Random Forest

Ensemble Learning

- Ensemble learning is when you take multiple machine learning algorithms and put them together to create one bigger machine learning model so that the final model can leveraging many outputs and combine them to give a better output.
- There are four steps to construct a Random Forest:
    1. Randomly pull-out K number of datapoints from the dataset. Also ensure that you randomly select data from only F number of features. This is called the bootstrap dataset
    2. Train a Decision Tree for the K number of datapoints. When we train the decision tree, instead of calculating the Gini Impurity for all the Features, we randomly select F number of features from the dataset and calculate gini impurity just for those F features and select the root.
    3. Go back to Step 1 to pull another set of datapoints. Repeat Step 1 and 2 to build N number of trees
    4. While prediction:
        1. Take the datapoint and it pass it to all the N trees. You'll get N outputs.
        2. Do a majority vote on the given datasets to identify the predicted class

Ensemble learning relies on many weak models.

Let’s take a toy 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

Let’s take a toy 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

Good blood circ	Blocked Arteries	Heart Disease

The bootstrapped dataset is generally supposed to be the same size as that of the original dataset.
We randomly select datapoints from the original dataset. To ensure that we only pick a subset and we also ensure that the bootstrapped dataset is of the same size, we’re allowed to pick the same datapoint multiple times from the original dataset to the bootstrapped dataset.

Let’s take a toy 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

$\rightarrow$

Chest Pain	Good blood circ	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes

Let’s take a toy 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

$\Rightarrow$

Chest Pain	Good blood circ	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No

Let's take a toy 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

$\rightarrow$

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

Let’s take a toy 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

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

Notice that we’ve repeatedly used the fourth datapoint from the original dataset twice in the bootstrapped dataset

We’ve obtained the 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 that we have our bootstrapped dataset, we should start building a decision tree to fit this data.

However, before we start identifying the root node, we must first randomly select F (=2) number of candidates for the root node instead of computing the gini impurity for all features.

We've obtained the bootstrapped dataset

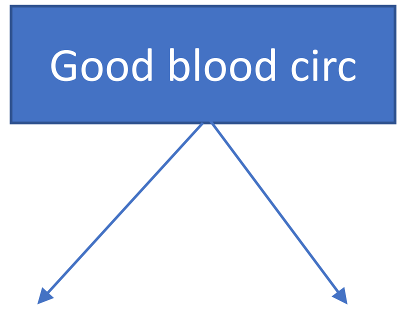
Chest Pain	Good blood circ	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Here, let's choose Good blood circulation and Blocked Arteries as the candidates for the root node.

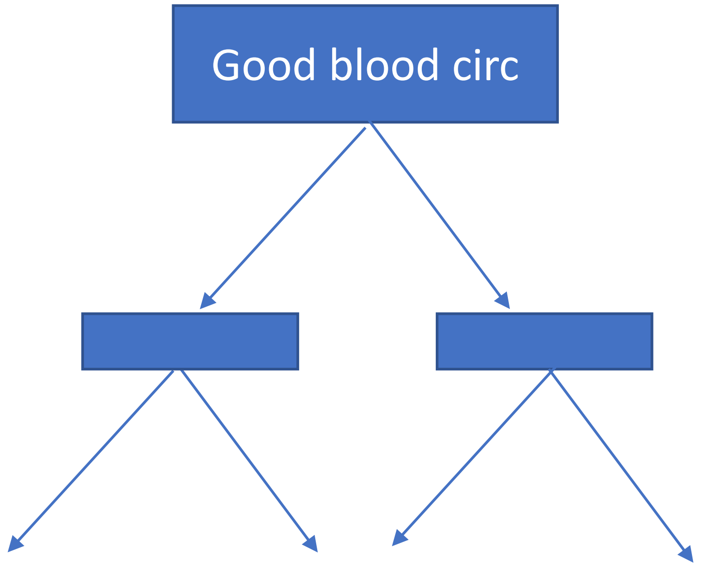
We’ve obtained the bootstrapped dataset

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, the feature ‘Good Blood Circulation’ has a lower gini impurity in comparison to the other candidate feature “Blocked artery”, we’ll choose ‘Good Blood Circulation’ as the root node.



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

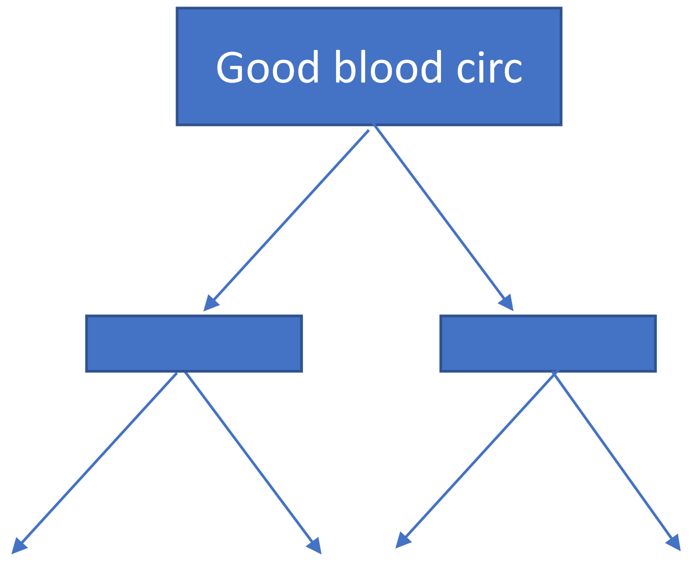


What have we done so far?

- We build a boostrapped dataset.
- We then create a tree only considering a random number of features at each step of building the tree.

What have we done so far?

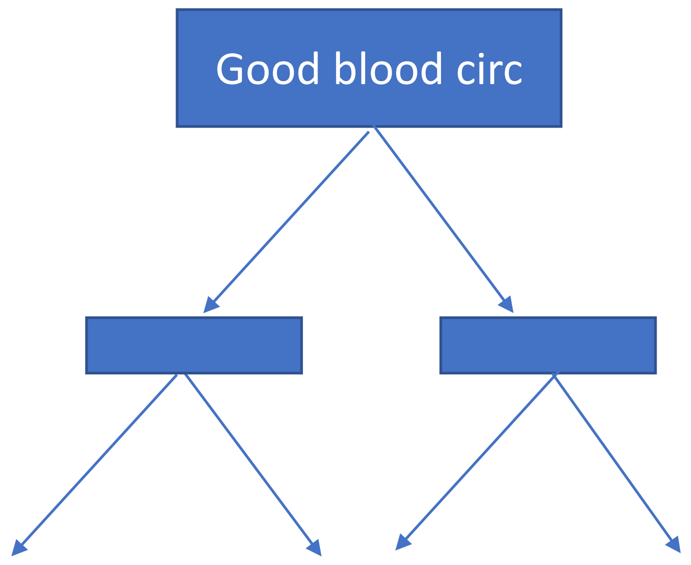
- We build a boostrapped dataset.
- We then create a tree only considering a random number of features at each step of building the tree.



What do we do next?

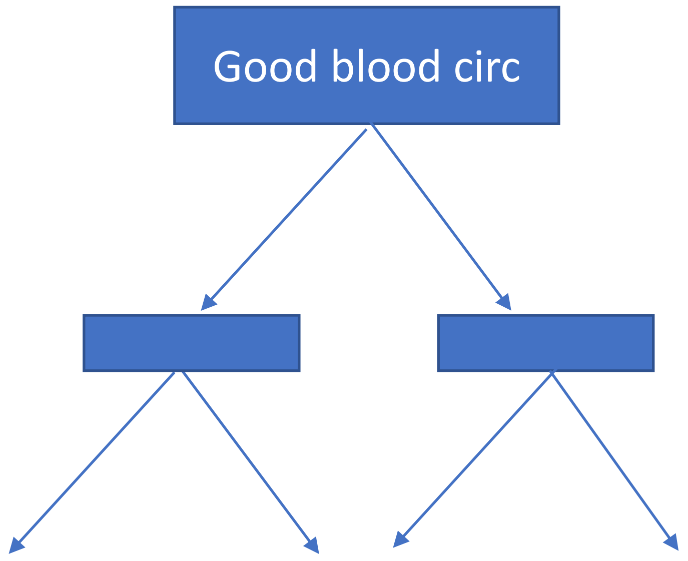
What do we do next?

Well, go back to step 1, build a new bootstrapped dataset with random selection, build a new tree by randomly selecting feature sets at each step.



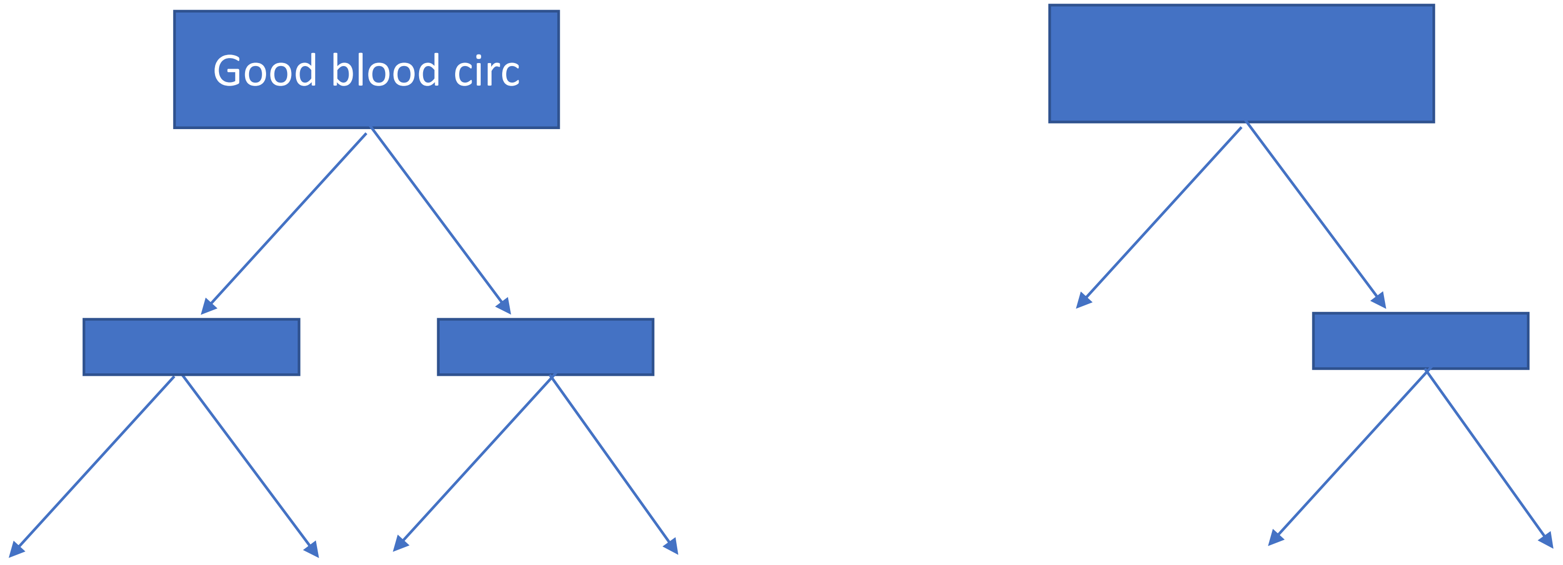
Well, go back to step 1, build a new bootstrapped dataset with random selection, build a new tree by randomly selecting feature sets at each step.

This way, we'll can get Ntrees number of decision trees



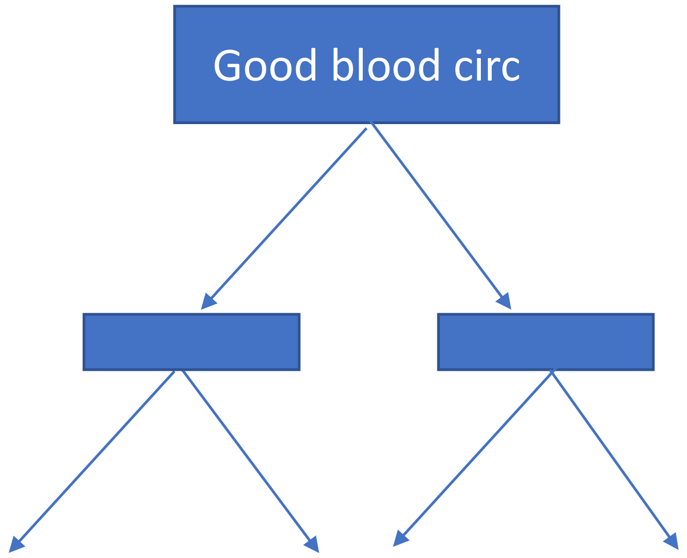
Well, go back to step 1, build a new bootstrapped dataset with random selection, build a new tree by randomly selecting feature sets at each step.

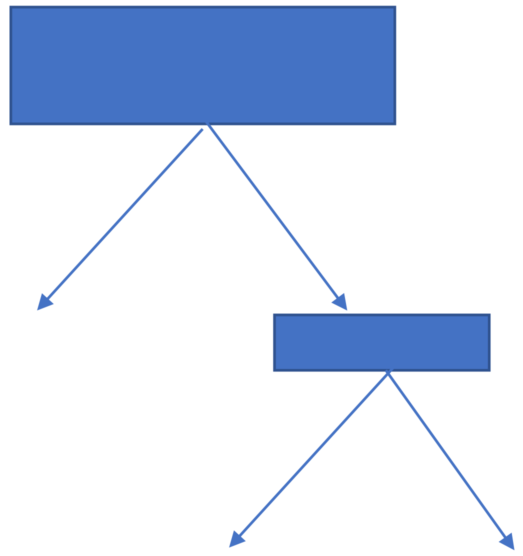
This way, we'll can get Ntrees number of decision trees

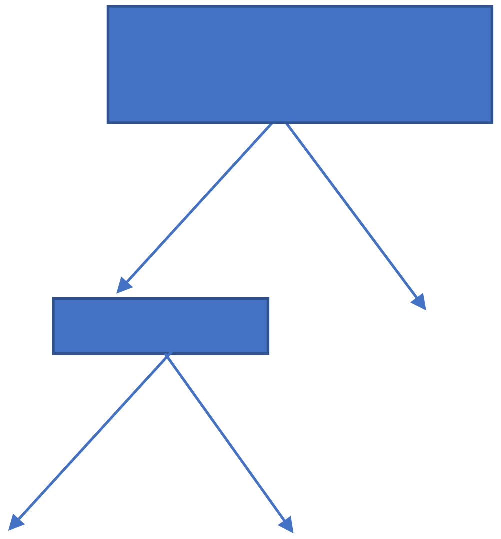


Well, go back to step 1, build a new bootstrapped dataset with random selection, build a new tree by randomly selecting feature sets at each step.

This way, we'll can get Ntrees number of decision trees

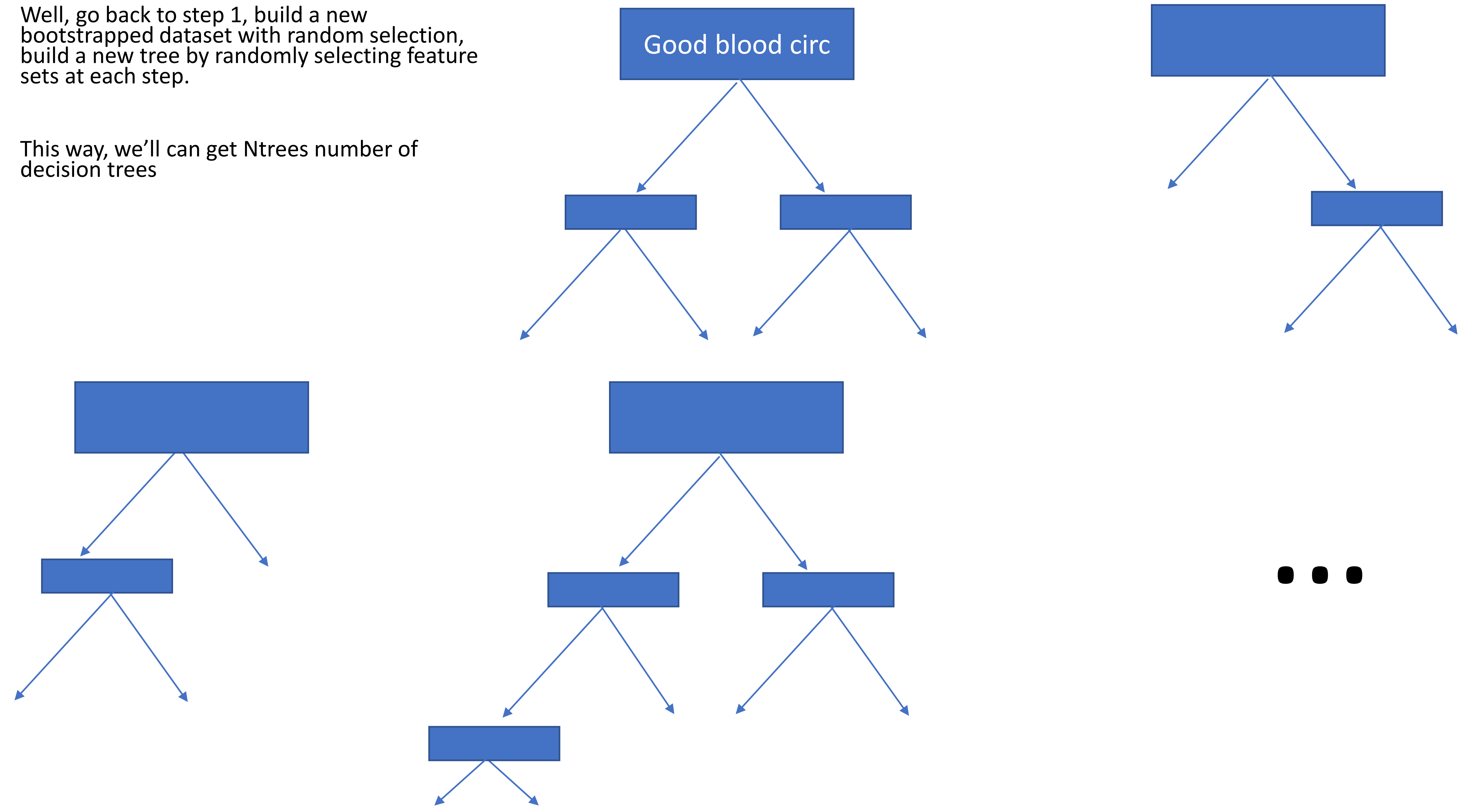




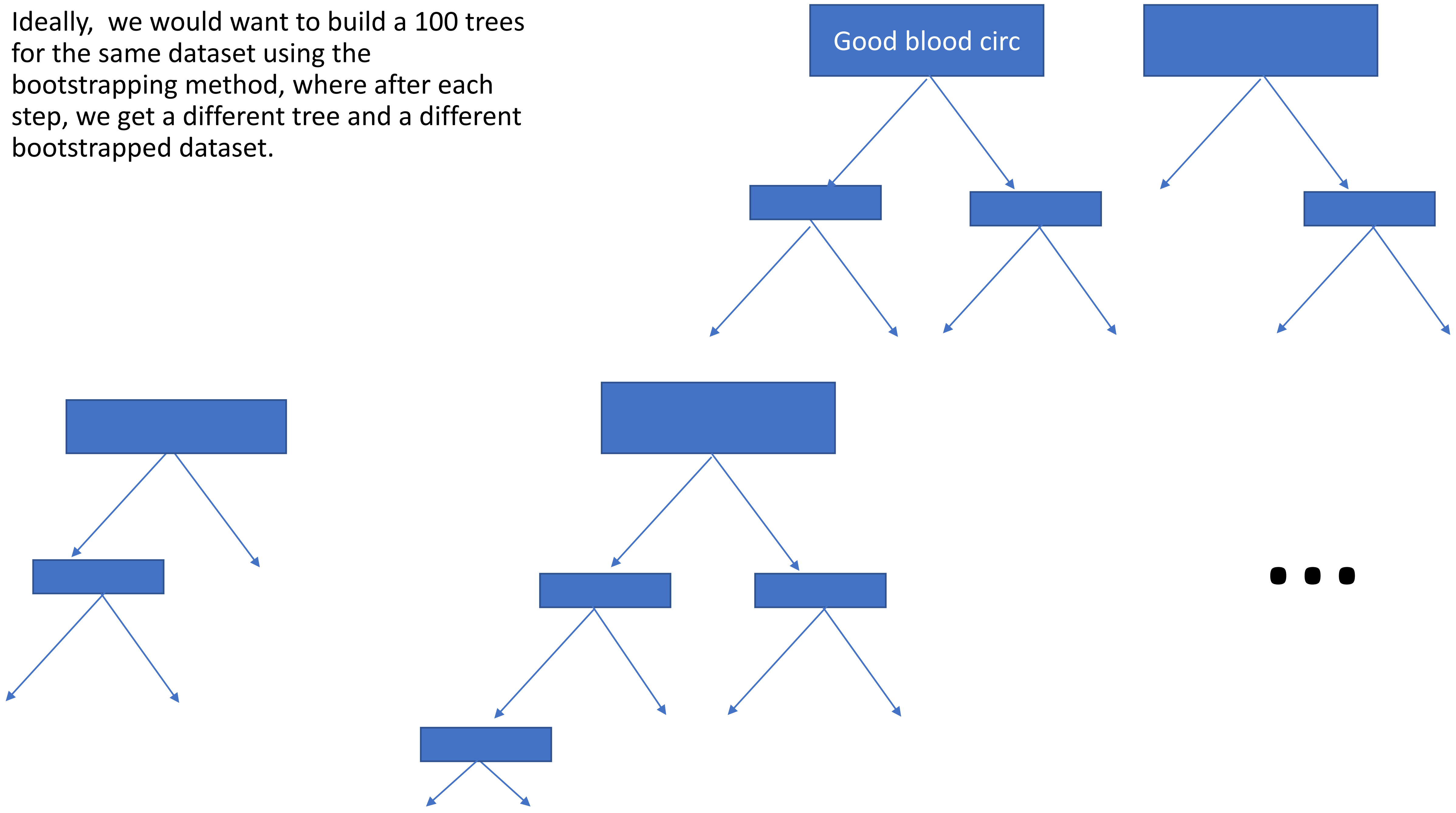


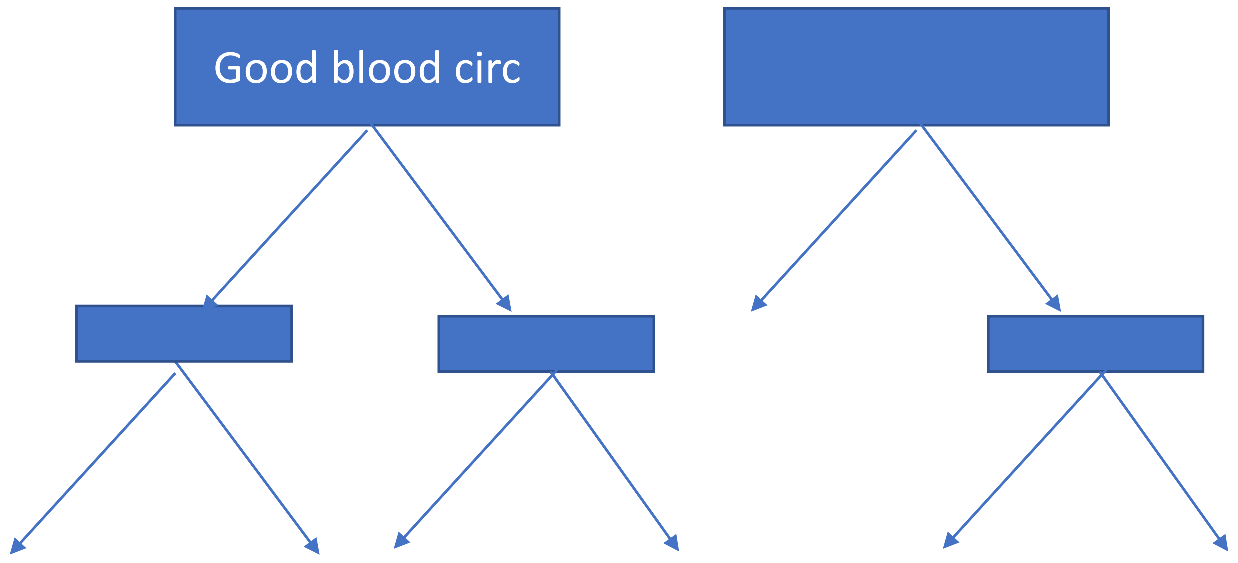
Well, go back to step 1, build a new bootstrapped dataset with random selection, build a new tree by randomly selecting feature sets at each step.

This way, we'll can get Ntrees number of decision trees

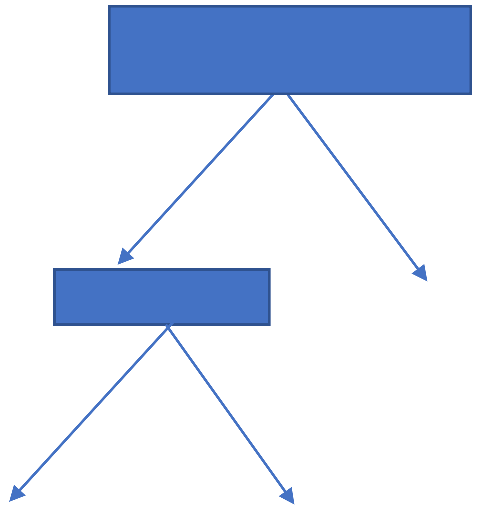


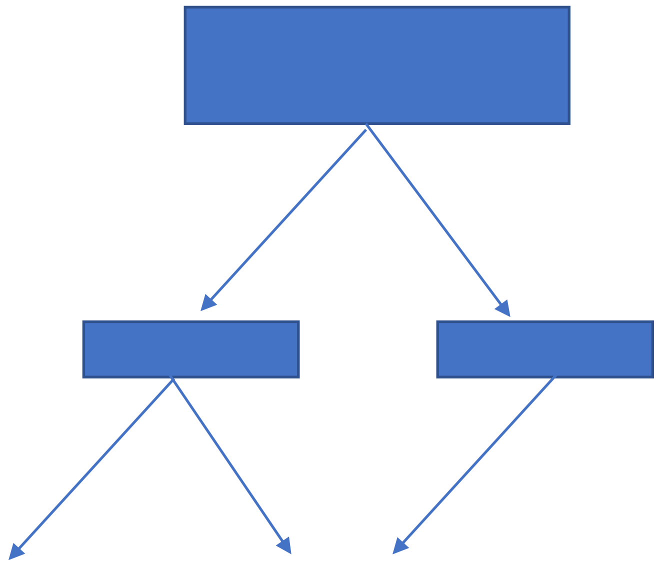
Ideally, we would want to build a 100 trees for the same dataset using the bootstrapping method, where after each step, we get a different tree and a different bootstrapped dataset.





Getting this variety of trees is what makes random forests so effective. Each tree will see a different kind of dataset, hence we have a different model for the same problem.





...

But how do we use Random forest?

Assume we are asked to find the output of this datapoint:

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

- In decision trees, getting an output was fairly simple. There was one tree, you pass in the input, and it gives you an output.
- In random forests however, we typically have a 100 trees or more where each tree will give an output.
- In that case, if we have a hundred trees, then we'll get a 100 outputs, one from each tree in the random forest.

So how do we get the output?

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

So, lets assume we pass this datapoint and find that out of the 100 trees, 80 of them predict that the person has a heart disease, while only 20 of them predict that the person doesn’t have a heart disease.

So, since majority of the trees say that the person has a heart disease, we predict that the person has a heart disease.

This prediction process is called bagging. It is so because, we are first **b**oostrapping and then **agg**regating.

Evaluating Random Forests

While creating a bootstrapped dataset, it is very likely that at each step we don't include a few datapoints from the original dataset for training the decision tree.

For eg, in our case, when we built the first decision tree, we did not use the circled datapoint

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 datapoint is called as the out-of-the-bag data

Evaluating Random Forests

While creating a bootstrapped dataset, it is very likely that at each step we don't include a few datapoints from the original dataset for training the decision tree.

For eg, in our case, when we built the first decision tree, we did not use the circled datapoint

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

We keep track of this datapoint and store its index somewhere.

Since, this datapoint isn't used to train the model, we can test the model with this point itself.

Evaluating Random Forests

While creating a bootstrapped dataset, it is very likely that at each step we don't include a few datapoints from the original dataset for training the decision tree.

For eg, in our case, when we built the first decision tree, we did not use the circled datapoint

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

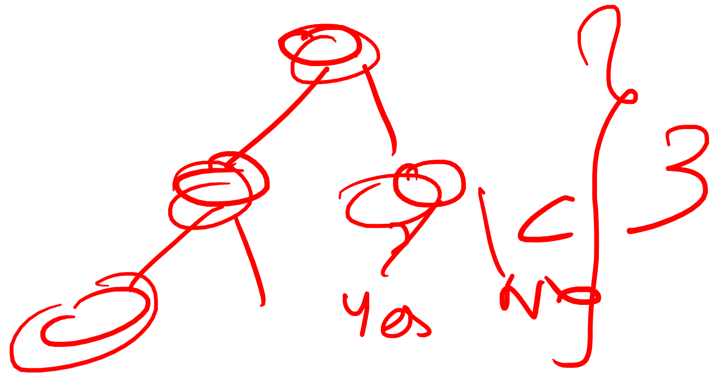
Similarly, for each decision tree we generate a bootstrapped dataset and in each bootstrapped dataset, we ignore a few datapoints from the original dataset.

So, each time, we take the left out datapoints to evaluate the tree and then finally aggregate the results to get final accuracy.

The accuracy of the entire random forest is the total number of times the individual models got the answer right divided by the total number of out-of-bag datapoints.

Validation sets and Hyper-parameters

- Every time we work on developing a Random Forest, we are supposed to manually decide a few values:
  - n_estimators: Total number of trees we want to build
  - max_depth: maximum depth of each tree
  - max_features: The number of features to consider when looking for the best split
    - sqrt : max_features = sqrt (total_number_of_features)
    - float: custom number that you can decide
    - Log2 : max_features = log2 (total_number_of_features)
    - None: max_features = total_number_of_features



Hyperparameters

- The hypermeters are the knobs that we, the Machine Learning Engineers, must use to tune the accuracy of the model.
- However, we aren’t completely on our own. There are some heuristics that we can use to understand better which knobs to use.
- We can train multiple random forests using varying hyperparameters and then check the accuracy of each on a validation set to choose which hyperparameters help model our data better

- We’ll learn how to do this easily in SKLearn!!