

Solution 1:

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

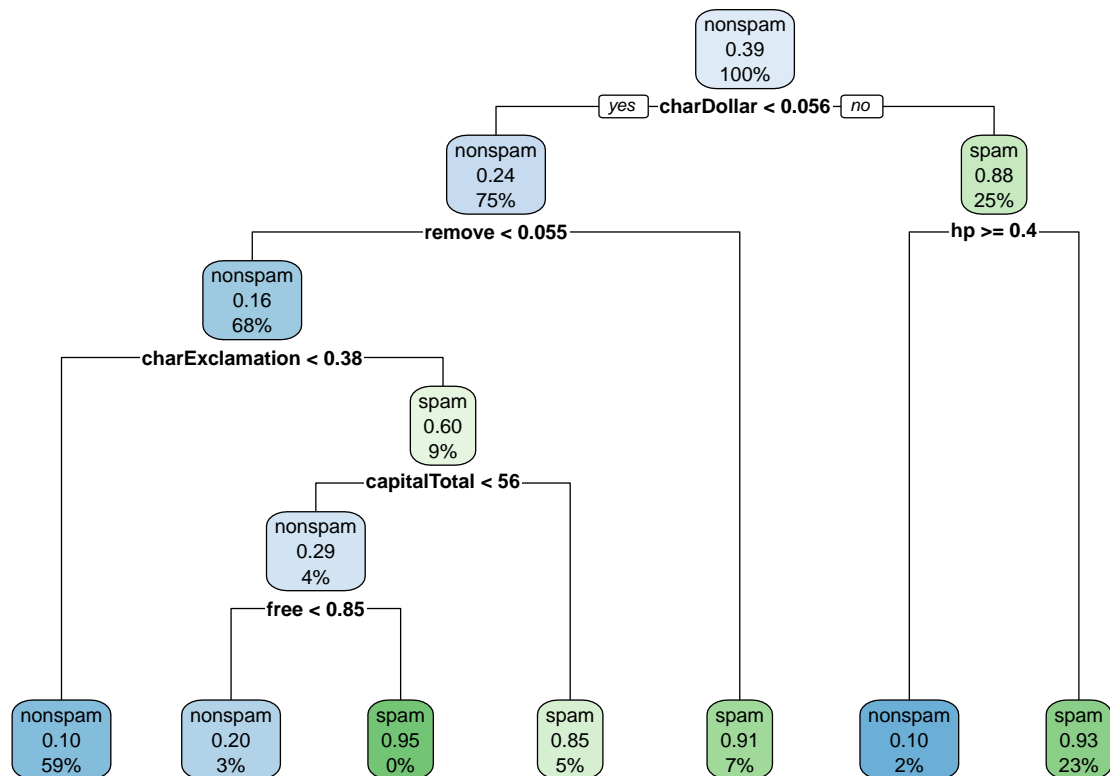
```
library(mlr)
spam.task

## Supervised task: spam-example
## Type: classif
## Target: type
## Observations: 4601
## Features:
##   numerics      factors    ordered functionals
##         57          0         0             0
## Missings: FALSE
## Has weights: FALSE
## Has blocking: FALSE
## Has coordinates: FALSE
## Classes: 2
## nonspam      spam
##      2788      1813
## Positive class: nonspam
```

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

lrn = makeLearner("classif.rpart")
model = train(lrn, spam.task)
mod = getLearnerModel(model)
rpart.plot(mod)

Warning: Cannot retrieve the data used to build the model (so cannot determine
roundint and is.binary for the variables).
To silence this warning:
Call rpart.plot with roundint=FALSE,
or rebuild the rpart model with model=TRUE.
```



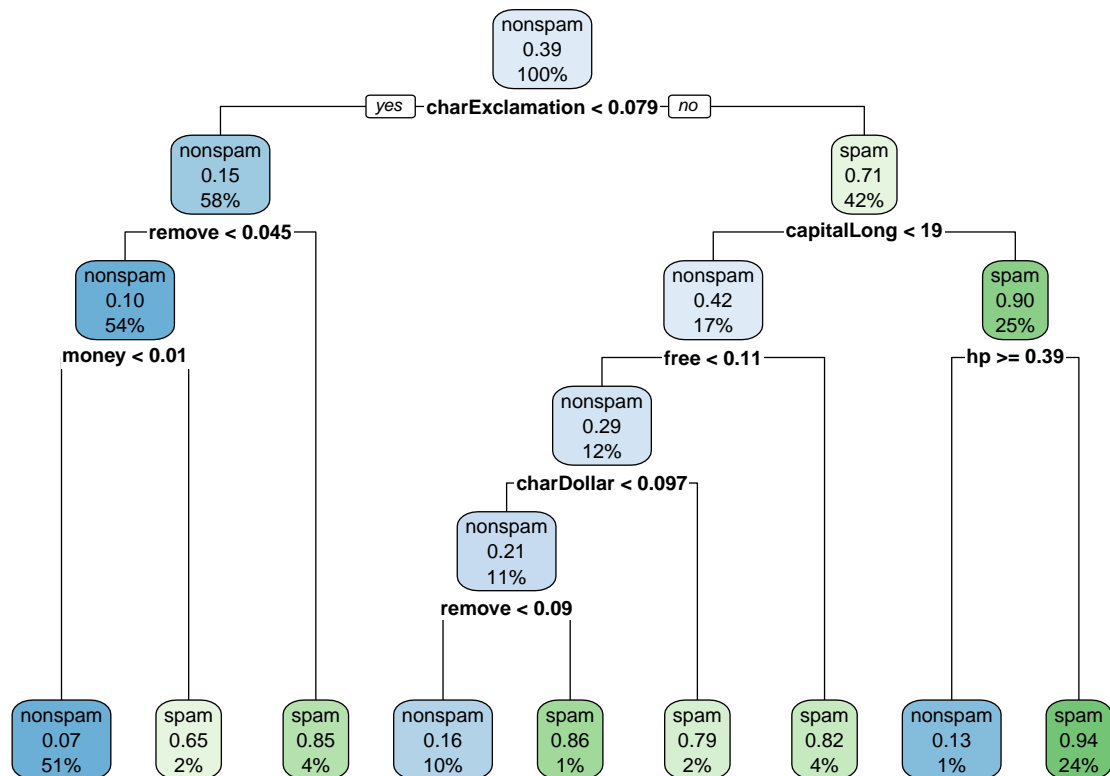
```

set.seed(42)
subset1 = sample.int(getTaskSize(spam.task), size = 0.8 * getTaskSize(spam.task))
subset2 = sample.int(getTaskSize(spam.task), size = 0.8 * getTaskSize(spam.task))

model = train(lrn, spam.task, subset = subset1)
mod = getLearnerModel(model)
rpart.plot(mod)

Warning: Cannot retrieve the data used to build the model (so cannot determine
roundint and is.binary for the variables).
To silence this warning:
Call rpart.plot with roundint=FALSE,
or rebuild the rpart model with model=TRUE.

```



```

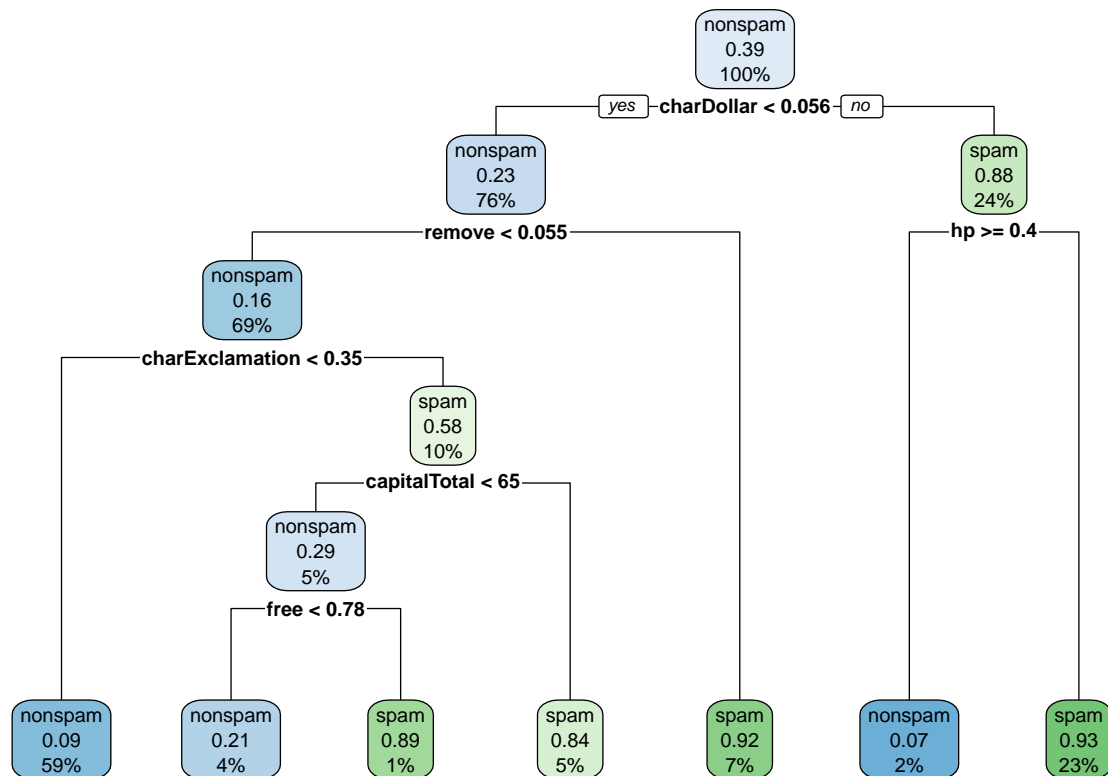
model = train(lrn, spam.task, subset = subset2)
mod = getLearnerModel(model)
rpart.plot(mod)

```

```

Warning: Cannot retrieve the data used to build the model (so cannot determine
roundint and is.binary for the variables).
To silence this warning:
Call rpart.plot with roundint=FALSE,
or rebuild the rpart model with model=TRUE.

```



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

```

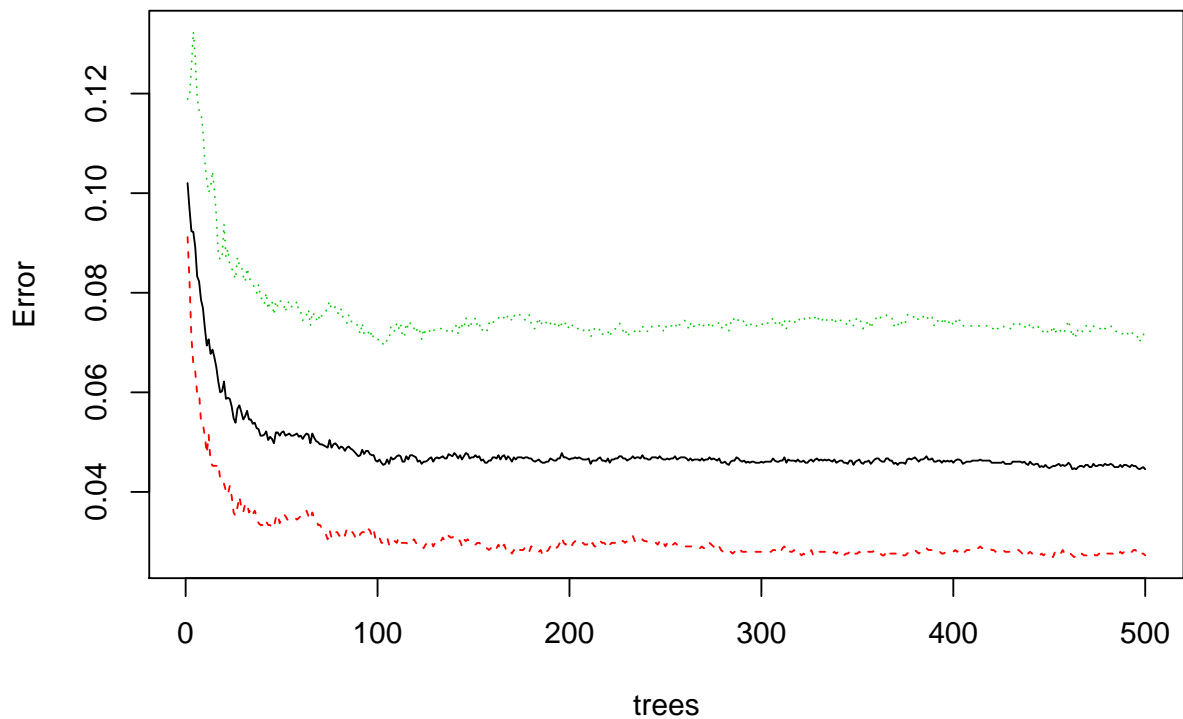
c) lrn = makeLearner("classif.randomForest")
model = train(lrn, spam.task)
mod = getLearnerModel(model)
mod

##
Call:
randomForest(formula = f, data = data, classwt = classwt, cutoff = cutoff)
##
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 7
##
OOB estimate of error rate: 4.46%
Confusion matrix:
nonsпам spam class.error
nonsпам 2712 76 0.02726
spam 129 1684 0.07115

plot(mod)

```

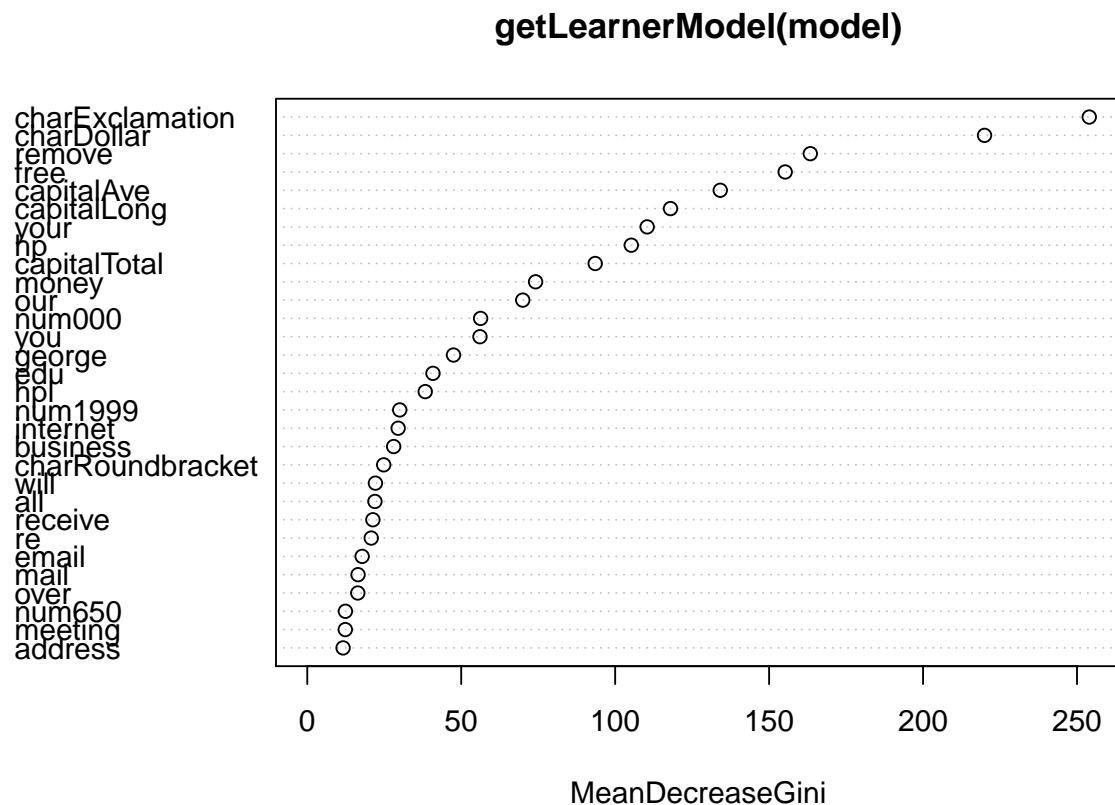
## mod



```
d) imp = getFeatureImportance(model)
 sort(imp$res, decreasing = TRUE)
```

```
charExclamation charDollar remove free capitalAve capitalLong your
1 254 220 163.4 155.2 134.1 118 110.4
hp capitalTotal money our num000 you george edu hpl num1999
1 105.2 93.52 74.15 69.98 56.3 56.09 47.5 40.82 38.31 30.03
internet business charRoundbracket will all receive re email mail
1 29.51 28.06 24.82 22.13 21.98 21.31 20.77 17.82 16.49
over num650 meeting address charSemicolon order credit make labs
1 16.41 12.38 12.36 11.63 10.85 9.045 8.255 8.01 7.917
charHash num85 people technology pm data charSquarebracket font
1 7.689 7.663 7.415 6.834 6.589 5.838 5.443 5.09
project report lab telnet original addresses conference direct cs
1 4.693 4.506 4.175 3.96 3.63 2.94 2.724 2.449 2.025
num415 num3d num857 parts table
1 1.689 1.686 1.625 0.9221 0.5594
```

```
as alternative, the randomForest package provides a plotting function
randomForest::varImpPlot(getLearnerModel(model))
```



### Solution 2:

See R code **randomForest\_1.2.R**

### Solution 3:

a) Proceed as follows:

- (i) 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)
- (ii) For each possible split point compute the sum of squares in both groups.
- (iii) 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:

$$x \leq 1.5$$

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

$$x \leq 4.5 \quad \mathcal{R}(1, 4.5) = 13.43$$

$$x \leq 8.5 \quad \mathcal{R}(1, 8.5) = 0.13$$

$$x \leq 15 \quad \mathcal{R}(1, 15) = 12.64$$

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

Doing the same for the log-transformation gives:

$$x \leq 0.3 \quad \mathcal{R}(1, 0.3) = 19.14$$

$$x \leq 1.3 \quad \mathcal{R}(1, 1.3) = 13.43$$

$$x \leq 2.1 \quad \mathcal{R}(1, 2.1) = 0.13$$

$$x \leq 2.6 \quad \mathcal{R}(1, 2.6) = 12.64$$

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

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

calculateMSE = function (y) mean((y - mean(y))^2)
calculateTotalMSE = function (yleft, yright) {
 n_left = length(yleft)
 n_right = length(yright)

 w_mse_left = n_left / (n_left + n_right) * calculateMSE(yleft)
 w_mse_right = n_right / (n_left + n_right) * calculateMSE(yright)

 return(w_mse_left + w_mse_right)
}

split = function(x, y) {
```

```

try out all points as potential split points and ...
split_points = 0.5 * diff(sort(x)) + x[-length(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 = calculateTotalMSE(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] 0.0000 0.6931 1.9459 2.3026 2.9957

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] 2.124

```

#### Solution 4:

The fractions of the classes  $k = 1, \dots, g$  in node  $\mathcal{N}$  of a decision tree are  $p(1|\mathcal{N}), \dots, p(g|\mathcal{N})$ . Assume we replace the classification rule in node  $t$

$$\hat{k}|\mathcal{N} = \arg \max_k p(k|\mathcal{N})$$

with a randomizing rule, in which we draw the classes in one node from their estimated probabilities.

Derive an estimator for the misclassification rate in node  $\mathcal{N}$ . What do you (hopefully) recognize?

**Solution:**



- For the target space  $\mathcal{Y} = \{1, \dots, g\}$  we assume the distribution  $P_Y$ . The distribution  $P_{\hat{Y}}$  of our classifier  $\hat{Y} = h(x)$  is defined as the individual class frequencies in a node  $\mathcal{N}$  (estimated probability, that an object of class  $k$  is in node  $\mathcal{N}$ ):

$$P(\hat{Y} = k \mid \mathcal{N}) = p(k|\mathcal{N})$$

- As estimate of the target distribution  $P_Y$  we use the distribution of the classifier  $h(x)$  which we assume to be independent of  $P_{\hat{Y}}$ :

$$P_Y \stackrel{\text{ind.}}{\sim} P_{\hat{Y}}$$

- The individual error rate of wrongly predicting a true label  $k$  in node  $\mathcal{N}$  ( $\text{err}_{k|\mathcal{N}}$ ) can be written as probability that  $Y = k$  and  $\hat{Y} \neq k$ :

$$\begin{aligned} P(\text{err}_{k|\mathcal{N}}) &= P(Y = k, \hat{Y} \neq k \mid \mathcal{N}) \\ &= P(Y = k \mid \mathcal{N})P(\hat{Y} \neq k \mid \mathcal{N}) \\ &= p(k|\mathcal{N})(1 - p(k|\mathcal{N})) \end{aligned}$$

- The error rate is the combination of all individual error rates:

$$\text{err}_{\mathcal{N}} = \bigcup_{k=1}^g \text{err}_{k|\mathcal{N}}$$

- Finally, we are interested in the probability of the error rate  $\text{err}_{\mathcal{N}}$  as estimator for the missclassification rate:

$$\begin{aligned} P(\text{err}_{\mathcal{N}}) &= P\left(\bigcup_{k=1}^g \text{err}_{k|\mathcal{N}}\right) \\ &= \sum_{k=1}^g P(\text{err}_{k|\mathcal{N}}) \\ &= \sum_{k=1}^g p(k|\mathcal{N})(1 - p(k|\mathcal{N})) \\ &= \sum_{k=1}^g p(k|\mathcal{N}) - \sum_{k=1}^g p(k|\mathcal{N})^2 \\ &= 1 - \sum_{k=1}^g p(k|\mathcal{N})^2 \end{aligned}$$

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

#### Solution 5:

$f(x) = \frac{1}{B} \sum_{b=1}^B f_b(x)$  is the bagging estimator based on  $B$  bootstrap samples. Then we can easily calculate:

$$\begin{aligned} \text{Var}(f(x)) &= \frac{1}{B^2} \left( \sum_{b=1}^B \text{Var}(f_b(x)) + \sum_{i \neq j}^B \text{Cov}(f_i(x), f_j(x)) \right) \\ &= \frac{1}{B^2} (B\sigma^2 + (B^2 - B)\rho\sigma^2) \\ &= \frac{1}{B}\sigma^2 + \rho\sigma^2 - \frac{1}{B}\rho\sigma^2 \\ &= \rho\sigma^2 + \frac{\sigma^2}{B}(1 - \rho) \end{aligned}$$

In the first line the rules for variance of a non-independent sum of random variables is used. All other steps are trivial.