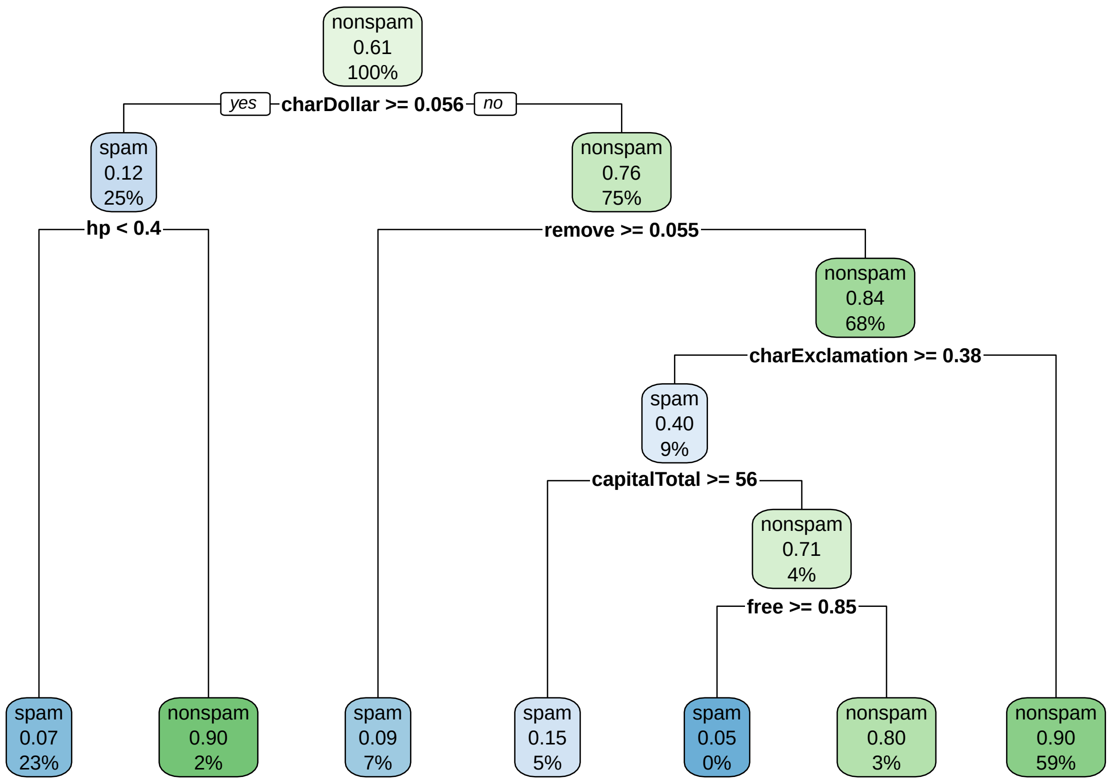
Solution 1: Classifying spam

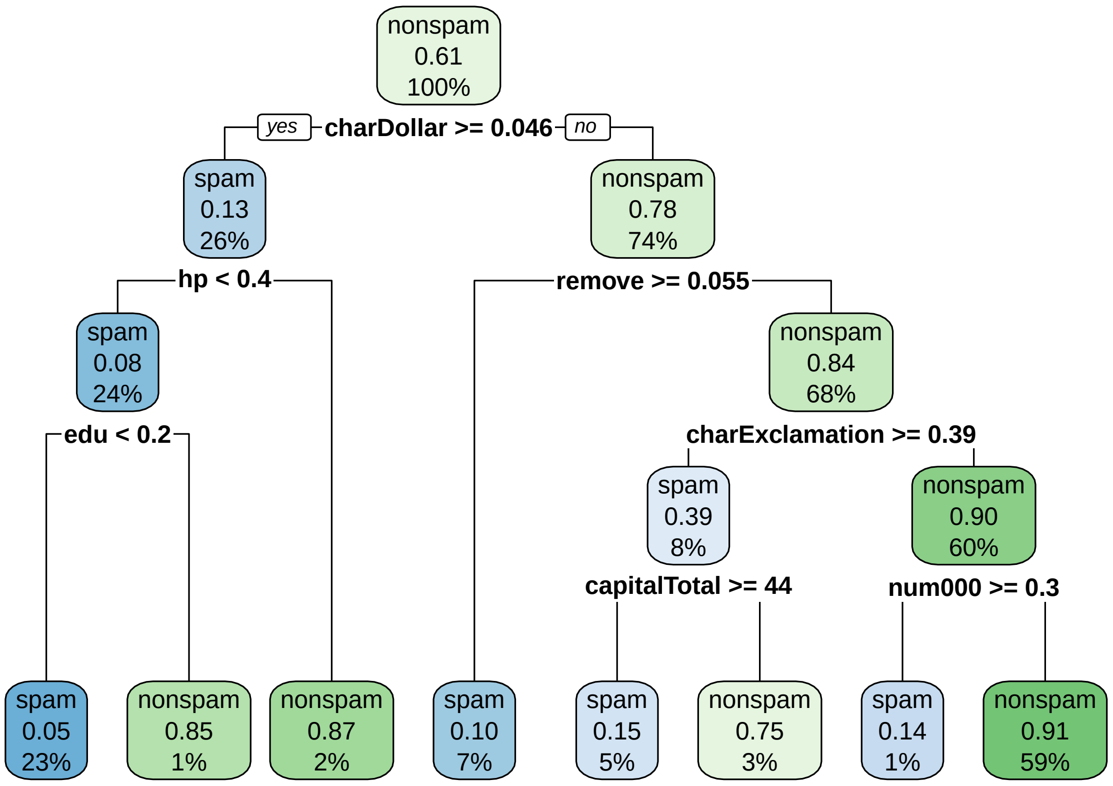
a) The spam data is a binary classification task where the aim is to classify an e-mail as spam or non-spam.

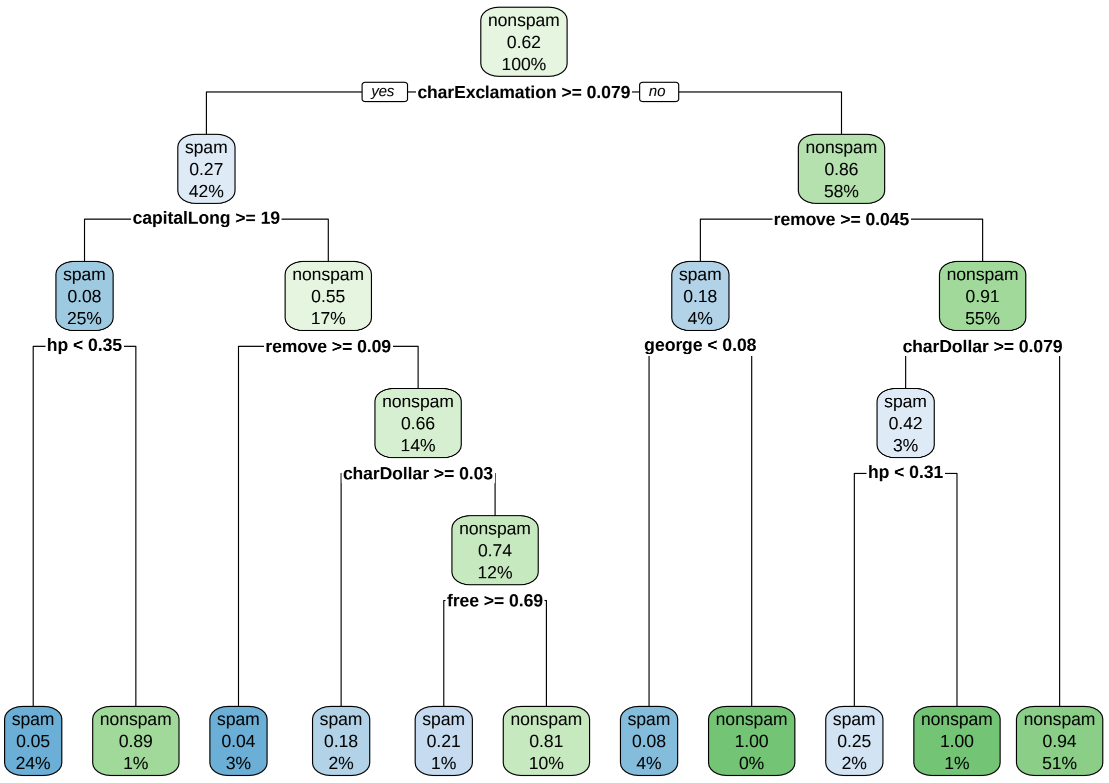
```r
library(mlr3)
tsk("spam")

## <TaskClassif:spam> (4601 x 58)
## * Target: type
## * Properties: twoclass
## * Features (57):
##   - dbl (57): address, addresses, all, business, capitalAve,
##     capitalLong, capitalTotal, charDollar, charExclamation, charHash,
##     charRoundbracket, charSemicolon, charSquarebracket, conference,
##     credit, cs, data, direct, edu, email, font, free, george, hp, hpl,
##     internet, lab, labs, mail, make, meeting, money, num000, num1999,
##     num3d, num415, num650, num85, num857, order, original, our, over,
##     parts, people, pm, project, re, receive, remove, report, table,
##     technology, telnet, will, you, your
```

b)
```r
library(rpart.plot)
## Loading required package:  rpart

task_spam <- tsk("spam")

learner <- lrn("classif.rpart")
learner$train(task_spam)

set.seed(123)
rpart.plot(learner$model, roundint = FALSE)
```



```r
set.seed(456)
subset_1 <- sample.int(task_spam$nrow, size = 0.6 * task_spam$nrow)
set.seed(789)
subset_2 <- sample.int(task_spam$nrow, size = 0.6 * task_spam$nrow)

for (i in list(subset_1, subset_2)) {
  learner$train(task_spam, row_ids = i)
  rpart.plot(learner$model, roundint = FALSE)
}
```





Observation: trees trained on different samples differ considerably in their structure, regarding split variables as well as thresholds (recall, though, that the split candidates are a further source of randomness).

c) i) This is actually quite easy when we recall that the exponential function at an arbitrary input x can be characterized via

$$e^x = \lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n,$$

which already resembles the limit expression we are looking for. Setting x to -1 yields:

$$\lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^n = e^{-1} = \frac{1}{e}.$$

ii)
```r
# generate IDs and bootstrap saples
ids <- seq_len(1000)
bootstrap_samples <- lapply(
  seq_len(1000),
  function(i) sample(ids, size = length(ids), replace = TRUE))

# for each ID, assert whether it is OOB for a given sample (i.e., not in ir),
# and take the relative frequencies across all samples
freqs <- sapply(ids, function(i) {
  mean(sapply(bootstrap_samples, function(j) 1 - (i %in% j)))})

# compute the mean OOB frequencies across all IDs
mean(freqs)

## [1] 0.368051
```

iii)
```r
library(mlr3learners)

learner <- lrn("classif.ranger", "oob.error" = TRUE)
learner$train(tsk("spam"))
learner$model$prediction.error

## [1] 0.0458596
```

d) Variable importance in general measures the contributions of features to a model. One way of computing the variable importance of the j-th variable is based on permuting it for the OOB observations and calculating the mean increase in OOB error this permutation entails.

In order to determine the with the biggest influence on prediction quality, we can choose the k variables with the highest importance score, e.g., for $k = 5$:

```r
library(mlr3filters)

learner <- lrn("classif.ranger", importance = "permutation", "oob.error" = TRUE)
filter <- flt("importance", learner = learner)
filter$calculate(tsk("spam"))
head(as.data.table(filter), 5)

##            feature      score
## 1:     capitalLong 0.04472184
## 2:              hp 0.04171921
## 3: charExclamation 0.04094333
## 4:          remove 0.03724485
## 5:    capitalTotal 0.03410045
```

Solution 2: Decision boundaries

See R code