﻿---

output:

pdf\_document: default

html\_document: default

---

# Random Forest Classification

## Introduction

If you have played the child's game \*20 Questions\* [^71] where the answers are \*Yes/No\* (true/false) then this analogy may help you understand Random Forest classification? Consider for a moment all the questions that could be asked are all branches that you could traverse. This tree of all possible questions is akin to a decision tree. Now imagine a person plays this game tens, hundreds or even thousands of times. After many games, one would learn the \*best\* questions to ask and the quickest way to win. Random Forests are very much like the child's game \*20 Questions\* which is played over and over in the hope of finding the best questions such that you may sort through all the possibilities to win quickly.

[^71]:https://en.wikipedia.org/wiki/Twenty\_Questions

A Random Forest classifier is a meta classifier built on nothing more than a large number of Decision Trees \*grown\* under certain conditions. Similar to a meta-analysis of clinical data, the Random Forest uses a large set of previously constructed Decision Trees processed into an ensemble learning tool.

Let us start with the more basic explanation of a Classification and Regression Tree (CART) learning system. A Decision Tree is a set of nested \*If...Then\* logical questions designed to provide an efficient way of steering a course through a set of narrowing branches to come to an answer or a class. The simplest decision tree has a root node which is the first question which branches or splits to only 2 offspring, daughter nodes or leaves. A leaf is simple the terminal point for the branching tree.

It is common to discuss decision trees in terms of how deep it may be built. Depending on the application, it may be advantageous to build a tree with a very small number of nodes or a fully articulated tree with every item on its leaf allowing the decision-maker to fully differentiate to every observation. For example, a tree which stops?? after 1 or 2 nodes may be produced in the hope of finding a strong classifier. Alternatively, fully defined trees with one item per one leaf may not classify well and may need to be pruned like a wild tree out of control. As you see the tree analogy has its roots in everyday circumstances.

The algorithm that I choose to work with utilizes the ranger software package, which is a fast implementation of Random Forests, particularly suited for high dimensional data.[^72]

[^72]:https://cran.r-project.org/web/packages/ranger/index.html

## Outline of Ranger Random Forest Algorithm

As mentioned earlier, a Random Forest Classifier is a Supervised learning technique. Therefore requiring a labeled pair of observations.

\*\*Data set\*\*: $(X\_1, y\_1), (X\_2 , y\_2), ~. . ., ~(X\_N , y\_N); ~~~y \in \{1, ..., ~C\}$

Where $C$ is the number of classes, $X$ has $m$ observations (rows) and $n$ features (columns), $x\_{ij} \in \Re$.

The Random Forest process starts by choosing a sample from the original data set for a complete run. This process is called Bootstrapping. Bootstrapping uses a random number generator to sample from the set of original observations producing a new subset, i.e. drawing a sample of size N with replacement. The subset is then partitioned into further 2 sets. One set is approximately 66% the size of N and is the training set. The second set, which is 33% of N in size, is used as a testing set later on. Although it is possible to change this proportion in other software packages this value is 'factory set' in Ranger.

Once the training set is produced the splitting process may start. The first step in this process is to determine the number of features which will be used for optimum splitting. Using R, the hyperparameter is called the `mtry`. Using a simple grid type search it has been found that an optimum value of `mtry` is the square root of the number of features in the experiment, i.e. `mtry` = $\sqrt{M}$, $M$ is the number of features.

The splitting process has two additional hyperparameters which can be set by the researcher. These hyperparameters are called `splitrule` and `min.node.size`. The `splitrule` determines which calculation will be used to measure the split purity. In general, there are two methods for calculating the purity of splits for decision trees. One method is the \*Gini coefficient\* while the second common method is the \*entropy coefficient\*.

Gini coefficient:

$$G\_1 ~=~ 1 - \sum\_{k=1}^{n} (X\_k - X\_{k-1}) (Y\_k + Y\_{k-1})$$

Where:

- $X\_k$ is the cumulated proportion of the population variable, for $k$ = 0,..., n, with $X\_0$ = 0, $X\_n$ = 1.

- $Y\_k$ is the cumulated proportion of the income variable, for $k$ = 0,..., n, with $Y\_0$ = 0, $Y\_n$ = 1.

- $Y\_k$ should be indexed in non-decreasing order ($Y\_k > Y\_k - 1$)

entropy coefficient:

$$H(X) ~=~ - \sum\_x P\_x(x) \cdot log P\_x(x)$$

Subsequently, utilizing the training set discussed above a complete Decision Tree is built to the largest extent possible, not employing pruning.

---------------------

https://github.com/imbs-hl/ranger

CART trees

Tree-based and rule-based models are popular modeling tools for many reasons. First, they generate a set of conditions that are highly interpretable and are easy to implement.

Furthermore, these models can effectively handle missing data and implicitly

conduct feature selection, characteristics that are desirable for many real-life

modeling problems.

---

Two well-known weaknesses are (1) model instability (i.e., slight

changes in the data can drastically change the structure of the tree or rules

and, hence, the interpretation) and (2) less-than-optimal predictive performance.

---

If the relationship

between predictors and the response cannot be adequately defined by rectangular subspaces of the predictors, then tree-based or rule-based models will

have larger prediction errors than other kinds of models.

---

Random Forest

ranger

Classification, Regression

e1071, ranger, dplyr

mtry, splitrule, min.node.size

- Trees are fast to learn and very fast for making predictions. They are also often accurate for a broad range of problems and do not require any special preparation for your data.

---------------

# Classification - CART

## Decision Tree Classification in R

See Youtube: Melvin L. [big data](https://youtu.be/JFJIQ0\_2ijg)

\* Decision Trees using rpart library [https://cran.r-project.org/web/packages/rpart/index.html]

\*\*Limitations, considerations and alternatives: oldest

\*\*Used for Regression or Classification (CART)

\*\*Example uses Iris dataset from R.Fisher

\* See5/C5.0 has been designed to analyze substantial databases containing thousands to millions of records and tens to hundreds of numeric, time, date, or nominal fields. See5/C5.0 also takes advantage of computers with up to eight cores in one or more CPUs (including Intel Hyper-Threading) to speed up the analysis.

\* To maximize interpretability, See5/C5.0 classifiers are expressed as decision trees or sets of if-then rules, forms that are generally easier to understand than neural networks.

\* is easy to use (easy to understand) and does not presume any special knowledge of Statistics or Machine Learning