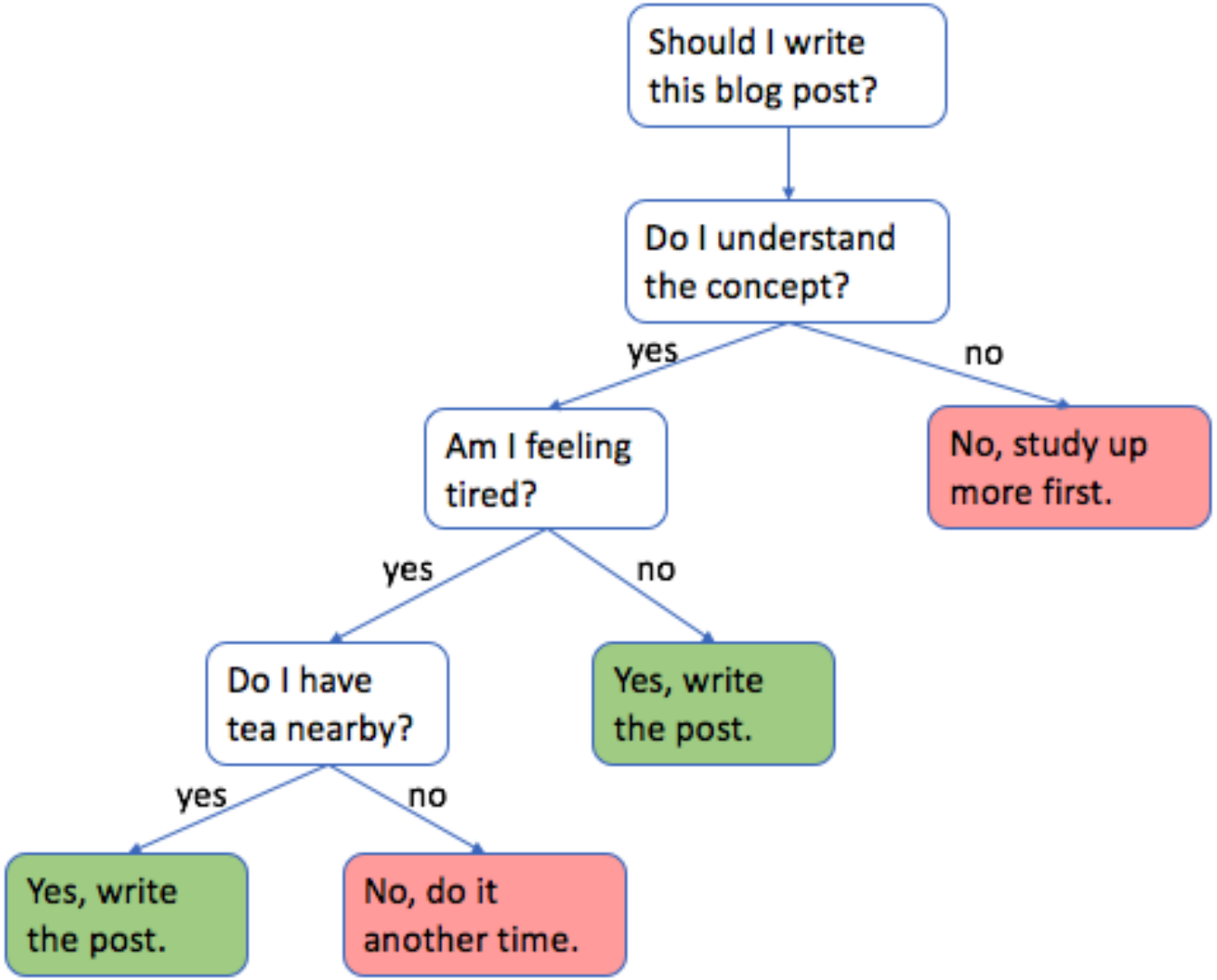
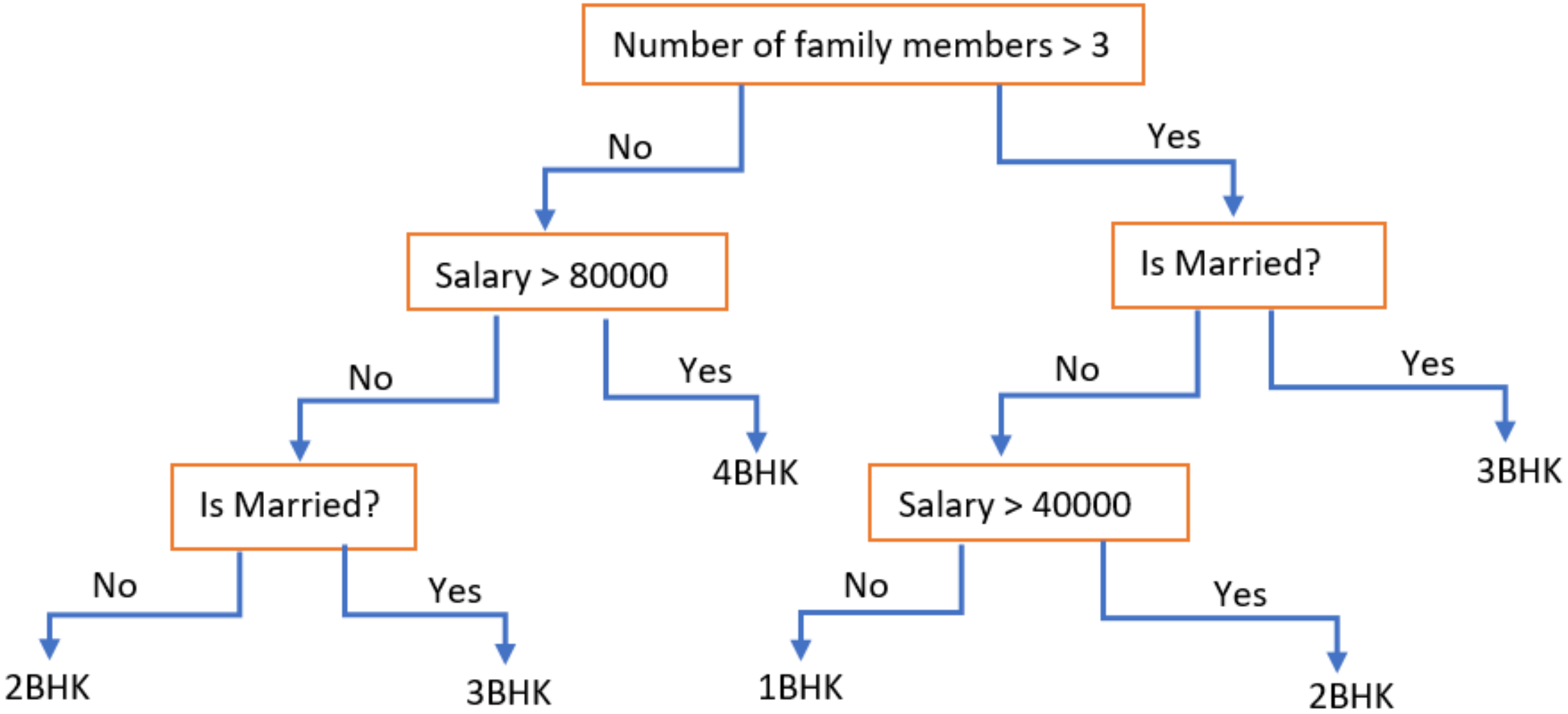
Decision Tree, Random Forests

Decision Tree

Decision making process - example



Decision making process - example



Gini Impurity

Gini index is the chance of getting different classes in a group

This way, it provides a measure of "Impurity" in the group

Gini Index = 0.32 < Gini Index = 0.72

Purer group

Less pure group

Gini Impurity Index = $1 - \sum_{i=1}^{n} p_i^2$

n: number of class in the data

p_i: probability of each class

Random Forest Classifier

Random Forests

- Decision Trees have one aspect that prevents them from being the ideal tool for predictive learning
- They work great with the data used to create them, but they are not flexible when it comes to classifying new samples – Overfitting
- Random Forests combines the simplicity of Decision Trees with flexibility resulting in a vast improvement in accuracy

Disadvantages of Random Forest

- Three Major Disadvantages - Usage of random forest is lesser in production
  - It is time consuming to predict on production - Because each trees have to separately predict on the same observation - slow during prediction on production data
  - Consulting scenario - used less - because u cannot visualize a random forest - something that u cannot visualize. Very difficult to communicate to stakeholders -
  - If your class is too imbalanced , random forest predictions cannot be trusted. If in your training data one or more of the class labels (Dep variable labels) is rare - that will mean even after bagging - there will be very few trees which will know about the class.

Impurity – 2 implementations – Gini & Entropy

- Gini (Binary classification) - CART
- Gini = 1- [Prob of (1)]^2 - [Prob of (0)]^2

- Entropy / Information Gain (Multiclass classification) - C4.5/ID3
- Entropy = Degree of randomness (Impurity)
- Information gain = Reduction in entropy
- Your predictive accuracy will improve as you lower the entropy
- Entropy = -Prob(1)*log2Prob(1) - Prob(0).log2Prob(0)

Creating a RF

- Step 1: Create a bootstrapped dataset – create samples with replacement

Original Dataset

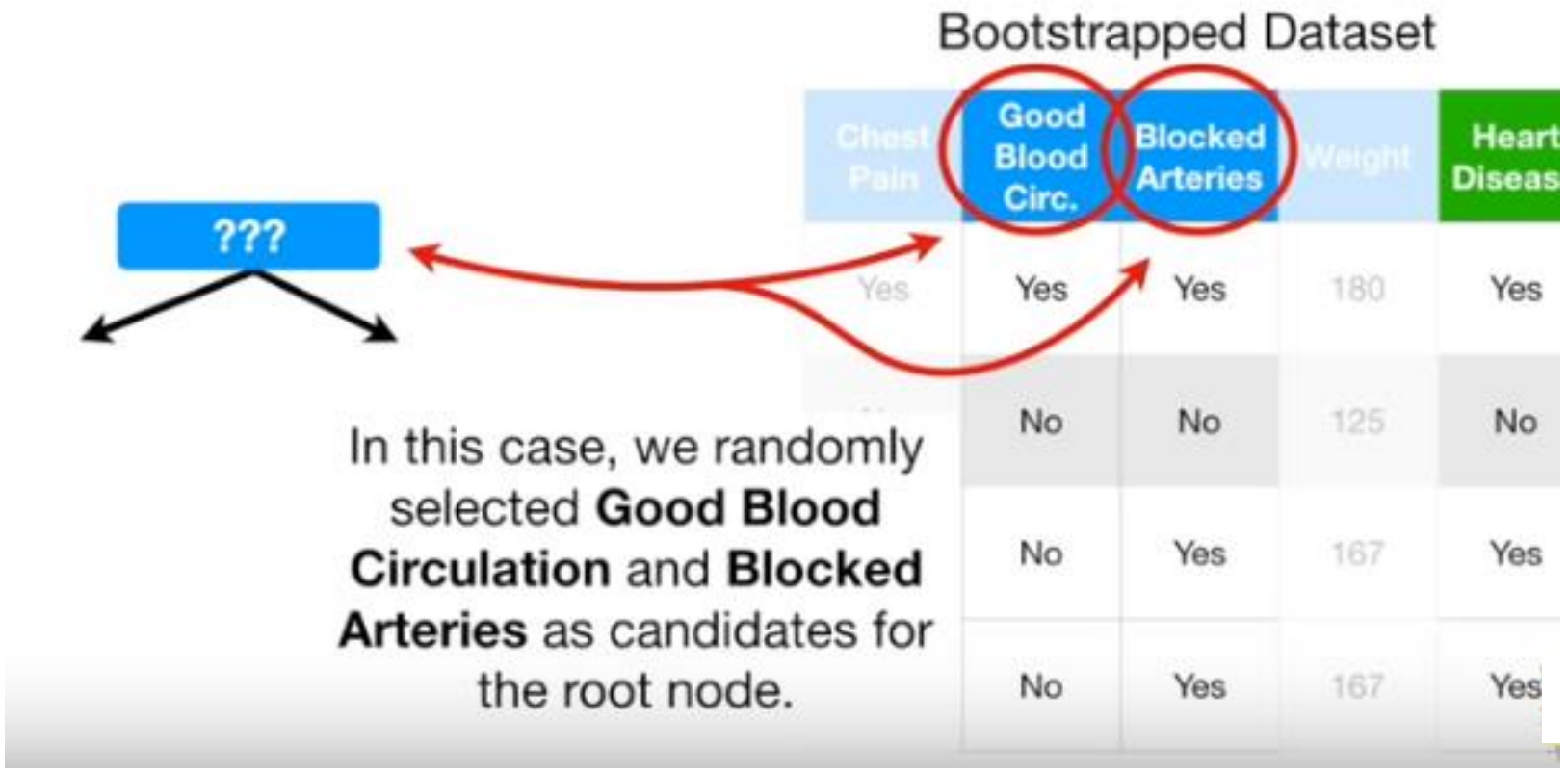
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

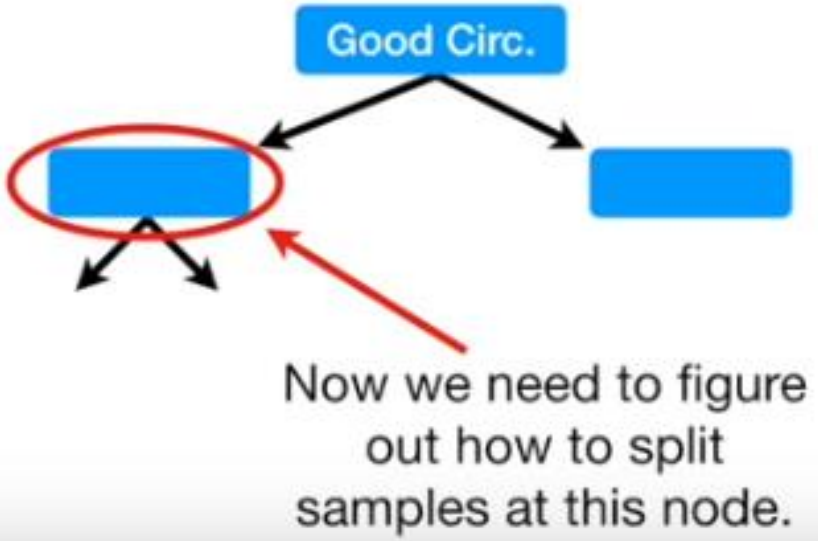
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Creating a RF

- Step 2: Create a Decision Tree with the bootstrapped dataset, but only use a random subset of variables (or columns) at each step



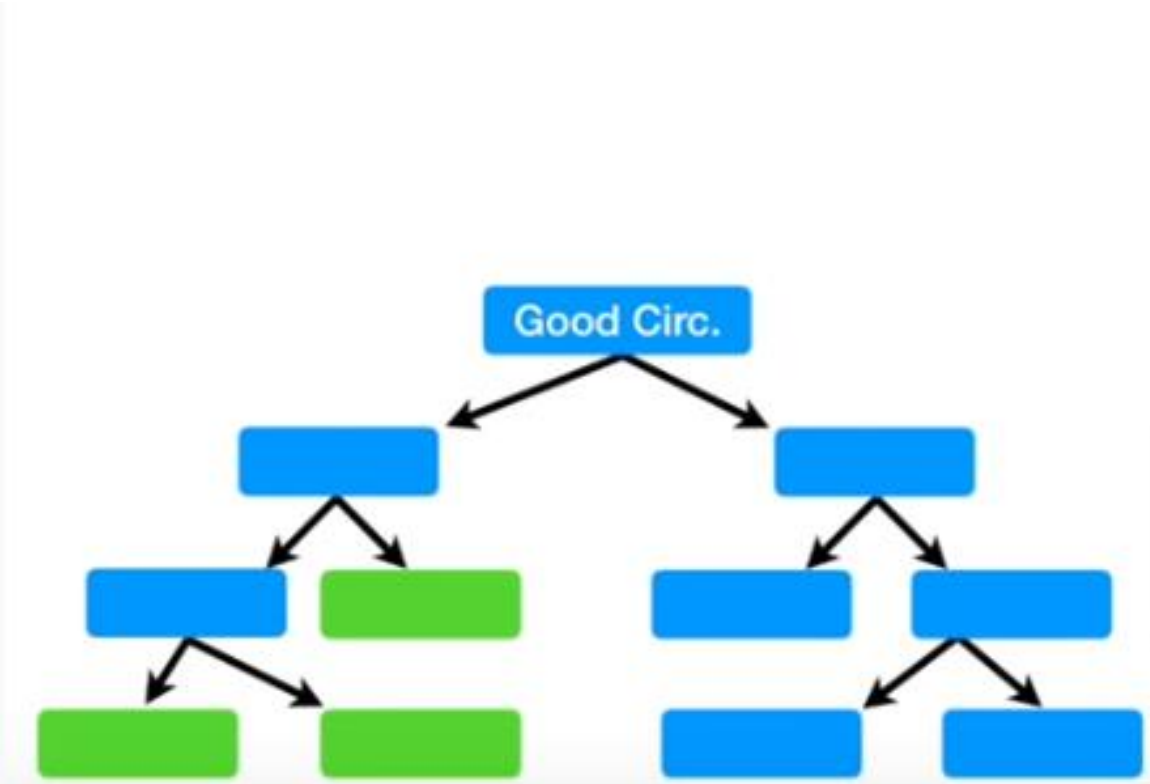
Creating a RF



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

Creating a RF



Bootstrapped Dataset

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

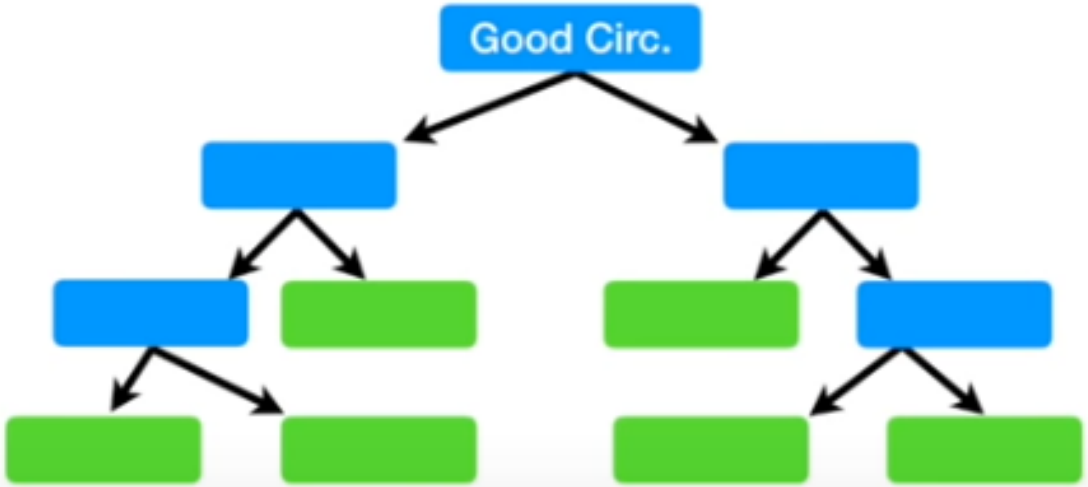
And we just build the tree as usual, but only considering a random subset of variables at each step.

Creating a RF

We built a tree…

1) Using a bootstrapped dataset

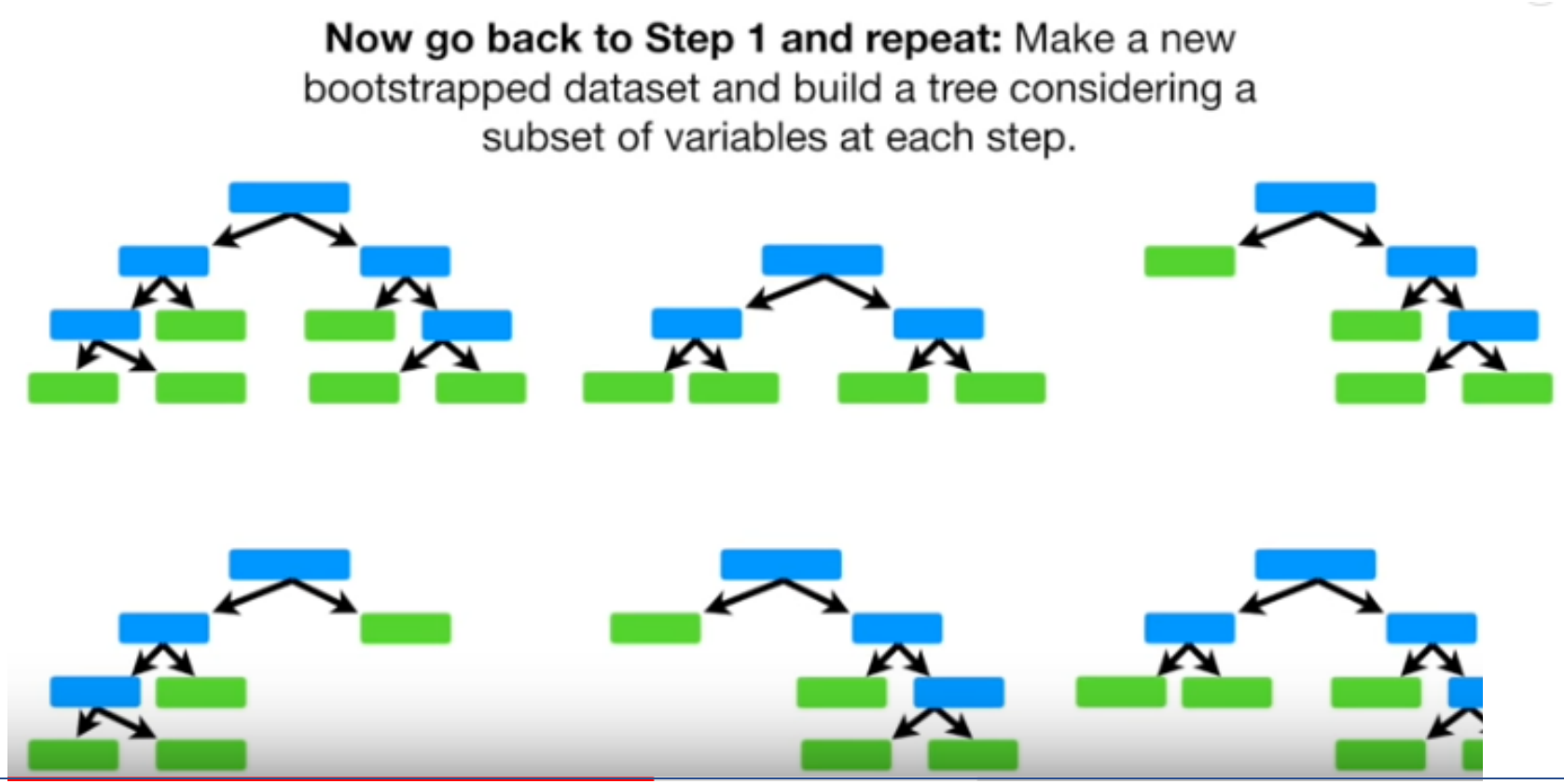
2) Only considering a random a subset of variables at each step.



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

Creating a RF



- Ideally you do this 100's of times
- Using a bootstrapped sample and considering a subset of features at each step results in a wide variety of trees
- The variety is what makes Random Forests more effective than Decision Trees

Using a RF

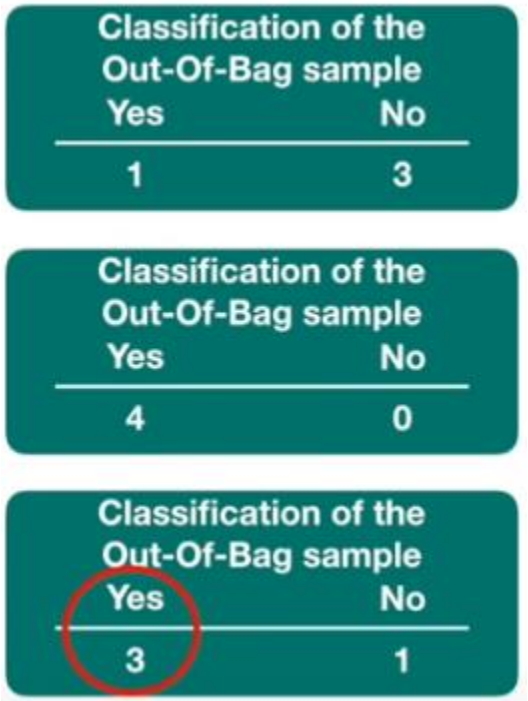
- New Data

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

- We take this data and run it down the first tree that we made
- Lets say the first tree says “Yes”. The patient has a heart disease
- Now we run the data down the second tree
- The second tree also says “Yes”
- We repeat this for all the trees and find out which options received the maximum votes
- In this case “Yes” received the most votes, so we conclude that the patient has a probability for heart disease

Bagging, Out of Bag dataset, Out of Bag Error

- Bootstrapping the data plus using the aggregate to make a decision is called "**Bagging**"
- The bootstrapped dataset allows duplicate entries. As a result we end up having observations which don't make it to the bootstrapped dataset. Typically about 1/3rd of the observations don't make it to the bootstrapped dataset
- This is called the "**Out of Bag**" dataset
- We can run the Out of Bag observation through the DT, as a Test dataset. It is similarly run through all the other DTs where the OOB observation did not make it.
- We again take a voting of all outcomes using the OOB dataset
- Ultimately, we can measure how accurate our random forest is by the proportion of OOB observations that were correctly classified by the Random Forest
- The proportion of OOB observations that were incorrectly classified is the "**Out-of-Bag Error**"



Ramdom Forest - Summary

Training and Testing of Random Forests:

1. Create bootstrapped data sets
2. Build decision trees for each data set using different set of variables
3. Test Random Forest using Out-Of-Bag Data set