Exercise 6

Nikolaus Czernin

```r
library("MASS")
library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
```

```r
# install.packages("ROCit")
library("ROCit")
library("knitr")

# install.packages("klaR")
library("klaR")

set.seed(11721138)
```

```r
# loading data
load("Loan.Rdata")
```

```r
Loan %>% head()
```

```
##   Amount Term IntRate   ILR EmpLen     Home Income Status    Score
## 1  67.57   36  0.1838 0.035      D     RENT 126400     CO 200.9581
## 2  22.97   36  0.1198 0.032      D     RENT  30900     CO 179.6058
## 3  54.05   36  0.1166 0.032      D MORTGAGE 111900     FP 161.7622
## 4  24.32   36  0.1733 0.034      A     RENT  66000     FP 196.6619
## 5  43.24   36  0.1723 0.034      A MORTGAGE  71900     CO 203.4912
## 6  16.22   36  0.1355 0.033      B     RENT  27614     FP 186.3070
```

```r
Loan <- Loan %>%
  # Scale only numeric columns
  mutate(across(where(is.numeric), ~ if (sd(.) > 0) scale(.) else .)) %>%
  # remove the constant variable Term
```

```r
  dplyr::select(-Term) %>%
  # remove the variables with colearity
  dplyr::select(-Score) %>%
  # 1-hot encode the response levels
  mutate(Status = ifelse(Status == "CO", 1, 0))
```

LDA assumes that variable are normally distributed and therfore, scaling should not be required. To be safe, I scaled all numeric variables anyway. I removed Term because it is constant. I also removed Score because is is highly correlated with other variables.

```r
N <- nrow(Loan)
train_ids <- sample(1:N, (N %/% 3) * 2)

train <- Loan[train_ids, ]

test <- Loan[-train_ids, ]

summary(Loan)
```

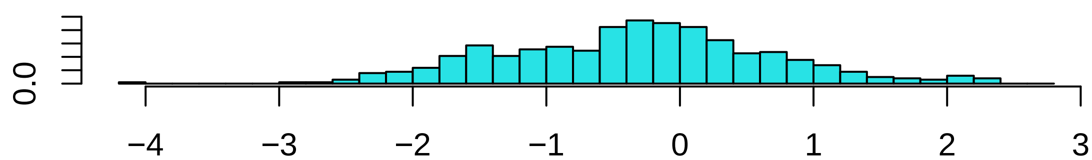
```
##       Amount.V1           IntRate.V1            ILR.V1         EmpLen
##  Min.   :-1.4501320   Min.   :-1.740472   Min.   :-1.885735   A:198
##  1st Qu.:-0.7410807   1st Qu.:-0.772345   1st Qu.:-0.816657   B:198
##  Median :-0.3257248   Median :-0.040650   Median :-0.282118   C:141
##  Mean   : 0.0000000   Mean   : 0.000000   Mean   : 0.000000   D:305
##  3rd Qu.: 0.4281549   3rd Qu.: 0.611405   3rd Qu.: 0.786960   U: 58
##  Max.   : 2.7965501   Max.   : 3.224602   Max.   : 3.459655
##        Home           Income.V1           Status
##  MORTGAGE:429   Min.   :-1.359480   Min.   :0.0000
##  OWN     : 95   1st Qu.:-0.646346   1st Qu.:0.0000
##  RENT    :376   Median :-0.220695   Median :0.0000
##                 Mean   : 0.000000   Mean   :0.1456
##                 3rd Qu.: 0.311927   3rd Qu.:0.0000
##                 Max.   : 9.562607   Max.   :1.0000
```

```r
eval_ <- function(y, yhat){
  conf.mat <- table(y, yhat)
  TP <- conf.mat[2, 2]
  FP <- conf.mat[1, 2]
  FN <- conf.mat[2, 1]
  TN <- conf.mat[1, 1]
  return (list(TP=TP, FP=FP, FN=FN, TN=TN))
}

# misclassification rate: (FP+FN)/(FP+TN+FN+TP)
MR <- function(y, yhat){
  n <- length(y)
  metrics <- eval_(y, yhat)
  (metrics$FP + metrics$FN) / n
}

# balanced accuracy: (TPR+TNR)/2
BACC <- function(y, yhat){
```

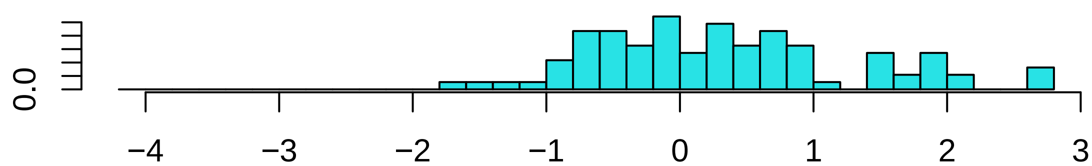
```
  metrics <- eval_(y, yhat)
  TPR <-  metrics$TP / (metrics$TP + metrics$FN)
  TNR <-  metrics$TN / (metrics$TN + metrics$FP)
  (TPR + TNR) / 2
}
```

LDA

```
model.lda <- lda(Status~., data=train)
plot(model.lda)
```



group 0



group 1

```
prediction <- predict(model.lda, train)$class
observed <- train$Status
MR(observed, prediction) %>% round(4) %>% paste("Misclassification rate:", .)
```

```
## [1] "Misclassification rate: 0.1483"
```

```
BACC(observed, prediction) %>% round(4) %>% paste("Balanced accuracy", .)
```

```
## [1] "Balanced accuracy 0.5163"
```

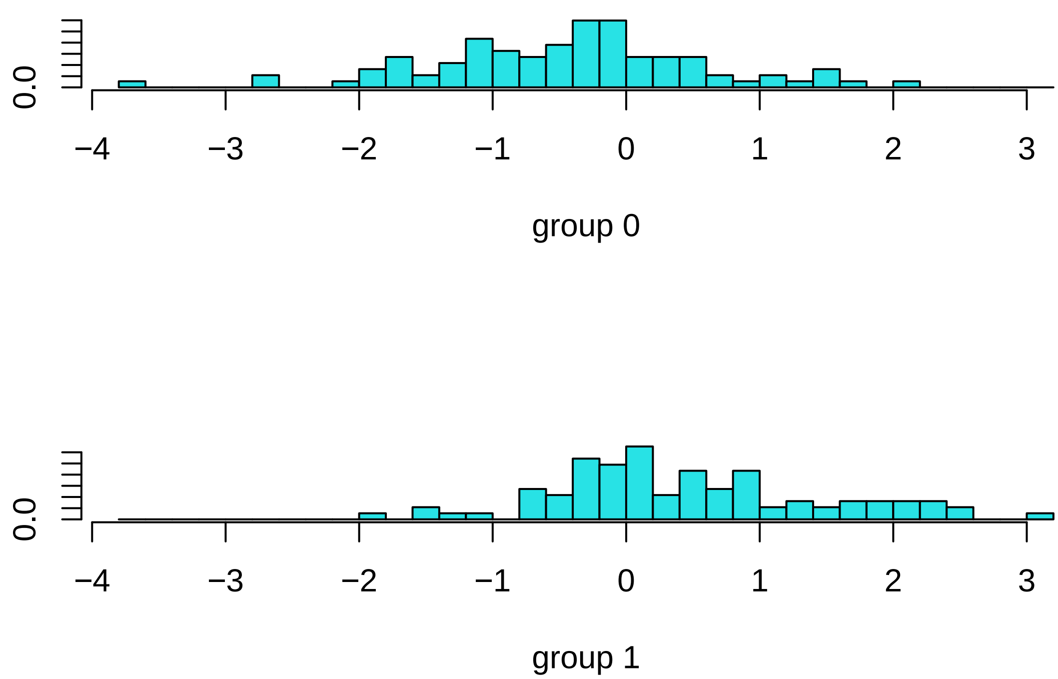
```r
table(observed, prediction)
```

```
##          prediction
## observed   0   1
##        0 508   0
##        1  89   3
```

LDA with undersampled balancing

```r
# get the number of the smaller group of classes
n.min <- min(train %>% filter(Status==1) %>% nrow(), train %>% filter(Status==0) %>% nrow())
# now sample both groups in the training data and create the undersampled training dataset
train.us <- rbind(
  train %>% filter(Status==0) %>% sample_n(n.min),
  train %>% filter(Status==1) %>% sample_n(n.min)
)
```

```r
model.lda.us <- lda(Status~., data=train.us)
plot(model.lda.us)
```



```r
prediction.us <- predict(model.lda.us, train)$class
observed.us <- train$Status
MR(observed.us, prediction.us) %>% round(4) %>% paste("Misclassification rate (undersampled):", .)
```

```
## [1] "Misclassification rate (undersampled): 0.405"
```

```r
BACC(observed.us, prediction.us) %>% round(4) %>% paste("Balanced accuracy (undersampled)", .)
```

```
## [1] "Balanced accuracy (undersampled) 0.614"
```

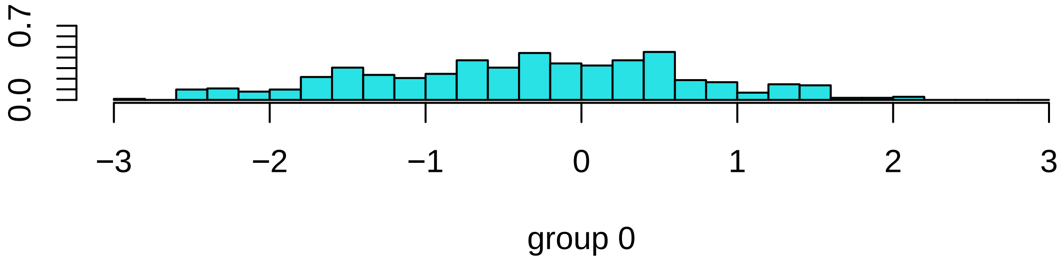
```r
table(observed.us, prediction.us)
```

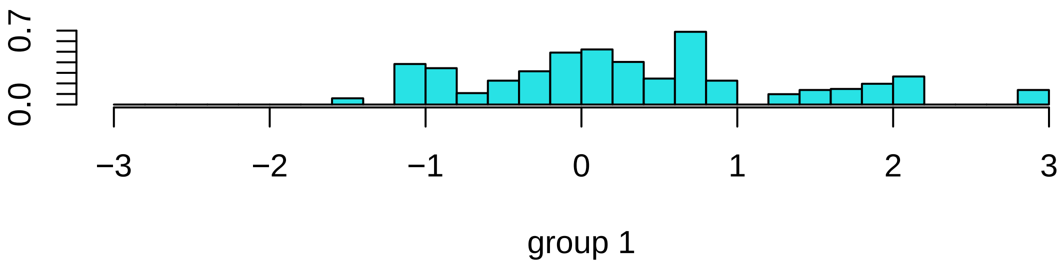
```
##              prediction.us
## observed.us    0   1
##             0 298 210
##             1  33  59
```

LDA with oversampled balancing

```r
# get the number of the smaller group of classes
n.max <- max(train %>% filter(Status==1) %>% nrow(), train %>% filter(Status==0) %>% nrow())
# now sample both groups in the training data and create the undersampled training dataset
train.os <- rbind(
  train %>% filter(Status==0) %>% sample_n(n.max, replace = T),
  train %>% filter(Status==1) %>% sample_n(n.max, replace = T)
)
```

```r
model.lda.os <- lda(Status~., data=train.os)
plot(model.lda.os)
```






```r
prediction.os <- predict(model.lda.os, train)$class
observed.os <- train$Status
MR(observed.os, prediction.os) %>% round(4) %>% paste("Misclassification rate (oversampled):", .)
```

```
## [1] "Misclassification rate (oversampled): 0.375"
```

```r
BACC(observed.os, prediction.os) %>% round(4) %>% paste("Balanced accuracy (oversampled)", .)
```

```
## [1] "Balanced accuracy (oversampled) 0.6094"
```

```r
table(observed.os, prediction.os)
```

```
##            prediction.os
## observed.os   0   1
##           0 321 187
##           1  38  54
```

Interestingly, the oversampling-balanced training data has both a lower misclassification rate AND a lower balanced accuracy, but only by a small margin. Generally, they judged only slightly differently, so I would be reluctant to judge either to be better. Generally, they outperformed the model trained on the unbalanced data.

Quadratic Discriminant Analysis

```
model.qda <-    qda(Status~., data=train.us)
model.qda.us <- qda(Status~., data=train.us)
model.qda.os <- qda(Status~., data=train.os)

yhat.qda <-    predict(model.qda, train)$class
yhat.qda.us <- predict(model.qda.us, train)$class
yhat.qda.os <- predict(model.qda.os, train)$class

observed <- train$Status

data.frame(
  Data=c("Unbalanced", "Undersampling-balanced", "Oversampling-balanced"),
  `Misclassification Rate` =c(
    MR(observed, yhat.qda) %>% round(4),
    MR(observed, yhat.qda.us) %>% round(4),
    MR(observed, yhat.qda.os) %>% round(4)
  ),
  `Balanced Accuracy`=c(
    BACC(observed, yhat.qda) %>% round(4),
    BACC(observed, yhat.qda.us) %>% round(4),
    BACC(observed, yhat.qda.os) %>% round(4)
  )
) %>% kable(caption="QDA results")
```

Table 1: QDA results

Data	Misclassification.Rate	Balanced.Accuracy
Unbalanced	0.4350	0.6096
Undersampling-balanced	0.4350	0.6096
Oversampling-balanced	0.3883	0.6550

```
table(observed.os, yhat.qda)
```

```
##            yhat.qda
## observed.os   0   1
##           0 277 231
##           1  30  62
```

```
table(observed.os, yhat.qda.us)
```

```
##            yhat.qda.us
## observed.os   0   1
##           0 277 231
##           1  30  62
```

```r
table(observed.os, yhat.qda.os)
```

```
##              yhat.qda.os
## observed.os    0   1
##             0 301 207
##             1  26  66
```

Here again, oversampling has both a higher misclassification rate and a higher balanced accuracy than the other methods. The undersampling-balanced model performed equally as the unbalanced model.

Regularized Discrimant Analysis

```r
model.rda <- rda(Status~., data=train.us)
model.rda.us <- rda(Status~., data=train.us)
model.rda.os <- rda(Status~., data=train.os)

yhat.rda <- predict(model.rda, train)$class
yhat.rda.us <- predict(model.rda.us, train)$class
yhat.rda.os <- predict(model.rda.os, train)$class

observed <- train$Status

data.frame(
  Data=c("Unbalanced", "Undersampling-balanced", "Oversampling-balanced"),
  `Misclassification Rate` =c(
    MR(observed, yhat.rda) %>% round(4),
    MR(observed, yhat.rda.us) %>% round(4),
    MR(observed, yhat.rda.os) %>% round(4)
  ),
  `Balanced Accuracy`=c(
    BACC(observed, yhat.rda) %>% round(4),
    BACC(observed, yhat.rda.us) %>% round(4),
    BACC(observed, yhat.rda.os) %>% round(4)
  ),
  Gamma=c(
    model.rda$regularization[1],
    model.rda.us$regularization[1],
    model.rda.os$regularization[1]
  ),
  Lambda=c(
    model.rda$regularization[2],
    model.rda.us$regularization[2],
    model.rda.os$regularization[2]
  )
) %>% kable(caption="rda results")
```

Table 2: rda results

Data	Misclassification.Rate	Balanced.Accuracy	Gamma	Lambda
Unbalanced	0.4117	0.6100	0.9661390	0.9224731
Undersampling-balanced	0.4167	0.6026	0.8772991	0.8281936
Oversampling-balanced	0.3550	0.6034	0.6041179	0.9986638

```r
table(observed.os, yhat.rda)
```

```
##              yhat.rda
## observed.os    0   1
##            0 294 214
##            1  33  59
```

```r
table(observed.os, yhat.rda.us)
```

```
##              yhat.rda.us
## observed.os    0   1
##            0 292 216
##            1  34  58
```

```r
table(observed.os, yhat.rda.os)
```

```
##              yhat.rda.os
## observed.os    0   1
##            0 337 171
##            1  42  50
```

```r
# ?qda
```

Both balanced models, again, have higher misclassifciation rates and balanced accuracies than the model trained on the unbalanced data.
The balanced accuracy of the oversampling-balanced model this time around has a higher advantage on the other's than in the previous methods.

The hyperparameters `lambda` and `gamma` determine the assumptions made about the covariances of the different groups and how they interact.

We did not specify any values for the parameters, so the function uses simulated annealing to tune them to minimize the misclassification rate.

In all 3 cases, lambda was very high, meaning that generally the models shrink the covariance matrices down to be diagonal and assume common covariance across the 2 groups.
Except for the oversampling-balanced dataset model, the models also converge at a high gamma, meaning they assume a linear independence of the variables, so they are not correlated.
For the oversampling-balanced dataset model, gamma was = 0.6, so the model assumed some linear dependence between the variables, but tended more to independence.