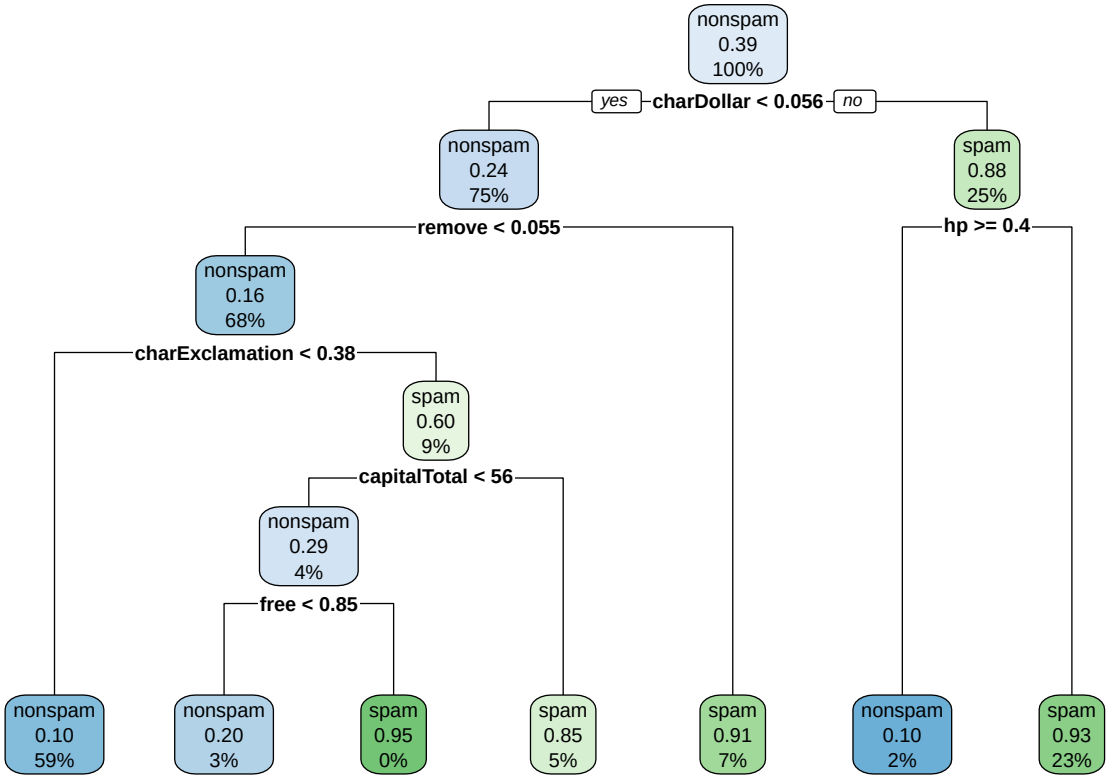
Solution 1:

a) The spam data is a binary classification task where the aim is to classify an email as spam or no-spam.

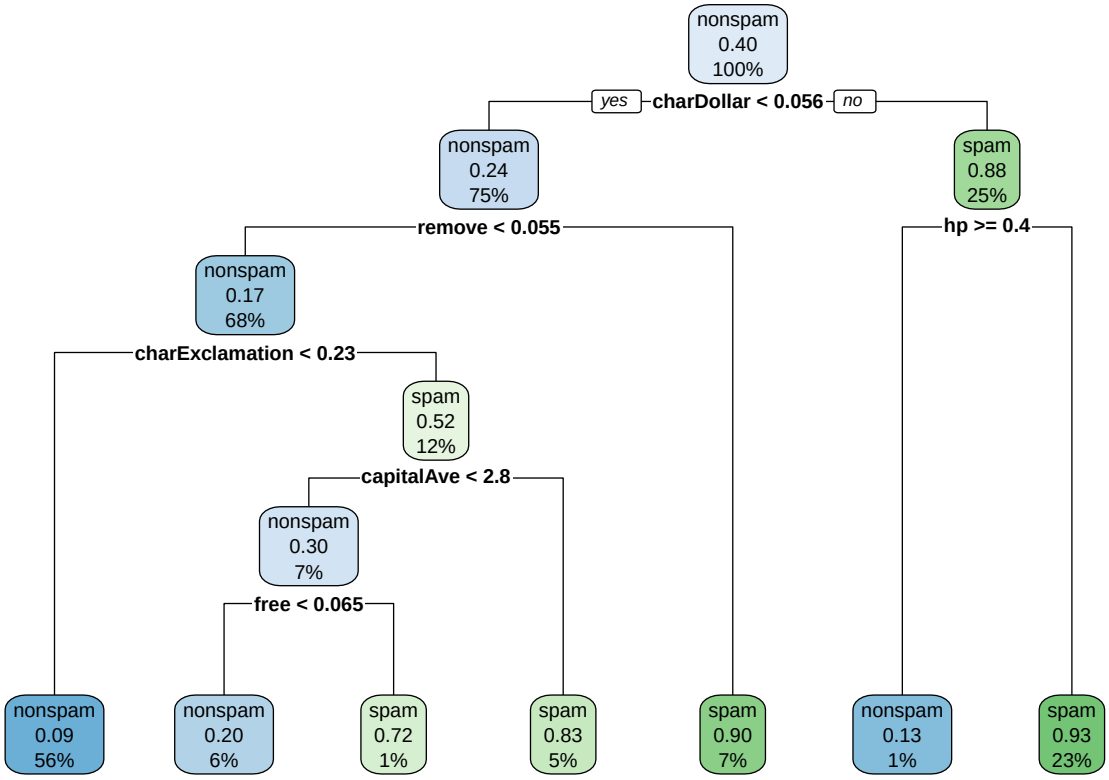
```
library(mlr3)
library(mlr3learners)
library(mlr3filters)

## Error in library(mlr3filters):  there is no package called 'mlr3filters'

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)

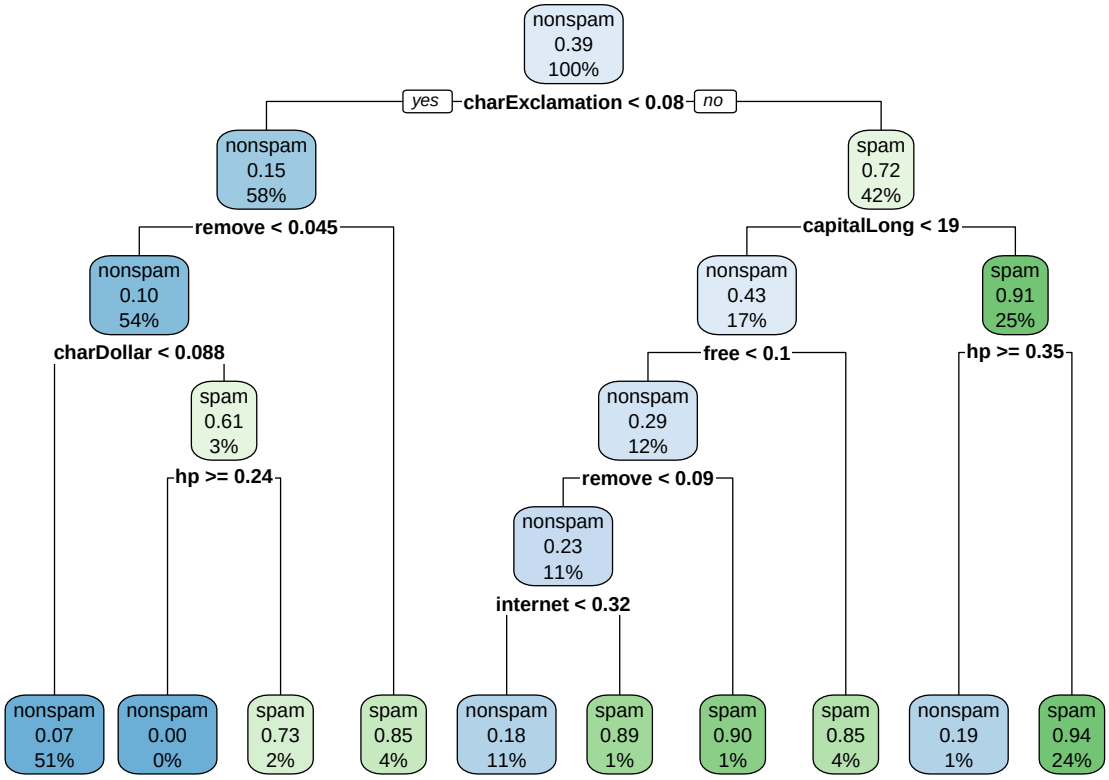
```
library(rpart.plot)
## Loading required package:  rpart

task_spam <- tsk("spam")

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

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




```r
set.seed(42)

subset1 <- sample.int(task_spam$nrow, size = 0.8 * task_spam$nrow)
subset2 <- sample.int(task_spam$nrow, size = 0.8 * task_spam$nrow)

learner$train(task_spam, row_ids = subset1)
rpart.plot(learner$model, roundint=FALSE)
```




```
learner$train(task_spam, row_ids = subset2)
rpart.plot(learner$model, roundint=FALSE)
```



Observation: Trees with different sample find different split points and variables, leading to different trees!

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

model <- learner$model

model$prediction.error

## [1] 0.04542491
```

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 permutations of the OOB observations of the j-th variable, which measures the mean deacrease of the predictive accuracy induced by this permutation. To determine the n variables with the biggest influence on the prediction quality, one can choose the n variables with the highest variable importance based on permutations of the OOB, e.g. for $n = 5$:

```
learner <- lrn("classif.ranger", importance = "permutation", "oob.error" = TRUE)
filter <- flt("importance", learner = learner)

## Error in flt("importance", learner = learner):  could not find function "flt"

filter$calculate(tsk("spam"))

## Error in filter$calculate:  object of type 'closure' is not subsettable

head(as.data.table(filter), 5)

## Error in as.data.frame.default(x, ...):  cannot coerce class '"function"' to a
data.frame
```

Solution 2:
See R code **randomForest_l_2.R**

Solution 3:

- Proceed as follows, when solving manually:
  (a) Split x in two groups using the following split points.
    - (1), $(2, 7, 10, 20)$ (splitpoint 1.5)
    - $(1, 2)$, $(7, 10, 20)$ (splitpoint 4.5)
    - $(1, 2, 7)$, $(10, 20)$ (splitpoint 8.5)
    - $(1, 2, 7, 10)$, (20) (splitpoint 15)

  (b) For each possible split point compute the sum of squares in both groups.

  (c) Use as split point the point that splits both groups best w.r.t. minimizing the sum of squares in both groups.

  Here, we have only one split variable x. A split point t, leads to the following half-spaces:

$$\mathcal{N}_1(t) = \{(x, y) \in \mathcal{N} : x \leq t\} \text{ and } \mathcal{N}_2(t) = \{(x, y) \in \mathcal{N} : x > t\}.$$

  Remember the minimization Problem (here only for one split variable x):

$$\min_t \left(\min_{c_1} \sum_{(x,y)\in\mathcal{N}_1} (y - c_1)^2 + \min_{c_2} \sum_{(x,y)\in\mathcal{N}_2} (y - c_2)^2 \right).$$

  The inner minimization is solved through: $\hat{c}_1 = \bar{y}_1$ and $\hat{c}_2 = \bar{y}_2$
  Which results in:

$$\min_t \left(\sum_{(x,y)\in\mathcal{N}_1} (y - \bar{y}_1)^2 + \sum_{(x,y)\in\mathcal{N}_2} (y - \bar{y}_2)^2 \right).$$

  The sum of squares error of the parent is:

$$Impurity_{parent} = MSE_{parent} = \frac{1}{5}\sum_{i=1}^{5}(y_i - 4.7)^2 = 22.56$$

  Calculate the risk for each split point:

  $x \leq 1.5$

$$\begin{aligned}\mathcal{R}(1, 1.5) &= \frac{1}{5}\text{MSE}_{left} + \frac{4}{5}\text{MSE}_{right} = \\ &= \frac{1}{5} \cdot \frac{1}{1}(1 - 1)^2 + \frac{4}{5} \cdot \frac{1}{4}((1 - 5.625)^2 + (0.5 - 5.625)^2 + (10 - 5.625)^2 + (11 - 5.625)^2) \\ &= 19.1375\end{aligned}$$

  $x \leq 4.5$ $\mathcal{R}(1, 4.5) = 13.43$
  $x \leq 8.5$ $\mathcal{R}(1, 8.5) = 0.13$
  $x \leq 15$ $\mathcal{R}(1, 15) = 12.64$

Minimal empirical risk is obtained by choosing the split point 8.5.

Doing the same for the log-transformation gives:

$x \leq 0.3 \;\; \mathcal{R}(1, 0.3) = 19.14$
$x \leq 1.3 \;\; \mathcal{R}(1, 1.3) = 13.43$
$x \leq 2.1 \;\; \mathcal{R}(1, 2.1) = 0.13$
$x \leq 2.6 \;\; \mathcal{R}(1, 2.6) = 12.64$

Minimal empirical risk is obtained by choosing the split point 2.1.

- Code example:

```
x = c(1,2,7,10,20)
y = c(1,1,0.5,10,11)

calculate_mse <- function (y) mean((y - mean(y))^2)
calculate_total_mse <- function (yleft, yright) {
  num_left <- length(yleft)
  num_right <- length(yright)

  w_mse_left <- num_left / (num_left + num_right) * calculate_mse(yleft)
  w_mse_right <- num_right / (num_left + num_right) * calculate_mse(yright)

  return(w_mse_left + w_mse_right)
}

split <- function(x, y) {
  # try out all unique points as potential split points and ...
  unique_sorted_x <- sort(unique(x))
  split_points <- unique_sorted_x[1:(length(unique_sorted_x) - 1)] +
    0.5 * diff(unique_sorted_x)
  node_mses <- lapply(split_points, function(i) {
    y_left <- y[x <= i]
    y_right <- y[x > i]

    # ... compute SS in both groups
    mse_split <- calculate_total_mse(y_left, y_right)
    print(sprintf("Split at %.1f: empirical Risk = %.2f", i, mse_split))

    return(mse_split)
  })
  # select the split point yielding the maximum impurity reduction
  best <- which.min(node_mses)
  split_points[best]
}

x

## [1]  1  2  7 10 20

split(x, y) # the 3rd observation is the best split point

## [1] "Split at 1.5: empirical Risk = 19.14"
## [1] "Split at 4.5: empirical Risk = 13.43"
## [1] "Split at 8.5: empirical Risk = 0.13"
## [1] "Split at 15.0: empirical Risk = 12.64"
## [1] 8.5

log(x)
```

```
## [1] 0.0000000 0.6931472 1.9459101 2.3025851 2.9957323

split(log(x), y) # also here, the 3rd observation is the best split point

## [1] "Split at 0.3: empirical Risk = 19.14"
## [1] "Split at 1.3: empirical Risk = 13.43"
## [1] "Split at 2.1: empirical Risk = 0.13"
## [1] "Split at 2.6: empirical Risk = 12.64"
## [1] 2.124248
```

Solution 4:

According to the lecture for a target y with target space $\mathcal{Y} = \{1, \dots, g\}$ the target class proportion $\pi_k^{(\mathcal{N})}$ of class $k \in \mathcal{Y}$ in a node can be computed, s.t.

$$\pi_k^{(\mathcal{N})} = \frac{1}{|\mathcal{N}|} \sum_{(x^{(i)}, y^{(i)}) \in \mathcal{N}} [y^{(i)} = k].$$

Now for any $n \in \mathbb{N}$ let $Y^{(1)}, \dots, Y^{(n)}, \hat{Y}^{(1)}, \dots, \hat{Y}^{(n)}$ be i.i.d. random variables, where $Y^{(i)}$ and $\hat{Y}^{(i)}$ are categorically distributed with

$$\mathbb{P}(Y^{(i)} = k | \mathcal{N}) = \mathbb{P}(\hat{Y}^{(i)} = k | \mathcal{N}) = \pi_k^{(\mathcal{N})} \quad \forall i \in \{1, \dots, n\}, \quad k \in \mathcal{Y}.$$

The random variables $Y^{(1)}, \dots, Y^{(n)}$ represent data distributed like the training data[1] of size n and the random variables $\hat{Y}^{(1)}, \dots, \hat{Y}^{(n)}$ the corresponding estimators using the randomizing rule. With these we can define the misclassification rate $\mathsf{err}_{\mathcal{N}}$ of node $\mathcal{N}$ for data distributed like the training data, s.t

$$\mathsf{err}_{\mathcal{N}} = \frac{1}{n} \sum_{i=1}^{n} [Y^{(i)} \neq \hat{Y}^{(i)}].$$

We're interested in the expected misclassification rate $\mathsf{err}_{\mathcal{N}}$ of node $\mathcal{N}$ for data distributed like the training data, i.e.,

$$\begin{aligned}
\mathbb{E}_{Y^{(1)}, \dots, Y^{(n)}, \hat{Y}^{(1)}, \dots, \hat{Y}^{(n)}} (\mathsf{err}_{\mathcal{N}}) &= \mathbb{E}_{Y^{(1)}, \dots, Y^{(n)}, \hat{Y}^{(1)}, \dots, \hat{Y}^{(n)}} \left(\frac{1}{n} \sum_{i=1}^{n} [Y^{(i)} \neq \hat{Y}^{(i)}] \right) \\
&= \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{Y^{(i)}, \hat{Y}^{(i)}} \left([Y^{(i)} \neq \hat{Y}^{(i)}] \right) \\
&= \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{Y^{(i)}} \left(\mathbb{E}_{\hat{Y}^{(i)}} \left([Y^{(i)} \neq \hat{Y}^{(i)}] \right) \right) \\
&= \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{Y^{(i)}} \left(\sum_{k=1}^{g} [Y^{(i)} \neq k] \pi_k^{(\mathcal{N})} \right) \\
&= \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{Y^{(i)}} \left(\sum_{k \in \mathcal{Y} \setminus \{Y^{(i)}\}} \pi_k^{(\mathcal{N})} \right) \\
&= \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{Y^{(i)}} \left(1 - \pi_{Y^{(i)}}^{(\mathcal{N})} \right) \\
&= \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{g} (1 - \pi_k^{(\mathcal{N})}) \pi_k^{(\mathcal{N})} \\
&= \frac{n}{n} \sum_{k=1}^{g} (1 - \pi_k^{(\mathcal{N})}) \pi_k^{(\mathcal{N})} \\
&= 1 - \sum_{k=1}^{g} \left(\pi_k^{(\mathcal{N})} \right)^2.
\end{aligned}$$

This is exactly the Gini-Index which CART uses for splitting the tree.

[1] under the independence assumption