Exercise 1:

Shortly answer the following questions:

- What is the difference between inner and outer loss?
- Which model is more likely to overfit the training data:
  - knn with 1 or with 10 neighbours?
  - logistic regression with 10 or 20 features?
  - lda or qda?
- 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
- Which problem does resampling of training and test data solve?
- Which problem does nested resampling solve?

Exercise 2:

We fit a knn model to predict the pixel classes in the Satellite dataset. The performance is evaluated with the kappa coefficient, which measures the agreement between two scorers: The higher the kappa, the better the agreement (max 1) between predicted and actual classes.
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?

```
library(mlr)
library(mlbench)

data(Satellite)
Satellite.task = makeClassifTask(data = Satellite, target = "classes")
knn.learner = makeLearner("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)

# Training data performance estimate
mod = train(knn.learner, task = Satellite.task, subset = train_indices)
pred = predict(mod, task = Satellite.task, subset = train_indices)
performance(pred, measure = mlr::kappa)
```

```
## kappa
##     1

# Test data performance estimate
pred = predict(mod, task = Satellite.task, subset = test_indices)
performance(pred, measure = mlr::kappa)

##  kappa
## 0.8709

# CV performance estimate
rdesc = makeResampleDesc(method = "CV", iters = 10)
res = resample(knn.learner, Satellite.task, rdesc, measures = list(mlr::kappa))
res$measures.test

##    iter  kappa
## 1     1 0.8671
## 2     2 0.8737
## 3     3 0.8936
## 4     4 0.8929
## 5     5 0.9256
## 6     6 0.8867
## 7     7 0.8799
## 8     8 0.8659
## 9     9 0.8770
## 10   10 0.8815

mean(res$measures.test$kappa)

## [1] 0.8844
```

Exercise 3:

In preparing this course you already learned about `mlr`. If you need to refresh your knowledge you can find help at `https://mlr.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 `mlr`?

b) Use the `bh.task` regression task from `mlr` and split the data into 50 % training data and 50 % test data while training and predicting (i. e., use the `subset` argument of the `train` and `predict` function). Fit a prediction model (e. g. knn) to the training set and make predictions for the test set.

c) Compare the performance on training and test data. Use the `performance` 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 `makeResampleDesc` and `resample`.

Exercise 4:

ID	Actual Class	Score
1	0	0.33
2	0	0.27
3	1	0.11
4	1	0.38
5	1	0.17
6	0	0.63
7	1	0.62
8	1	0.33
9	0	0.15
10	0	0.57

a) Create a confusion matrix assuming the decision boundary at 0.5.

b) Calculate: precision, sensitivity, negative predictive value, specificity, accuracy, error rate and F-measure.

c) Draw the ROC curve.

d) Calculate the AUC.