A decision tree algorithm starts with a directional graph, where each node of the graph uses a feature (e.g. color) from the data to sort the records into groups (e.g. red, blue, green). An effective node is one that happens to sort the data in a way that strongly predicts the target feature (e.g. 97% of reds are "guilty" and 4% of blues are "guilty"). A great number of paths through this graph can exist. A decision tree algorithm determines the path through the graph resulting in the most predictive decision tree.

A random forest algorithm is a collection of — like it sounds — many trees. The individual decision trees of a random forest are trained on a subset of the records and a subset of the features. The final algorithm is a combined prediction of all the decision trees. A major strength of a random forest algorithm is that the trees complement each other, like a professional team of individuals with diverse sets of skills. If the color feature from the earlier example consistently misdiagnoses 3% of "reds" as "guilty", then this error might be corrected by one of the trees whose subset of features doesn't include color at any of its nodes.

Random forest algorithms generally perform better but are more computationally complex and can occupy a lot of memory.