

# Random Forest

```
library(tree)
library(tidyverse)

## Registered S3 method overwritten by 'cli':
##   method      from
##   print.tree tree

## -- Attaching packages ----- tidyverse 1.3.0 --

## v ggplot2 3.2.1    v purrr  0.3.3
## v tibble  2.1.3    v dplyr  0.8.4
## v tidyr   1.0.2    v stringr 1.4.0
## v readr   1.3.1    v forcats 0.4.0

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

## Tree

We only illustrate it via a classification tree. Much of the followings are also true for regression tree.

```
library(kernlab) # for the data spam

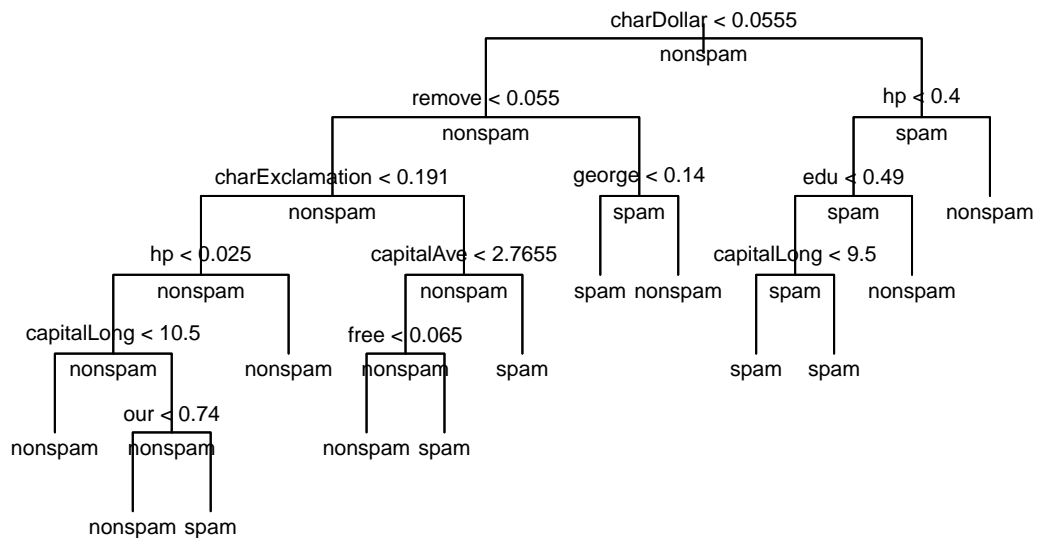
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:purrr':
##
##   cross

## The following object is masked from 'package:ggplot2':
##
##   alpha

data(spam)

tree_spam <- tree(type ~ ., spam)
plot(tree_spam, type = "uniform")
text(tree_spam, pretty = 1, all = TRUE, cex = 0.7)
```



## Random forest

In R, there are packages like `randomForest` and `ranger` to perform random forest. However, we want to implement it from scratch.

The main idea behind random forest is to resampling the dataset by sampling the row with replacement and selecting the columns randomly.

For the `spam` data, suppose we want to predict the class of a random observation. (We randomly draw an observation from the original data and pretend that we do not know its class)

```
set.seed(141)
new_data <- spam %>% sample_n(1)
```

With the whole tree, we could do

```
predict(tree_spam, new_data)
```

```
##      nonsпам      spam
## 1 0.812709 0.187291
```

that gives the probability of nonsпам about 0.81.

However it is known that a single tree is not predictive. We want to use bootstrap to increase the predictivity.

```

r <- 1000 # in practise, we need a larger value, say 10000
m <- 8
n <- nrow(spam)
all_col_names <- names(spam)[1:57] # skip "type"

probs <- map_dbl(seq_len(r), function(i) {
  col_names <- c("type", sample(all_col_names, 8))
  spam_boot <- spam[sample(n, n, replace = TRUE), col_names]
  tree_spam_boot <- tree(type ~ ., spam_boot)
# we only need the probabliiy of spam, because the sum of the two values is always 1
  predict(tree_spam_boot, new_data)[2]
})

```

There are two ways to yield the final predicted class, either by consensus or by averaging probabilities. Either way, we need the baseline,

```
(baseline <- mean(spam$type == "spam"))
```

```
## [1] 0.3940448
```

### Consensus

```
mean(probs > baseline)
```

```
## [1] 0.513
```

Since more than 50% of the trees predicted **spam**, by consensus, the predicted class for the new data is spam.

### By averaging probabilities

```
mean(probs)
```

```
## [1] 0.4475824
```

The average probability across all trees is  $0.45 > \text{baseline}$  so the predicted class is “spam”. For this new data, we have the same prediction using average probability.

In general, it is more stable to use average probability rather than consensus.