Solution 1:

- The inner loss is the loss that is optimized directly by the machine learning model. The outer loss is the loss (or performance measurement) used to evaluate the model.
- Which model is more likely to overfit the training data:
  - k-NN with 1 or with 10 neighbors? **1 neighbor**, because it's an exact memorization of training data.
  - Logistic regression with 10 or 20 features? **20 features**, because the more features, the more coefficients the learner estimates. More coefficients mean more degrees of freedom, which make overfitting more likely.
  - LDA or QDA? **QDA**, because it has more parameters to possibly overfit the data. LDA is more likely to underfit more complex relationships.
- Which of the following methods yield an unbiased generalization error estimate? Performance estimation ...
  - on training data: **Biased, too optimistic**
  - on test data: **Unbiased / Biased, too pessimistic** (Test data is *not included* / *included* in the final model)
  - on training and test data combined: **Biased, too optimistic** (But a little bit less than only using training data).
  - using cross validation: **Biased, too pessimistic** (The higher the ratio of folds / number of observation, the smaller the pessimistic bias)
  - using subsampling: **Biased, too pessimistic** (The smaller the subsampling rate, the larger the pessimistic bias)
- Resampling strategies solve the problem that comes from the randomness of the training and test data split: Error estimation using a single split has a high variance. Resampling estimates are more robust because they average over different splits.
- Nested resampling solves the problem of simultaneously conducting tuning/model selection and performance estimation. When we use the performance estimates from the same data that were used for model selection (as done in simple, not-nested resampling), the final error estimate is too optimistic.

Solution 2:

- The training performance is too optimistic (mmce of 0), because the mmce is higher on new data.
- The test performance is unbiased (if the final model is only trained on the training data), but it depends on the split, as can be seen in the CV folds: Each CV fold represents a training test split and the mmce measure varies between folds.
- The CV estimate averages over the different splits and gives an slightly pessimistic, more robust estimate.
- The CV estimate is preferable over the other two, but more computationally expensive.

Solution 3:

a) Each loss function we have learned so far to fit the model (inner loss) can also be used as performance measure (outer loss).

For classification:

- 0-1 loss (= mean misclassification error),
- Logistic loss (bernoulli loss), ...

For regression:

- L_2-loss (= mean squared error),
- L_1-loss (= mean absolute error), ...

To get a list of all measures you can use `mlr_measures`.

b)
```
# look at the task
task <- tsk("boston_housing")
task

## <TaskRegr:boston_housing> (506 x 19)
## * Target: medv
## * Properties: -
## * Features (18):
##   - dbl (13): age, b, cmedv, crim, dis, indus, lat, lon, lstat, nox,
##     ptratio, rm, zn
##   - int (3): rad, tax, tract
##   - fct (2): chas, town

n <- task$nrow

# select index vectors to subset the data randomly
set.seed(123)
train_ind <- sample(seq_len(n), 0.5*n)
test_ind <- setdiff(seq_len(n), train_ind)

# specify learner
learner <- lrn("regr.kknn", k = 3)

# train model to the training set
learner$train(task, row_ids = train_ind)

# predict on the test set
pred <- learner$predict(task, row_ids = test_ind)
pred

## <PredictionRegr> for 253 observations:
##     row_ids truth response
##           1  24.0 23.22445
##           2  21.6 19.98830
##           3  34.7 34.97419
## ---
##         504  23.9 22.22775
##         505  22.0 21.76531
##         506  11.9 20.88958
```

c)
```
# predict on the train set
pred_train <- learner$predict(task, row_ids = train_ind)
pred_train$score(list(msr("regr.mse"), msr("regr.mae")))

##  regr.mse  regr.mae
## 1.2322560 0.7564092

# predict on the test set
pred_test <- learner$predict(task, row_ids = test_ind)
pred_test$score(list(msr("regr.mse"), msr("regr.mae")))

##  regr.mse  regr.mae
## 12.424958  2.596332
```

Unsurprisingly the model performs better on the training data (smaller loss) then on the test data.

d)
```
# select different index vectors to subset the data randomly
set.seed(321)
train_ind <- sample(seq_len(n), 0.5*n)
test_ind <- setdiff(seq_len(n), train_ind)

# specify learner
learner <- lrn("regr.kknn", k = 3)

# train model to the training set
learner$train(task, row_ids = train_ind)

# predict on the test set
pred_test <- learner$predict(task, row_ids = test_ind)
pred_test

## <PredictionRegr> for 253 observations:
##     row_ids truth response
##           2  21.6 29.45474
##           5  36.2 33.14900
##           6  28.7 32.95574
## ---
##         501  16.8 19.61312
##         505  22.0 23.29286
##         506  11.9 21.28301

pred_test$score(list(msr("regr.mse"), msr("regr.mae")))

##  regr.mse  regr.mae
## 12.507468  2.458798
```

Effect: We will predict different observations since the test set is different. The same observations get a slightly different prediction (e.g. observation with id 2). This affects the final error estimation.

e)
```
rdesc <- rsmp("cv", folds = 10)
r <- resample(task, learner, rdesc)

## INFO  [10:36:19.867] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 9/10)
## INFO  [10:36:19.914] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 10/10)
## INFO  [10:36:19.938] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 7/10)
## INFO  [10:36:19.958] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 6/10)
```

```
## INFO  [10:36:19.978] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 4/10)
## INFO  [10:36:19.997] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 8/10)
## INFO  [10:36:20.020] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 1/10)
## INFO  [10:36:20.040] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 5/10)
## INFO  [10:36:20.059] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 3/10)
## INFO  [10:36:20.078] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 2/10)
```

```
r$aggregate(list(msr("regr.mse"), msr("regr.mae")))
```

```
##  regr.mse  regr.mae
## 10.045363  2.229458
```