Solution 1:

a) NB: these are very general scenarios and we can only state tendencies – there might be additional factors at play that encourage under- or overfitting in the specific situation.

  i) Large training set & linear regression learner: we would rather expect *underfitting* behavior here, as the model class is fairly simple and the model will not easily fit all training points closely if there are many.

  ii) Noisy training data & high-degree polynomial regression learner: this situation risks *overfitting.* The learner is quite flexible and the regression model might interpolate between the training samples, creating a wiggly curve that generalizes poorly.

  iii) Few observations of many features & linear regression learner: even though the model class is simple, we might run into *overfitting* here. The data situation, which we frequently encounter in biostatistics (e.g., genomics data), creates a high-dimensional and sparsely populated input space. In particular, some combinations of features will be represented by only few or even no observations at all. This means we have a large number of hypotheses, i.e., linear models with many covariates (possibly even interactions if we allow for it), and only few data to discern between them.

  iv) Data with class overlap & LDA learner: here we might expect *underfitting.* Clearly, the data are not linearly separable if the classes overlap in the input space, so LDA will not be able to learn the shape of the underlying function perfectly.

b) Overfitting and underfitting are always connected to a particular fixed *model*, even though attributes of the underlying hypothesis space typically influence the tendency toward one or the other behavior, as we have seen in the previous question. In order to understand this, think of a classification problem with linearly separable data. Applying a QDA learner, which is able to learn more complex decision boundaries, poses a risk of overfitting with the chosen model, but the degree of overfitting depends on the model itself. In theory, the QDA learner is set to choose equal covariances for the Gaussian class densities, amounting to LDA and *not* overfitting the data. A QDA model with distinctly different covariances, on the other hand, will probably invoke overfitting. Under- and overfitting are therefore properties of a specific model and not of an entire learner, though flexibility of the latter impacts the propensity toward either behavior.

A common strategy is to choose a rather flexible model class and encourage simplicity in the actual model by *regularization* (e.g., take a higher-degree polynomial but drive as many coefficients a possible toward zero).

c) That will be hardly possible. Recall how we defined the two variants of ill fit:

$$UF(\hat{f}, L) = GE(\hat{f}, L) - GE(f^*, L)$$

$$OF(\hat{f}, L) = GE(\hat{f}, L) - \mathcal{R}_{\text{emp}}(\hat{f}, L)$$

In order to avoid underfitting we would need to always find the universally loss-optimal model across arbitrary hypothesis spaces (the so-called *Bayes-optimal* model), which is obviously not something we can hope to achieve in general. Zero overfitting would mean to exactly balance theoretical generalization error and empirical risk, but the way empirical risk minimization is designed, our model will likely fare a bit worse on unseen test data.

In practice we will always experience these phenomena to some degree and finding a model that trades them off well is the holy grail in machine learning.

Solution 2:

a) The two main advantages of resampling are:

- We are able to use larger training sets (at the expense of test set size) because the high variance this incurs for the resulting estimator is smoothed out by averaging across repetitions.
- Repeated sampling reduces the risk of getting lucky (or not so lucky) with a particular data split, which is especially relevant with few observations.

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

# create task and learner
(task <- tsk("german_credit"))

## <TaskClassif:german_credit> (1000 x 21)
## * Target: credit_risk
## * Properties: twoclass
## * Features (20):
##   - fct (14): credit_history, employment_duration, foreign_worker,
##     housing, job, other_debtors, other_installment_plans,
##     people_liable, personal_status_sex, property, purpose, savings,
##     status, telephone
##   - int (3): age, amount, duration
##   - ord (3): installment_rate, number_credits, present_residence

learner <- lrn("classif.log_reg")

# train, predict and compute train error
learner$train(task)
preds <- learner$predict(task)
preds$score()

## classif.ce
##      0.211
```

c)
```r
# create different resampling strategies
set.seed(123)
resampling_3x10_cv <- rsmp("repeated_cv", folds = 10, repeats = 3)
resampling_10x3_cv <- rsmp("repeated_cv", folds = 3, repeats = 10)
resampling_ho <- rsmp("holdout", ratio = 0.9)

# evaluate without stratification
result_3x10_cv <- resample(task, learner, resampling_3x10_cv, store_models = TRUE)
result_10x3_cv <- resample(task, learner, resampling_10x3_cv, store_models = TRUE)
result_ho <- resample(task, learner, resampling_ho, store_models = TRUE)

# evaluate with stratification
task_stratified <- task$clone()
task_stratified$set_col_roles("foreign_worker", roles = "stratum")
result_stratified <- resample(
  task_stratified, learner, resampling_3x10_cv, store_models = TRUE)
```

```r
# aggregate results over splits (mce is default)
print(sapply(
  list(result_3x10_cv, result_10x3_cv, result_stratified, result_ho),
  function(i) i$aggregate()))

## classif.ce classif.ce classif.ce classif.ce
##  0.2486667  0.2557977  0.2525512  0.1800000
```

d) We see that generalization error estimates are pretty stable for the different resampling strategies because we have a fairly large number (1000) of observations. Still, the pessimistic bias of small training sets is visible: 10x3-CV, using roughly 67% of data for training in each split, estimates a higher generalization error than 3x10-CV with roughly 90% training data. Stratification by `foreign_worker` does not seem to have much effect on the estimate. However, we see a glaring optimistic bias when we use a single 90%-10% split, where the estimated GE is roughly 7 percentage points lower than with 3x10-CV.

   Comparing the results (except for the unreliable one produced by a single split) with the training error from b) indicates no serious overfitting.

e) LOO is not a very good idea here – with 1000 observations this would take a very long time, and the results from c) show that repeated CV with a sufficient number of folds yields pretty stable estimates.