# Data and Text Mining Exercises

1. ISLR2 Ch 4 Ex 6.
2. ISLR2 Ch 4 Ex 7.
3. ISLR2 Ch 4 Ex 8.
4. ISLR2 Ch 4 Ex 9.
5. ISLR2 Ch 4 Ex 13, at least until (d).
6. ISLR2 Ch 4 Ex 16.
7. ISLR2 Ch 5 Ex 5.
8. ISLR2 Ch 5 Ex 6.
9. ISLR2 Ch 5 Ex 7.
10. ISLR2 Ch 5 Ex 8.
11. ISLR2 Ch 5 Ex 9.
12. For the German dataset, obtain $B = 1234$ bootstrap replications of the regression coefficients for the linear regression of Amount on Duration and Job using the non-parametric bootstrap.

    (I.e., sample the rows of the data frame with replacement.)

    Compare the bootstrap confidence intervals for the regression coefficients with the ones obtained "directly".
13. For the German dataset, obtain $B = 1234$ bootstrap replications of the regression coefficient for the logistic regression of Class on Age and Status_of_checking_account using the parametric bootstrap.

    (I.e., generate good/bad classifications according to the fitted model probabilities.)

    Compare the bootstrap confidence intervals for the regression coefficients with the ones obtained "directly".
14. Investigate the variability of the 10-CV prediction error estimate for the linear regression model of Amount on Duration and Job by repeatedly generating such error estimates.
15. For the German data, consider logistic regression models of Class by suitable sets of predictors. The maximal model uses all possible predictors:

    ```
    R> m <- glm(Class ~ ., data = german, family = binomial())
    ```

    (This has many non-significant coefficient/variables, which ideally would be dropped from the model, e.g., using a stepwise procedure.)

    The fitted values of this model are the probabilities for class 1, so we can implement classification into the more probable class by simply rounding the fitted values. This gives confusion matrix

    ```
    R> with(german, table(Class, round(fitted(m))))
    ```

    ```
    Class    0   1
      good 626  74
      bad  140 160
    ```

Using classification error

```
R> MCE <- function(y, yhat) mean(y != yhat)
```

we would obtain the apparent (in-sample) prediction error as

```
R> Classhat <- factor(round(fitted(m)), labels = c("good", "bad"))
R> with(german, MCE(Class, Classhat))
```

```
[1] 0.214
```

Perform improved estimation of the prediction error of the logistic regression model of your choice via cross-validation.

16. Redo the previous problems using stratified 10-fold cross-validation. To generate the folds, you can use:

```
R> folds <- function(n, k, f = NULL) {
+     if(is.null(f))
+         ids <- sample(rep(1 : k, length.out = n))
+     else {
+         ids <- integer(n)
+         for(l in levels(f)) {
+             ind <- (f == l)
+             ids[ind] <- sample(rep(1 : k, length.out = sum(ind)))
+         }
+     }
+     split(1 : n, ids)
+ }
```

17. ISLR2 Ch 6 Ex 7.

18. ISLR2 Ch 6 Ex 8.

19. ISLR2 Ch 6 Ex 9 excluding (e) and (f).

20. ISLR2 Ch 6 Ex 10.

21. ISLR2 Ch 6 Ex 11.

22. ISLR2 Ch 8 Ex 10.

23. ISLR2 Ch 8 Ex 11.

24. Perform a simulation experiment to compare the stability of trees to those of linear models.

    For simplicity, consider the linear regression of Amount on Duration and Job, and do not attempt to optimally prune the trees. E.g., use the `cp` value giving the "best" tree for the original data set.

    Note that for bootstrap data sets one can get different tree structures (different splits and numbers thereof), so one cannot compare fitted model "coefficients". Instead, compare the variability of the (in-sample) predictions.

25. For the German data, try to find better classification trees for `Class` using the available predictors.

26. For the German data, compare the performance of predicting `Class` by logistic regression and random forests.

27. For the German data, try to find better generalized boosted models for `Class` using the available predictors, e.g., by increasing interaction depth.

28. In an earlier problem, we obtained (for the German dataset) $B = 1234$ bootstrap replications of the regression coefficients for the logistic regression of Class on Age and Status_of_checking_account using the parametric bootstrap.

    How much can this computation be sped up by using several cores?