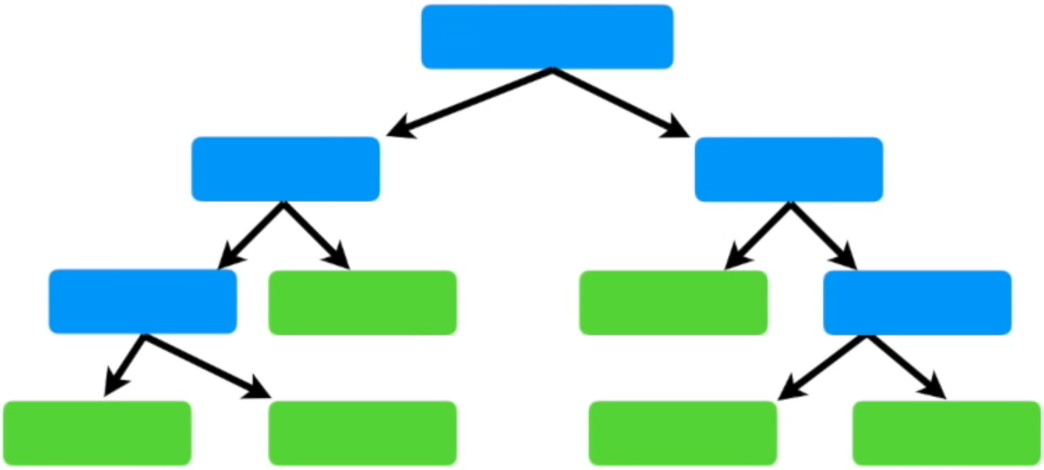
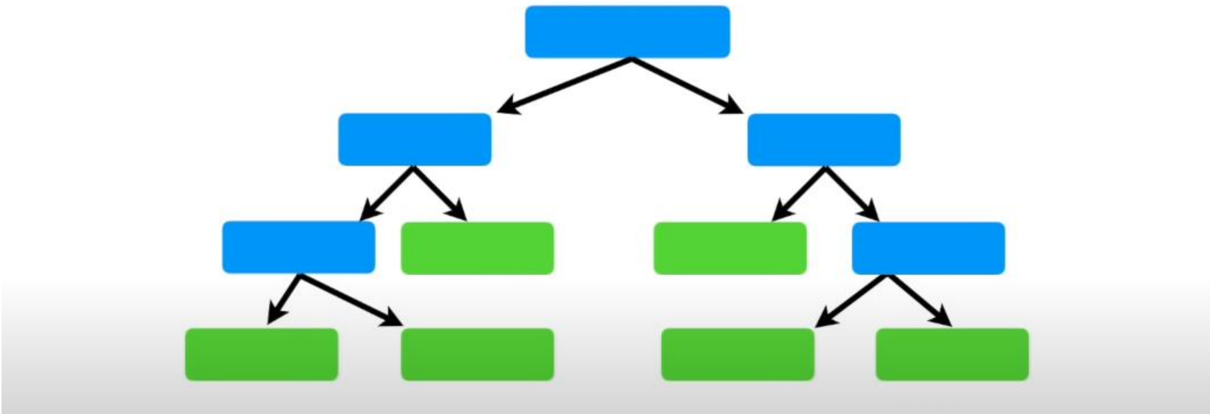
Decision Trees are easy to build, easy to use and easy to interpret…

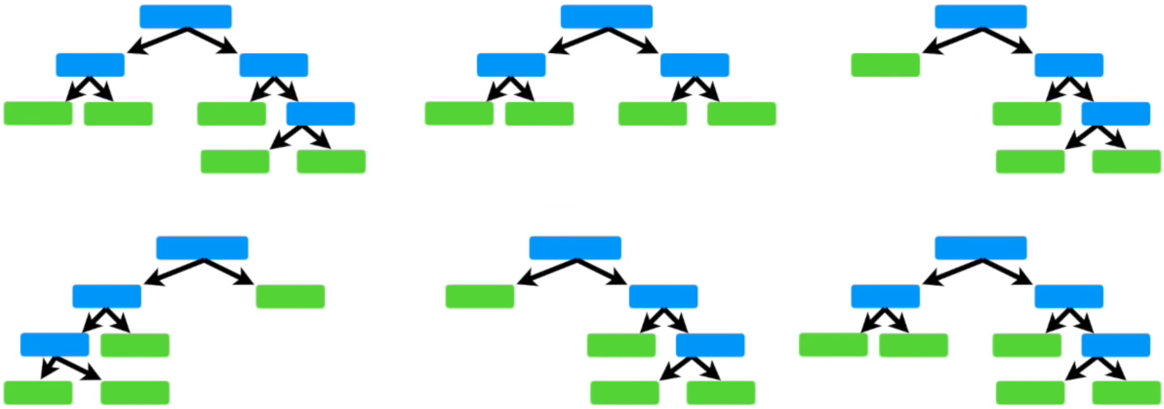


…but in practice they are not that awesome.

To quote from ***The Elements of Statistical Learning*** (aka The Bible of Machine Learning), "Trees have one aspect that prevents them from being the ideal tool for predictive learning, namely **inaccuracy**."



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



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

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

… I know it's crazy small, but just pretend for now.

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.

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

Here's the fourth randomly selected sample (**Note**: it's the same as the third)…

Bam!!! We've created 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
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…

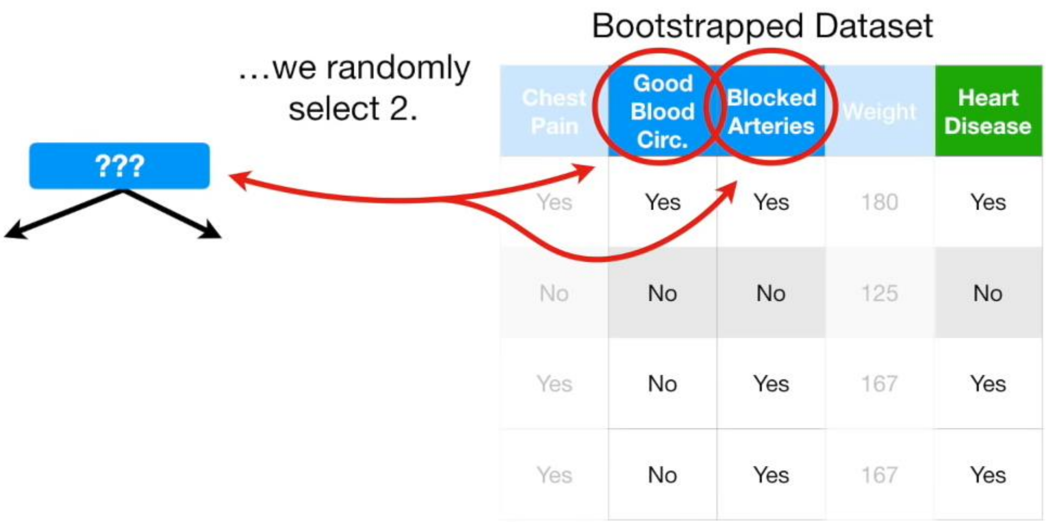
Bootstrapped Dataset

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

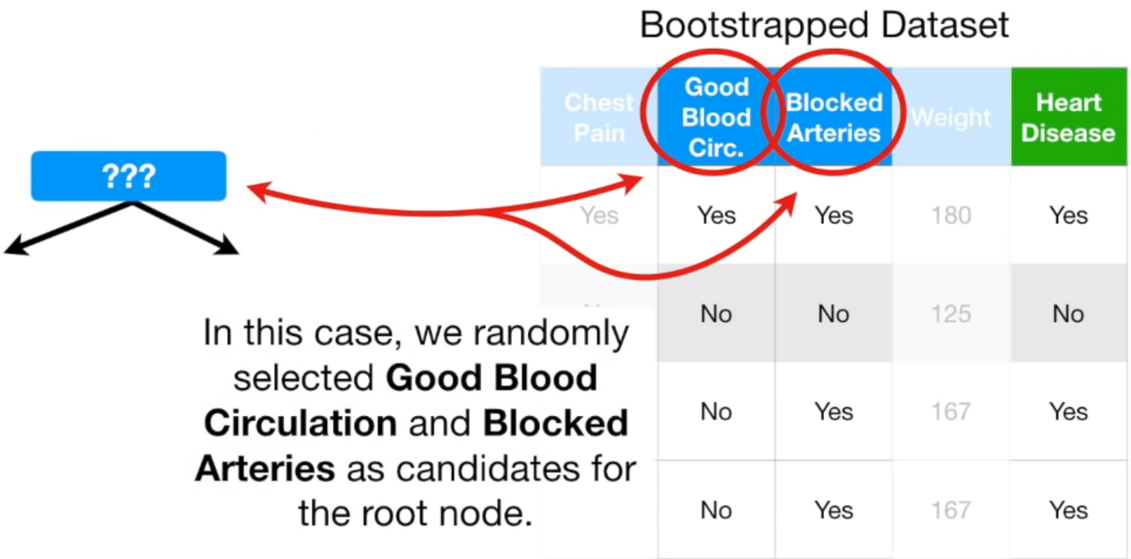
Thus, instead of considering all 4 variables to figure out how to split the root node…

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



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



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

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

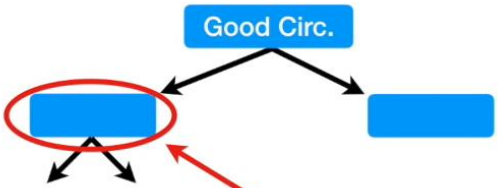
Since we used **Good Blood Circulation**, I'm going to grey it out so that we focus on the remaining variables.

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

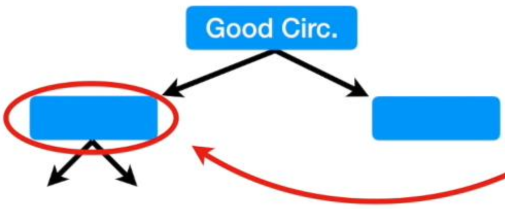
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes



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

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

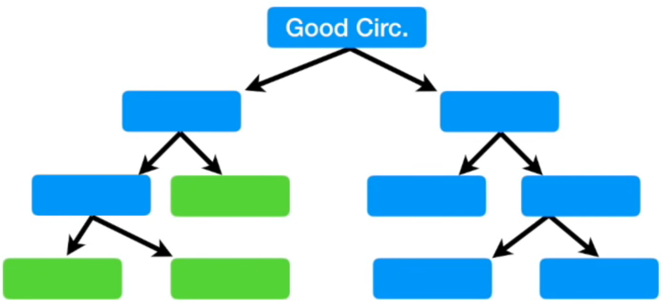


Just like for the root, we randomly select 2 variables as candidates, instead of all 3 remaining columns.

Bootstrapped Dataset

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

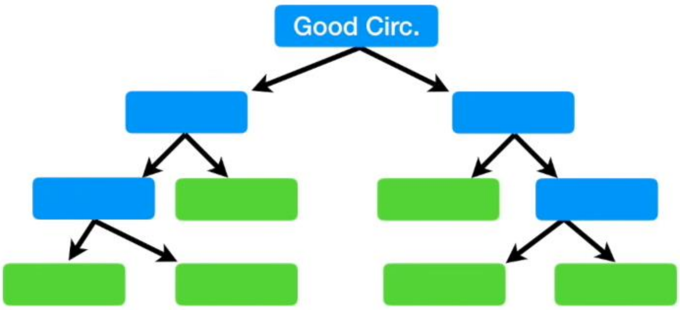
And we just build the tree as usual, but only considering a random subset of variables at each step.



We built a tree…

1) Using a bootstrapped dataset

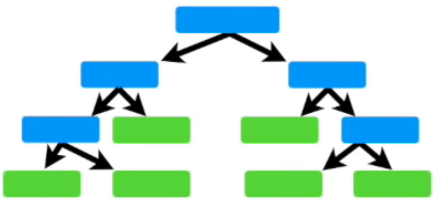
2) Only considering a random a subset of variables at each step.



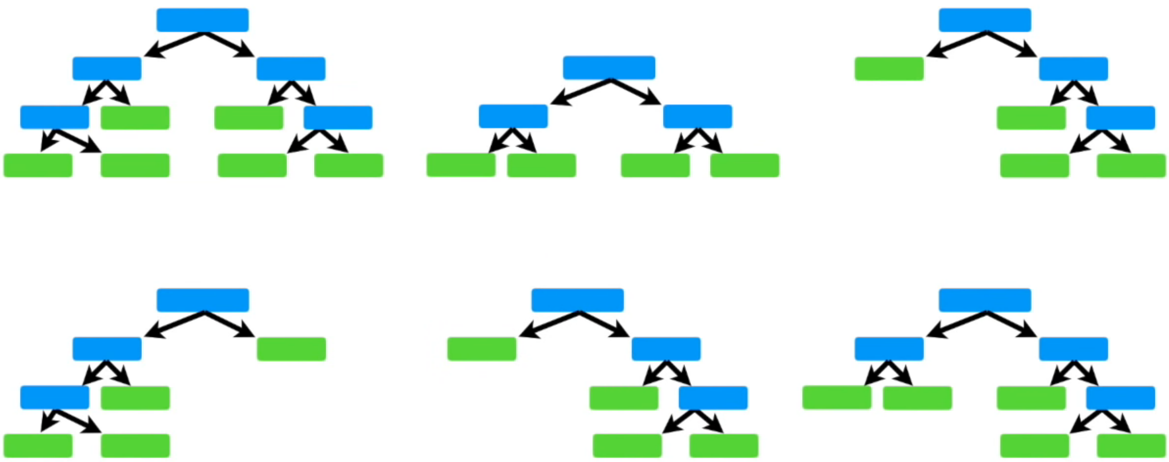
Bootstrapped Dataset

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

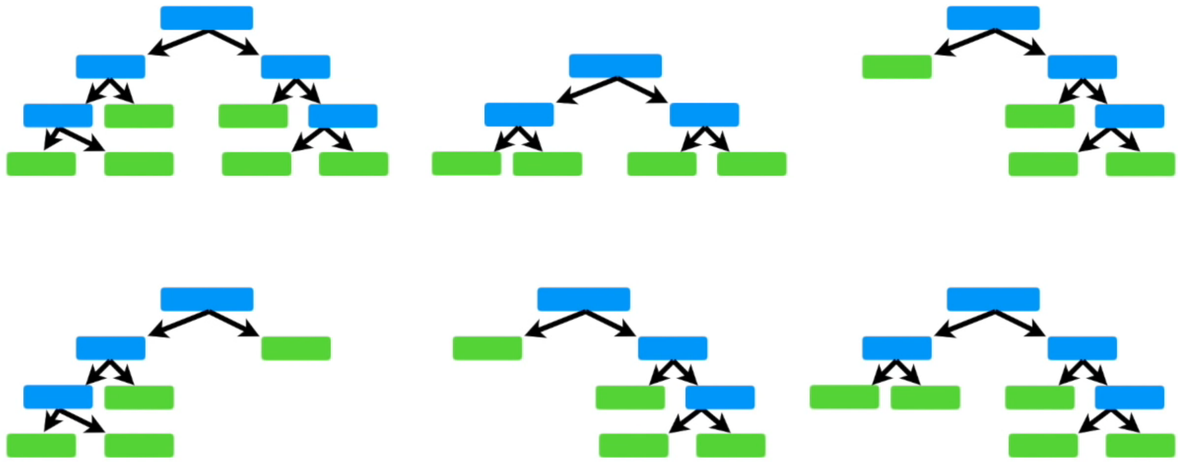
Here's the tree we just made…



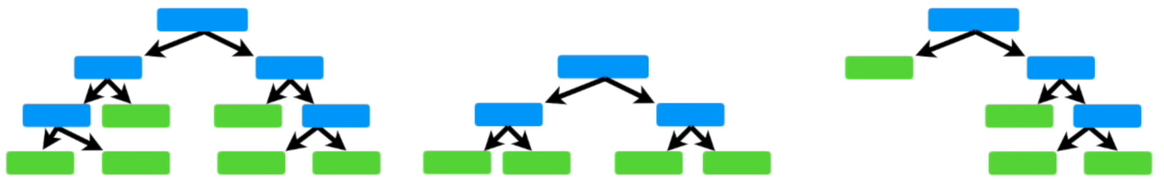
Now go back to Step 1 and repeat: Make a new bootstrapped dataset and build a tree considering a subset of variables at each step.



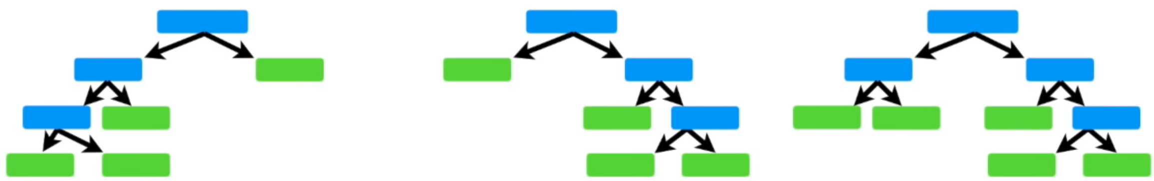
Ideally, you'd do this 100's of times, but we only have space to show 6… but you get the idea.



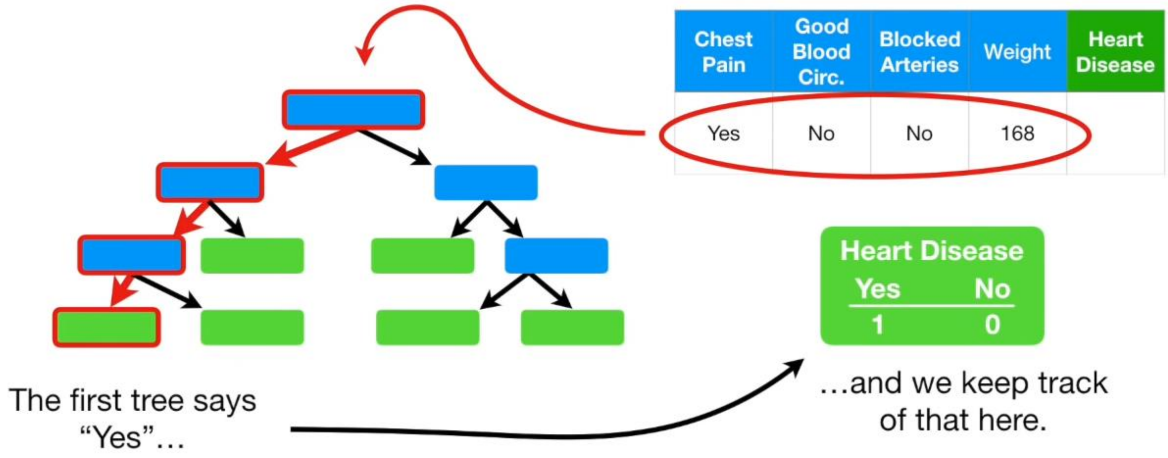
Using a bootstrapped sample and considering only a subset of the variables at each step results in a wide variety of trees.

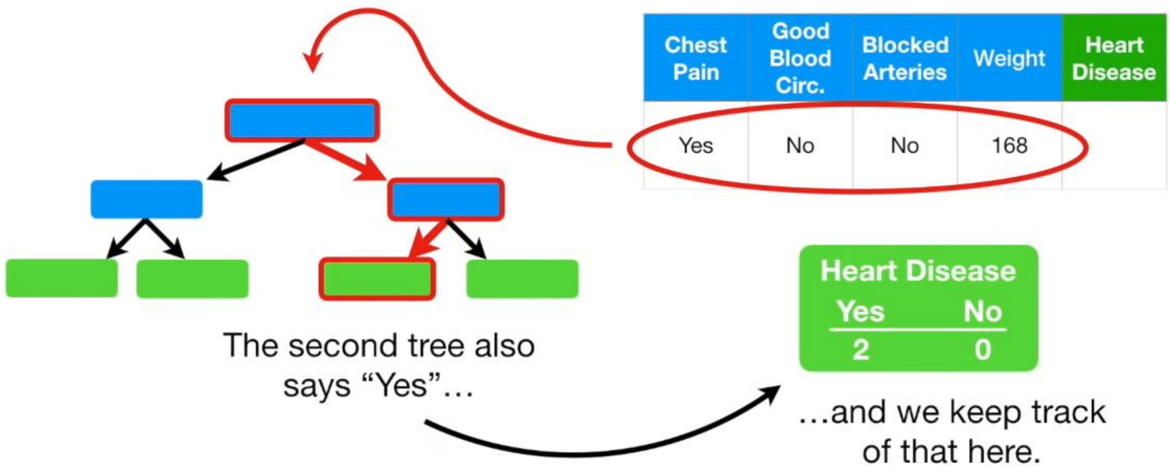


The variety is what makes random forests more effective than individual decision trees.

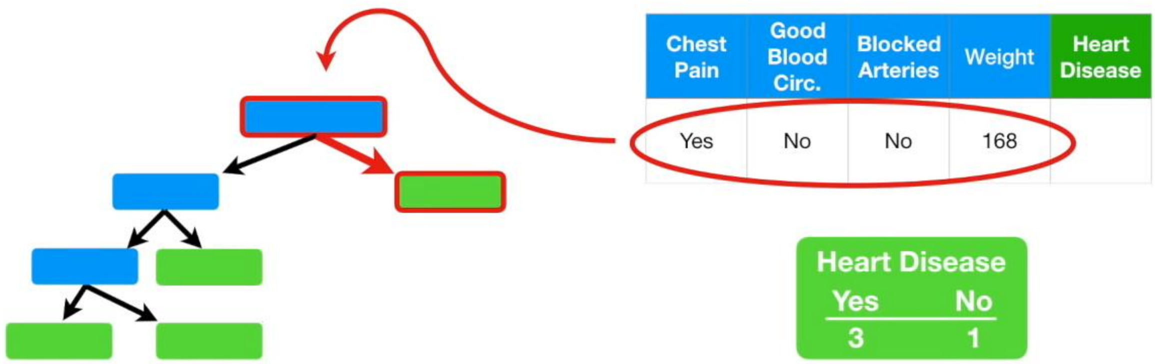


Sweet!!! Now that we've created a random forest, how do we use it?





Then we repeat for all the trees that we made…



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

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

Heart Disease	
Yes	No
5	1

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.

Heart Disease	
Yes	No
5	1

Terminology Alert!!!

Bootstrapping the data plus using the **agg**regate to make a decision is called "**Bagging**"

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

How do we know if it's any good?

Remember when we created 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

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

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

Typically, about 1/3 of the original data does not end up 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

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

Here is the entry that didn't end up in the bootstrapped dataset..

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

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

(psst… if the original dataset were larger, we'd have more than just 1 entry over here…)

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

Original Dataset

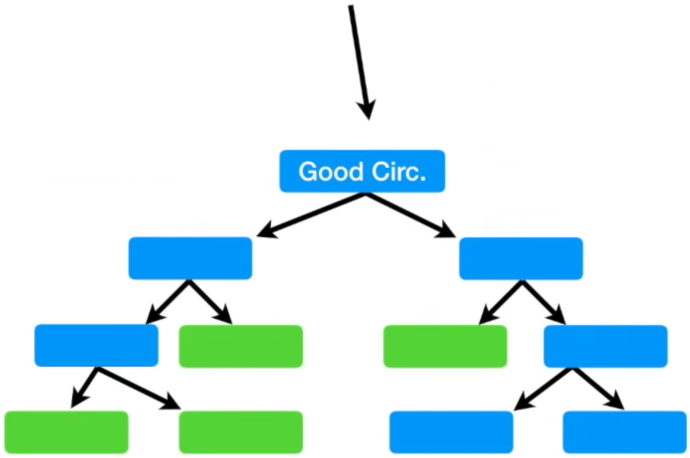
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

This is called the "**Out-Of-Bag Dataset**"

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

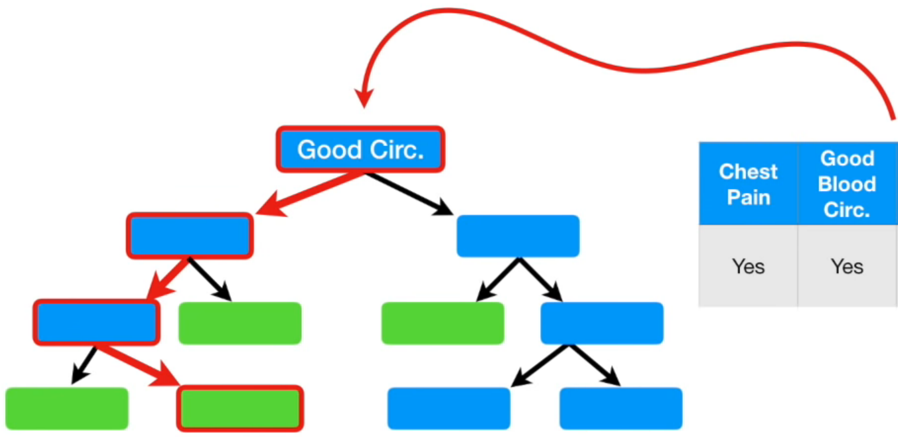
(If it were up to me, I would have named it the "**Out-Of-Boot Dataset**", since it's the entries that didn't make it into the bootstrap dataset.)

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

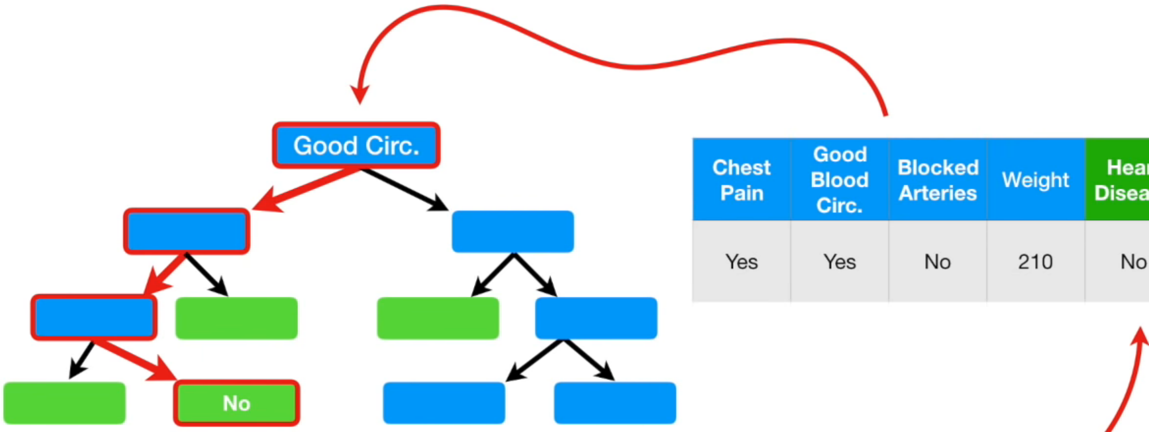


Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No

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



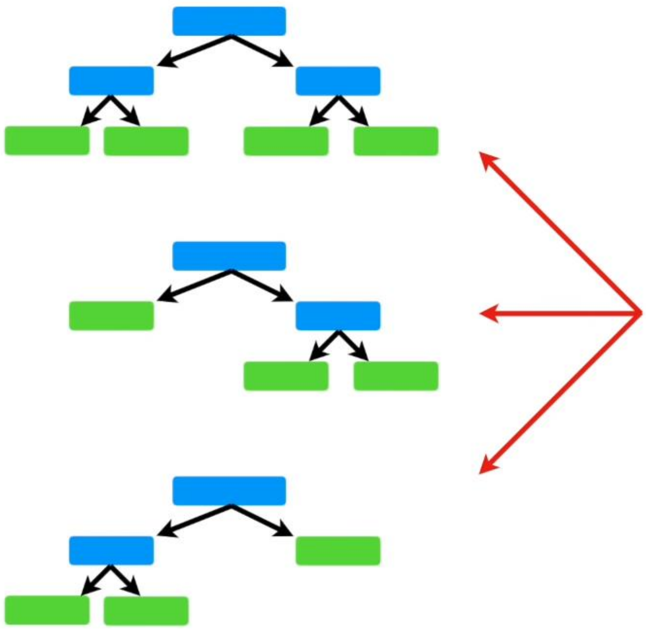
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No



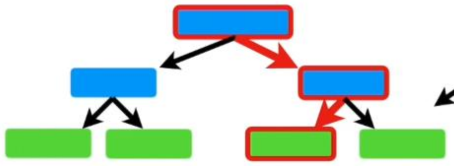
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No

In this case, the tree correctly labels the Out-of-Bag sample "**No**".

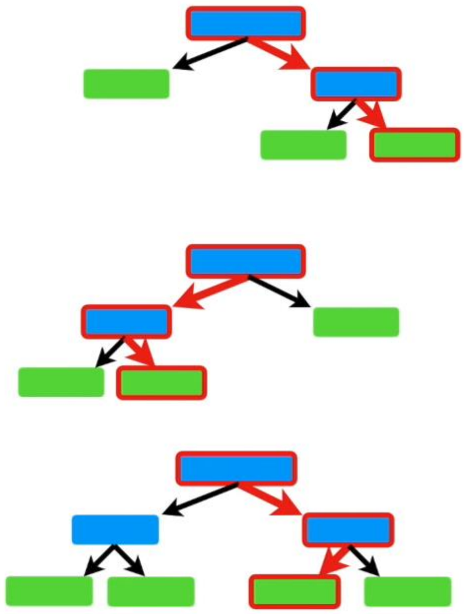
Then we run this Out-Of-Bag sample through all of the other trees that were built without it…



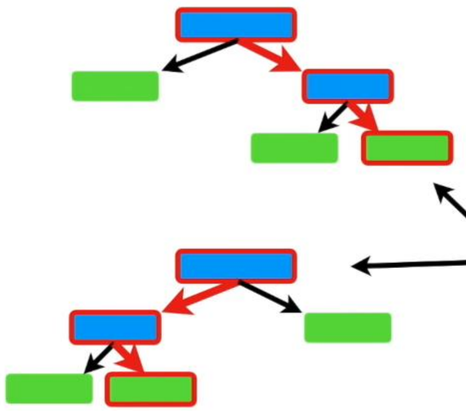
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No



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



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	No
1	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

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

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

Classification of the Out-Of-Bag sample

Yes	No
3	1

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**"

OK, we now know how to:

1) Build a Random Forest

2) Use a Random Forest

3) Estimate the accuracy of a Random Forest.

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

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…

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

…to random forest built using 3 variables per step…

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

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

In other words…

1) Build a Random Forest

…change the number of variables used per step…

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 a few settings above and below that value.

Hyperparameter Tuning with Cross Validation