Exercise 1:

Shortly answer the following questions:

(a) What is the difference between inner and outer loss?

(b) Which model is more likely to overfit the training data:

- k-NN with 1 or with 10 neighbours?
- Logistic regression with 10 or 20 features?
- LDA or QDA?

(c) Which of the following methods yield an unbiased generalization error estimate?
Performance estimation ...

- on training data
- on test data
- on training and test data combined
- using cross validation
- using subsampling

(d) Which problem does resampling of training and test data solve?

(e) Which problem does nested resampling solve?

Exercise 2:

The Satellite dataset consists of pixels in 3x3 neighbourhoods in a satellite image, where each pixel is described by 4 spectral values, and the classification label of the central pixel. (for further information see `?Satellite`) We fit a k-NN model to predict the class of the middle pixel. The performance is evaluated with the mmce.
Look at the following R code and output: The performance is estimated in different ways: using training data, test data and then with cross validation. How do the estimates differ and why? Which one should be used?

```r
library(mlr3)
library(mlr3learners)
library(mlbench)

data(Satellite)
satellite_task <-
  TaskClassif$new(id = "satellite_task",
                  backend = Satellite,
                  target = "classes")
knn_learner <- lrn("classif.kknn", k = 3)

# Train and test subsets:
set.seed(42)
train_indices <-
  sample.int(nrow(Satellite), size = 0.8 * nrow(Satellite))
test_indices <- setdiff(1:nrow(Satellite), train_indices)
```

```r
# Training data performance estimate
knn_learner$train(task = satellite_task, row_ids = train_indices)
pred <-
  knn_learner$predict(task = satellite_task, row_ids = train_indices)
pred$score()

## classif.ce
##          0

# Test data performance estimate
pred <-
  knn_learner$predict(task = satellite_task, row_ids = test_indices)
pred$score()

## classif.ce
## 0.09246309

# CV performance estimate
rdesc <- rsmp("cv", folds = 10)
res <- resample(satellite_task, knn_learner, rdesc)

## INFO  [15:19:36.440] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 1/10)
## INFO  [15:19:36.623] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 4/10)
## INFO  [15:19:36.755] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 7/10)
## INFO  [15:19:36.879] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 5/10)
## INFO  [15:19:37.013] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 3/10)
## INFO  [15:19:37.142] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 9/10)
## INFO  [15:19:37.382] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 6/10)
## INFO  [15:19:37.505] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 10/10)
## INFO  [15:19:37.634] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 2/10)
## INFO  [15:19:37.761] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 8/10)

res$score()

##                   task        task_id                  learner   learner_id
##  1: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  2: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  3: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  4: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  5: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  6: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  7: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  8: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  9: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
## 10: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##              resampling resampling_id iteration              prediction
##  1: <ResamplingCV[19]>            cv         1 <PredictionClassif[19]>
##  2: <ResamplingCV[19]>            cv         2 <PredictionClassif[19]>
##  3: <ResamplingCV[19]>            cv         3 <PredictionClassif[19]>
##  4: <ResamplingCV[19]>            cv         4 <PredictionClassif[19]>
##  5: <ResamplingCV[19]>            cv         5 <PredictionClassif[19]>
##  6: <ResamplingCV[19]>            cv         6 <PredictionClassif[19]>
##  7: <ResamplingCV[19]>            cv         7 <PredictionClassif[19]>
##  8: <ResamplingCV[19]>            cv         8 <PredictionClassif[19]>
##  9: <ResamplingCV[19]>            cv         9 <PredictionClassif[19]>
## 10: <ResamplingCV[19]>            cv        10 <PredictionClassif[19]>
##     classif.ce
```

```
##  1: 0.09627329
##  2: 0.07919255
##  3: 0.08540373
##  4: 0.09472050
##  5: 0.10559006
##  6: 0.08553655
##  7: 0.08864697
##  8: 0.10108865
##  9: 0.08864697
## 10: 0.10419907

res$aggregate()

## classif.ce
## 0.09292983
```

Exercise 3:

In preparing this course you already learned about `mlr3`. If you need to refresh your knowledge you can find help at `https://mlr3book.mlr-org.com/` under 'Basics'.

a) How many performance measures do you already know? Try to explain some of them. How can you see which of them are available in `mlr3`?

b) Use the `boston_housing` regression task from `mlr3` and split the data into 50 % training data and 50 % test data while training and predicting (i. e., use the `row_ids` argument of the `train` and `predict` function). Fit a prediction model (e. g. k-NN) to the training set and make predictions for the test set.

c) Compare the performance on training and test data. Use the `score` function.

d) Now use different observations (but still 50 % of them) for the training set. How does this affect the predictions and the error estimates of the test data?

e) Use 10 fold cross-validation to estimate the performance. Hint: Use the mlr functions `rsmp` and `resample`.