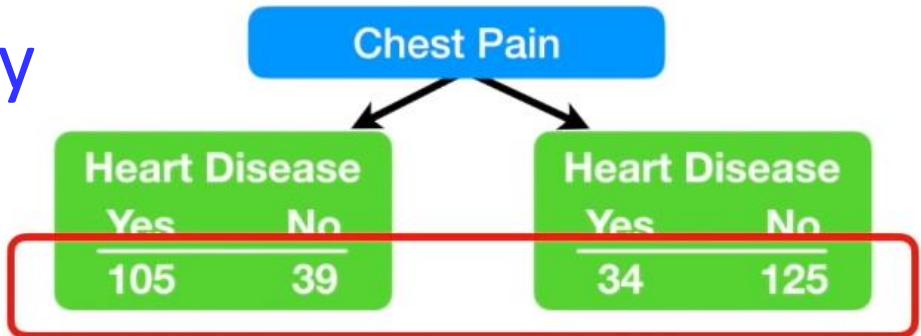


Decision trees: Missing data

However, there are alternatives that I'll discuss in a follow up video.

<https://www.youtube.com/watch?v=wpNI-JwwplA&t=183s>

Decision trees: Impurity



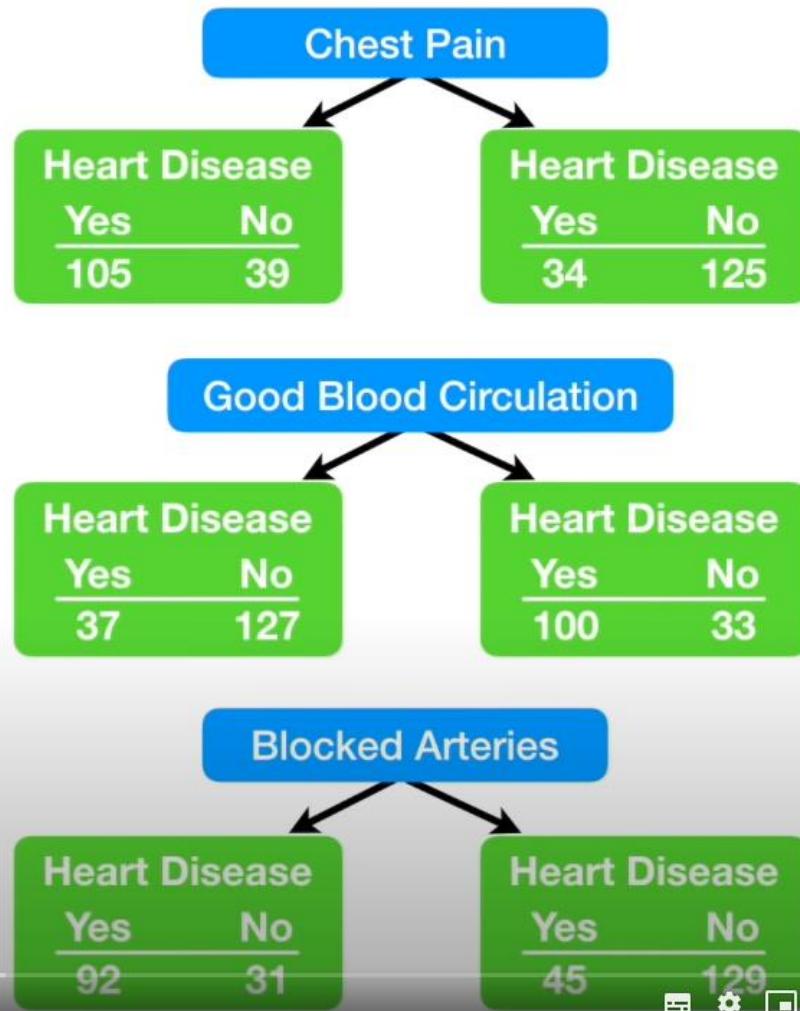
NOTE: The total number of patients with heart disease is different for Chest Pain, Good Blood Circulation and Blocked Arteries because some patients had measurements for Chest Pain, but not for Blocked Arteries, etc.

Because none of the leaf nodes are 100% “YES Heart Disease” or 100% “NO Heart Disease”, they are all considered “**impure**”.

Decision trees: Impurity

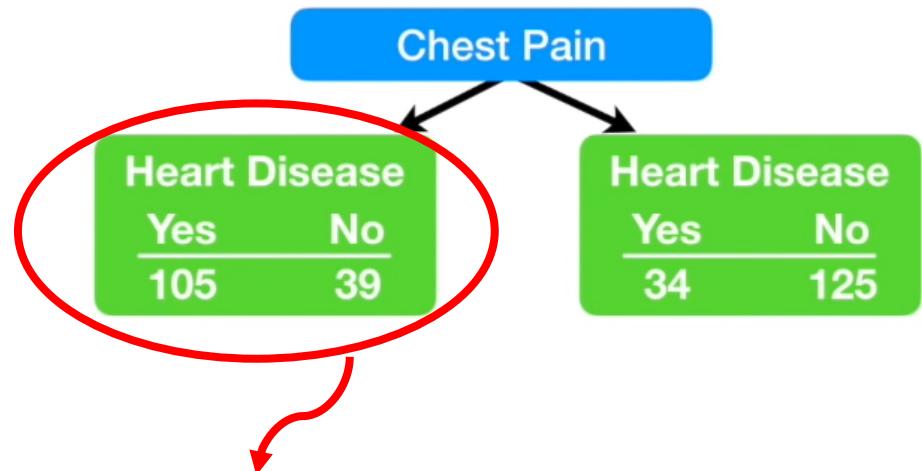
Because none of the leaf nodes are 100% “YES Heart Disease” or 100% “NO Heart Disease”, they are all considered “**impure**”.

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



Decision trees: Gini Impurity

Impurity=0 是最佳

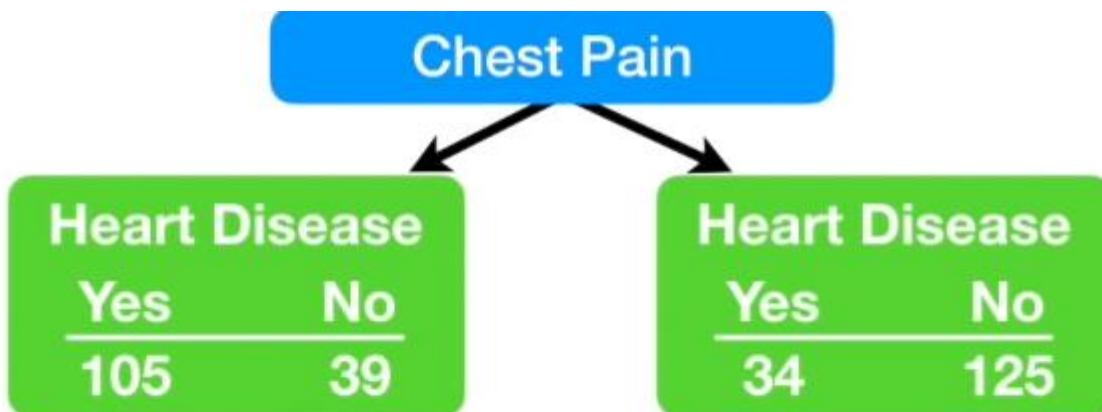


For this leaf, the Gini impurity = $1 - (\text{the probability of "yes"})^2 - (\text{the probability of "no"})^2$

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

$$= 0.395$$

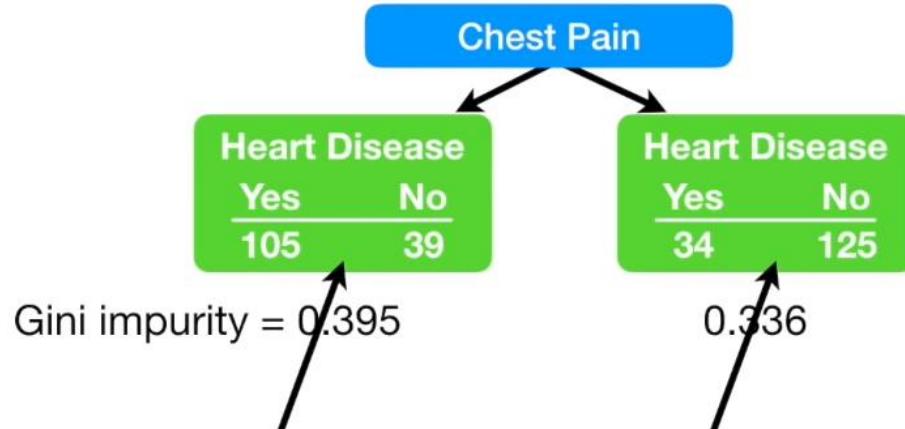




Gini impurity = 0.395

0.336

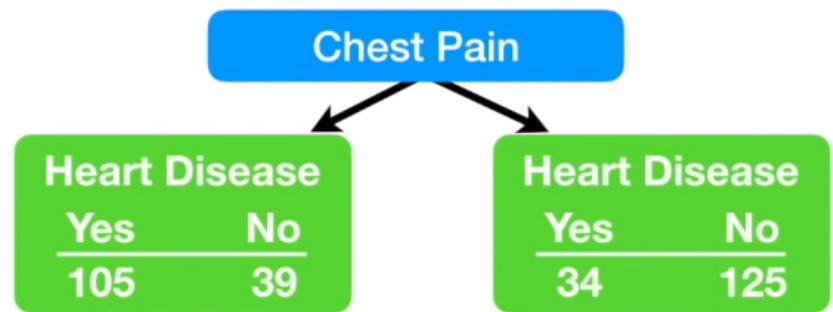
Now that we have measured the Gini impurity for both leaf nodes, we can calculate the total Gini impurity for using Chest Pain to separate patients with and without heart disease.



Because this leaf node ... and this leaf node
represents 144 patients... represents 159 patients...

...the leaf nodes do not
represent the same
number of patients.

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**.



Gini impurity = 0.395

0.336

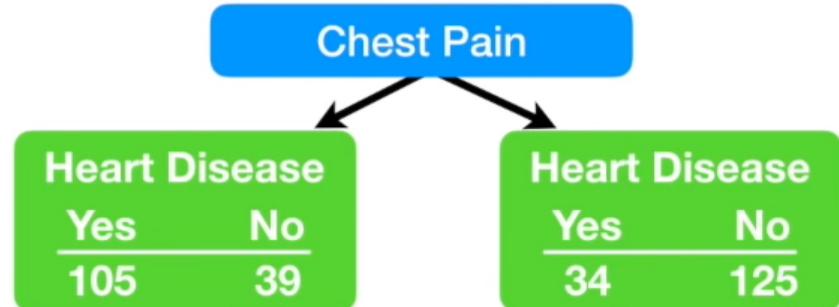
Gini impurity for Chest Pain = weighted average of Gini impurities for the leaf nodes

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

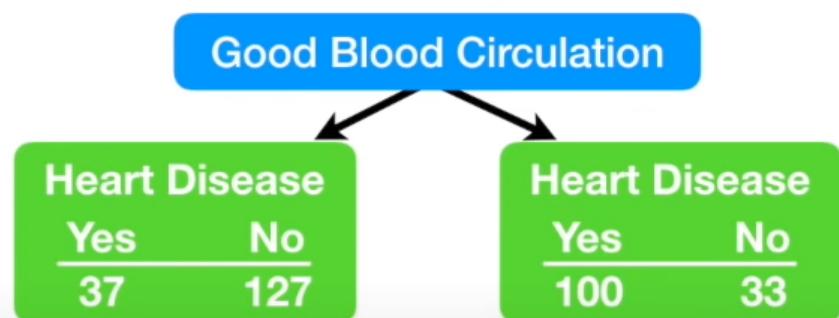
$$= 0.364$$



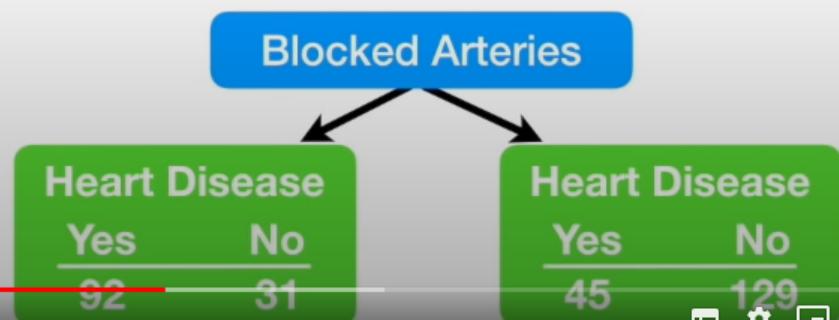
Gini impurity for Chest Pain = 0.364



Gini impurity for Good Blood Circulation = 0.360



Gini impurity for Blocked Arteries = 0.381

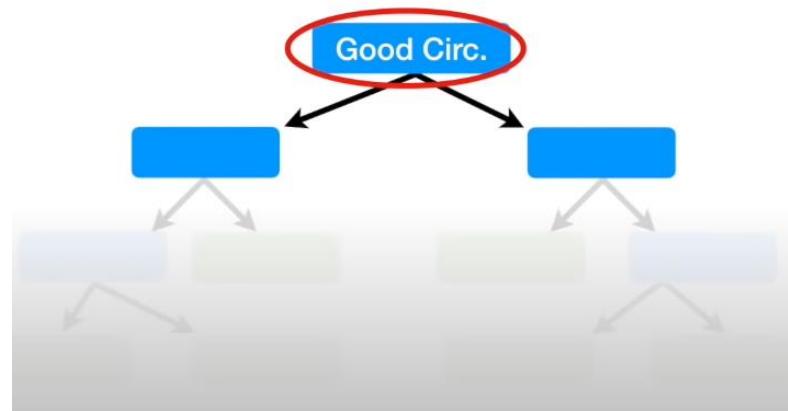
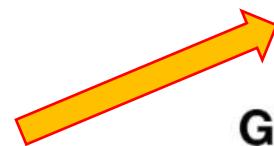


The lowest impurity separates the patients with/without HD the best

...so we will use it at the root of the tree.

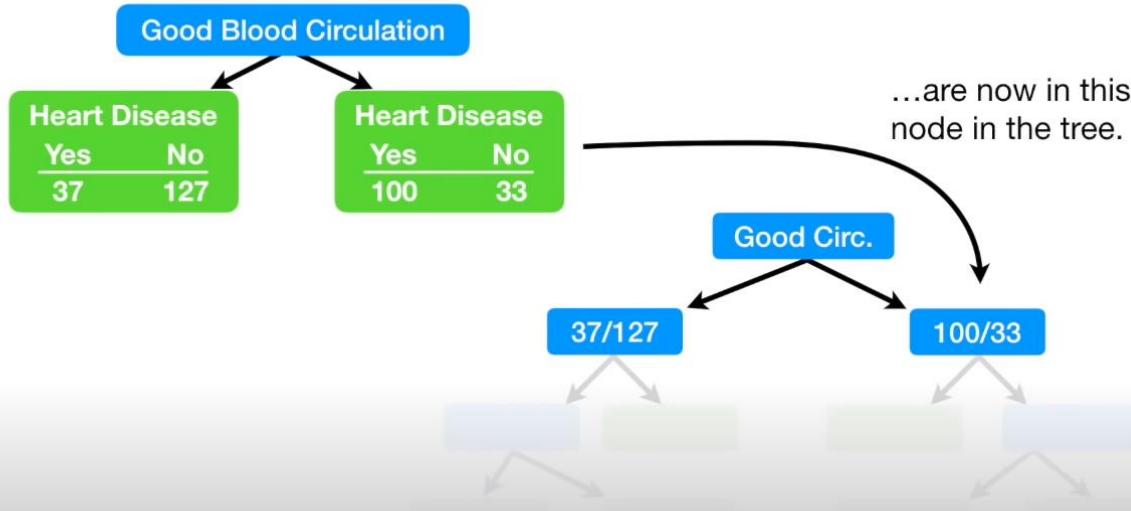
Gini impurity for Chest Pain = 0.364

Gini impurity for Good Blood Circulation = 0.360

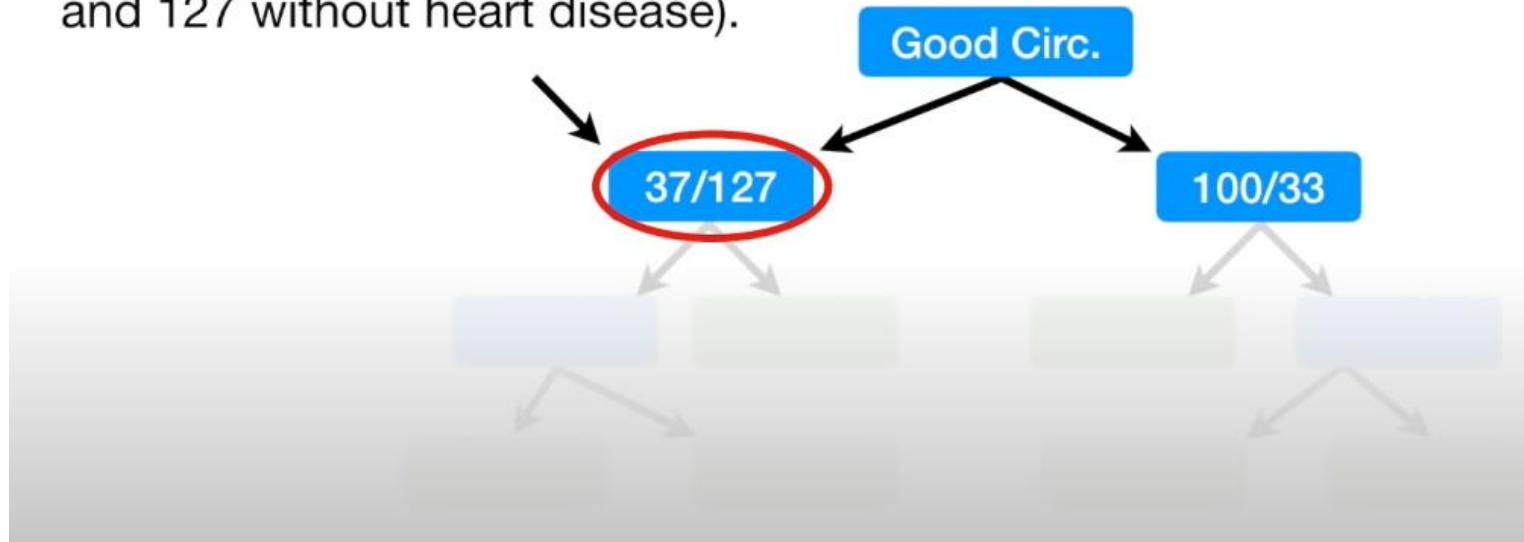


Good Blood Circulation has the lowest impurity (it separates patients with and without heart disease the best)...

Gini impurity for Blocked Arteries = 0.381

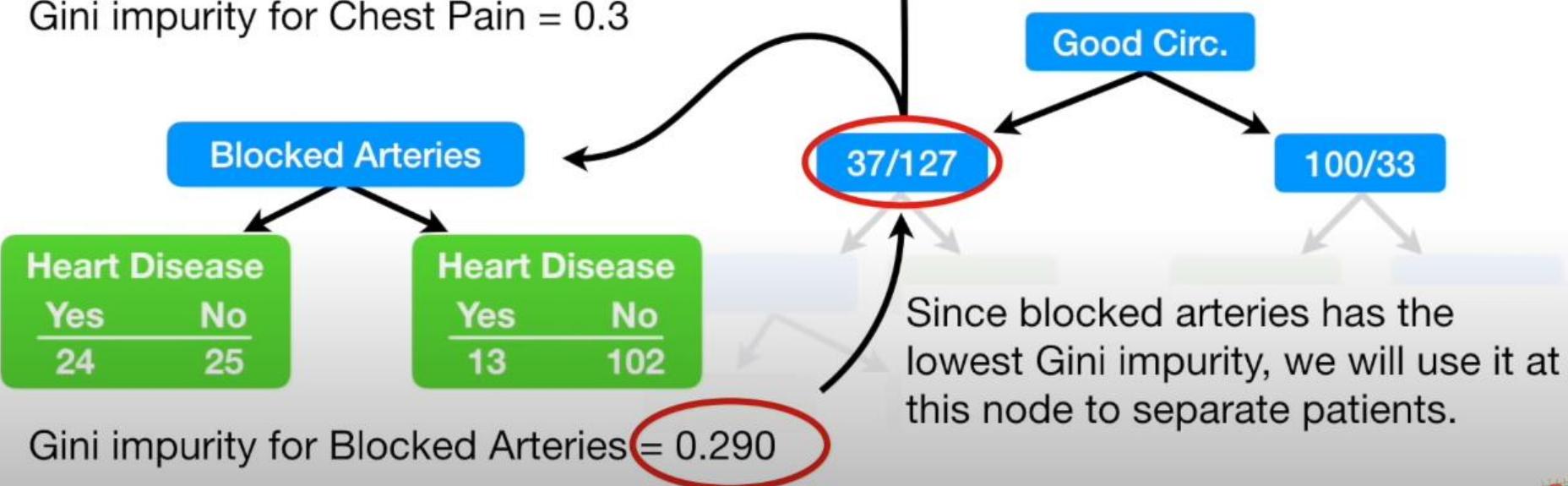


Now we need to figure how well **chest pain** and **blocked arteries** separate these 164 patients (37 with heart disease and 127 without heart disease).

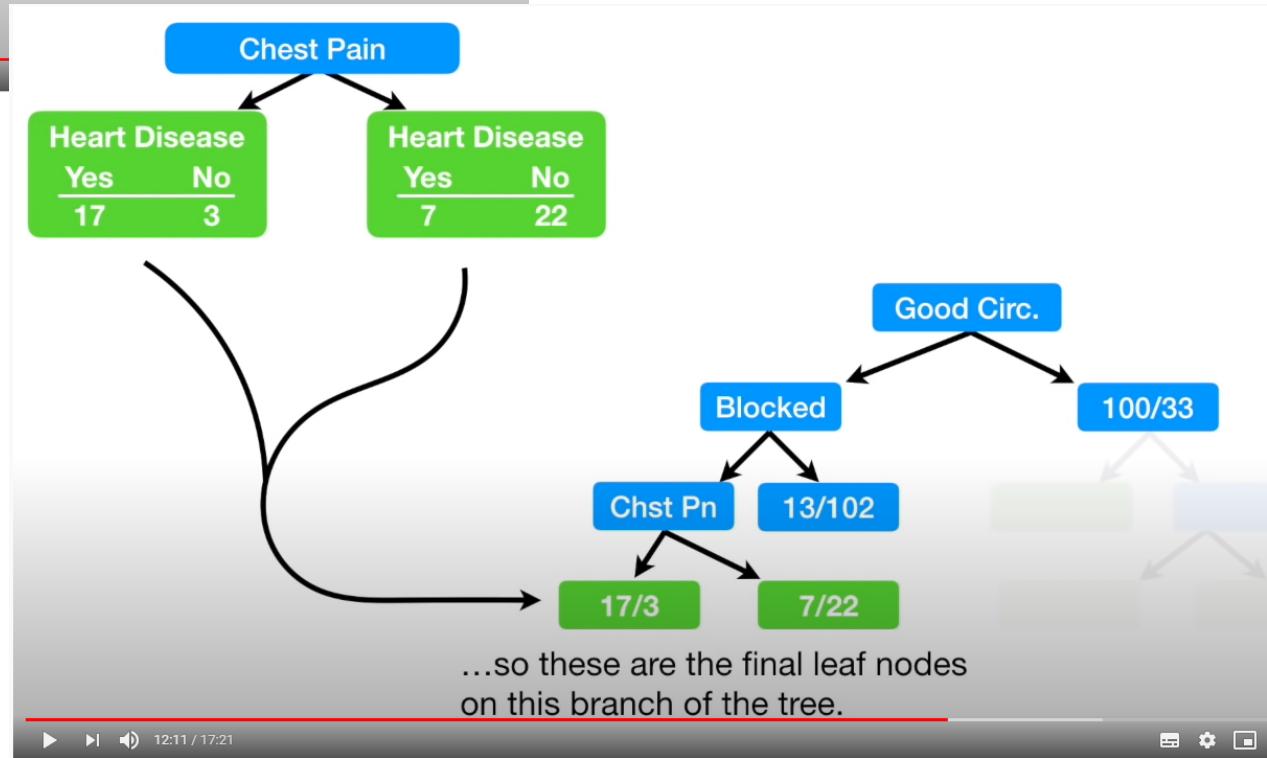
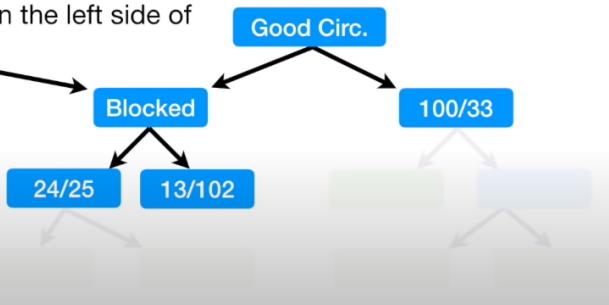




Gini impurity for Chest Pain = 0.3



...then we used Blocked Arteries to separate patients on the left side of the tree.



Chest Pain

Heart Disease	
Yes	No
7	26

Heart Disease	
Yes	No
6	76

Gini impurity for Chest Pain = 0.29

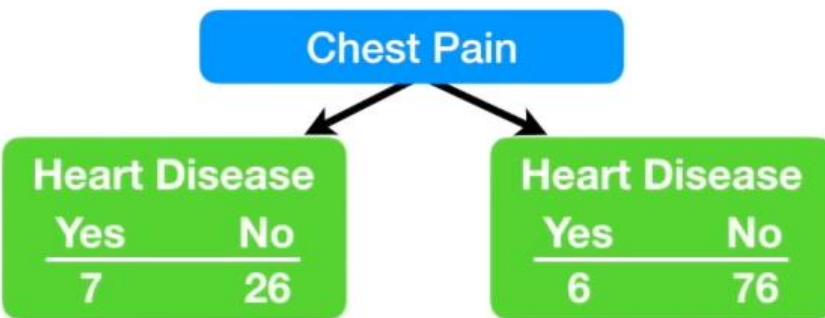
The Gini impurity for this node, before using chest pain to separate patients is...

$$= 1 - (\text{the probability of "yes"})^2 - (\text{the probability of "no"})^2$$

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

$$= 0.2$$





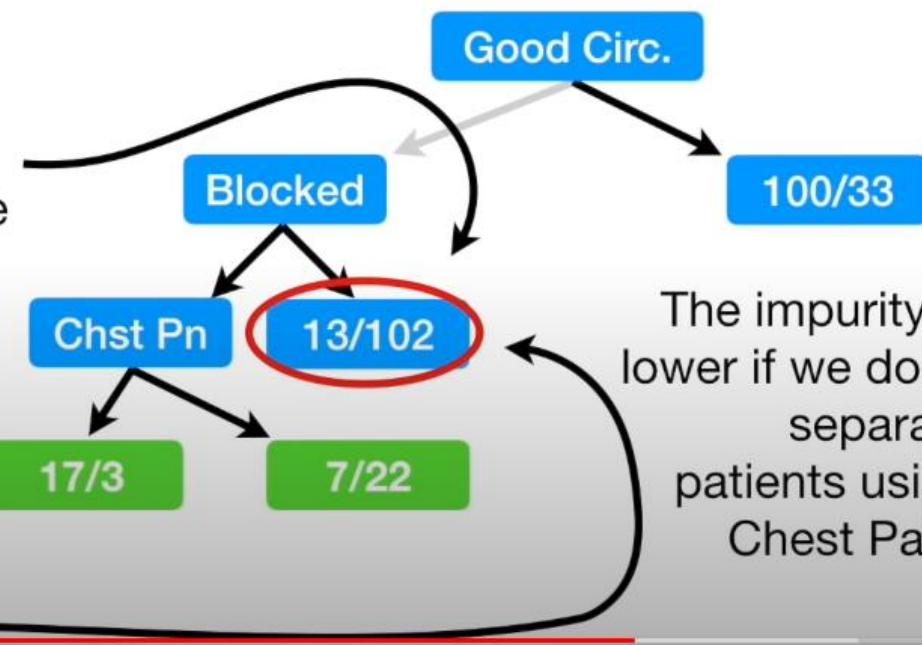
Gini impurity for Chest Pain = 0.29

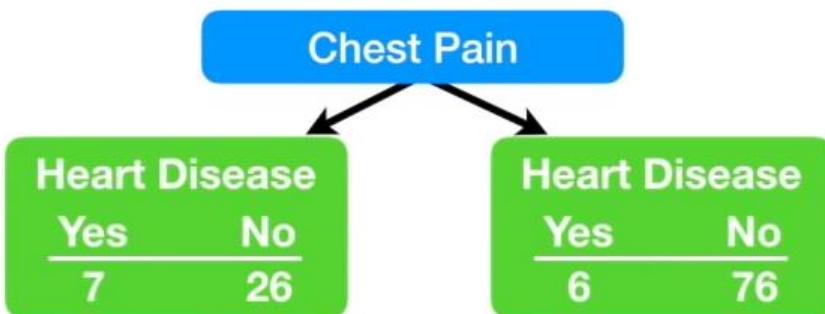
The Gini impurity for this node, before using chest pain to separate patients is...

$$= 1 - (\text{the probability of "yes"})^2 - (\text{the probability of "no"})^2$$

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

$$= 0.2$$





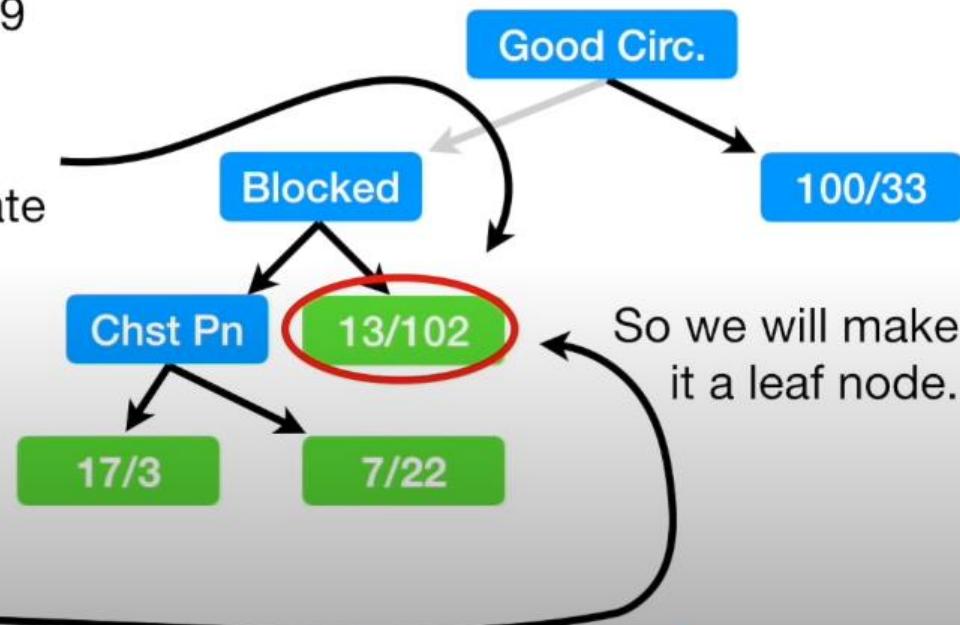
Gini impurity for Chest Pain = 0.29

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

$$= 1 - (\text{the probability of "yes"})^2 - (\text{the probability of "no"})^2$$

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

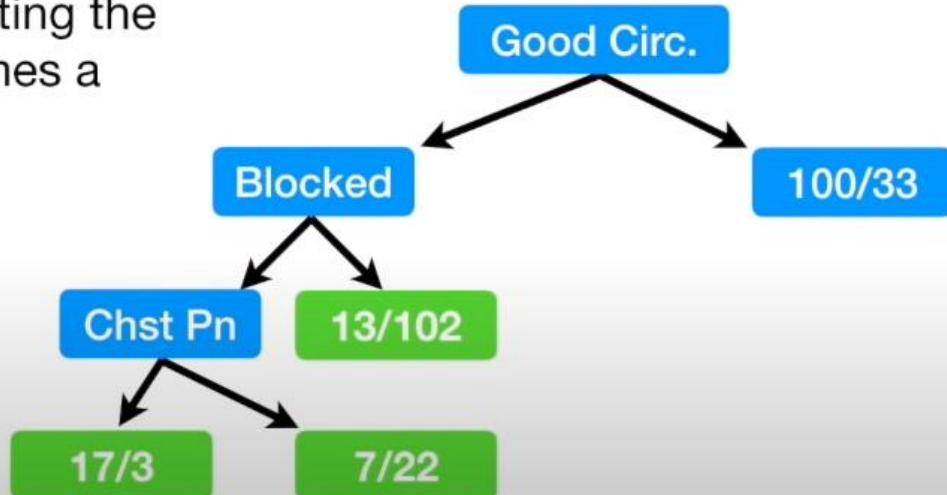
- 02



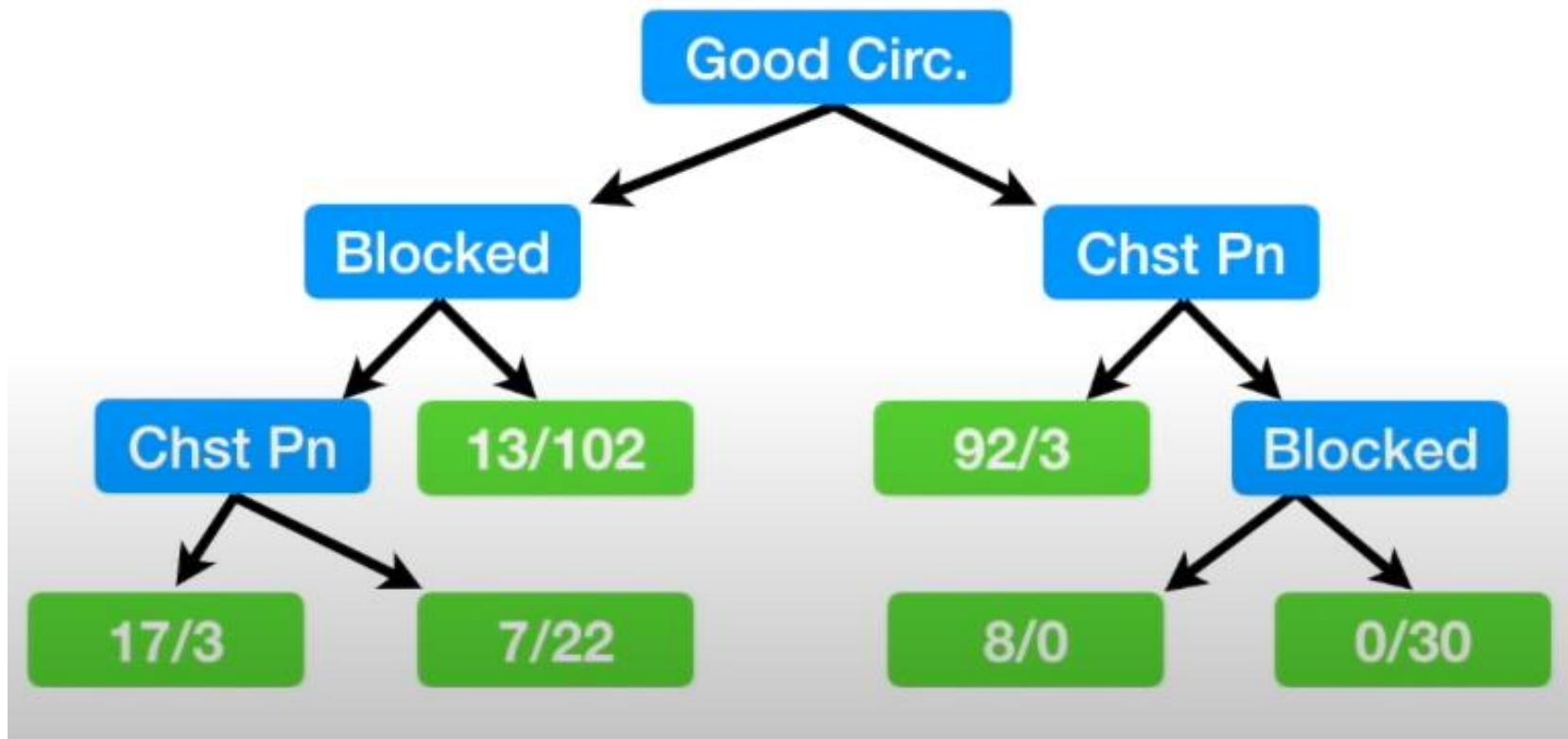
Right side in the example

The good news is that we follow the exact same steps as we did on the left side:

- 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.



Hooray!!! We made a decision tree!!!



So far we've seen how to build a tree with "yes/no" questions at each step...

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

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?

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

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

	Weight	Heart Disease
	155	No
	167.5	Yes
	185	No
	205	Yes
	222.5	Yes

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

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 = ?

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

No

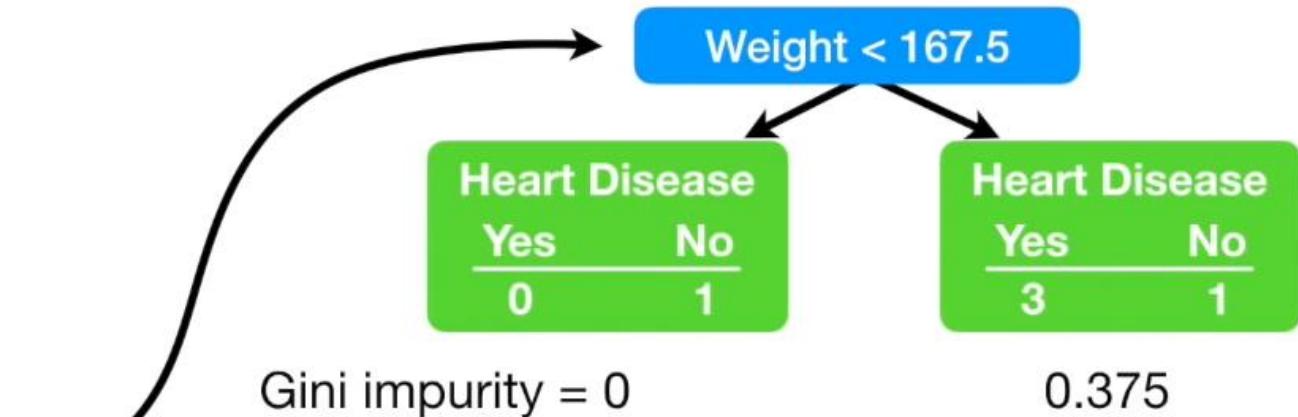
Yes

No

Yes

Yes

Yes



Gini impurity = 0

0.375

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

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

Weight	Heart Disease
155	No
167.5	 Gini impurity = 0.3
180	Yes
185	 Gini impurity = 0.47
190	No
205	 Gini impurity = 0.27
220	Yes
222.5	 Gini impurity = 0.4
225	Yes

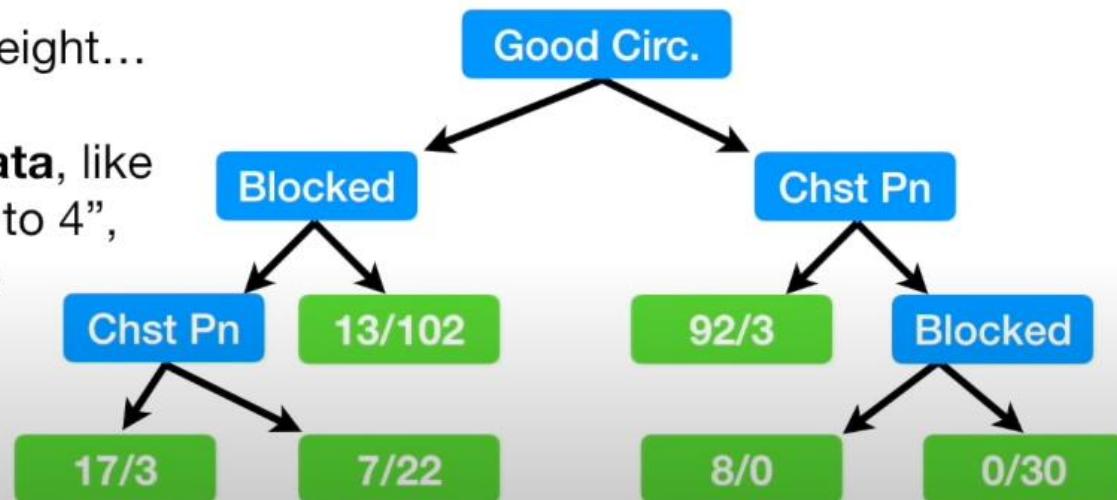
The lowest impurity occurs when we separate using **weight < 205**...

...so this is the cutoff and impurity value we will use when we compare weight to chest pain or blocked arteries.

Now we've seen how to build a tree
with...

- 1) "yes/no" questions at each step...
- 2) Numeric data, like patient weight...

Now let's talk about **ranked data**, like
“rank my jokes on a scale of 1 to 4”,
and **multiple choice data**, like
“which color do you like, red,
blue or green?”



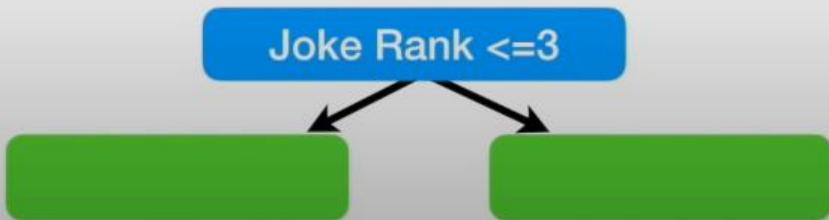
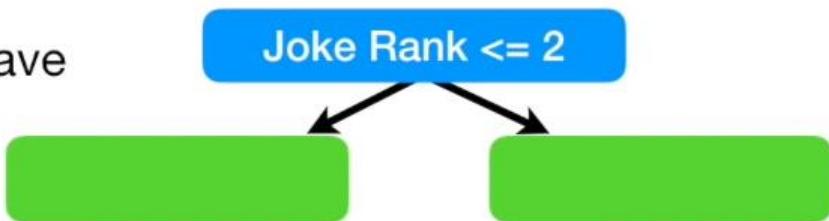
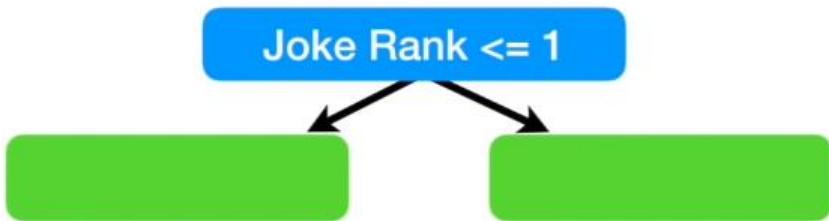
Rank my jokes...	Likes StatQuest
1	Yes
1	No
3	Yes
1	Yes
etc...	etc...

Ranked data is similar to numeric data, except instead now we calculate impurity scores for all of the possible ranks.

So if people could rank my jokes from 1 to 4 (4 being the funniest), we could calculate the following impurity scores...

Rank my jokes...	Likes StatQuest
1	Yes
1	No
3	Yes
1	Yes
etc...	etc...

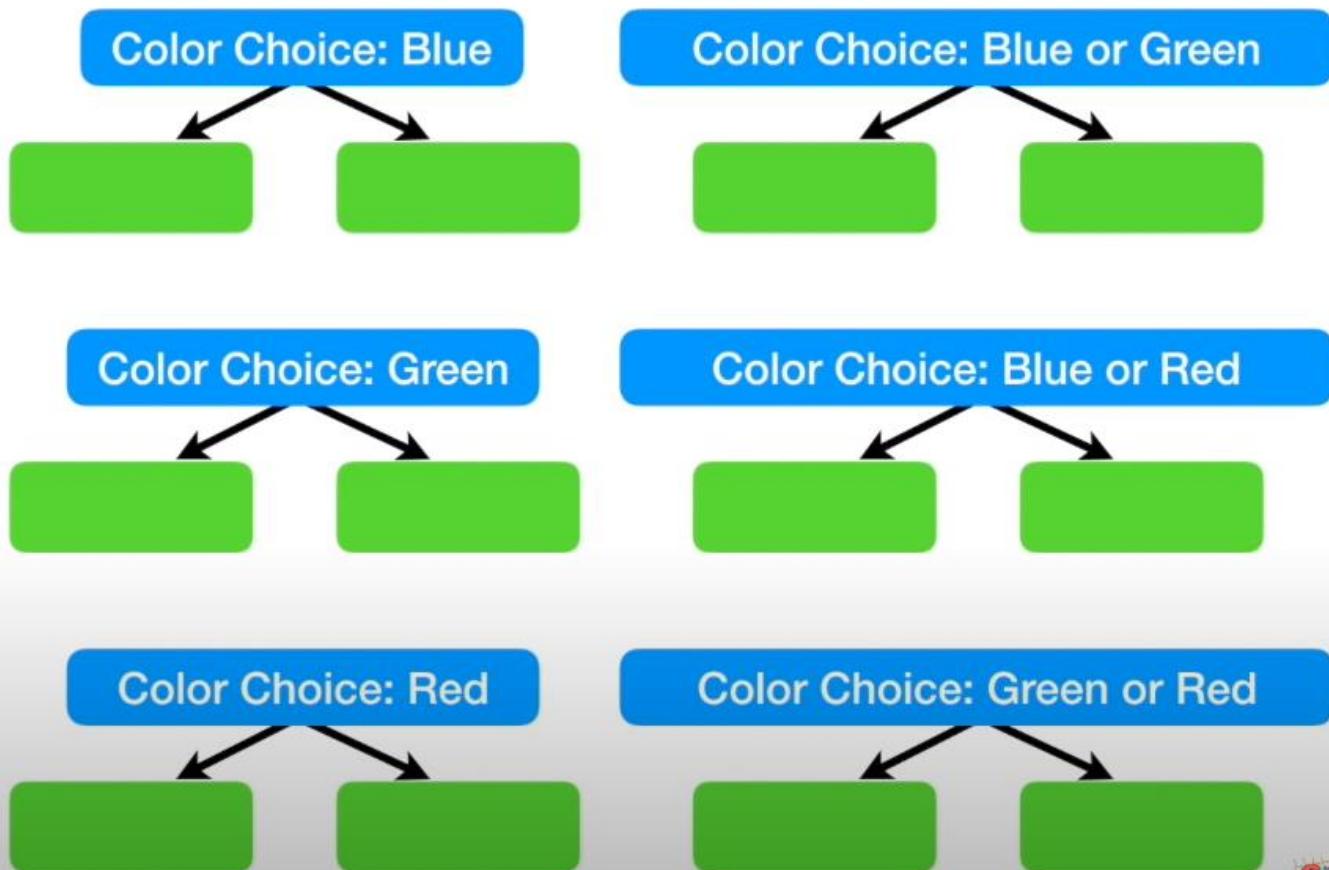
NOTE: We don't have to calculate an impurity score for Joke Rank ≤ 4 because that would include everyone.



Color Choice	Likes StatQuest
Green	Yes
Blue	No
Red	Yes
Green	Yes
etc...	etc...

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.

Color Choice	Likes StatQuest
Green	Yes
Blue	No
Red	Yes
Green	Yes
etc...	etc...



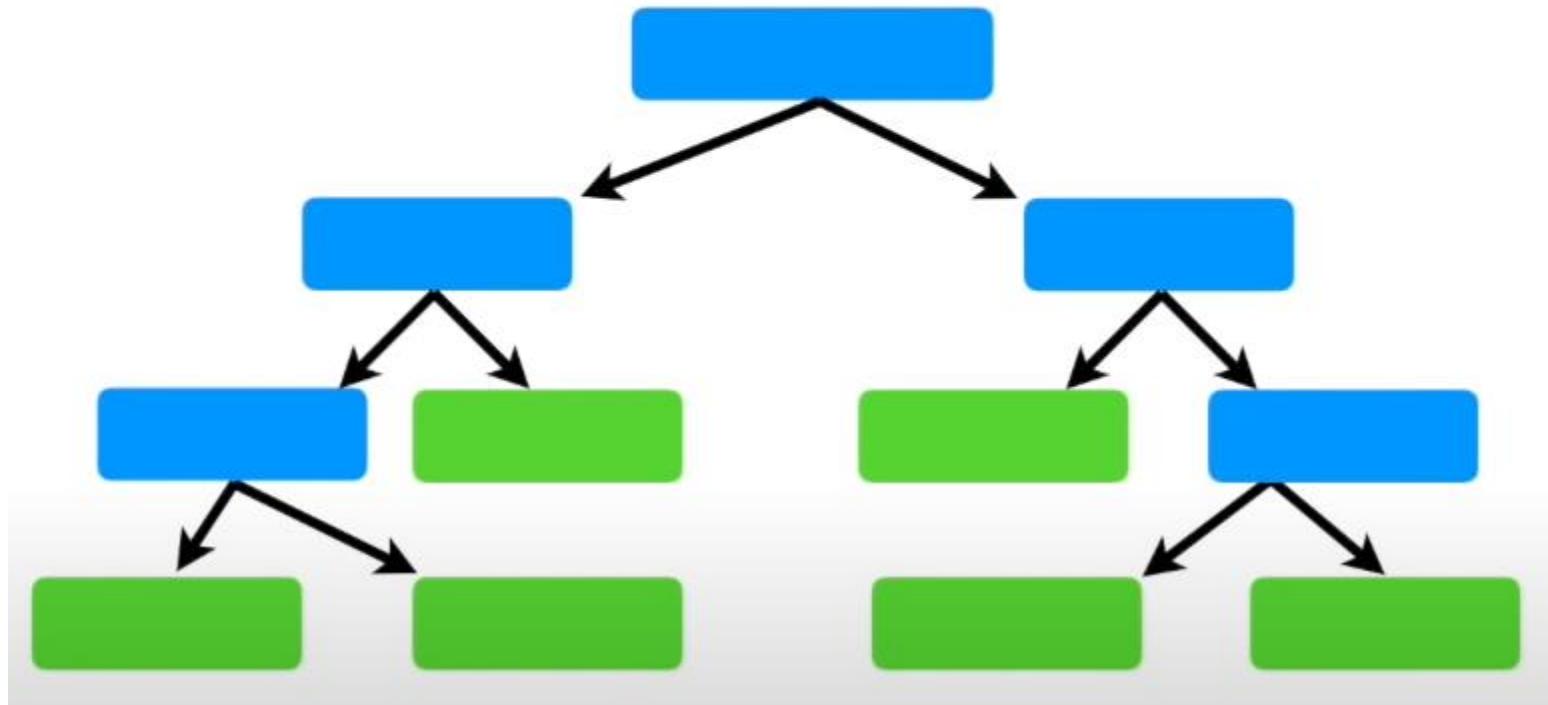
NOTE: We don't have to calculate an impurity score for "Color Choice: Blue or Green or Red" since that includes everyone.



Random Forests

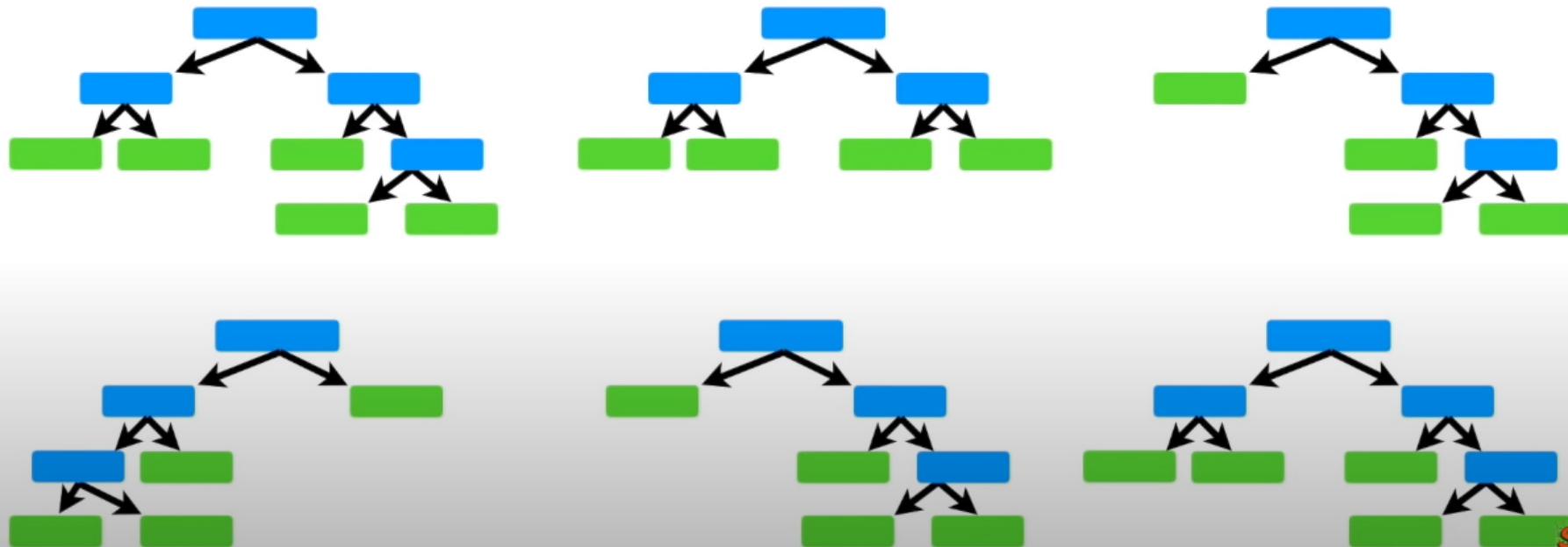
Decision trees 缺點:

In other words, they work great with the data used to create them, but **they are not flexible when it comes to classifying new samples.**





The good news is that **Random Forests** combine the simplicity of decision trees with flexibility resulting in a vast improvement in accuracy.



Random Forests steps:

- 1. Build a Random Forest (**training data**)
- 2. Use a Random Forest (**new data**)
- 3. Estimate the accuracy of a Random Forest (**test data**)

Random Forests

(1. Build a Random Forest)

Step 1: Create a “bootstrapped” dataset.

Original Dataset

Chest Pain	Good Blood Circ.	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 Circ.	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.

The important detail is that we're allowed to pick the same sample more than once.

Random Forests

(1. Build a Random Forest)

Step 1: Create a “bootstrapped” dataset.

Randomly selected samples from the Original Dataset

Original Dataset

Chest Pain	Good Blood Circ.	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 Circ.	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

Step 2: Create a decision tree using the bootstrapped dataset, but only use a random subset of variables (or columns) at each step.

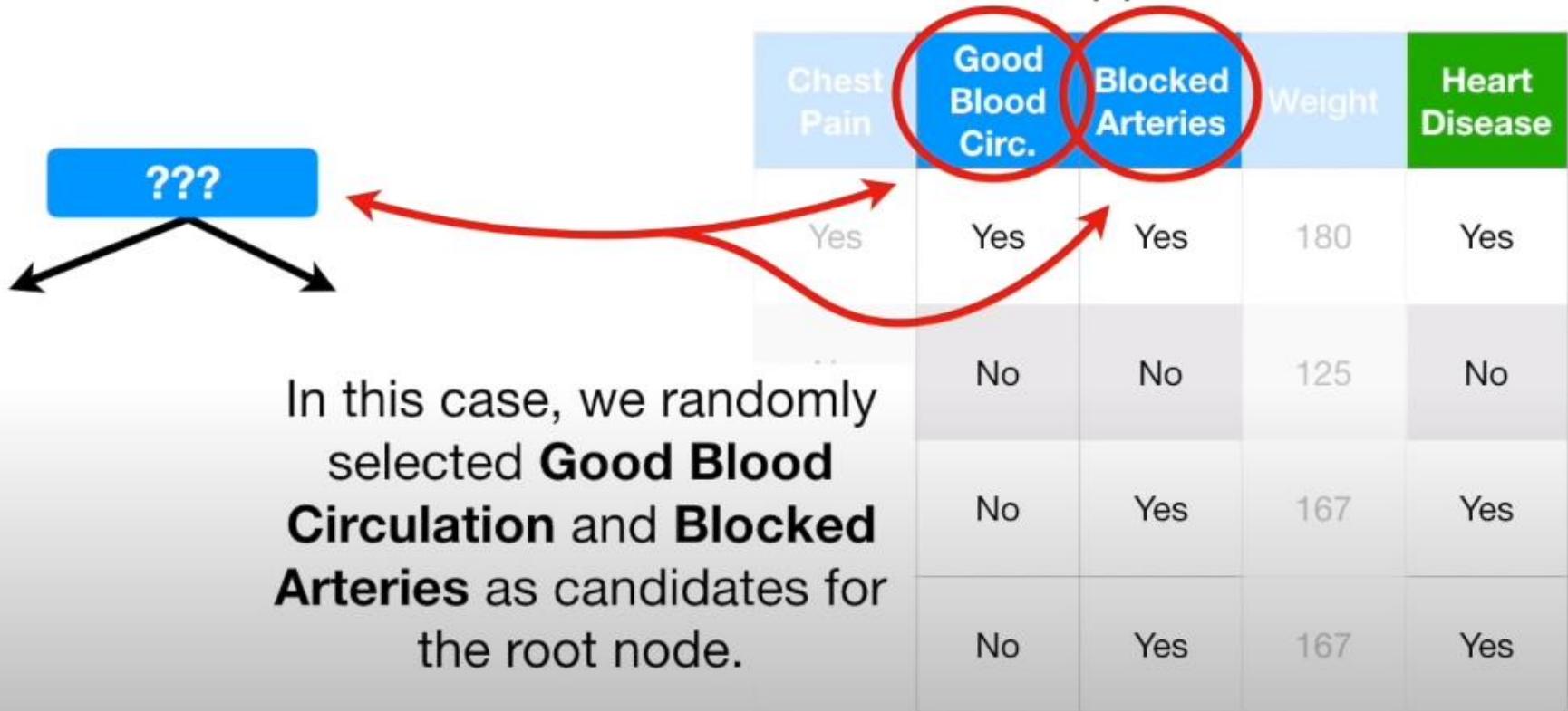
In this example, we will only consider 2 variables (columns) at each step.

NOTE: We'll talk more about how to determine the optimal number of variables to consider later...

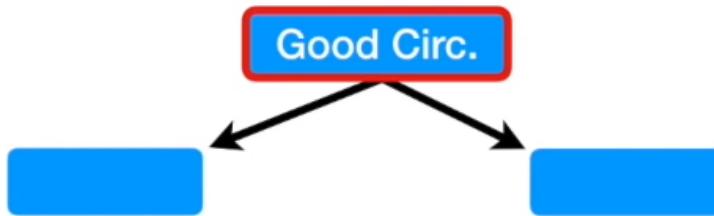
Bootstrapped Dataset

Chest Pain	Good Blood Circ.	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

Bootstrapped Dataset



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

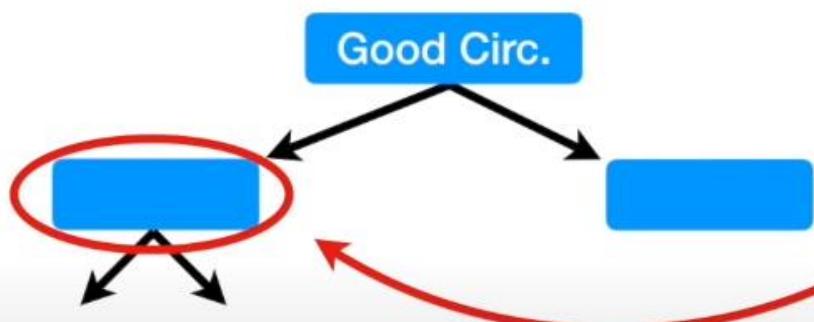


Bootstrapped Dataset

Chest Pain	Good Blood Circ.	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

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	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

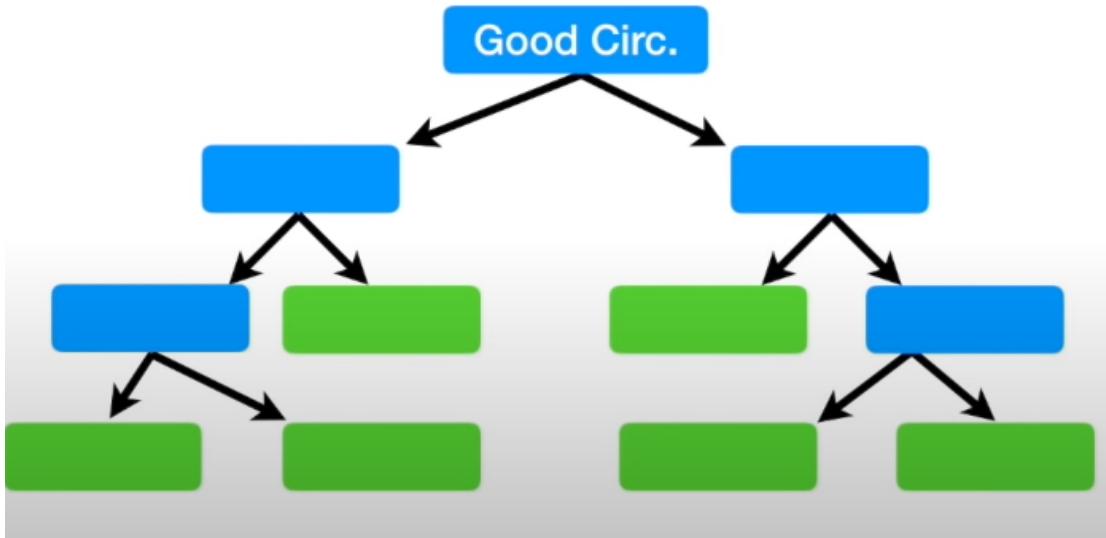


Just like for the root, we randomly select 2 variables as candidates, instead of all 3 remaining columns.

Bootstrapped Dataset

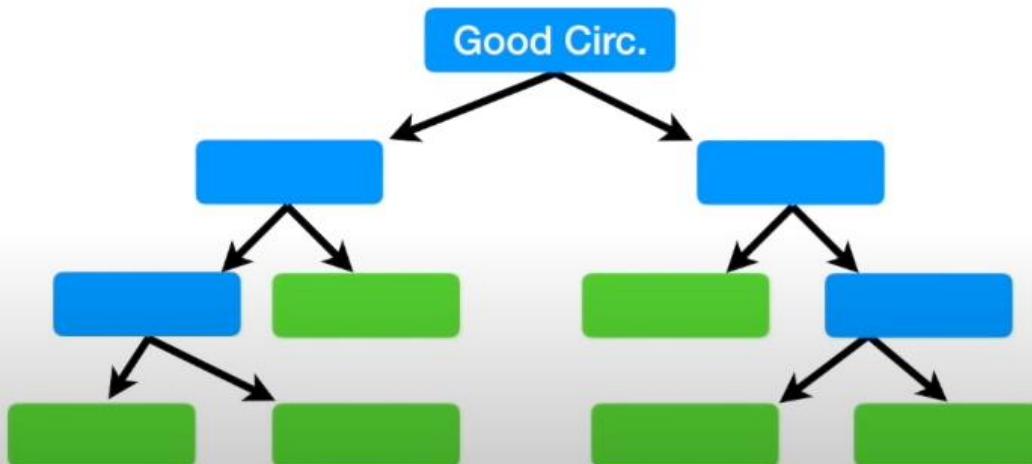
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
Yes	No	Yes	167	Yes

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



We built a tree...

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

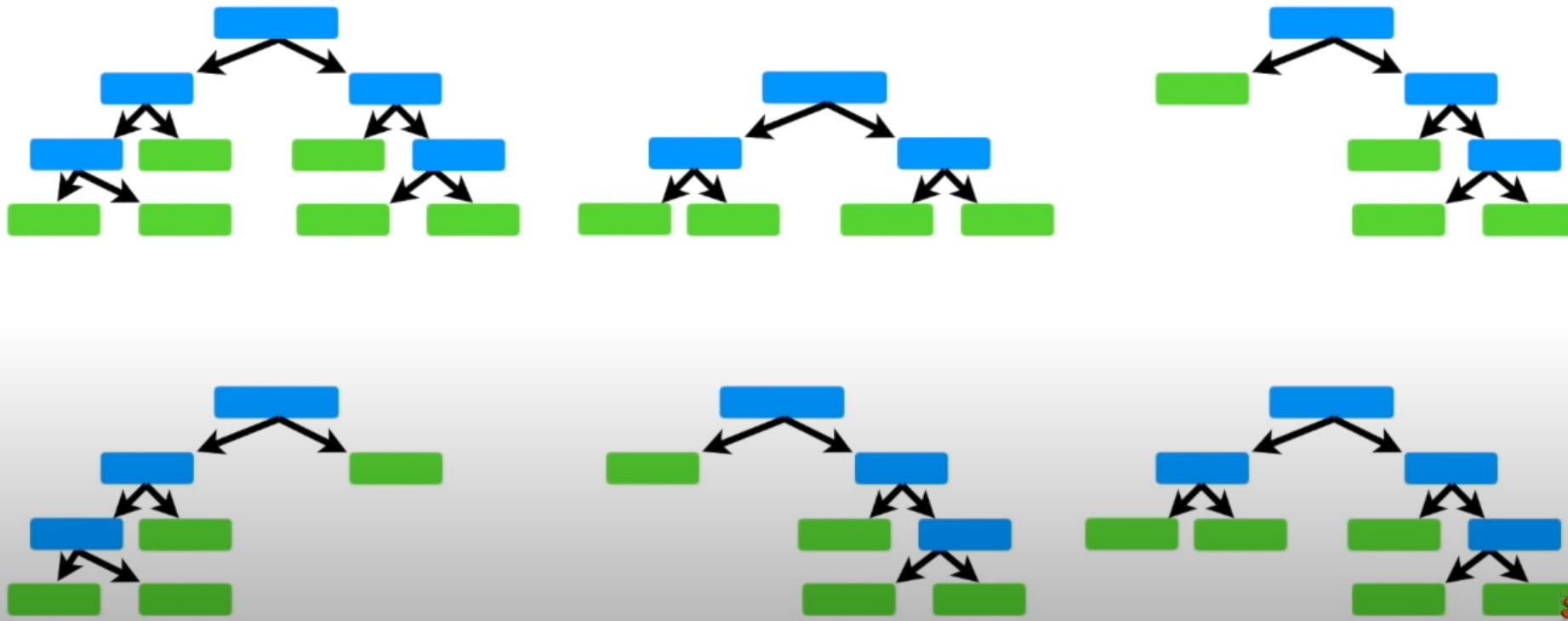


Bootstrapped Dataset

Chest Pain	Good Blood Circ.	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



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.



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

Ideally, you'd do this 100's of times.

The variety is what makes random forests more effective than individual decision trees.

Random Forests

(2. Use a Random Forest)

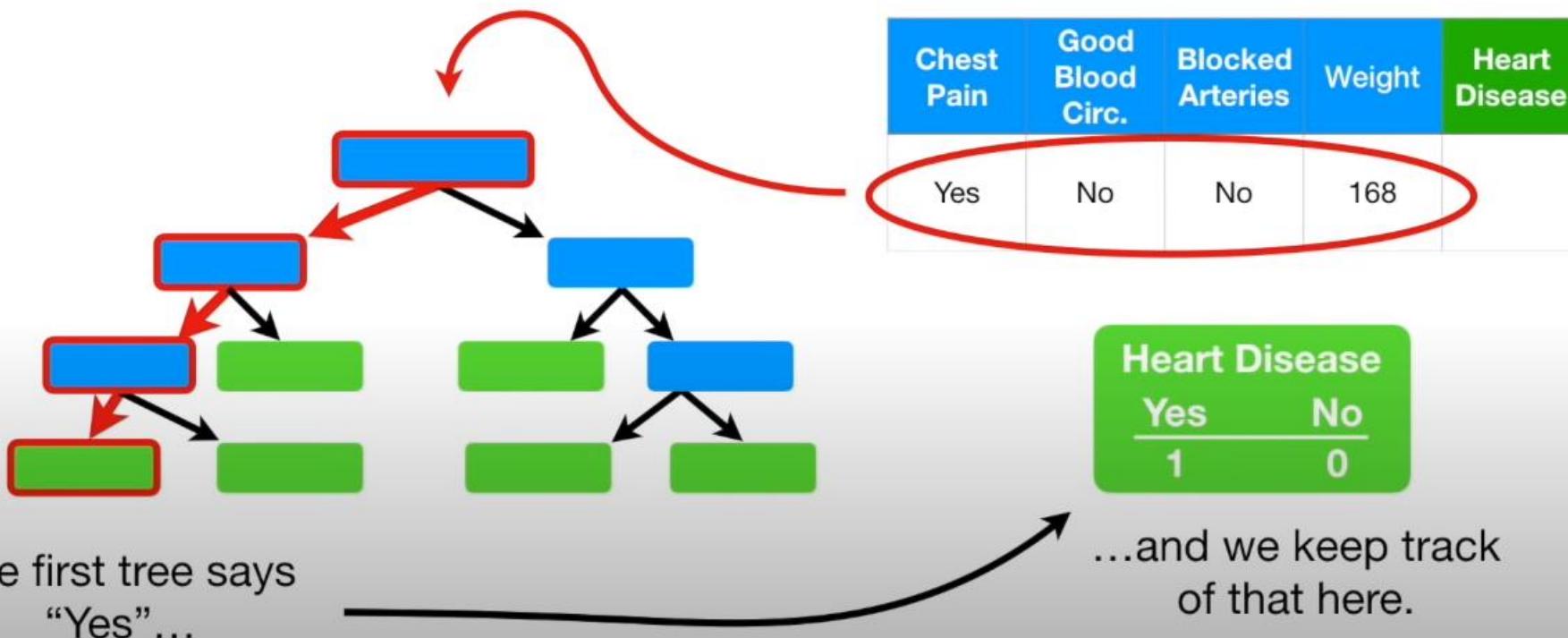
How to use a Random Forest?

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	

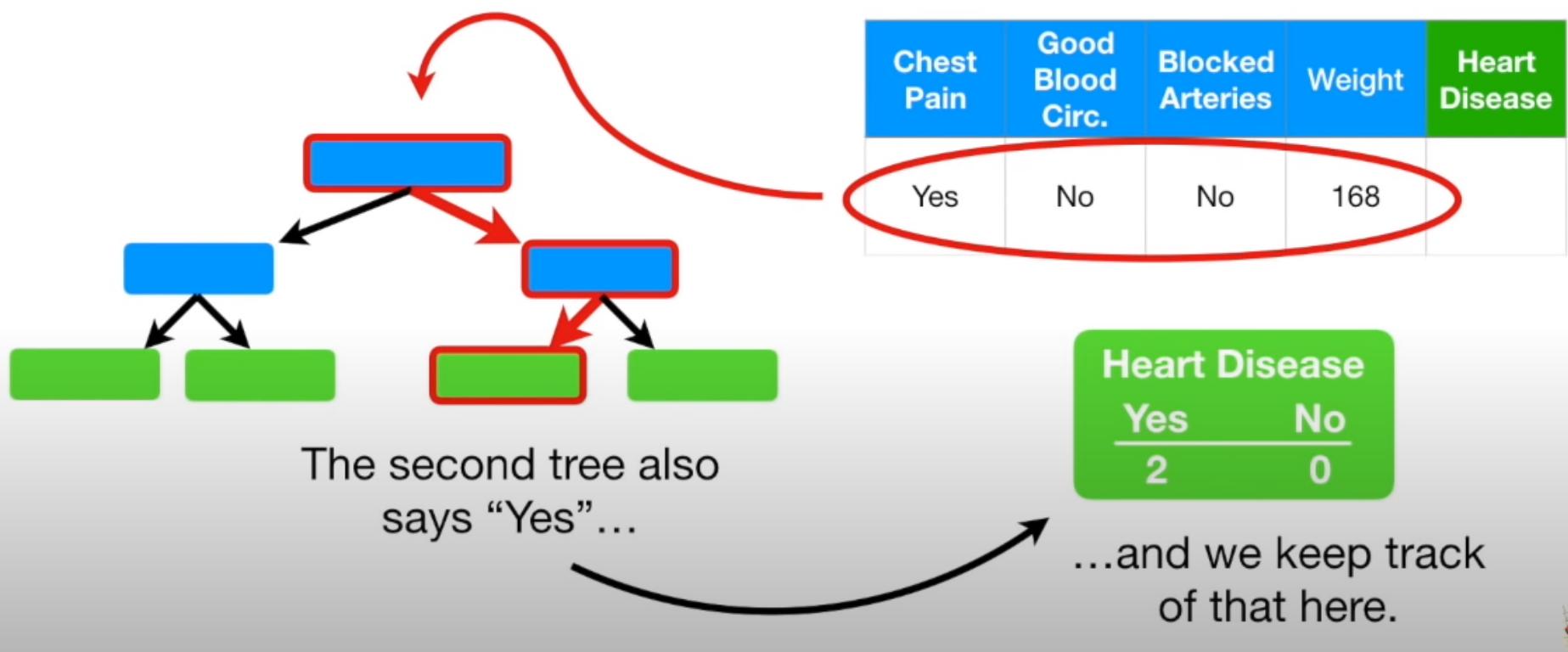
...we've got all the measurements...

...and now we want to know if they have heart disease or not.

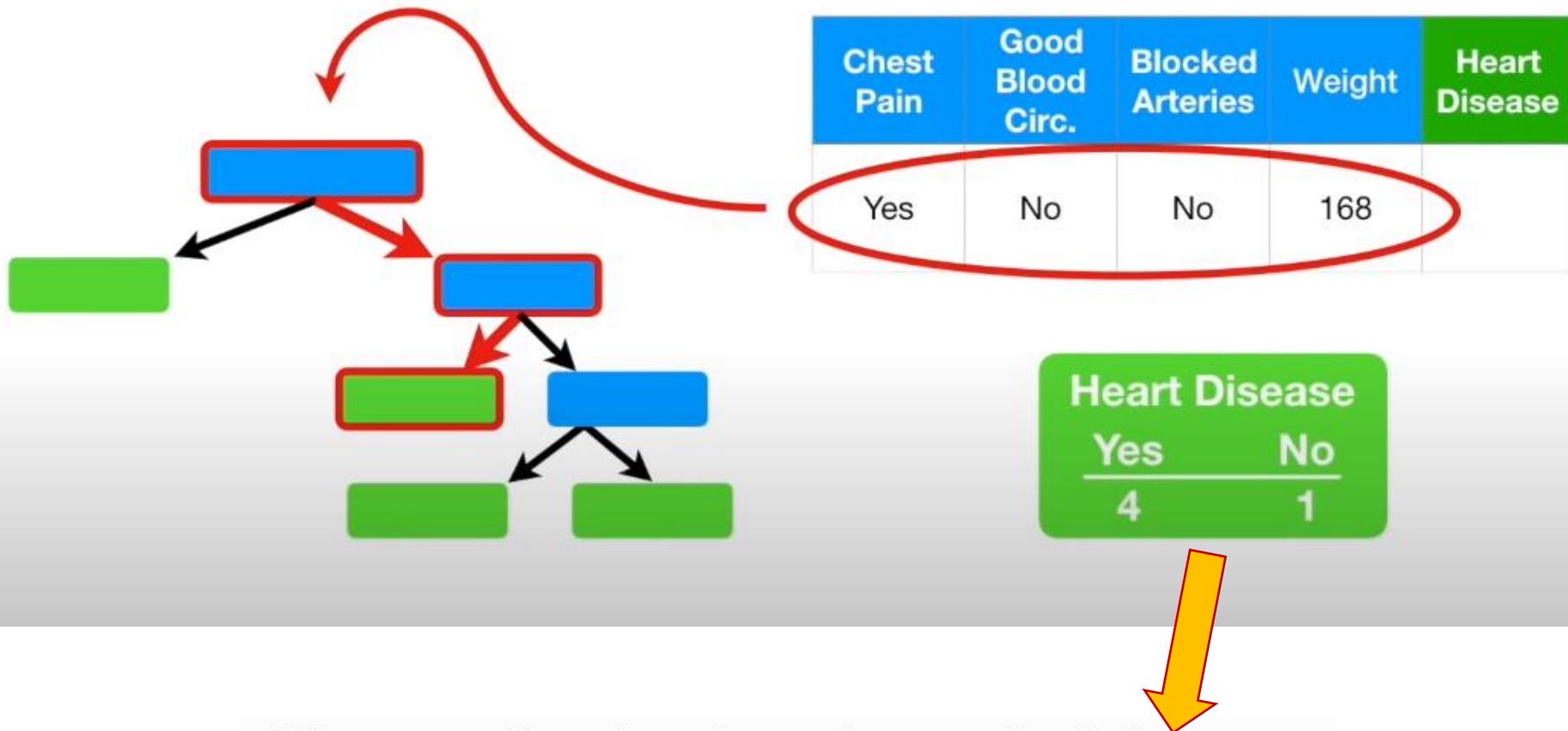
So we take the data
and run it down the
first tree that we
made...



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



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



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

How to use a Random Forest?

→ To predict the patient has heart disease

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	YES

In this case, “**Yes**” received the most votes, so we will conclude that this patient has heart disease.



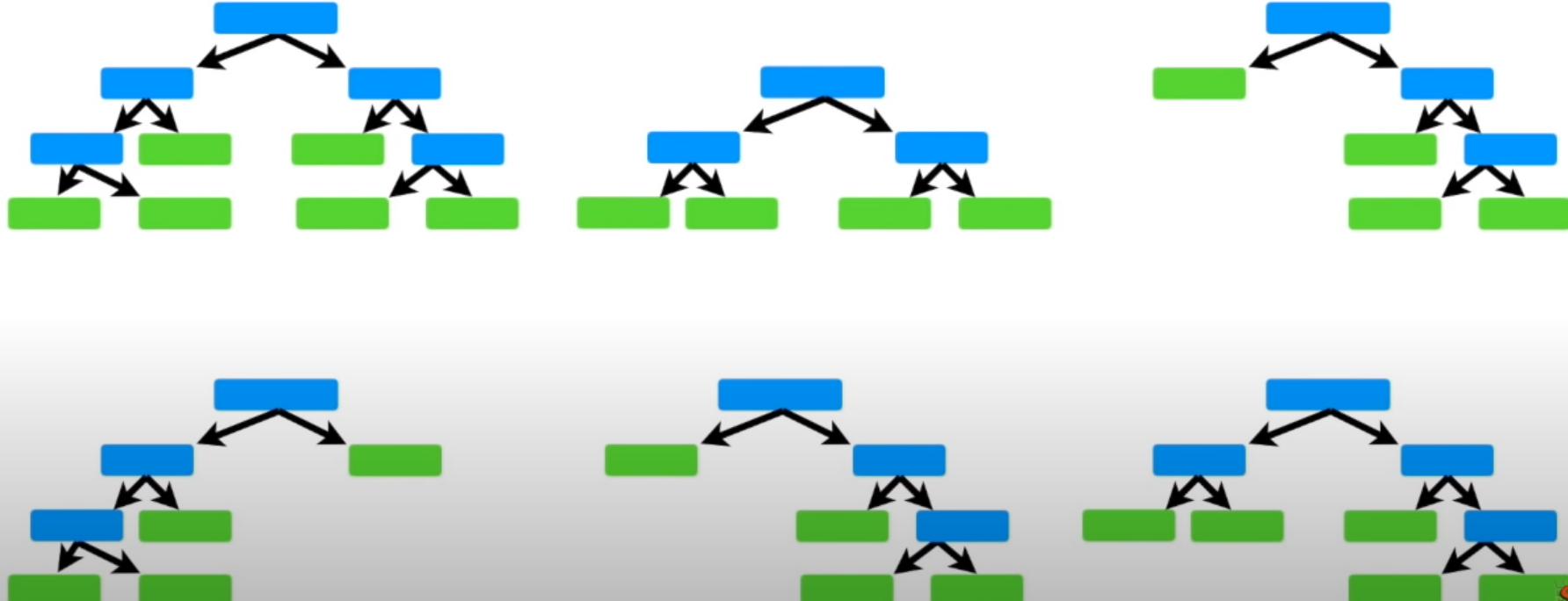
Bootstrapping the data plus using the aggregate to make a decision is called
“**Bagging**”

Random Forests

(3. Estimate the accuracy of a Random Forest)

OK, now we've seen how to create and use a random forest...

How do we know if it's any good?



We allowed duplicate entries in
the bootstrapped dataset...

Original Dataset

Chest Pain	Good Blood Circ.	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 Circ.	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

So!

Typically, about 1/3 of the original data does not end up in the bootstrapped dataset.

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

Original Dataset

Chest Pain	Good Blood Circ.	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

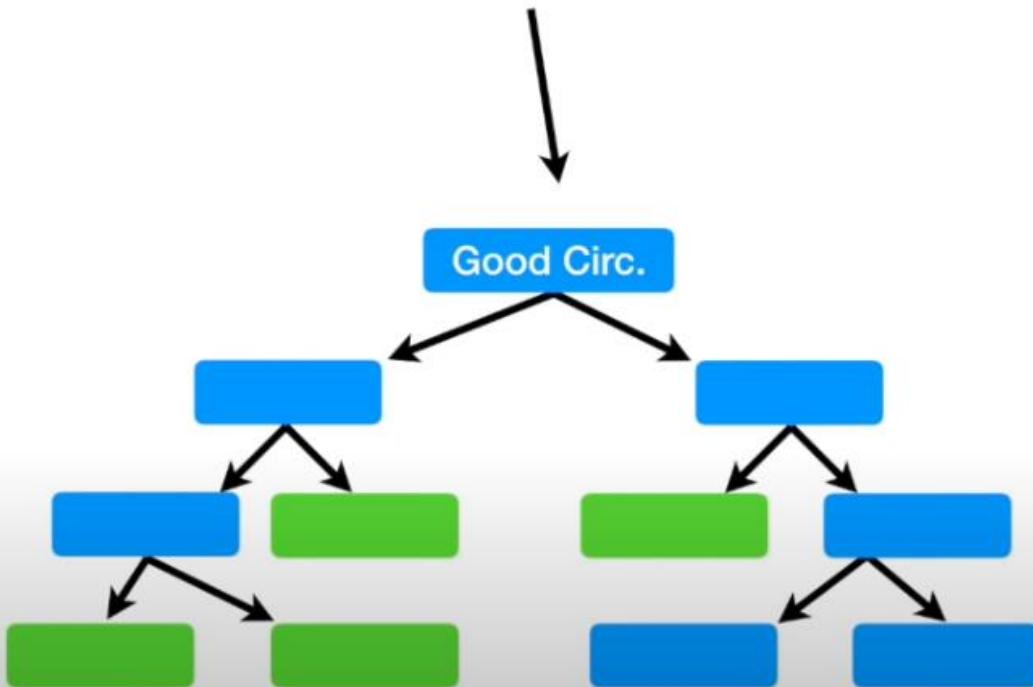
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

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

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No

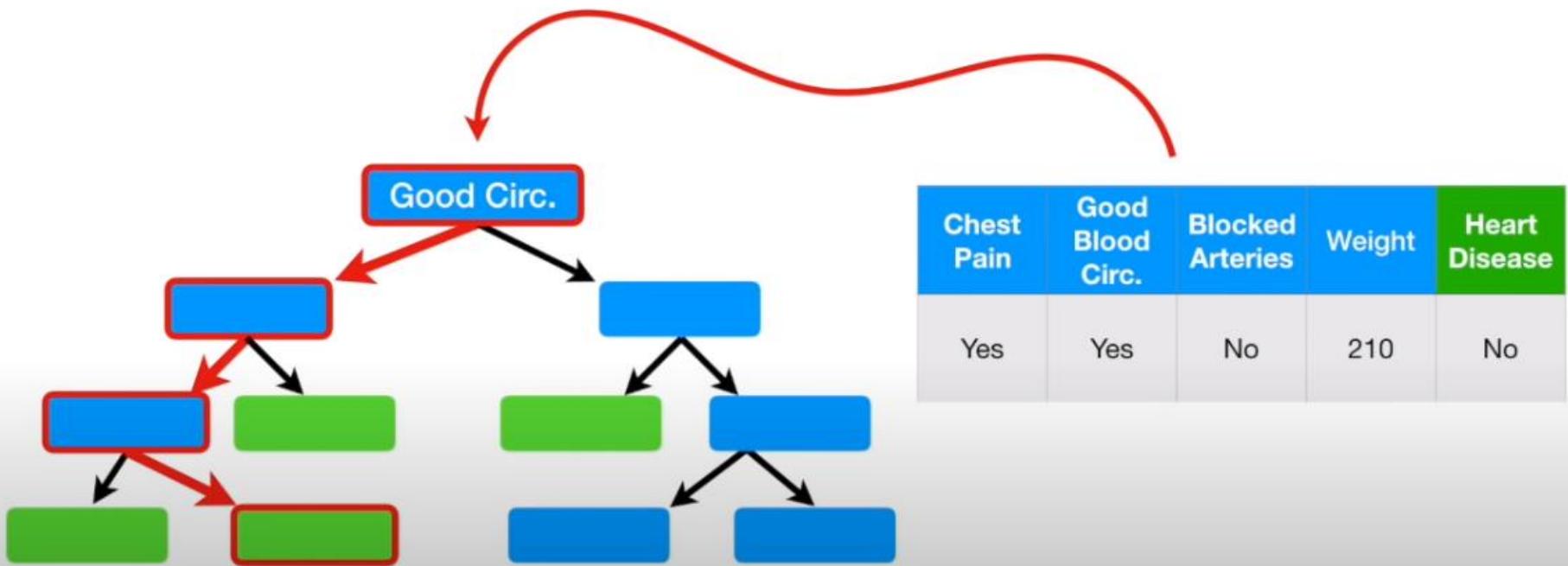
Test data

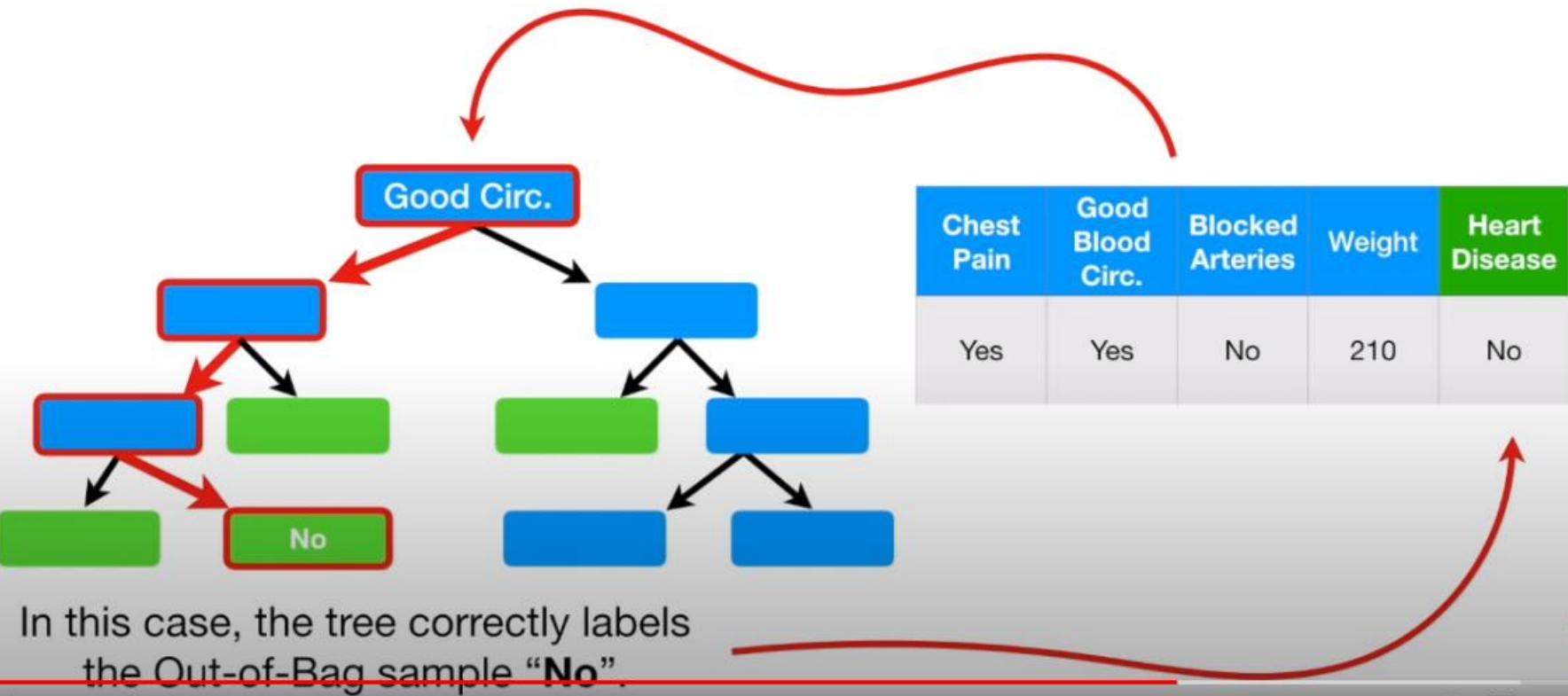
Since the Out-Of-Bag data was
not used to create this tree...

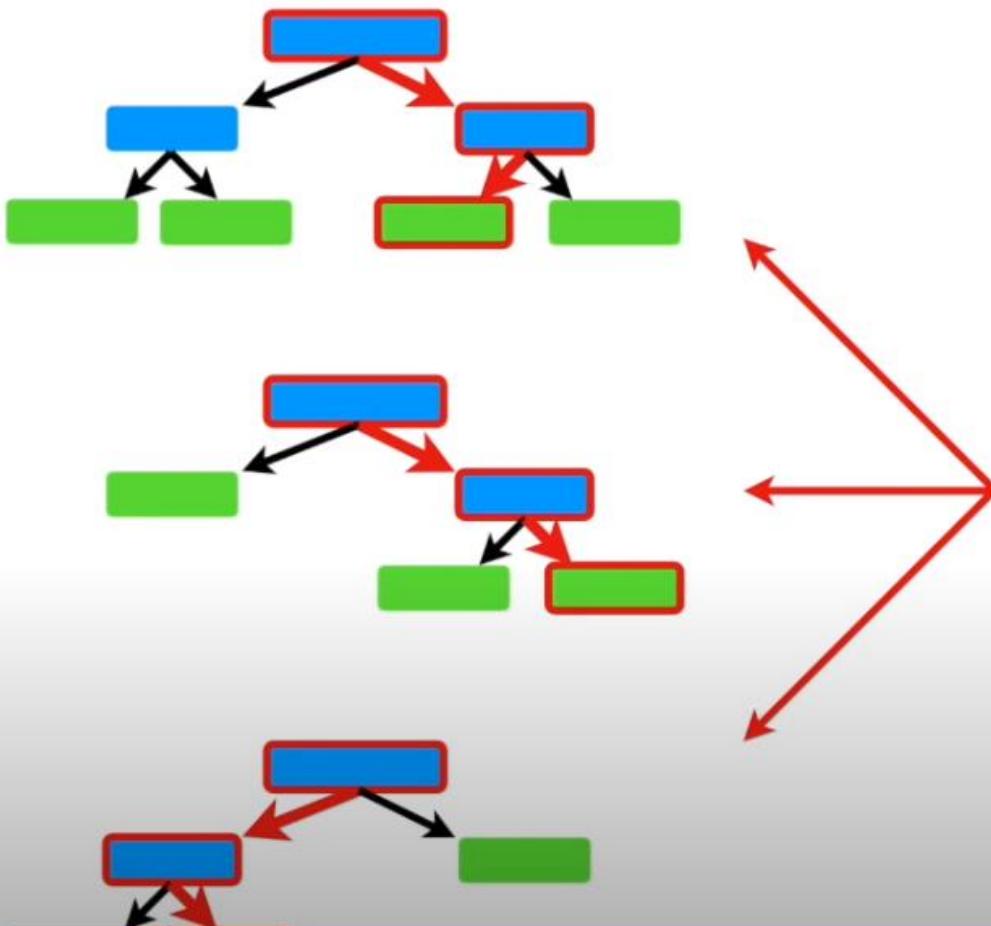


Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No

...we can run it through and see if it correctly classifies the sample as
“No Heart Disease”

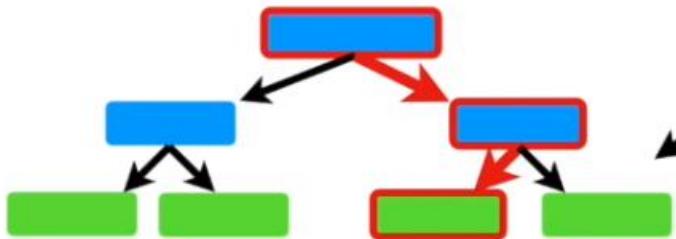




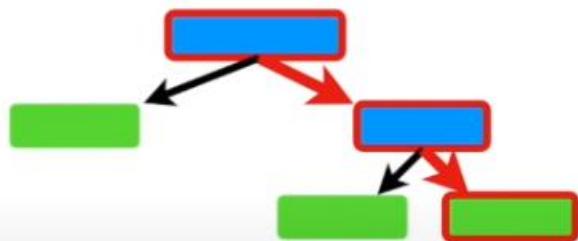


Then we run this Out-Of-Bag sample through all of the other trees that were built without it...

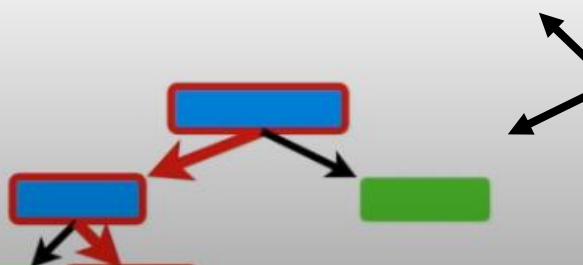
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No



This tree incorrectly labeled the Out-of-Bag sample “**Yes**”.



Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No



These trees correctly labeled the Out-of-Bag sample “**No**”.

Classification of the Out-Of-Bag sample

Yes

1

No

3

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.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No

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

Classification of the Out-Of-Bag sample	
Yes	No
1	3

Classification of the Out-Of-Bag sample	
Yes	No
4	0

This Out-of-Bag sample was also correctly labeled...

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes

(i)

Classification of the Out-Of-Bag sample

Yes	No
-----	----

1	3
---	---

Classification of the Out-Of-Bag sample

Yes	No
-----	----

4	0
---	---

Classification of the Out-Of-Bag sample

Yes	No
-----	----

3	1
---	---

etc... etc... etc...

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No

This Out-of-Bag sample was
incorrectly labeled...



Classification of the Out-Of-Bag sample

Yes	No
1	3

Classification of the Out-Of-Bag sample

Yes	No
4	0

Classification of the Out-Of-Bag sample

Yes	No
3	1

etc... etc... etc...

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**”

...we can talk a little
more about how to
do this!

OK, we now know how to:

- 1) Build a Random Forest
- 2) Use a Random Forest
- 3) Estimate the accuracy of a Random Forest.



However, now that
we know how to do
this...

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 Circ.	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

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

...to random forest built using 3 variables per step...



Bootstrapped Dataset

Chest Pain	Good Blood Circ.	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 we test a bunch of different settings and choose the most accurate random forest.

In other words...

...change the number of variables used per step...

1) Build a Random Forest

2) Estimate the accuracy of a Random Forest.

Do this for a 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 then try a few settings above and below that value.

Continue ... next week