Exercise #8

Classification - LDA and Logistic Regression

Brian Schweigler; 16-102-071

11/05/2022

Preliminaries

Load the required libraries

```
library(FNN)
```

```
## Warning: package 'FNN' was built under R version 4.0.5
```

```
library(MASS)
```

Set a seed for later:

```
set.seed(1786397)
```

Loading the vertebral dataset, set Status as a factor and show an overview:

```
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))



vertebral_df = read.csv("Vertebral.txt", header = TRUE, sep = ",", comment.char = "#")
vertebral_df$Status <-
    factor(
        vertebral_df$Status,
        levels = c("Normal", "Abnormal"),
        labels = c("Normal", "Abnormal")
    )
summary(vertebral_df)
```

```
##    Incidence          Tilt             Angle            Slope
##  Min.   : 26.15   Min.   :-6.555   Min.   : 14.00   Min.   : 13.37
##  1st Qu.: 46.43   1st Qu.:10.667   1st Qu.: 37.00   1st Qu.: 33.35
##  Median : 58.69   Median :16.358   Median : 49.56   Median : 42.40
##  Mean   : 60.50   Mean   :17.543   Mean   : 51.93   Mean   : 42.95
##  3rd Qu.: 72.88   3rd Qu.:22.120   3rd Qu.: 63.00   3rd Qu.: 52.70
##  Max.   :129.83   Max.   :49.432   Max.   :125.74   Max.   :121.43
##      Radius          Degree              Status
##  Min.   : 70.08   Min.   :-11.058    Normal  :100
##  1st Qu.:110.71   1st Qu.:  1.604    Abnormal:210
##  Median :118.27   Median : 11.768
##  Mean   :117.92   Mean   : 26.297
##  3rd Qu.:125.47   3rd Qu.: 41.287
##  Max.   :163.07   Max.   :418.543
```

As is mentioned in the PDF, no NAs are within the data. The range of the values can't be gauged without further information.

1a. Use logistic regression to predict variable Status.

Incidence, Tilt, Angle, Slope Radius, Degree are all derived from the shape and orientation of the pelvis. Thus we do not need all of them to predict status and will instead only use Incidence, Radius and Degree:

```
vert_df = subset(vertebral_df, select = c("Status", "Incidence", "Radius", "Degree"))
```

Now we divide this into test (30%) and train (70%) data set:

```
training = round(nrow(vert_df)*0.7)
training_index = sample( c(1:nrow(vert_df)), training)
training_data = vert_df[training_index,]
summary(training_data)
```

```
##       Status      Incidence          Radius           Degree
##  Normal   : 64   Min.   : 30.15   Min.   : 70.08   Min.   :-11.058
##  Abnormal:153   1st Qu.: 47.32   1st Qu.:110.86   1st Qu.:  1.663
##                 Median : 59.17   Median :118.69   Median : 12.393
##                 Mean   : 60.59   Mean   :118.35   Mean   : 25.346
##                 3rd Qu.: 72.56   3rd Qu.:125.43   3rd Qu.: 40.511
##                 Max.   :115.92   Max.   :163.07   Max.   :148.754
```

```
testing_data = vert_df[-training_index,]
summary(testing_data)
```

```
##       Status     Incidence          Radius          Degree
##  Normal   :36   Min.   : 26.15   Min.   : 79.0   Min.   :-10.0931
##  Abnormal:57   1st Qu.: 45.54   1st Qu.:110.7   1st Qu.:  0.9709
##                Median : 56.10   Median :117.2   Median :  7.0448
##                Mean   : 60.28   Mean   :116.9   Mean   : 28.5153
##                3rd Qu.: 74.85   3rd Qu.:125.5   3rd Qu.: 42.8105
##                Max.   :129.83   Max.   :145.6   Max.   :418.5431
```

Now we can perform the logistic regression model:

```
vert_df_LR = glm(Status ~ ., data = training_data, family = binomial(link = "logit"))
summary(vert_df_LR)
```

```
##
## Call:
## glm(formula = Status ~ ., family = binomial(link = "logit"),
##     data = training_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.87663  -0.53112   0.04795   0.31713   1.95500
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 12.78065    3.63976   3.511 0.000446 ***
## Incidence   -0.03701    0.01882  -1.966 0.049281 *
## Radius      -0.09522    0.02555  -3.727 0.000194 ***
## Degree       0.15796    0.03026   5.220 1.79e-07 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 263.22  on 216  degrees of freedom
## Residual deviance: 136.50  on 213  degrees of freedom
## AIC: 144.5
##
## Number of Fisher Scoring iterations: 8
```

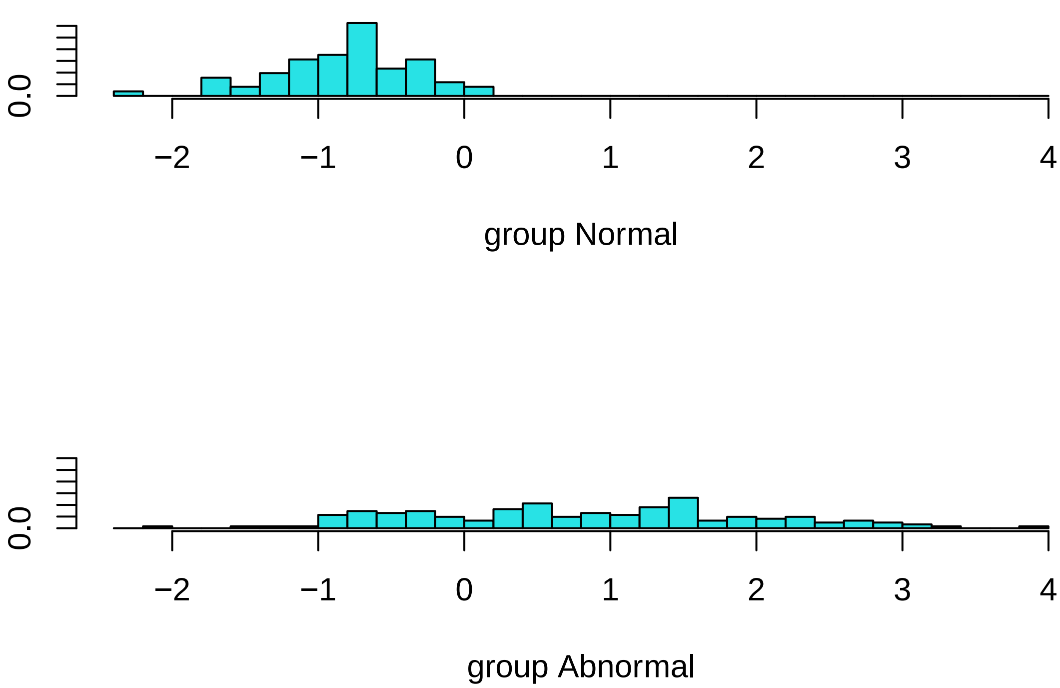
Degree seems the most useful out of the 3 looked at, while Radius is also useful but Incidence is not a good predictor of status on its own.

1b. Use LDA to predict the variable Status

```
vert_df_LDA = lda(Status ~ ., data = training_data)
vert_df_LDA
```

```
## Call:
## lda(Status ~ ., data = training_data)
##
## Prior probabilities of groups:
##    Normal  Abnormal
## 0.2949309 0.7050691
##
## Group means:
##          Incidence   Radius    Degree
## Normal    51.49257 124.2805  1.469409
## Abnormal  64.39495 115.8644 35.333417
##
## Coefficients of linear discriminants:
##                    LD1
## Incidence -0.007574224
## Radius    -0.051407494
## Degree     0.038566255
```

```
plot(vert_df_LDA)
```



From the coefficients, we can tell that radius has the highest impact in LDA on the model. Its impact is higher than the sum of incidence and degree (as coefficients of Incidence + Degree < Coefficient of Radius).

3. Compare the predictions you obtained with the logistic regression and LDA. Use a fair methodology to compare the classifiers (and explain your choice). Can you estimate the error rate for those strategies? Which classifier is the best? Why?

Some stats for the LR model:

```
LR_Test_Output = predict(vert_df_LR, testing_data, type = "response")
threshold = 0.5
LR_Test_Results = as.factor(ifelse(LR_Test_Output < threshold, "Abnormal", "Normal"))
LR_Correct_Pred = sum (testing_data$Status == LR_Test_Results)
LR_Correct_Pred
```

```
## [1] 21
```

```
LR_Accuracy = LR_Correct_Pred / nrow(testing_data)
LR_Accuracy
```

```
## [1] 0.2258065
```

Stats for the LDA model:

```
LDA_Test_Output = predict(vert_df_LDA, testing_data, type = "response")$class
LDA_Correct_Predictions = sum(testing_data$Status == LDA_Test_Output)
LDA_Correct_Predictions
```

```
## [1] 72
```

```
LDA_Accuracy = LDA_Correct_Predictions/nrow(testing_data)
LDA_Accuracy
```

```
## [1] 0.7741935
```

The LDA has a way higher accuracy than the LR. Removing "Incidence" from the LR (or changing the threshold) might improve its performance, but in this case the LDA is the better choice.

Alternatively we can also use the confusion matrix:

```
confusion_mat_LR <-
    table(testing_data$Status, LR_Test_Results)[2:1, 2:1]
confusion_mat_LR
```

```
##             LR_Test_Results
##              Normal Abnormal
##    Abnormal      49        8
##    Normal        13       23
```

```
TP_LR = confusion_mat_LR[1]
TN_LR = confusion_mat_LR[4]
FP_LR = confusion_mat_LR[2]
FN_LR = confusion_mat_LR[3]

precision_LR = TP_LR / (TP_LR + FP_LR)
print(sprintf("Precision = %f", precision_LR))
```

```
## [1] "Precision = 0.790323"
```

```
recall_LR = TP_LR / (TP_LR + FN_LR)
print(sprintf("Recall = %f", recall_LR))
```

```
## [1] "Recall = 0.859649"
```

```r
confusion_mat_LDA <-
    table(testing_data$Status, LDA_Test_Output)[2:1, 2:1]
confusion_mat_LDA
```

```
##           LDA_Test_Output
##            Abnormal Normal
##   Abnormal       52      5
##   Normal         16     20
```

```r
TP_LDA = confusion_mat_LDA[1]
TN_LDA = confusion_mat_LDA[4]
FP_LDA = confusion_mat_LDA[2]
FN_LDA = confusion_mat_LDA[3]

precision_LDA = TP_LDA / (TP_LDA + FP_LDA)
print(sprintf("Precision = %f", precision_LDA))
```

```
## [1] "Precision = 0.764706"
```

```r
recall_LDA = TP_LDA / (TP_LDA + FN_LDA)
print(sprintf("Recall = %f", recall_LDA))
```

```
## [1] "Recall = 0.912281"
```

If we go with precision and recall, we see that LDA has better recall but its precision is worse than LR. Even more interesting is the fact that the precision of the LDA is roughly its accuracy, but the accuracy of the LR is much, much worse. The LR might require some fine-tuning before I'd vouch for it to be publicly used.