Week 6

gherardo varando

January 9, 2019

CORIS data

```
coris <- read.table("coris.dat", skip = 4, sep = ",",
col.names = c("row.names", "sbp", "tobacco",
"ldl", "adiposity",
"famhist", "typea", "obesity",
"alcohol",
"age", "chd"))[,-1]
```

Ex 1

1.1

Fit the full logistic regression model

```
coris$chd <- as.factor(coris$chd)
fitAll <- glm(chd ~ ., data = coris, family = "binomial")
```

We estimate accuracy of the model using leave-one -out cross validation, first of all we copy from week 5 solutions the function to transfrom the prediction output to the class values.

```
toClass <- function(predictions,
                    levels,
                    linkinv =
                    binomial()$linkinv){

  ## threshold the prob of success
  a <- linkinv(predictions) > 0.5
  b <- array(dim =
             c(length(predictions)))
  ## if prob succ > 0.5 => success
  ##                       (second lvl)
  b[a] <- levels[2]

  ## otherwise not success (first lvl)
  b[!a] <- levels[1]

  ## we should return a factor
  return(factor(b, levels = levels))
}
tt <- table(toClass(predict(fitAll), levels(coris$chd)), coris$chd)
```

```r
acc_train <- sum(diag(tt)) / sum(tt)
acc_train ### accuracy over the training set
```

```
## [1] 0.7337662
```

Now we can perform cross validation using leave-one-out

```r
n <- nrow(coris)
res_loo <- sapply(1:n, function(i){
  fit <- glm(chd ~ ., data = coris[-i, ], family = "binomial")
  pr <- predict(fit, newdata = coris[i,])
  pred.class <- toClass(pr, levels = levels(coris$chd))
  return(pred.class == coris$chd[i])
})
acc_loo <- sum(res_loo) / length(res_loo)
acc_loo
```

```
## [1] 0.7186147
```

And using 10-fold, first of all we need to create the groups, the easier way is to shuffle the index of the observations (rows of `coris`) and then create 10 consecutive groups. Moreover since `n = 462` we will create 8 groups of 46 elements and 2 of 47.

```r
k <- 10
r <- floor(n / k)
groups <- list()
t <- 0
s <- 0
for (i in 1:10){
  if (i > 8){
    t <- 1
    if (i > 9){
      s <- 1
    }
  }
  groups[[i]] <- ((i - 1) * r + 1 + s) : (i * r + t + s)
}
corisSHF <- coris[sample(1:n), ]
```

Now we can perform the validation:

```r
res_10 <- sapply(1:10, function(i){
  fit <- glm(chd ~ ., data = corisSHF[-groups[[i]], ], family = "binomial")
  pr <- predict(fit, newdata = corisSHF[groups[[i]],])
  pred.class <- toClass(pr, levels = levels(coris$chd))
  tt <- table(pred.class, corisSHF$chd[ groups[[i]] ] )
  acc <- sum(diag(tt)) / sum(tt)
  return(acc)
})
acc_10 <- mean(res_10)
acc_10
```

```
## [1] 0.7211378
```

We use backward stepwise selection

```r
fitst <- step(fitAll, direction = "backward", trace = 0)
```

And we evaluate the model with both leve-one-out and 10-fold cross validation,

```
res_loo_st <- sapply(1:n, function(i){
  fit <- glm(formula(fitst), data = coris[-i, ], family = "binomial")
  pr <- predict(fit, newdata = coris[i,])
  pred.class <- toClass(pr, levels = levels(coris$chd))
  return(pred.class == coris$chd[i])
})
acc_loo_st <- sum(res_loo_st) / length(res_loo_st)
acc_loo_st
```

```
## [1] 0.7359307
```

```
res_10_st <- sapply(1:10, function(i){
  fit <- glm(formula(fitst), data = corisSHF[-groups[[i]], ], family = "binomial")
  pr <- predict(fit, newdata = corisSHF[groups[[i]],])
  pred.class <- toClass(pr, levels = levels(coris$chd))
  tt <- table(pred.class, corisSHF$chd[ groups[[i]] ] )
  acc <- sum(diag(tt)) / sum(tt)
  return(acc)
})
acc_10_st <- mean(res_10_st)
acc_10_st
```

```
## [1] 0.7383441
```

We put everything in a matrix to compre it

```
matrix(c(acc_loo, acc_10, acc_loo_st, acc_10_st), nrow = 2, byrow = TRUE,
       dimnames = list(model = c("full", "step"), accuracy = c("loo", "10-fold")))
```

```
##       accuracy
## model        loo   10-fold
##   full 0.7186147 0.7211378
##   step 0.7359307 0.7383441
```

We can observe that the simpler model obtained with backward stepwise selection based on AIC, generalize better, that is has a better accuracy over unseen observations.

1.2

First of all we create a function to cross validate a model. By default it performs leave-one-out.

```
crossval <- function(object, data = object$data,
                     groups = as.list(1:nrow(data)),
                     shuffle = TRUE){
  if (shuffle){
    data <- data[sample(1:nrow(data)), ]
  }
  class <- as.character(object$formula[[2]])
  res <- sapply(groups, function(ix){
    fit <- glm(formula(object), data = data[-ix, ], family = object$family)
    pr <- predict(fit, newdata = data[ix,])
     pred.class <- toClass(pr, levels = levels(data[[class]]),
                       linkinv = object$family$linkinv)
    tt <- table(pred.class, data[[ class ]][ ix ] )
    acc <- sum(diag(tt)) / sum(tt)
    return(acc)
  })
  return(mean(res))
}
```

We test the function,

```
crossval(fitAll)
```

```
## [1] 0.7186147
```

```
crossval(fitAll, data = corisSHF, groups = groups, shuffle = FALSE)
```

```
## [1] 0.7211378
```

It seems that it works fine.

Now we copy and modify the stepwise algorithm from the solutions of week 5,

```
#### we create the 5 groups for validation
k <- 5
r <- floor(n / k)
groups <- list()
t <- 0
s <- 0
for (i in 1:5){
  if (i > 3){
    t <- 1
    if (i > 4){
      s <- 1
    }
  }
  groups[[i]] <- ((i - 1) * r + 1 + s) : (i * r + t + s)
}

### we start the model with only the intercept
fit <- glm(chd ~ 1, family = "binomial",
           data = coris) ## only the intercept
regressors <- colnames(coris)[-10]
selected <- c()
score <- crossval(object = fit, groups = groups, shuffle = FALSE)
score.best <- score
done <- FALSE
while (!done){
   for (reg in regressors[!(regressors %in% selected)]){
     tmp <- update(fit, formula = paste(". ~ . + ", reg))
     score.tmp <- crossval(tmp, groups = groups, shuffle = FALSE)
     if (score.tmp > score.best){
       score.best <- score.tmp
       best <- tmp
       selected.best <- c(selected, reg)
     }
   }
   if (score.best > score){
     fit <- best
     score <- score.best
     selected <- selected.best
  }else{ ### if there is no increase
    done <- TRUE
  }
}
#### when the while loop ends we will have the selected model in
#### fit

fit
```

```
##
## Call:  glm(formula = chd ~ tobacco + ldl + famhist + age, family = "binomial",
##     data = coris)
##
## Coefficients:
## (Intercept)      tobacco          ldl      famhist          age
##    -4.20428      0.08070      0.16758      0.92412      0.04404
##
## Degrees of Freedom: 461 Total (i.e. Null);  457 Residual
## Null Deviance:       596.1
## Residual Deviance: 485.4     AIC: 495.4
```

The 5-fold cross validation accuracy (using the same groups) of the model is

```
crossval(fit, groups = groups, shuffle = FALSE)
```

```
## [1] 0.7335671
```

And for the other models we have

```
crossval(fitst, groups = groups, shuffle = FALSE)
```

```
## [1] 0.729266
```

and

```
crossval(fitAll, groups = groups, shuffle = FALSE)
```

```
## [1] 0.727022
```

Wines quality

Ex 2

```
wines_red <- read.csv("winequality-red.csv", sep =";")
wines_white <- read.csv("winequality-white.csv", sep =";")
good <- wines_red$quality > 5
wines_red$quality <- "bad"
wines_red[good, "quality"] <- "good"
wines_red[,"quality"] <- as.factor(wines_red[, "quality"])
good <- wines_white$quality > 5
wines_white$quality <- "bad"
wines_white[good, "quality"] <- "good"
wines_white[,"quality"] <- as.factor(wines_white[, "quality"])
```

2.1

Fit a logistic regression model sing all the predictors and data for red wines.

```r
model_red <- glm(quality ~ ., family = "binomial", data = wines_red)
summary(model_red)
```

```
##
## Call:
## glm(formula = quality ~ ., family = "binomial", data = wines_red)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4025  -0.8387   0.3105   0.8300   2.3142
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)           42.949948  79.473979   0.540  0.58890
## fixed.acidity          0.135980   0.098483   1.381  0.16736
## volatile.acidity      -3.281694   0.488214  -6.722 1.79e-11 ***
## citric.acid           -1.274347   0.562730  -2.265  0.02354 *
## residual.sugar         0.055326   0.053770   1.029  0.30351
## chlorides             -3.915713   1.569298  -2.495  0.01259 *
## free.sulfur.dioxide    0.022220   0.008236   2.698  0.00698 **
## total.sulfur.dioxide  -0.016394   0.002882  -5.688 1.29e-08 ***
## density              -50.932385  81.148745  -0.628  0.53024
## pH                    -0.380608   0.720203  -0.528  0.59717
## sulphates              2.795107   0.452184   6.181 6.36e-10 ***
## alcohol                0.866822   0.104190   8.320  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2209.0  on 1598  degrees of freedom
## Residual deviance: 1655.6  on 1587  degrees of freedom
## AIC: 1679.6
##
## Number of Fisher Scoring iterations: 4
```

We compute the accuracy over the red wines data:

```r
## we can use again the toClass function we defined before
predicted_quality <- toClass(predictions = predict(model_red), levels = levels(wines_red$quality))
acc_red <- sum(diag(table(predicted_quality, wines_red$quality))) / nrow(wines_red)
acc_red
```

```
## [1] 0.7442151
```

```r
### or we can do it without using the toClass variable
pred <- predict(model_red)

predicted_quality <- rep(NA, length(pred))
predicted_quality[pred >= 0] <- "good"
predicted_quality[pred < 0] <- "bad"
acc_red <- sum(diag(table(predicted_quality, wines_red$quality))) / nrow(wines_red)
acc_red
```

```
## [1] 0.7442151
```

```
### you can see that the two methods return the same accuracy
```

The accuracy for the model over the white wines:

```
### we just replace wines_red with wines_white
pred <- predict(model_red, newdata = wines_white)
predicted_quality <- rep(NA, length(pred))
predicted_quality[pred >= 0] <- "good"
predicted_quality[pred < 0] <- "bad"
acc_white <- sum(diag(table(predicted_quality, wines_white$quality))) / nrow(wines_white)
acc_white
```

```
## [1] 0.6672111
```

As expected the accuracy over the white wines is lower, we used a model trained over red wines to predict the quality of white wines.

2.2

Is similar to ex 2.1 but with inverted roles of red and whites wines.

```
model_white <- glm(quality ~ ., family = "binomial", data = wines_white)
summary(model_white)
```

```
##
## Call:
## glm(formula = quality ~ ., family = "binomial", data = wines_white)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1731  -0.8946   0.4420   0.7994   2.9466
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)           2.582e+02  7.099e+01   3.638 0.000275 ***
## fixed.acidity         3.648e-02  7.178e-02   0.508 0.611271
## volatile.acidity     -6.459e+00  4.128e-01 -15.646  < 2e-16 ***
## citric.acid           1.158e-01  3.029e-01   0.382 0.702219
## residual.sugar        1.701e-01  2.704e-02   6.291 3.16e-10 ***
## chlorides             8.852e-01  1.671e+00   0.530 0.596379
## free.sulfur.dioxide   9.601e-03  2.782e-03   3.451 0.000560 ***
## total.sulfur.dioxide -1.333e-03  1.211e-03  -1.101 0.270982
## density              -2.709e+02  7.195e+01  -3.765 0.000167 ***
## pH                    1.090e+00  3.618e-01   3.013 0.002590 **
## sulphates             1.797e+00  3.595e-01   5.000 5.75e-07 ***
## alcohol               7.429e-01  9.361e-02   7.937 2.08e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6245.4  on 4897  degrees of freedom
## Residual deviance: 4932.6  on 4886  degrees of freedom
## AIC: 4956.6
##
## Number of Fisher Scoring iterations: 5
```

```
#
#
# .....
```

2.3

```
summary(model_red)
```

```
##
## Call:
## glm(formula = quality ~ ., family = "binomial", data = wines_red)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4025  -0.8387   0.3105   0.8300   2.3142
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)           42.949948  79.473979   0.540  0.58890
## fixed.acidity          0.135980   0.098483   1.381  0.16736
## volatile.acidity      -3.281694   0.488214  -6.722 1.79e-11 ***
## citric.acid           -1.274347   0.562730  -2.265  0.02354 *
## residual.sugar         0.055326   0.053770   1.029  0.30351
## chlorides             -3.915713   1.569298  -2.495  0.01259 *
## free.sulfur.dioxide    0.022220   0.008236   2.698  0.00698 **
## total.sulfur.dioxide  -0.016394   0.002882  -5.688 1.29e-08 ***
## density              -50.932385  81.148745  -0.628  0.53024
## pH                    -0.380608   0.720203  -0.528  0.59717
## sulphates              2.795107   0.452184   6.181 6.36e-10 ***
## alcohol                0.866822   0.104190   8.320  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2209.0  on 1598  degrees of freedom
## Residual deviance: 1655.6  on 1587  degrees of freedom
## AIC: 1679.6
##
## Number of Fisher Scoring iterations: 4
```

```
summary(model_white)
```

```
##
## Call:
## glm(formula = quality ~ ., family = "binomial", data = wines_white)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1731  -0.8946   0.4420   0.7994   2.9466
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)           2.582e+02  7.099e+01   3.638 0.000275 ***
## fixed.acidity         3.648e-02  7.178e-02   0.508 0.611271
## volatile.acidity     -6.459e+00  4.128e-01 -15.646  < 2e-16 ***
## citric.acid           1.158e-01  3.029e-01   0.382 0.702219
## residual.sugar        1.701e-01  2.704e-02   6.291 3.16e-10 ***
## chlorides             8.852e-01  1.671e+00   0.530 0.596379
## free.sulfur.dioxide   9.601e-03  2.782e-03   3.451 0.000560 ***
## total.sulfur.dioxide -1.333e-03  1.211e-03  -1.101 0.270982
## density              -2.709e+02  7.195e+01  -3.765 0.000167 ***
## pH                    1.090e+00  3.618e-01   3.013 0.002590 **
## sulphates             1.797e+00  3.595e-01   5.000 5.75e-07 ***
## alcohol               7.429e-01  9.361e-02   7.937 2.08e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6245.4  on 4897  degrees of freedom
## Residual deviance: 4932.6  on 4886  degrees of freedom
## AIC: 4956.6
##
## Number of Fisher Scoring iterations: 5
```

The model fitted over the two dataset show some differences:

- The p-value for the intercept in the model for red wines shows that we can not reject the hypothesis that the simpler model without intercept is sufficient, while for white wines we have enough data to show that the intercept should be included in the model.
- the residual sugar variable seems to be important for predicting quality of white wines but not for red wines.
- the total sulfur dioxid is significant in the model for white wines, while it can be probably omitted in the red wines model.
- ....

Ex 3

3.1

We want to perform stepwise forward selection using all the data available.

```
## the function rbind will join two dataframe with the same columns
## into a single dataframe
wines_all <- rbind(wines_red, wines_white)
dim(wines_all) ### you can check that the dimensions are correct
```

```
## [1] 6497   12
```

```
model_0 <- glm(quality ~ 1, data = wines_all, family = "binomial")
model_all <- glm(quality ~ ., data = wines_all, family = "binomial")
model_step <- step(model_0, scope = as.formula(model_all),
                       direction = "forward", trace = 0)
summary(model_step)
```

```
##
## Call:
## glm(formula = quality ~ alcohol + volatile.acidity + sulphates +
##     residual.sugar + total.sulfur.dioxide + free.sulfur.dioxide +
##     citric.acid + pH, family = "binomial", data = wines_all)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3123  -0.9079   0.4374   0.8161   2.6646
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -9.7229883  0.7803911 -12.459  < 2e-16 ***
## alcohol               0.9480857  0.0339259  27.946  < 2e-16 ***
## volatile.acidity     -4.6118529  0.2391270 -19.286  < 2e-16 ***
## sulphates             1.9394319  0.2320192   8.359  < 2e-16 ***
## residual.sugar        0.0669939  0.0075820   8.836  < 2e-16 ***
## total.sulfur.dioxide -0.0074298  0.0008427  -8.816  < 2e-16 ***
## free.sulfur.dioxide   0.0162968  0.0025035   6.510 7.54e-11 ***
## citric.acid          -0.5627994  0.2312561  -2.434   0.0149 *
## pH                    0.3831002  0.2109088   1.816   0.0693 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8541.0  on 6496  degrees of freedom
## Residual deviance: 6710.3  on 6488  degrees of freedom
## AIC: 6728.3
##
## Number of Fisher Scoring iterations: 4
```

3.2

```r
k <- 10
n <- nrow(wines_all)
m <- n %/% k
groups <- lapply(1:k, function(j){
  return( (m* (j-1) + 1): (m * j) )
})
crossval(model_step,data =  wines_all, groups = groups, shuffle = TRUE)
```

```
## [1] 0.7416025
```

3.3

We want to test the model `model_step` over the red wines

First of all we see how we can select 200 randomly red wines and then train the model on the remaining red and all the white wines, and finally test the model over the 200 selected red wines.

```r
### randomly select 200 red wines
red_test <- sample(nrow(wines_red), size = 200, replace = FALSE)

### train the model
model_trained <- glm(as.formula(model_step), family = "binomial",
                     data = rbind(wines_red[-red_test,], wines_white))

pred_qual <- toClass(predict(model_trained, newdata = wines_red[red_test,]),
                  levels(wines_red$quality))
## compute the accuracy
sum(diag(table(pred_qual, wines_red$quality[red_test]))) / 200
```

```
## [1] 0.75
```

now we repeat the process a certain number of times and we average the results:

```
accs <- replicate(100, {
  ### randomly select 200 red wines
red_test <- sample(nrow(wines_red), size = 200, replace = FALSE)

### train the model
model_trained <- glm(as.formula(model_step), family = "binomial",
                     data = rbind(wines_red[-red_test,], wines_white))

pred_qual <- toClass(predict(model_trained, newdata = wines_red[red_test,]),
                  levels(wines_red$quality))
## compute the accuracy
sum(diag(table(pred_qual, wines_red$quality[red_test]))) / 200
})

mean(accs)
```

```
## [1] 0.7393
```