Problem 6 - Data analysis II

alexaoh

27 mai, 2021

All needed packages are run first in a chunk with `echo = FALSE`.

Import data.

```
id <- "1cSVIJv-OoAwkhUAuun2qQyOfiuZzkmo3"
d.sparrows <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download",
    id), header = T)
# pairs(d.sparrows)
str(d.sparrows)
```

```
#> 'data.frame':    169 obs. of  11 variables:
#>  $ sex    : int  1 1 1 1 1 1 1 1 1 1 ...
#>  $ lnrhday: int  80 118 118 115 54 113 113 76 74 74 ...
#>  $ clsize : int  4 2 2 2 1 3 3 2 3 3 ...
#>  $ hyear  : int  1995 1995 1995 1997 1999 2000 2000 2001 2002 2002 ...
#>  $ f      : num  0 0 0 12.5 18.7 12.5 14.1 1.6 1.6 1.6 ...
#>  $ hisl   : int  20 20 20 20 20 20 20 20 20 20 ...
#>  $ H1     : num  0.571 0.75 0.875 0.875 0.625 ...
#>  $ GTloci : int  7 8 8 8 8 9 8 8 8 8 ...
#>  $ Hloci  : int  4 6 7 7 5 6 6 6 7 5 ...
#>  $ geno   : int  103 1 1 1 1 2 203 201 201 201 ...
#>  $ recruit: int  0 0 1 0 0 0 1 0 0 0 ...
```

a)

Logistic regression is fitted below.

```
d.sparrows$hisl <- as.factor(d.sparrows$hisl)

# First model.
log.fit <- glm(recruit ~ . + sex:f, data = d.sparrows, family = "binomial")
summary(log.fit)
```

```
#>
#> Call:
#> glm(formula = recruit ~ . + sex:f, family = "binomial", data = d.sparrows)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -1.5536  -0.7744  -0.5230   0.7866   2.1427
#>
#> Coefficients:
#>               Estimate Std. Error z value Pr(>|z|)
```

```
#> (Intercept) 423.153540 349.742787   1.210   0.2263
#> sex          -0.821220   0.447836  -1.834   0.0667 .
#> lnrhday       0.020683   0.010802   1.915   0.0555 .
#> clsize        0.196824   0.190545   1.033   0.3016
#> hyear        -0.222854   0.179995  -1.238   0.2157
#> f            -0.132373   0.087483  -1.513   0.1302
#> hisl20       -0.137605   0.788102  -0.175   0.8614
#> hisl26       -0.532817   0.567052  -0.940   0.3474
#> hisl28       -1.876165   1.150416  -1.631   0.1029
#> hisl38        1.237168   0.811106   1.525   0.1272
#> H1           16.215035  22.584974   0.718   0.4728
#> GTloci        2.204134   2.472942   0.891   0.3728
#> Hloci        -1.673924   2.753501  -0.608   0.5432
#> geno          0.002901   0.005606   0.518   0.6048
#> sex:f         0.048489   0.058147   0.834   0.4043
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 193.82  on 168  degrees of freedom
#> Residual deviance: 166.20  on 154  degrees of freedom
#> AIC: 196.2
#>
#> Number of Fisher Scoring iterations: 5
```

```
# Second model.
log.fit2 <- glm(recruit ~ . + sex:f - hisl, data = d.sparrows, family = "binomial")
summary(log.fit2)
```

```
#>
#> Call:
#> glm(formula = recruit ~ . + sex:f - hisl, family = "binomial",
#>     data = d.sparrows)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -1.4365  -0.8059  -0.6188   1.0589   2.1147
#>
#> Coefficients:
#>              Estimate Std. Error z value Pr(>|z|)
#> (Intercept) 205.804018 291.514177   0.706  0.48020
#> sex          -0.721028   0.431196  -1.672  0.09449 .
#> lnrhday       0.026776   0.010274   2.606  0.00915 **
#> clsize        0.255770   0.175858   1.454  0.14583
#> hyear        -0.110760   0.150265  -0.737  0.46107
#> f            -0.101429   0.077906  -1.302  0.19293
#> H1            8.300599  20.966793   0.396  0.69218
#> GTloci        1.302837   2.266553   0.575  0.56542
#> Hloci        -0.753998   2.558580  -0.295  0.76823
#> geno          0.003053   0.004864   0.628  0.53026
#> sex:f         0.047542   0.054361   0.875  0.38181
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
```

```
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 193.82  on 168  degrees of freedom
#> Residual deviance: 176.92  on 158  degrees of freedom
#> AIC: 198.92
#>
#> Number of Fisher Scoring iterations: 5
```

```
# Evidence?
anova(log.fit, log.fit2, test = "Chisq")
```

```
#> Analysis of Deviance Table
#>
#> Model 1: recruit ~ sex + lnrhday + clsize + hyear + f + hisl + H1 + GTloci +
#>     Hloci + geno + sex:f
#> Model 2: recruit ~ sex + lnrhday + clsize + hyear + f + hisl + H1 + GTloci +
#>     Hloci + geno + sex:f - hisl
#>   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
#> 1       154     166.21
#> 2       158     176.92 -4  -10.713  0.02999 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Depending on the level of significance chosen, there might be evidence that the survival probabilities differed between hatch islands, based on the p-value shown in the output above. Since $p \approx 0.03$ one might want to conclude that there is evidence, e.g. if the significance level is chosen at 0.05 (which is pretty normal to do). Hence, based on this significance level, I would conclude that there is evidence that the survival probabilities differed between hatch islands.

b)

Split the dataset.

```
set.seed(123456)
samples <- sample(1:169, 120, replace = F)
d.sparrows.train <- d.sparrows[samples, ]
d.sparrows.test <- d.sparrows[-samples, ]
```

Logistic regression without hatch island and without the interaction between f and sex.

```
# Fit the model.
log.fit3 <- glm(recruit ~ . - hisl, data = d.sparrows.train, family = "binomial")
summary(log.fit3)
```

```
#>
#> Call:
#> glm(formula = recruit ~ . - hisl, family = "binomial", data = d.sparrows.train)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -1.3137  -0.7523  -0.5275  -0.1111   2.2028
#>
#> Coefficients:
#>               Estimate Std. Error z value Pr(>|z|)
#> (Intercept) 201.162333 376.824881   0.534  0.59346
#> sex          -0.563411   0.486574  -1.158  0.24690
#> lnrhday       0.040225   0.013519   2.975  0.00293 **
```

```
#> clsize        0.354978   0.220955   1.607  0.10815
#> hyear        -0.113676   0.195048  -0.583  0.56002
#> f            -0.094384   0.045990  -2.052  0.04014 *
#> H1           22.410929  25.406707   0.882  0.37773
#> GTloci        2.574752   2.814479   0.915  0.36028
#> Hloci        -2.746031   3.126325  -0.878  0.37975
#> geno          0.001902   0.006709   0.283  0.77683
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 132.72  on 119  degrees of freedom
#> Residual deviance: 114.79  on 110  degrees of freedom
#> AIC: 134.79
#>
#> Number of Fisher Scoring iterations: 5
```

```
# Predict with cutoff p = 0.5.
glm.probs <- predict(log.fit3, newdata = d.sparrows.test, type = "response")
glm.preds <- ifelse(glm.probs > 0.5, "1", "0")

# Confusion table.
(glm.conf <- table(true = d.sparrows.test$recruit, predicted = glm.preds))
```

```
#>     predicted
#> true  0  1
#>    0 30  4
#>    1 14  1
```

```
sens <- glm.conf[2, 2]/(sum(glm.conf[2, ]))
spes <- glm.conf[1, 1]/sum(glm.conf[1, ])
(spes.sens.log <- c(sensitivity = sens, specificity = spes))
```

```
#> sensitivity specificity
#>  0.06666667  0.88235294
```

The confusion matrix, as well as the sensitivity and the specificity can be seen in the output above.

c)

Same as *b)*, but with QDA instead.

```
# Fit the model.
qda.fit <- qda(recruit ~ . - hisl, data = d.sparrows.train)
summary(qda.fit)
```

```
#>         Length Class  Mode
#> prior     2    -none- numeric
#> counts    2    -none- numeric
#> means    18    -none- numeric
#> scaling 162    -none- numeric
#> ldet      2    -none- numeric
#> lev       2    -none- character
#> N         1    -none- numeric
#> call      3    -none- call
#> terms     3    terms  call
```

```
#> xlevels   1    -none- list
```

```
# Predict with cutoff p = 0.5.
qda.prob <- predict(qda.fit, newdata = d.sparrows.test)$posterior
qda.preds <- ifelse(qda.prob > 0.5, "1", "0")[, 2]  # Choose probs for 1.
qda.classes <- predict(qda.fit, newdata = d.sparrows.test)$class  # This gives the same result.


# Confusion table.
(qda.conf <- table(true = d.sparrows.test$recruit, predicted = qda.preds))
```

```
#>     predicted
#> true  0  1
#>    0 29  5
#>    1 10  5
```

```
sens <- qda.conf[2, 2]/(sum(qda.conf[2, ]))
spes <- qda.conf[1, 1]/sum(qda.conf[1, ])
(spes.sens.qda <- c(sensitivity = sens, specificity = spes))
```

```
#> sensitivity specificity
#>   0.3333333   0.8529412
```

The confusion matrix, as well as the sensitivity and the specificity can be seen in the output above.

d)

Neural network for classification.

First, prepare the data.

```
library(keras)
library(caret)
x_train <- d.sparrows.train[, -c(6, 11)]
x_test = d.sparrows.test[, -c(6, 11)]

mean = apply(x_train, 2, mean)
std = apply(x_train, 2, sd)
x_train = scale(x_train, center = mean, scale = std)
x_test = scale(x_test, center = mean, scale = std)

y_train = as.numeric(d.sparrows.train$recruit)
y_test = as.numeric(d.sparrows.test$recruit)
```

Next, perform the classification with the neural network.

```
set.seed(1234)
# Build the model
model <- keras_model_sequential() %>% layer_dense(units = 32, activation = "relu",
    input_shape = ncol(x_train)) %>% layer_dropout(rate = 0.2) %>% layer_dense(units = 64,
    activation = "relu") %>% layer_dropout(rate = 0.2) %>% layer_dense(units = 1,
    activation = "sigmoid")

model %>% compile(optimizer = "rmsprop", loss = "mse", metrics = c("mae"))

# Train
history <- model %>% fit(x_train, y_train, epochs = 25, batch_size = 16, validation_split = 0.5)
```

```r
# Predict Assuming that the method 'predict_classes' uses cut-off of 0.5.
predictionsNN <- model %>% predict_classes(x_test)

# Confusion table.
(NN.conf <- table(true = d.sparrows.test$recruit, predicted = predictionsNN))
```

```
#>     predicted
#> true  0  1
#>    0 30  4
#>    1 12  3
```

```r
sens <- NN.conf[2, 2]/(sum(NN.conf[2, ]))
spes <- NN.conf[1, 1]/sum(NN.conf[1, ])
(spes.sens.NN <- c(sensitivity = sens, specificity = spes))
```

```
#> sensitivity specificity
#>   0.2000000   0.8823529
```

The confusion matrix, as well as the sensitivity and the specificity can be seen in the output above.

All the different performances are summarized below.

```r
(perf <- data.frame(log = spes.sens.log, qda = spes.sens.qda, NN = spes.sens.NN))
```

```
#>                    log       qda        NN
#> sensitivity 0.06666667 0.3333333 0.2000000
#> specificity 0.88235294 0.8529412 0.8823529
```

As is apparent, all three methods perform a bit differently. Depending on if one wantt to prioritize higher sensitivity or specificity (or both) one can pick and choose between the methods. Their specificities are relatively similar however, while the sensitivities differ quite dramatically.