Exercise 2

a) [0.5 points]

First of all, we read the data from the csv file and then run `str` to see the classes of the columns.

```
heart <- read.csv( file = paste0("C:/Users/leaz9/OneDrive/Dokumente/StatLearn WS22", "/data/heart.csv"))
str(heart)
```

```
## 'data.frame':    297 obs. of  11 variables:
##  $ age     : int  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : chr  "M" "M" "M" "M" ...
##  $ cp      : int  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: int  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : int  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : int  1 0 0 0 0 0 0 0 0 1 ...
##  $ thalach : int  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : int  0 1 1 0 0 0 0 1 0 1 ...
##  $ ca      : int  0 3 2 0 0 0 2 0 1 0 ...
##  $ thal    : int  6 3 7 3 3 3 3 3 7 7 ...
##  $ hd      : chr  "Healthy" "Unhealthy" "Unhealthy" "Healthy" ...
```

According to the description of the variables in the exercise, the columns `sex, cp, fbs, exang, ca, thal, hd` contain categorial variables. Since they are either of class `int` or `chr`, we transform them class `factor`. The function `mutate_at` from the `dplyr` package is very convenient for this task.

```
library(dplyr)
?mutate_at
heart <- mutate_at(heart, vars(sex, cp, fbs, exang, ca, thal, hd ), as.factor)
str(heart)
```

```
## 'data.frame':    297 obs. of  11 variables:
##  $ age     : int  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : Factor w/ 2 levels "F","M": 2 2 2 2 1 2 1 1 2 2 ...
##  $ cp      : Factor w/ 4 levels "1","2","3","4": 1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: int  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : int  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 2 ...
##  $ thalach : int  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 1 2 1 2 ...
##  $ ca      : Factor w/ 4 levels "0","1","2","3": 1 4 3 1 1 1 3 1 2 1 ...
##  $ thal    : Factor w/ 3 levels "3","6","7": 2 1 3 1 1 1 1 1 3 3 ...
##  $ hd      : Factor w/ 2 levels "Healthy","Unhealthy": 1 2 2 1 1 1 2 1 2 2 ...
```

Of course, there are many ways to do it, you could e.g. also transform each column manually or use the `lapply()` function.

b) [0.5 points]

Now we want to fit several logistic regression models. For this, we will use the function `glm` (generalized linear model). The first paragraph in the section 'Details' of the function documentation of `glm` states that,

when fitting a logisitc regression model, the response variable (here: `hd`) can be a factor, and that in this case the second factor level is considered as success. Since here the second factor level is " Unhealthy' ' (see above) we are thus modelling $p(\mathbf{x}) = P(Unhealthy|\mathbf{x})$.

We fit the three models with `sex, sex` and `thal`, all features as explanatory variables as follows. Here, it is important to set `family = binomial`, such that indeed a logistic regression is performed.

```
mod.sex <- glm(hd ~ sex  , data = heart, family = "binomial")
mod.sexthal <- glm(hd ~ sex + thal , data = heart, family = "binomial")
mod.full <-  glm(hd ~. , data = heart, family = "binomial")
```

For predicting the disease status based on the above models, we can use the `predict()` function. We have to specify `type ="response"` to obtain estimates of$p(\mathbf{x})$ and not of logit$(p(\mathbf{x}))$, which is the default.

```
pr.sex <- predict(mod.sex, newdata = heart, type = "response")
pr.sexthal <- predict(mod.sexthal, newdata = heart, type = "response")
pr.full <- predict(mod.full,newdata = heart, type = "response" )
```

Based on these estimated probabilities, we predict the individual to be unhealthy whenever the estimated probability is larger than or equal to 0.5. We do it for all three models at once using `lapply()`.

```
pr.disease <- lapply(list( pr.sex =  pr.sex, pr.sexthal = pr.sexthal,
                           pr.full = pr.full ),
                     function(x) ifelse( x >= 0.5, "Unhealthy", "Healthy"))
```

From this, we can compute the in-sample misclassification rate.

```
lapply(pr.disease, function(x){
  conf_mat <- table(x, heart$hd)
  1 - sum(diag(conf_mat))/nrow(heart)})
```

```
## $pr.sex
## [1] 0.3838384
##
## $pr.sexthal
## [1] 0.2356902
##
## $pr.full
## [1] 0.1380471
```

The model using all features has by far the smallest in-sample error.

c) [1 point]

We write a function that computes the out-of-sample error based on leaving-one-out cross-validation. The argument model can be one of full, sex, sexthal or reduced (for part (f)).

```
compute_loocv <- function(index, model = "full", data){
  if( !(model %in% c("full", "sex", "sexthal", "reduced"))){stop("cv not implemented for this model.")}
  # reduce dataset according to features used in the respective model
  if(model == "sex"){ data <- select(data, c("sex", "hd"))}
  if(model == "sexthal"){ data <- select(data, c("sex", "thal", "hd"))}
  # for part (f)
  if(model == "reduced"){
    data <- select(data, -c("age", "chol", "fbs"))}

  data_train <- data[ -index, ]
  data_test <- data[ index, ]
```

```
  fitted.model <- glm(hd ~ . , data = data_train, family = "binomial")

  p.pred <- predict(fitted.model, newdata = data_test, type = "response")

  disease.pred <- ifelse(p.pred >= 0.5, "Unhealthy", "Healthy")
  disease.pred != data_test$hd  # returns TRUE ( = 1) whenever observation is misclassified
}
```

We apply the above function for every index in $\{1, \ldots, \text{nrow(heart)}\}$ and compute the mean of the corresponding errors.

```
cv_sex <- mean(sapply( 1:nrow(heart), compute_loocv,  model = "sex",  data = heart))
cv_sexthal <- mean(sapply( 1:nrow(heart), compute_loocv,  model = "sexthal",  data = heart))
cv_full <- mean(sapply( 1:nrow(heart), compute_loocv,  model = "full",  data = heart))

cv_sex
```

```
## [1] 0.3838384
```

```
cv_sexthal
```

```
## [1] 0.2356902
```

```
cv_full
```

```
## [1] 0.1649832
```

d) [0.5 points]

We take a look at the estimated coefficients of the model fitted in (b)(ii)

```
mod.sexthal$coefficients
```

```
## (Intercept)        sexM       thal6       thal7
##  -1.5226079   0.5276093   1.7194443   2.2534114
```

The fitted model for the log odds is thus

$$\text{logit}(p(\mathbf{x})) = \log(\text{odds}(p(\mathbf{x}))) = \beta_0 + \beta_1 \cdot 1(\text{sex} = \text{''}male\text{''}) + \beta_2 \cdot 1(\text{Thal} = 6) + \beta_3 \cdot 1(\text{Thal} = 7).$$

All coefficients (except intercept) are positive, i.e. being male and having thalium heart scan values of 6 or 7 is associated with an increase in log odds of being unhealthy. \ Taking exp() on the euqation above, we see that for fixed value of Thal, being male increases the odds of being unhealthy by $\exp(\beta_1)$, while for fixed value of sex, having value "thal = 6" increases the odds of being unhealthy by $\exp(\beta_2)$, and having value "thal = 7" increases the odds of being unhealthy by $\exp(\beta_3)$. These values are

```
expbeta <- exp(mod.sexthal$coefficients[2:4])
names(expbeta) <- paste0("exp(beta", 1:3, ")")
expbeta
```

```
## exp(beta1) exp(beta2) exp(beta3)
##   1.694875   5.581426   9.520158
```

e) [0.5 points]

```
summary(mod.full)
```

```
##
## Call:
## glm(formula = hd ~ ., family = "binomial", data = heart)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1314  -0.4809  -0.1482   0.3840   2.4398
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.088790   2.671287  -1.156  0.24756
## age         -0.023783   0.024667  -0.964  0.33497
## sexM         1.461947   0.499575   2.926  0.00343 **
## cp2          0.425111   0.734291   0.579  0.56263
## cp3          0.043774   0.673503   0.065  0.94818
## cp4          1.712085   0.643823   2.659  0.00783 **
## trestbps     0.028251   0.010935   2.584  0.00978 **
## chol         0.006506   0.003823   1.702  0.08882 .
## fbs1        -0.540918   0.542331  -0.997  0.31857
## thalach     -0.031942   0.010828  -2.950  0.00318 **
## exang1       0.852279   0.429640   1.984  0.04729 *
## ca1          1.920808   0.473827   4.054 5.04e-05 ***
## ca2          3.005052   0.666601   4.508 6.54e-06 ***
## ca3          2.214018   0.830983   2.664  0.00771 **
## thal6        0.406862   0.767774   0.530  0.59616
## thal7        1.585748   0.407368   3.893 9.91e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 409.95  on 296  degrees of freedom
## Residual deviance: 200.87  on 281  degrees of freedom
## AIC: 232.87
##
## Number of Fisher Scoring iterations: 6
```

The intercept, `age, chol` and `fbs1` might not have a big influence on the log odds, since the p-values of the tests with null hypothesis $H_0 : \beta_j = 0$ are all larger than 0.05, i.e. the hypothesis that the corresponding feature does not have any effect on the log odds cannot be rejected at a level of 5%. Note that this is a multiple testing problem, however, so p-values would have to be adjusted if you would want to rely on the test decisions.

Furhter, `cp3` and `cp4` as well as `thal6` don't have significant p-values. One could therefore also try to only use the dummy variables `cp4` and `thal7` in the model.

f) [0.5 points]

Reduced model:

```
mod.red <- glm( hd ~ . - age - chol - fbs ,
                data = heart,  family = "binomial")

pr.red <- predict( mod.red, newdata = heart, type = "response")
pr.dis.red <- ifelse(pr.red >= 0.5,
```

```
                        "Unhealthy", "Healthy")
conf_red <- table(pr.dis.red, heart$hd)
1 - sum(diag(conf_red))/nrow(heart)  # in-sample misclassification rate
```

```
## [1] 0.1414141
```

The computation of oos error based on LOOCV for the reduced model is already implemented in the function `compute_loocv` above.

```
cv_red <- mean(sapply( 1:nrow(heart), compute_loocv,  model = "reduced",  data = heart))
cv_red
```

```
## [1] 0.1683502
```

```
cv_full
```

```
## [1] 0.1649832
```

The reduced model has almost the same cv error as the full model, but it uses 3 features less.

g) [0.5 points]

We assemble a data frame for making the plot.

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

```
.df <- data.frame( prob = pr.red, status = heart$hd, x = rank(pr.red))

ggplot( .df, aes( x= x, y = prob, color = status)) + geom_point()+
  labs( x = "rank(p(x))", y = "p(x)")
```