Random Forest Introduction

What is ensemble learning ?

Ensemble learning is when you take multiple machine learning algorithms and put them together to create one better machine learning algorithm so that this final machine learning algorithm is leveraging many different other machine learning algorithms.

The algorithms in an Ensemble can be the same or it can be different as well.

Random Forest is one of the Ensemble learnings/method.

“Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction”

Random Forest Introduction

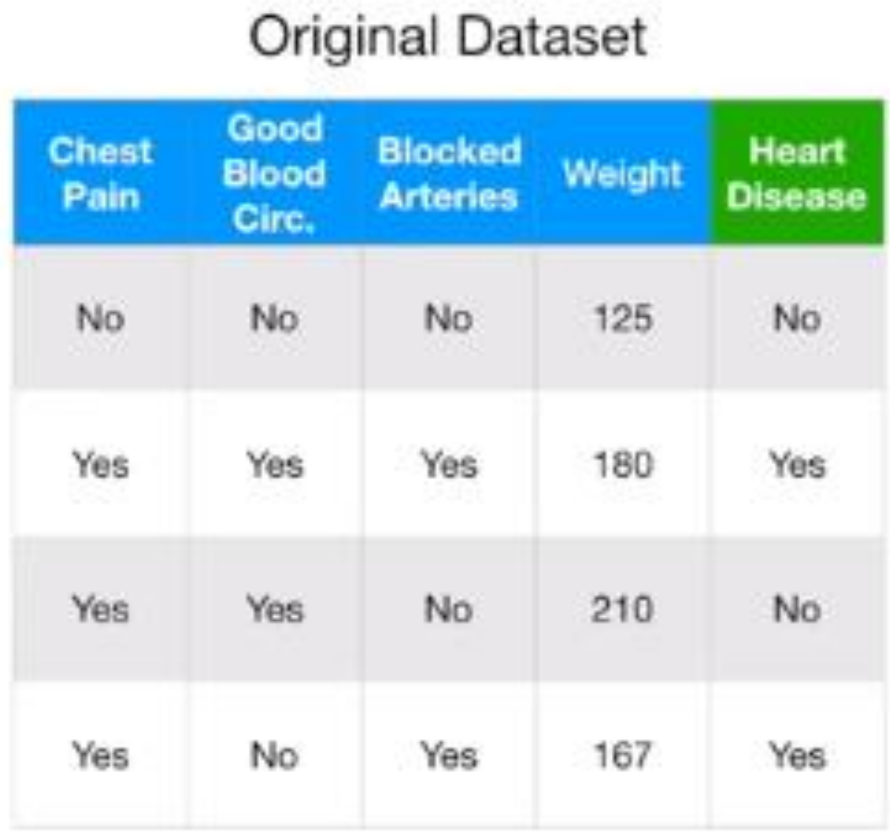
Decision Trees have one aspect that prevents them from being the ideal tool for predictive learning **, namely Inaccuracy.**

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

Good news is that **Random Forest** combines the simplicity of decision tress with flexibility resulting in a vast improvement in accuracy.

Random Forest – How it works ?

Suppose we have the following 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

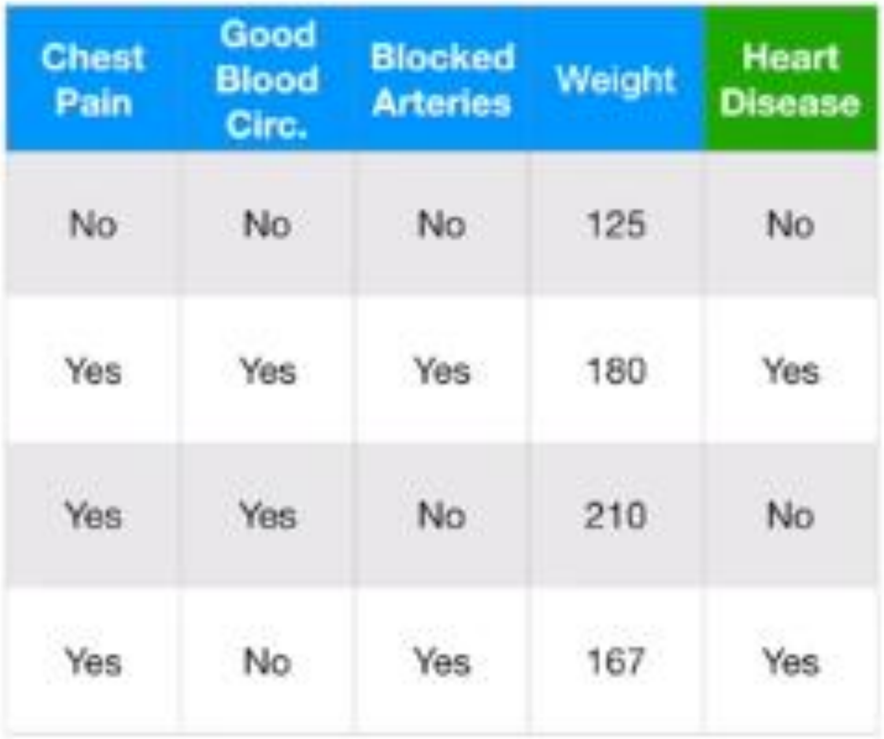
Step 1: Create a Bootstraped Dataset

- To create a bootstrapped dataset that is the same as the original, we just randomnly select samples from the original dataset.
- The important detail is that, one sample is allowed to be picked more than once

Random Forest – How it works ?

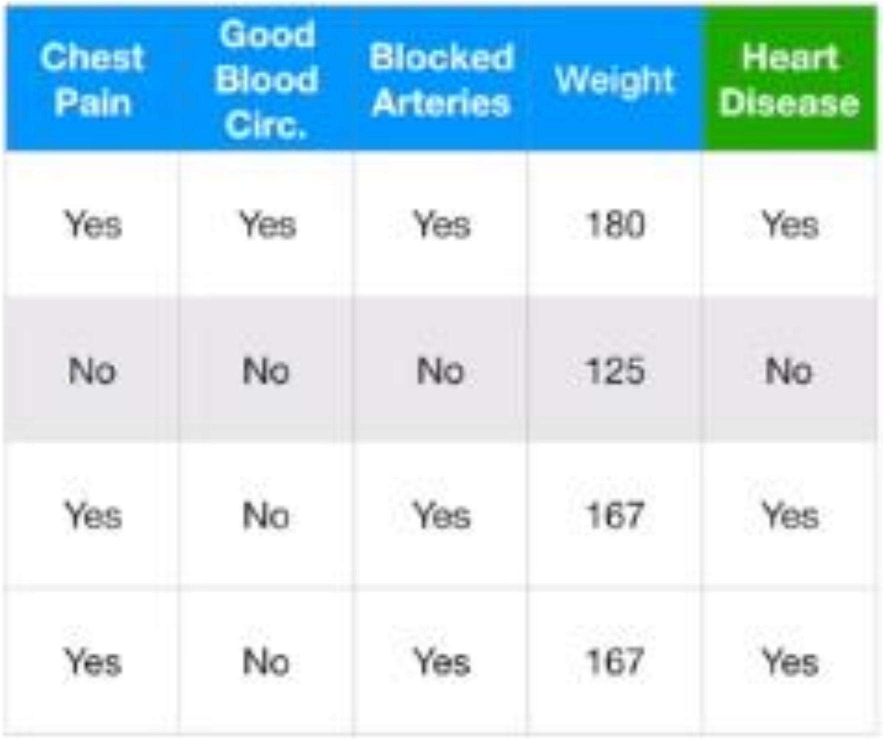
Step 1: Create a Bootstraped Dataset

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

Random Forest – How it works ?

Step 2: Create a decision tree using the bootstraped dataset.
But only use a random subset of variables (features/columns) at each step.

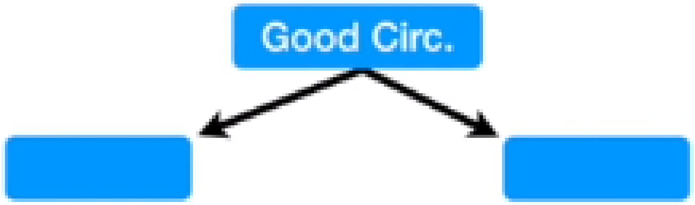
Note: Here we are considering only two features randomly.
We will see how to determine the optimal number of variables to consider shortly.

So for now, lets say we consider **‘Good Blood Circulation’** and **‘Blocked Arteries’** as candidates for the root node.

2.1: Assuming, ‘Good Blood Circulation’ did the best job separating the samples (ID3 Entropy/ Gini), we make it root node.

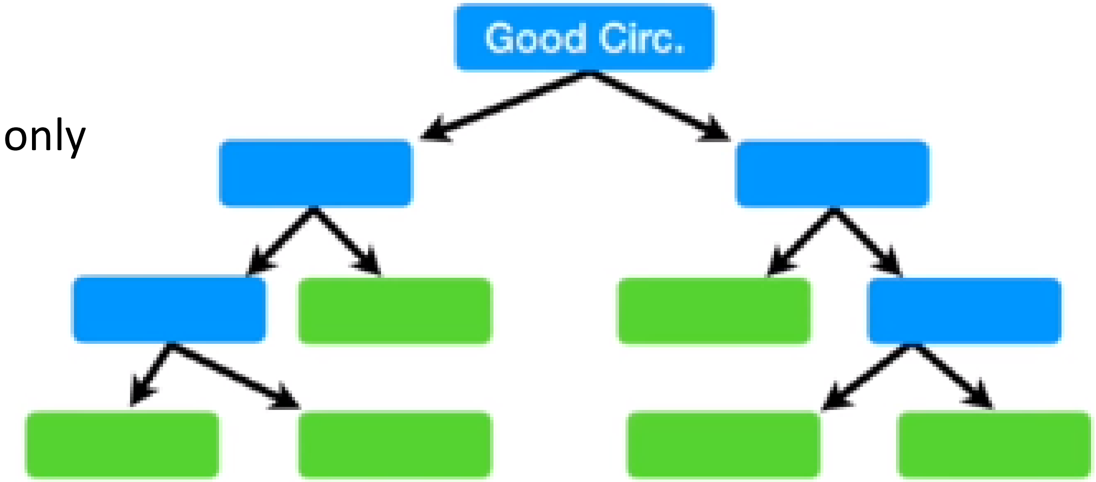
Random Forest – How it works ?

2.1: Assuming, ‘Good Blood Circulation’ did the best job separating the samples (ID3 Entropy/ Gini), we make it root node.



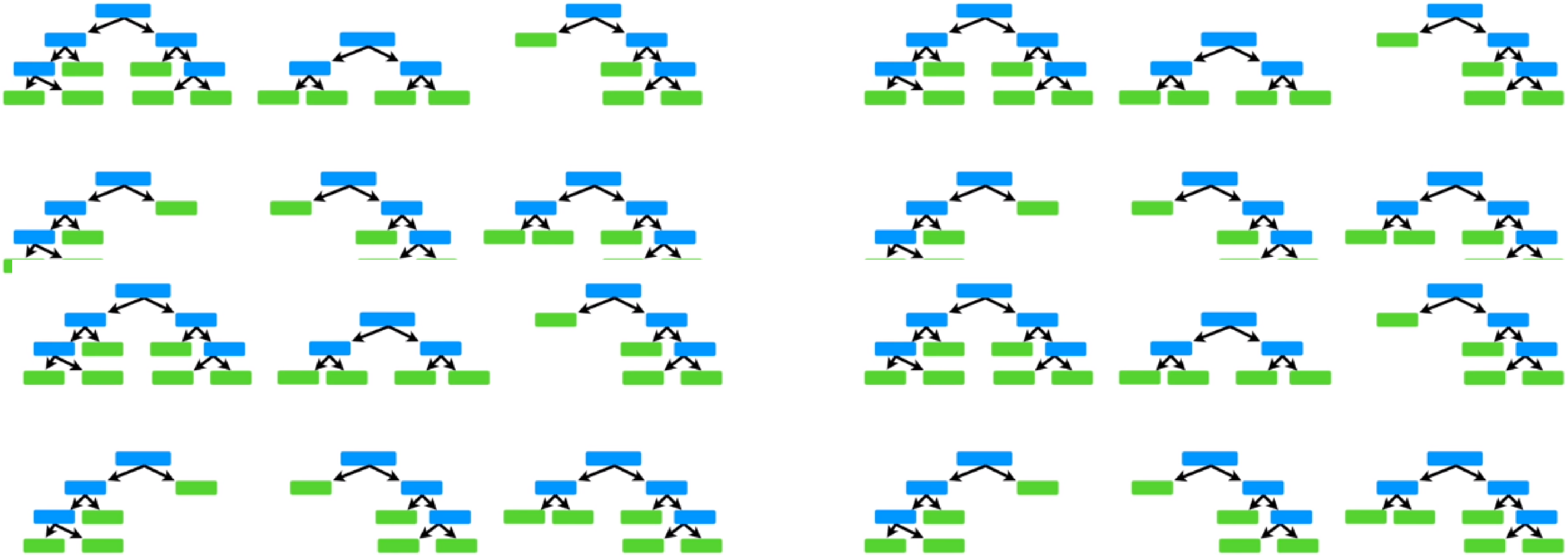
2.2: Just like for the root, we randomly select 2 variables as candidates, instead of all 3 remaining columns for second left node.

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



Random Forest – How it works ?

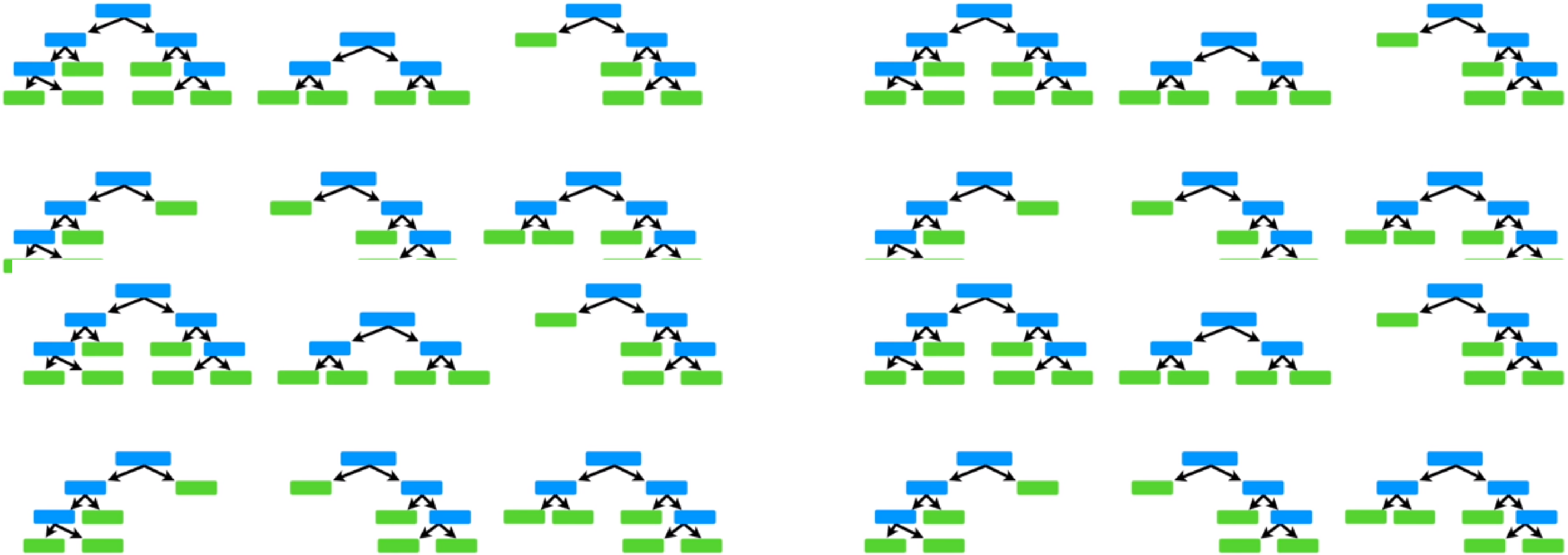
Step 3: Now go back to Step 1 and repeat.
Make new bootstrapped dataset and build a tree considering a subset of variables at each step.
You do this 100s of time.



Random Forest – How it works ?

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

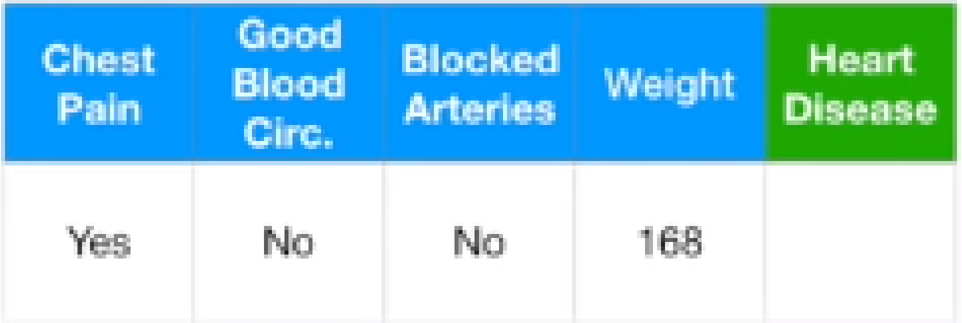
The variety is what makes random forest more effecive than individual decision trees.



Random Forest – How it works ?

How to use the trees for predictions ?

So now we have our forest ready for prediction, lets say we got a new data on which we need to predict if the patient has **Heart Disease** or not.



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

We take data and run it down the first tree in the forest and note the classification (yes/no)
We repeat this for all the trees and note down al the respective observations.

After running the data down all the trees in the random forest, we see which option received more votes.
If **Yes** received most votes, then the patient have heart disease otherwise patient do not have heart disease.

Random Forest – How it works ?

What is Bagging ?
Bootstraping the data using aggragate to make a decision is called Bagging.

Accuracy:
As we allow duplicates in bootstrapping, typically 1/3rd of the data original data does not end up in the bootstrap dataset.

This is called **“Out-Of-Bag-Dataset”**
We run all these out of bag samples down through all the trees in the forest and check if they are labelled correctly.

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.

Random Forest – How it works ?

Out-Of-Bag-Error:

The proportion of Out-Of-Samples incorrectly classified is called as Out-Of-Bag-Error

Summary: What Random Forest does ?
Step 1: Build Random Forest
Step 2: Estimates the accuracy of Random Forest
Step 3: Change the number of variables used per step
Step 4: Repeat from Step 1 a bunch of times.
Step 5: Choose the Random Forest that is most accurate.

Random Forest – Hyperparameters

Below are some of the important hyper parameters of Random Forest:

<u>To increase the Predictive Power:</u>

1. **n_estimators:** It is just the number of trees the algorithm builds before taking the maximum voting or taking averages of predictions. **Higher the number , makes predictions more stable, slows down computation.**

2. **max_features:** It is the maximum number of features Random Forest considers to split a node.

3. **min_sample_leaf:** It is the minimum number of features Random Forest considers to split a node.

Random Forest – Hyperparameters

Below are some of the important hyper parameters of Random Forest:

<u>To increase the Model Speed:</u>

1. **n_jobs**: tells the engine how many processors it is allowed to use. If it has a value of 1, it can only use one processor. A value of "-1" means that there is no limit
2. **random_state:** makes the model's output replicable. The model will always produce the same results when it has a definite value of random_state and if it has been given the same hyperparameters and the same training data.

3. **oob_score:** Also called as **oob (Out Of Bag) sampling.** It is a random forest cross validation method. This tells the accuracy of Random Forest.

Random Forest – Good or Bad

Below are some of important points of Random Forest:

Advantages:

- Can be used for both Classification as well as Regression
- Easy to use and tune algorithm. The number of hyperparameters for tuning are less.
- Resistant to overfitting due to large number of random trees.

Disadvantages:

- Large number of trees make the algorithm slow and inefficient for real-time predictions.
- These algorithms are fast to train, but quite slow to predict.
- More accurate model requires more trees which results in slower model.