Exercise Collection – Random Forests

Contents

Lecture exercises

Exercise 1: random forests vs CART

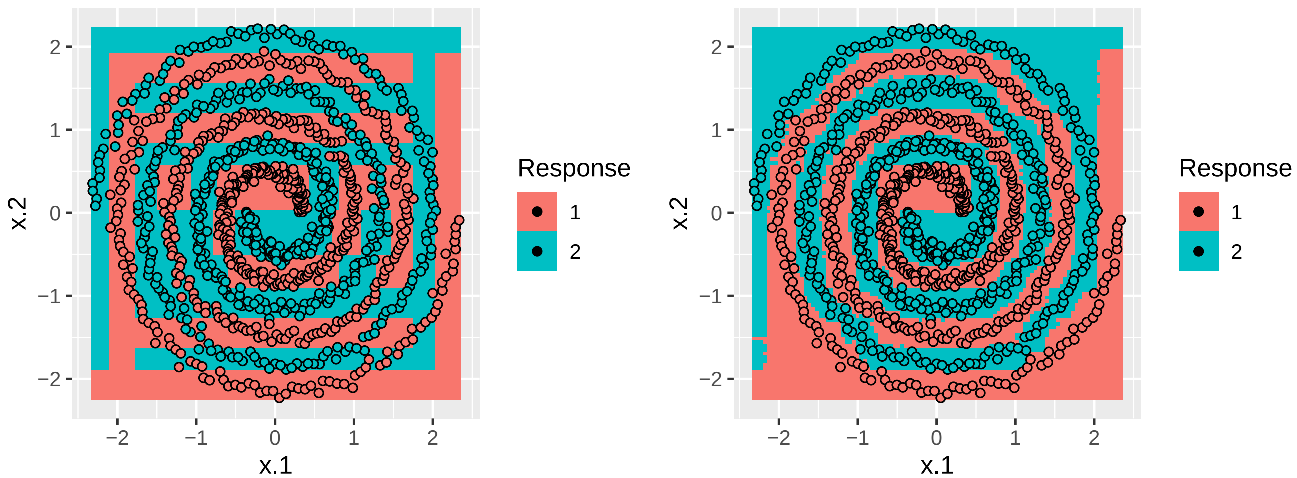
Try to find or simulate (at least) one 2 dimensional classification dataset as an example in which random forests can separate both classes well but CART is problematic. Hint: Have a look at the `mlbench` package.

Solution 1:

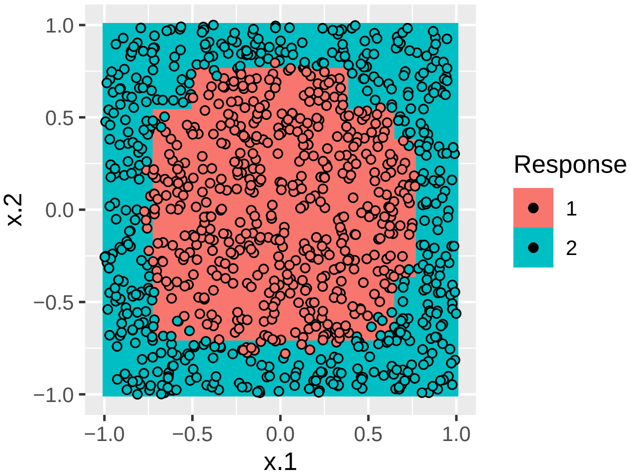
Try to find or simulate (at least) one 2 dimensional classification dataset as an example in which random forests can separate both classes well but CART is problematic. Hint: Have a look at the `mlbench` package.

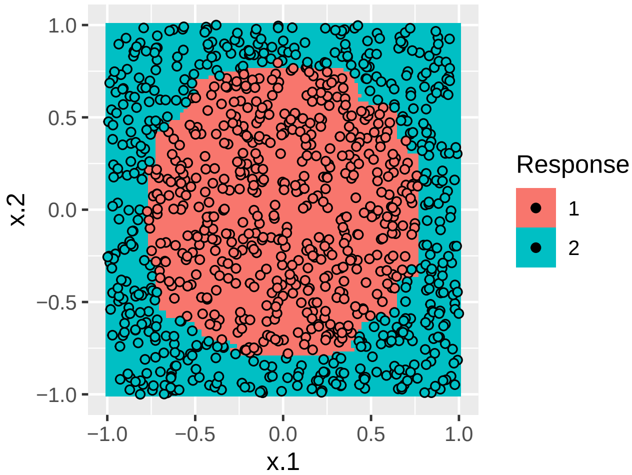
```
library(mlr3)
library(mlr3learners)
library(mlr3viz)
library(mlbench)
library(ggplot2)
library(gridExtra)

set.seed(123)

# create learners

rp = mlr3::lrn("classif.rpart")
rf = mlr3::lrn("classif.ranger")
```

```r
# create two example data and corresponding tasks for mlr3

data.spirals = data.table::as.data.table(
  mlbench.spirals(n = 1000, cycles = 3, sd = 0.05))
task.spirals = mlr3::TaskClassif$new(
  "spirals",
  backend = data.spirals,
  target = "classes")
data.circle = data.table::as.data.table(mlbench.circle(1000, d = 2))
task.circle = mlr3::TaskClassif$new(
  "circle",
  backend = data.circle,
  target = "classes")

# show how learners perform

spirals1 = mlr3viz::plot_learner_prediction(rp, task.spirals) +
  guides(shape = FALSE)

## INFO  [15:00:29.781] [mlr3]  Applying learner 'classif.rpart' on task 'spirals' (iter 1/1)

spirals2 = mlr3viz::plot_learner_prediction(rf, task.spirals) +
  guides(shape = FALSE)

## INFO  [15:00:30.124] [mlr3]  Applying learner 'classif.ranger' on task 'spirals' (iter 1/1)

grid.arrange(spirals1, spirals2, ncol = 2)
```




```r
circle1 = mlr3viz::plot_learner_prediction(rp, task.circle) +
  guides(shape = FALSE)

## INFO  [15:00:31.919] [mlr3]  Applying learner 'classif.rpart' on task 'circle' (iter 1/1)

circle2 = mlr3viz::plot_learner_prediction(rf, task.circle) +
  guides(shape = FALSE)

## INFO  [15:00:32.022] [mlr3]  Applying learner 'classif.ranger' on task 'circle' (iter 1/1)

grid.arrange(circle1, circle2, ncol = 2)
```





Exercise 2: decision boundaries

Generate an artificial dataset with the function call `mlbench.spirals(n = 500, sd = 0.1)`. (The function `mlbench.spirals` is part of the `mlbench` package.) Visualize the decision boundaries of a random forest using the `classif.ranger` learner from `mlr3learners`. Create plots with `plot_learner_prediction` from `mlr3viz` for an increasing number of trees. (Start with `num.trees = 1`) Explain what you see.

Solution 2:

See R code

Exercise 3: random forest implementations

Compare two implementations of random forests. One from the package `randomForest` and one from the package `ranger`. Compare them on some datasets (**hint:** use `benchmark()`) and measure the test error as well as computation time. Don't use too small datasets or your results will be way to noisy to see meaningful differences.

Solution 3:

See R code

Exercise 4: spam classification

a) Take a look at the spam dataset (?mlr3::mlr_tasks_spam). Shortly describe what kind of classification problem this is and access the corresponding task predefined in mlr3.

b) Use a decision tree to predict spam. Try refitting with different samples. How stable are the trees?

Hint: Use rpart.plot() from the package rpart.plot to vizualize the trees. (You can access the model of a learner by its class attribute model)

c) Use the random forest learner classif.ranger to fit the model and state the oob-error.

d) Your boss wants to know which variables have the biggest influence on the prediction quality. Explain your approach in words as well as code.

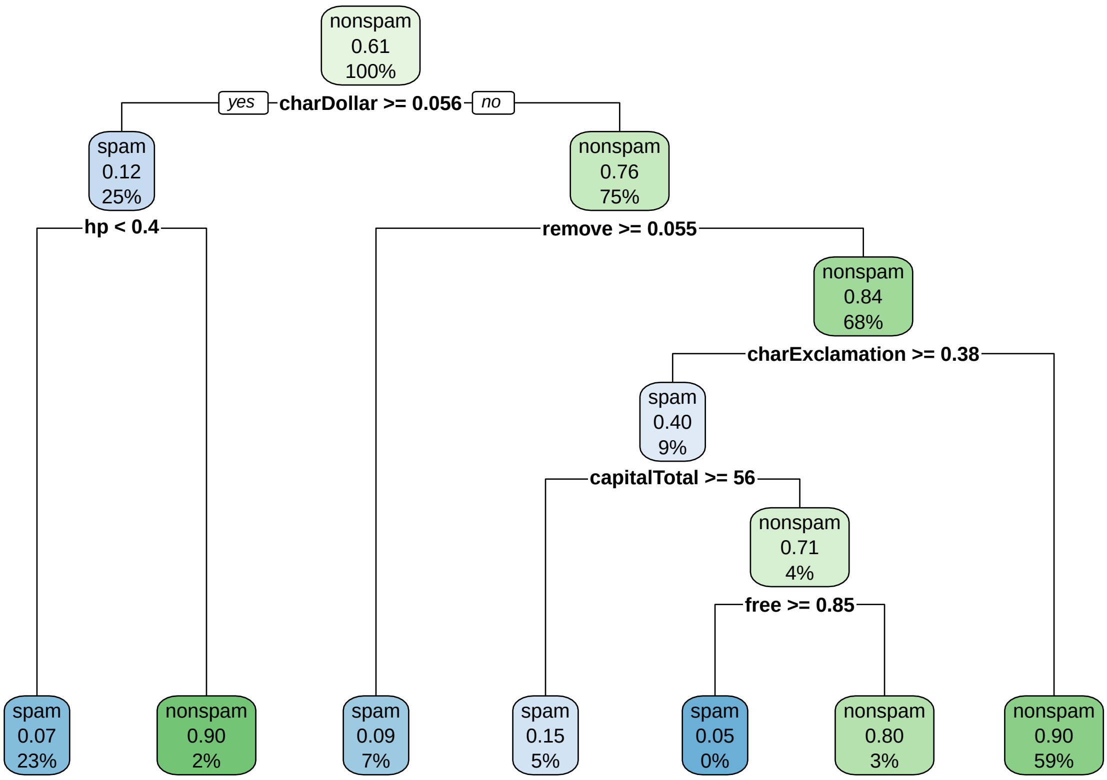
Hint: use an adequate variable importance filter as described in https://mlr3filters.mlr-org.com/#variable-importance-filters.

Solution 4:

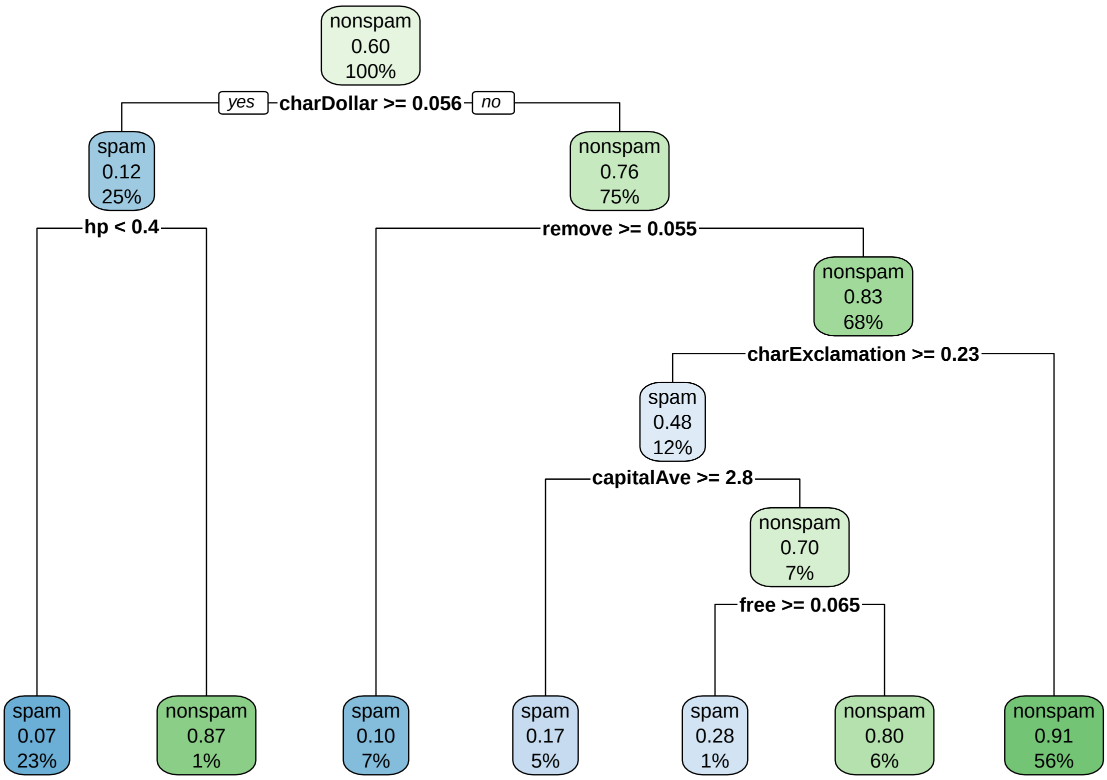
a) The spam data is a binary classification task where the aim is to classify an email as spam or no-spam.

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

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
```

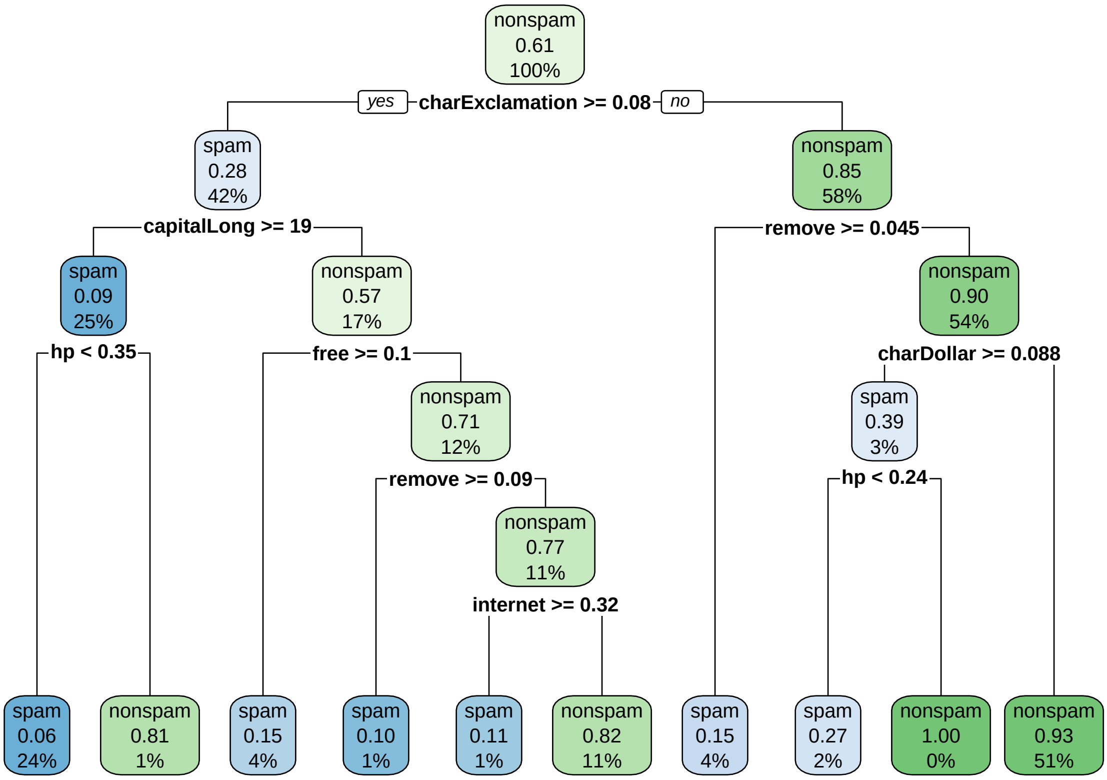
b)
```
library(rpart.plot)
## Loading required package:  rpart

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)
```




```r
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)
filter$calculate(tsk("spam"))
head(as.data.table(filter), 5)

##             feature      score
## 1:      capitalLong 0.04644338
## 2:               hp 0.04125252
## 3: charExclamation 0.03977957
## 4:           remove 0.03827180
## 5:       capitalAve 0.03424298
```

Further exercises

Exercise 5: WS2020/21, first, question 2

The table below shows $\mathcal{D} = \left(\left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(n)}, y^{(n)}\right)\right)$, a data set with $n = 5$ observations of a continuous target variable y and a continuous, 1-dimensional feature variable $\mathbf{x}$. In the following, we aim at predicting y with a machine learning model that takes $\mathbf{x}$ as input.

ID	$\mathbf{x}$	y
1	1.0	3.1
2	5.2	0.5
3	2.7	1.7
4	1.1	4.5
5	1.5	2.7

(a) We train a random forest with L2 loss $L(y, f(\mathbf{x})) = 0.5(y - f(\mathbf{x}))^2$ and `num.trees` = 3 trees. The results of the training and all estimated split points and predicted labels can be found in the R script `rf.R`. To prevent differing numbers due to technical reasons, you see the output of `ranger::treeInfo()` in the following table. Predict the label y for a new observation $\mathbf{x}_* = 2$ with the random forest as given in the table below. State the entire manual calculation, i.e., the entire path of the observation through the trees in detail. (You are allowed to use R to cross-check your solution, but we will only grade your manual computations – for full points, you have to describe your calculations thoroughly.)

(b) Compute the proximities of the 5 training observations, using the random forest. In `rf.R` you see the skeleton of a function `get_prox_matrix()`. Complete this function and apply it to the predictions of the individual trees of the random forest, which are precomputed in the R script for you and stored in the matrix `pred_mat`. Print the results. This is an R question. As a solution, hand in a completed version of `rf.R`. No hand-written solution is allowed here.

(c) Use the proximities matrix given below for outlier detection: Which observations are the most likely candidates for being an outlier and why? State the IDs of the respective observations.

Solution 5:

(a)

nodeID	leftChild	rightChild	splitvarID	splitvarName	splitval	terminal	prediction
0	1	2	0	x	1.30	FALSE	NA
1	3	4	0	x	1.05	FALSE	NA
2	NA	NA	NA	NA	NA	TRUE	2.7
3	NA	NA	NA	NA	NA	TRUE	3.1
4	NA	NA	NA	NA	NA	TRUE	4.5

nodeID	leftChild	rightChild	splitvarID	splitvarName	splitval	terminal	prediction
0	1	2	0	x	1.30	FALSE	NA
1	3	4	0	x	1.05	FALSE	NA
2	5	6	0	x	2.10	FALSE	NA
3	NA	NA	NA	NA	NA	TRUE	3.1
4	NA	NA	NA	NA	NA	TRUE	4.5
5	NA	NA	NA	NA	NA	TRUE	2.7
6	NA	NA	NA	NA	NA	TRUE	1.7

nodeID	leftChild	rightChild	splitvarID	splitvarName	splitval	terminal	prediction
0	1	2	0	x	1.30	FALSE	NA
1	3	4	0	x	1.05	FALSE	NA
2	NA	NA	NA	NA	NA	TRUE	2.7
3	NA	NA	NA	NA	NA	TRUE	3.1
4	NA	NA	NA	NA	NA	TRUE	4.5

- Tree 1:
  - Split 1: left child since $2 < 3.15$
  - End node: Prediction 4.5
- Tree 2:
  - Split 1: left child since $2 < 2.1$
  - Split 2: right child since $2 > 1.3$
  - End node: Prediction 2.7
- Tree 3:
  - Split 1: right child since $2 > 1.9$
  - Split 2: left child since $2 < 3.95$
  - End node: Prediction 1.7

  Final prediction is the mean of the 3 values: 2.97

(b) Model solution somewhere in exam folder?

(c)

	1	2	3	4	5
1	NA	0.00	0.00	0	0.00
2	0	NA	1.00	0	0.67
3	0	1.00	NA	0	0.67
4	0	0.00	0.00	NA	0.00
5	0	0.67	0.67	0	NA

Exercise 6: WS2020/21, second, question 2

The table below shows $\mathcal{D} = \left(\left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(n)}, y^{(n)}\right)\right)$, a data set with $n = 10$ observations of a binary target variable `PlayTennis` and two binary feature variables `Temperature` and `Weather`. In the following, we aim at predicting `PlayTennis` with a machine learning model that takes `Temperature` and `Weather` as input.

ID	1	2	3	4	5	6	7	8	9	10
Temperature	cool	cool	cool	hot	hot	cool	hot	cool	cool	hot
Weather	rain	rain	sunny	sunny	sunny	rain	rain	sunny	sunny	sunny
PlayTennis	no	no	yes	no	yes	no	yes	yes	yes	yes

(a) We train a random forest with `num.trees` = 3 trees. The results of the training and all estimated split points and predicted labels can be found in the R script `snd_rf.R`. To prevent differing numbers due to technical reasons, you see the output of `ranger::treeInfo()` in the following table. Predict the label `PlayTennis` for a new observation $\mathbf{x}_* = (cool, rain)^\top$ with the random forest as given in the table below. State the entire manual calculation, i.e., the entire path of the observation through the trees in detail. (You are allowed to use R to cross-check your solution, but we will only grade your manual computations – for full points, you have to describe your calculations thoroughly.)

Solution 6:

(a)

nodeID	leftChild	rightChild	splitvarID	splitvarName	splitval	terminal	prediction
0	1	2	0	temperature	1.5	FALSE	NA
1	NA	NA	NA	NA	NA	TRUE	no
2	NA	NA	NA	NA	NA	TRUE	yes

nodeID	leftChild	rightChild	splitvarID	splitvarName	splitval	terminal	prediction
0	1	2	0	temperature	1.5	FALSE	NA
1	NA	NA	NA	NA	NA	TRUE	yes
2	NA	NA	NA	NA	NA	TRUE	yes

nodeID	leftChild	rightChild	splitvarID	splitvarName	splitval	terminal	prediction
0	1	2	0	temperature	1.5	FALSE	NA
1	3	4	1	weather	1.5	FALSE	NA
2	5	6	1	weather	1.5	FALSE	NA
3	NA	NA	NA	NA	NA	TRUE	no
4	NA	NA	NA	NA	NA	TRUE	yes
5	NA	NA	NA	NA	NA	TRUE	yes
6	NA	NA	NA	NA	NA	TRUE	yes

In the R code you can see that cool is class 1, hot is class 2, for temperature and that rain is class 1, sun is class 2 for weather.

- Tree 1:
  - nodeID 0: left child since cool $= 1 < 1.5 \Rightarrow$ nodeID $= 1$
  - nodeID 1: left child since rain $= 1 < 1.5 \Rightarrow$ nodeID $= 3$
  - nodeID 3 = End node: Prediction 'no'
- Tree 2:
  - nodeID 0: left child since rain $= 1 < 1.5 \Rightarrow$ nodeID $= 1$
  - nodeID 1: left child since cool $= 1 < 1.5 \Rightarrow$ nodeID $= 3$
  - nodeID 3 = End node: Prediction 'no'
- Tree 3:
  - nodeID 0: left child since cool $= 1 < 1.5 \Rightarrow$ nodeID $= 1$
  - nodeID 1 = End node: Prediction 'no'

  Final prediction is the majority vote of the 3 values: 'no'

Ideas & exercises from other sources