Computer Lab 3

Introduction to Machine Learning

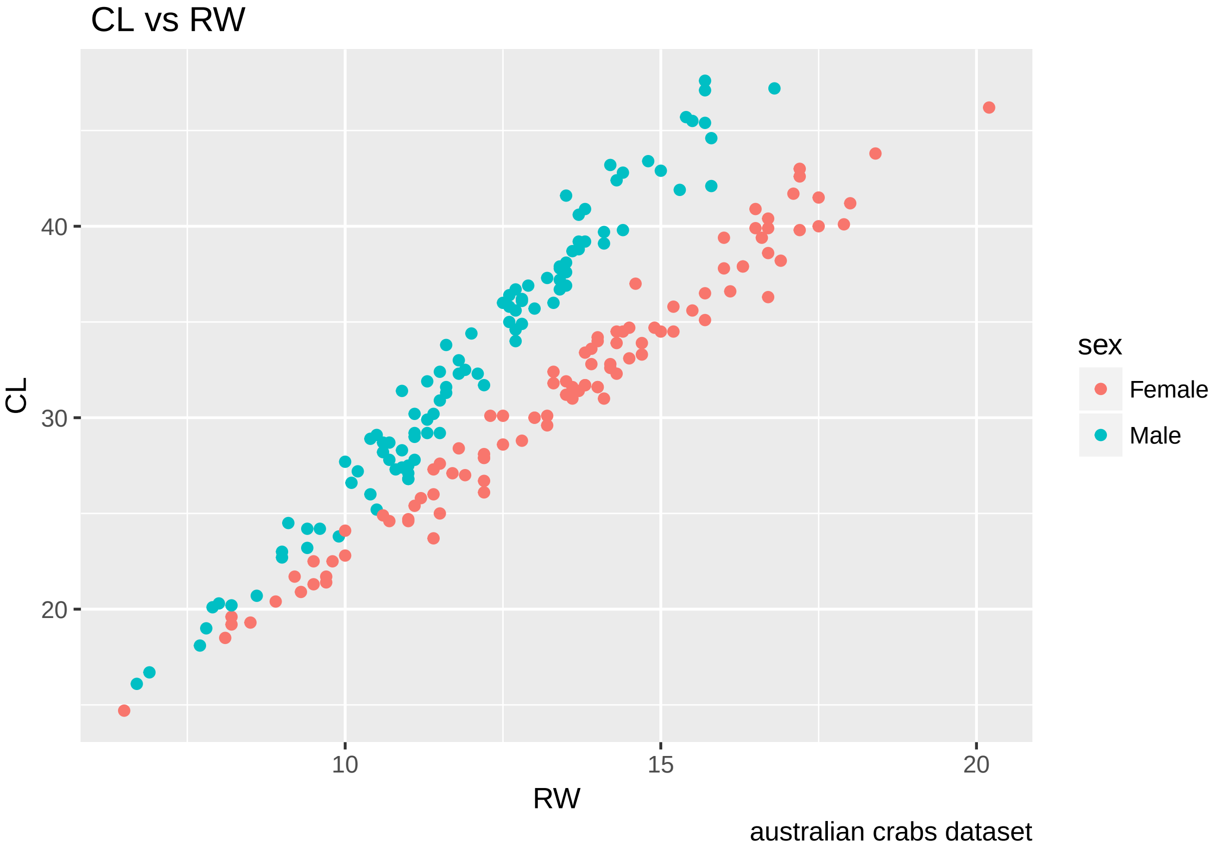
Emil K Svensson

2016-11-25

Assignment 1

1.1

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```



A line would be able to separate the genders pretty well. One could expect a couple of missclassifications when the variables has low values as the genders seem to have less separation there.

1.2

```
LDA <- function(X) {

    RW <- X[, 1]
    CL <- X[, 2]
    sex <- X[, 3]
    myMu <- aggregate(cbind(RW, CL), by = list(sex), FUN = mean, simplify = TRUE)
    myCov <- by(cbind(RW, CL), list(sex), cov, method = "pearson")
    myPi <- aggregate(cbind(RW, CL), by = list(sex), FUN = function(x) length(x)/nrow(cbind(RW,
        CL)), simplify = TRUE)


    mySig <- ((myCov[[1]] * myPi[2, 2] * length(RW)) + (myCov[[2]] * myPi[2,
        3] * length(RW)))/nrow(X)


    woMale <- -0.5 * as.matrix(myMu[2, 2:3], ncol = 2) %*% solve(mySig) %*%
        t(myMu[2, 2:3]) + log(myPi[2, 3])
    woFem <- -0.5 * (as.matrix(myMu[1, 2:3], ncol = 2)) %*% solve(mySig) %*%
        t(myMu[1, 2:3]) + log(myPi[1, 3])


    wM <- solve(mySig) %*% t(myMu[2, 2:3])
    wF <- solve(mySig) %*% t(myMu[1, 2:3])


    a <- (woMale - woFem)
    b <- wM - wF
    x <- cbind(X[, 1:2])

    # w0s is a w1s is b[1] w2s is b[2]
    myInter <- as.numeric(-a/b[2])
    mySlope <- as.numeric(-b[1]/b[2])

    X$myClass <- t(ifelse((a[1] + t(b) %*% t(x)) > 0, levels(X[, 3])[2], levels(X[,
        3])[1]))
    colnames(X)[4] <- "Predicted"
    retObj <- list(w0 = c(woMale, woFem), w1 = cbind(wM = wM, wF = wF), myClass = X,
        myModel = c(myInter = myInter, mySlope = mySlope))

    return(retObj)
}
```

The return object of the LDA-function returns a list with all answers and the decision boundary.

To get the decision boundary we have the set the two discriminantfunctions equal to each other and solve for one of the parameters depending on X so that we get where they intercect.

$\delta_k(x) = x^T \Sigma^{-1} \mu_k + (-1/2) \Sigma^{-1} \mu_k + log(\pi_k)$

$w_i = \Sigma^{-1} \mu_i$ $w_{oi} = (-1/2) \Sigma^{-1} \mu_k + log(\pi_k)$

$\delta_{male}(x) = \delta_{female}(x)$

$\delta_{male}(x) - \delta_{female}(x) = 0$

$x^T(w_{Male} - w_{Female}) + (w_{0Male} - w_{0Female}) = 0$

$x_{CL}^T(w_{Male} - w_{Female}) + x_{RW}^T(w_{Male} - w_{Female}) + (w_{0Male} - w_{0Female}) = 0$

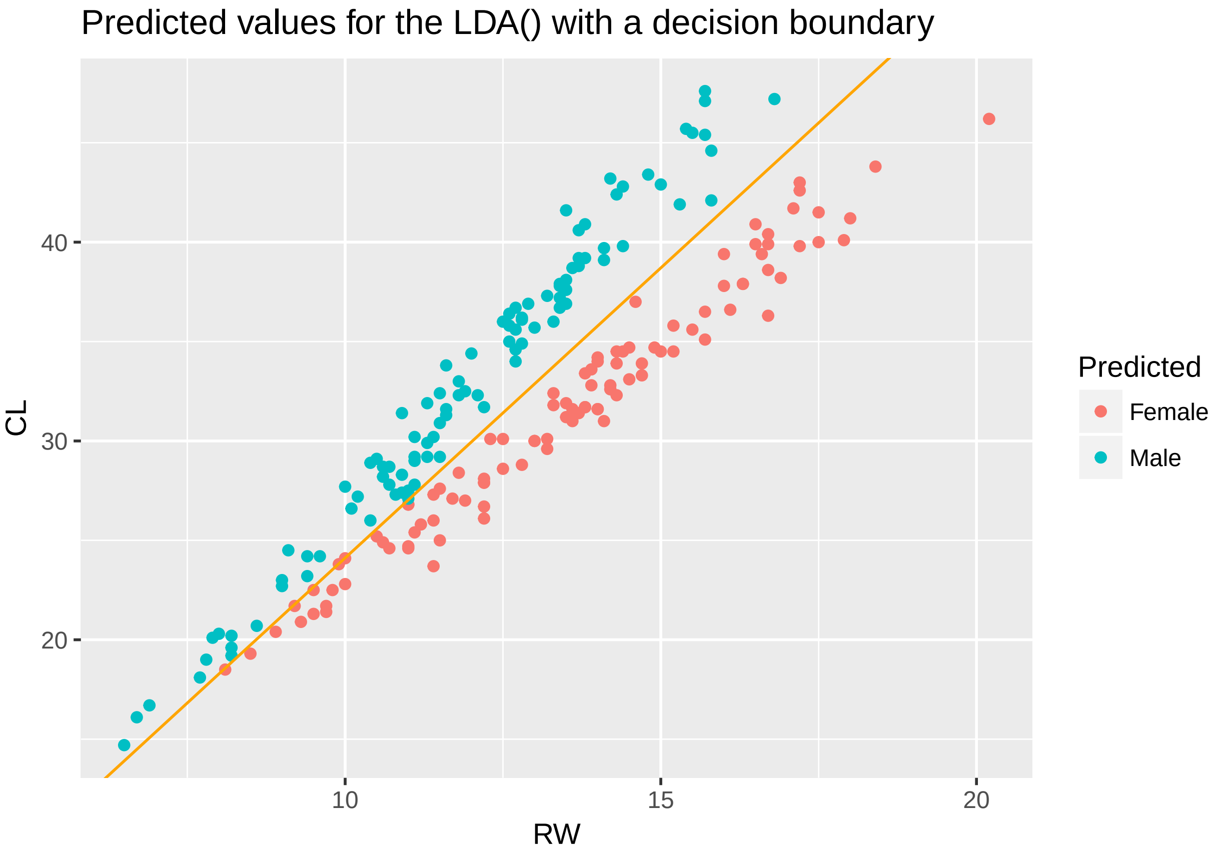
$x_{RW}(w_{Male} - w_{Female}) + (w_{0Male} - w_{0Female}) = -x_{CL}(w_{Male} - w_{Female})$

And we arrive the final result as.

$\frac{(x_{RW}(w_{Male} - w_{Female}) + (w_{0Male} - w_{0Female}))}{-(w_{Male} - w_{Female})} = x_{CL}$

This is expressed as y = CL and x = RW, which means this decision boundary needs to be plottet with the right variable on the right axis.

1.3



The line was calculated to

$$-5.0656387 + x_{RW}^T(2.9180154) = x_{CL}^T$$

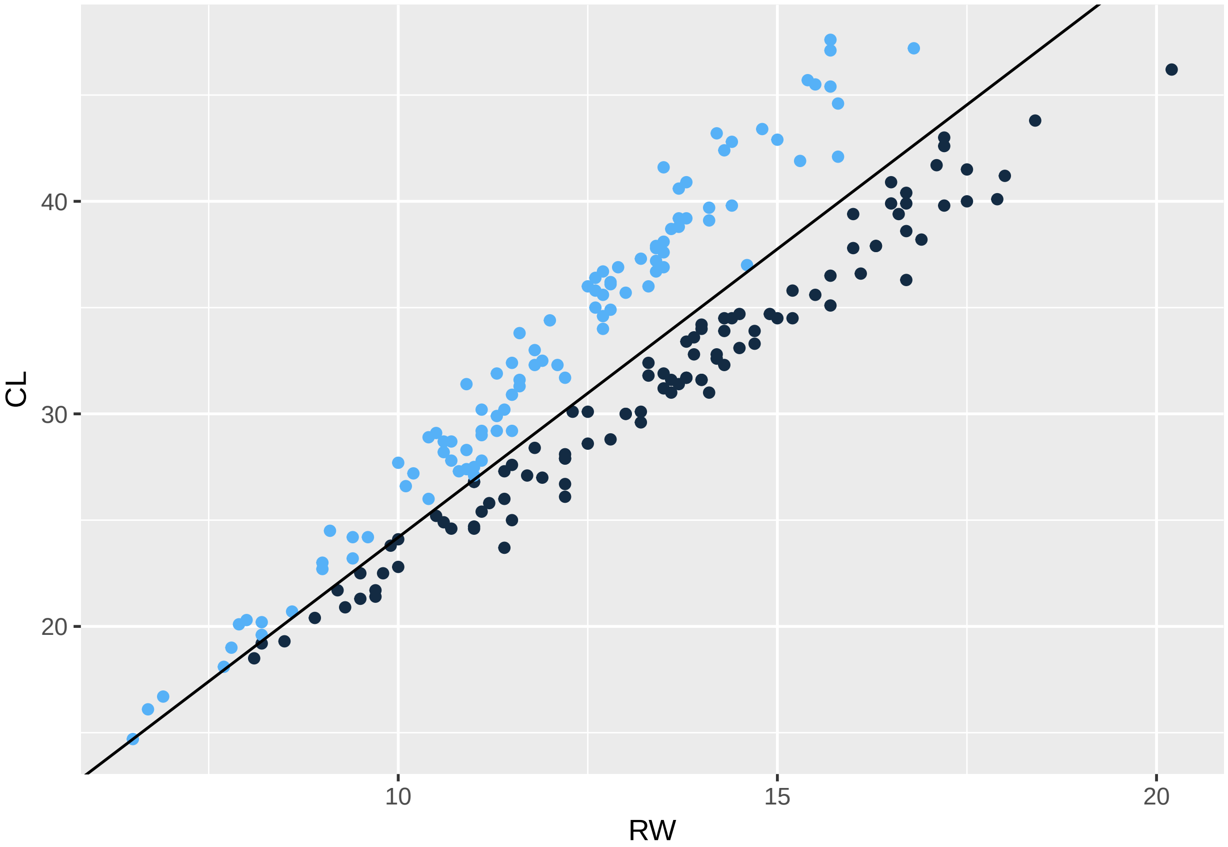
The decision line divides the data nicely but i has some issiues when RW is below 12 where the two groups are closer in distance.

The two discriminant functions were calculated to

$$\delta_{male}(x) = x^T(-12.5634175) + 2.5658514$$
$$\delta_{female}(x) = x^T(-22.4287694) + (-0.2138144)$$

1.4

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



One visible difference in the plots are that an observation located close to the line at CL = 37 and RW = 14 is now classified as a male wheras it was classified as a female. Other than this it is hard to disguinsh any visible diffrences.

The line was calculated as follows.

$$\frac{b_0 + b_{RW}}{b_{CL}}$$

```
## For the Logistic regression
```

```
##          Predicted
## Observed Female Male
##   Female     97    3
##   Male        4   96
```

```
## For the LDA:
```

```
##          Predicted
## Observed Female Male
##   Female     97    3
##   Male        4   96
```

Both classifiers has the same missclassificationrate 7/200 and has the same amount (but not necessery the same) of missclassifications in the anti-diagonals for the different categories.

Assignment 2

2.1

2.2

	Train	Test
Deviance	0.212	0.248
Gini	0.240	0.304

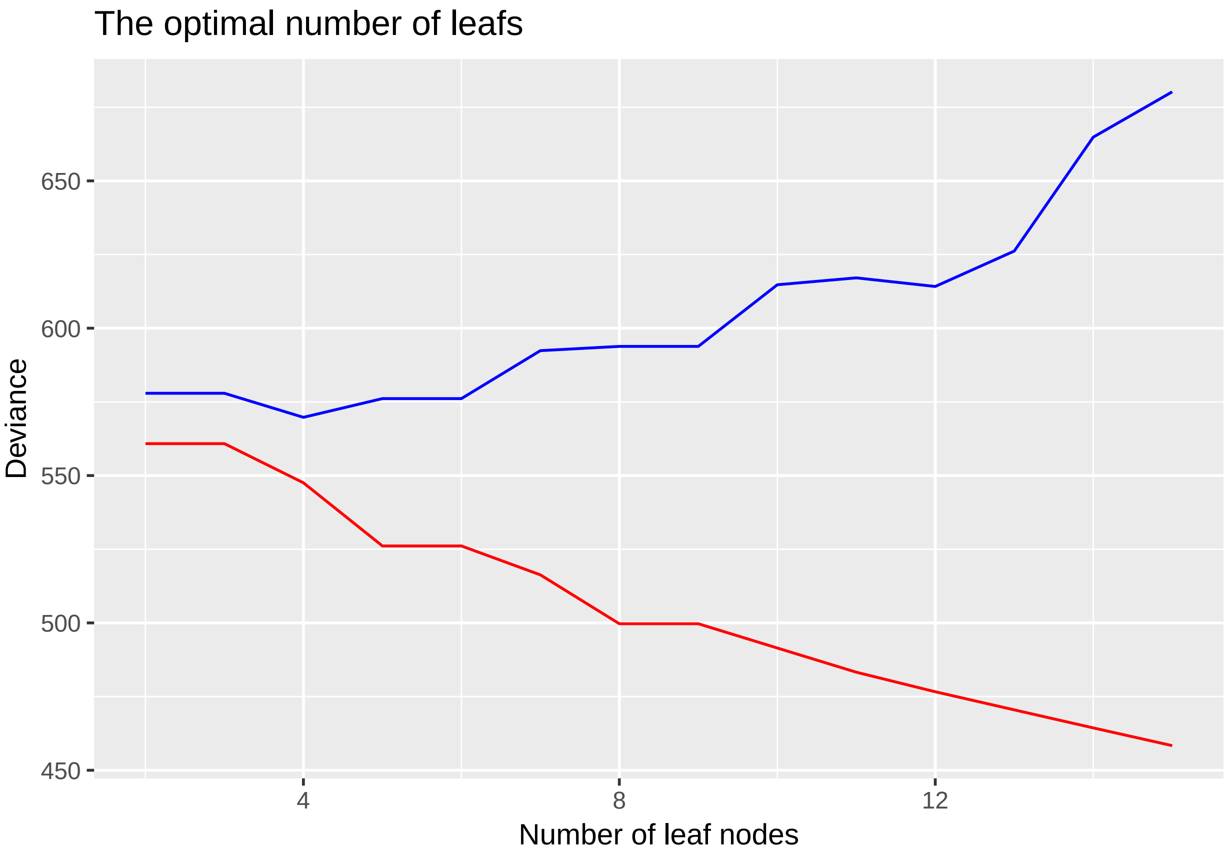
The Gini splitting criterion has a higher training and missclassification rate than the deviance splitting criterion. So for the following step the deviance criterion is used.

2.3

```r
valid23 <- data.frame(trainS = 1, testS = 1)
tDev2 <- tree(good_bad ~ ., data = csTrain, split = "deviance")
rad <- 1

for (i in 2:15) {
    tDev22 <- prune.tree(tDev2, best = i)
    valid23[rad, 1] <- deviance(tDev22)
    valid23[rad, 2] <- 2 * deviance(predict(tDev22, newdata = csValid, type = "tree"))
    rad <- rad + 1
}

valid23$best <- 2:15
ggplot(data = valid23, aes(x = best)) + geom_line(aes(y = trainS), col = "red") +
    geom_line(aes(y = testS), col = "blue") + labs(title = "The optimal number of leafs",
    x = "Number of leaf nodes", y = "Deviance")
```



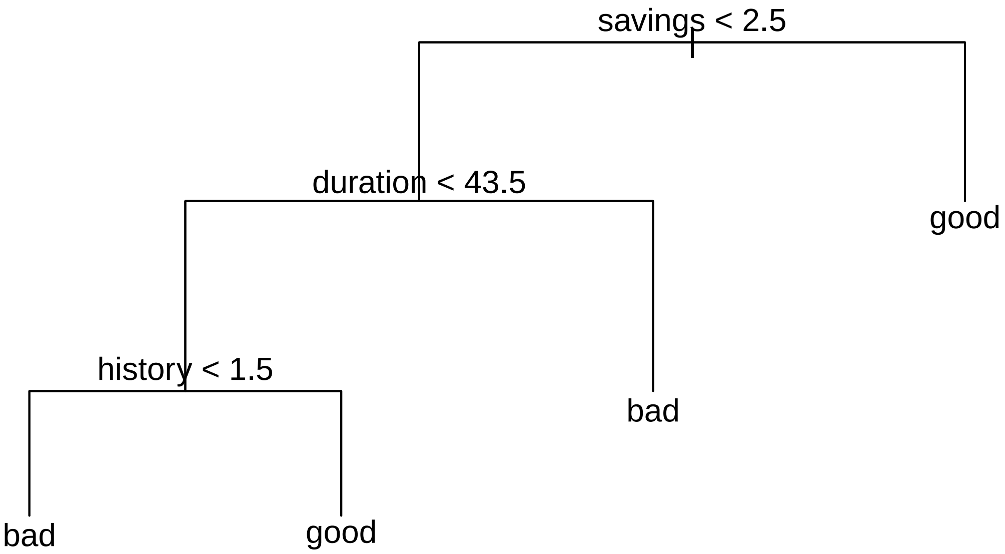
Validation is represented as the blue line and Traning as the red line, the deviance-fuction uses a summation over all observations in each data set. Since the validation is half the number of observations the validation deviance is doubled to get a more logical graph to interpret in a bias/variance trade off way.

The optimal tree is the the one with 4 leafs is the optimal tree as it has the lowest validation deviance weher the training deviance is the lowest before it starts to rise again.

```
##
## Classification tree:
## snip.tree(tree = tDev2, nodes = c(5L, 3L, 9L))
## Variables actually used in tree construction:
## [1] "savings"  "duration" "history"
## Number of terminal nodes:  4
## Residual mean deviance:  1.117 = 547.5 / 490
## Misclassification error rate: 0.251 = 124 / 494
```

```
## [1] "Missclassification rate for test data 0.248"
```

The tree uses the variables Savings, duration and history for the decision boundaries. The missclassification rate is 0.251 which is worse than the initial model, although the mcr for the test (2.48) is actualy lower than the mcr for the traning data.



We do not know about the variables so it is hard to see if these are resonable splits. One could guess that people with less than 2.5 (k is my interpretation) in savings are probably a person with a more straind economy and there for more likely to default.

2.4

```
## Training data
```

```
##          Observed
## Predicted   bad  good
##      bad  0.190 0.196
##      good 0.104 0.510
```

```
## [1] 0.3
```

```
##
##  Test data
```

```
##          Observed
## Predicted   bad  good
##      bad  0.208 0.212
##      good 0.088 0.492
```

```
## [1] 0.3
```

The mcr is the same for both traning and test data, In the confusion-matrix the proportions are displayed instead of the raw numbers. These proportions looks to be very similary distrubuted compared to each other. Overall these predictions are worse than the decision tree in 2.3.

2.5

```
##          Observed
## Predicted bad good
##       Bad  137  263
##       Good  10   90
```

```
## [1] 0.546
```

```
##          Observed
## Predicted bad good
##       Bad   66  134
##       Good   8   42
```

```
## [1] 0.568
```

We now classify more observations as Bad although they are Good. That has to do with the Lossfunction we implemented that says that we need to be 10 times as sure than normal. Because of this every observation that would normaly be classified as "good" when the probability for good is larger than bad now needs to be 10 times larger than bad for it to be good.

This is a powerfull tool if we want to minimize a certain amount of False positives or True positives depending on the case. In this case a bank don't want bad lenders to be predicted as good people as that would be more hurtful for the bussiness and the bank don't want to give people the benefit of the doubt and rather take secure earnings.

Code

```
knitr::opts_chunk$set(echo = FALSE, tidy = TRUE)
library(knitr)
crabs <- read.csv("australian-crabs.csv")
library(ggplot2)
p <- ggplot(data = crabs) + geom_point(aes(x = RW, y = CL, col = sex)) + labs(title = " CL vs RW",
    caption = "australian crabs dataset")
plot(p)

LDA <- function(X) {

    RW <- X[, 1]
    CL <- X[, 2]
    sex <- X[, 3]
    myMu <- aggregate(cbind(RW, CL), by = list(sex), FUN = mean, simplify = TRUE)
    myCov <- by(cbind(RW, CL), list(sex), cov, method = "pearson")
    myPi <- aggregate(cbind(RW, CL), by = list(sex), FUN = function(x) length(x)/nrow(cbind(RW,
        CL)), simplify = TRUE)


    mySig <- ((myCov[[1]] * myPi[2, 2] * length(RW)) + (myCov[[2]] * myPi[2,
        3] * length(RW)))/nrow(X)


    woMale <- -0.5 * as.matrix(myMu[2, 2:3], ncol = 2) %*% solve(mySig) %*%
        t(myMu[2, 2:3]) + log(myPi[2, 3])
    woFem <- -0.5 * (as.matrix(myMu[1, 2:3], ncol = 2)) %*% solve(mySig) %*%
        t(myMu[1, 2:3]) + log(myPi[1, 3])


    wM <- solve(mySig) %*% t(myMu[2, 2:3])
    wF <- solve(mySig) %*% t(myMu[1, 2:3])


    a <- (woMale - woFem)
    b <- wM - wF
    x <- cbind(X[, 1:2])

    # w0s is a w1s is b[1] w2s is b[2]
    myInter <- as.numeric(-a/b[2])
    mySlope <- as.numeric(-b[1]/b[2])

    X$myClass <- t(ifelse((a[1] + t(b) %*% t(x)) > 0, levels(X[, 3])[2], levels(X[,
        3])[1]))
    colnames(X)[4] <- "Predicted"
    retObj <- list(w0 = c(woMale, woFem), w1 = cbind(wM = wM, wF = wF), myClass = X,
        myModel = c(myInter = myInter, mySlope = mySlope))

    return(retObj)
}
results <- LDA(crabs[, c(5, 6, 2)])
## 2.3 actualdata + desicion boundaries p + geom_abline(intercept =
## results$myModel[1], slope = results$myModel[2], col = 'Red')

# predicted classes + desicion boundaries
ggplot(data = results$myClass) + geom_point(aes(x = RW, y = CL, col = Predicted)) +
    geom_abline(intercept = results$myModel[1], slope = results$myModel[2],
        col = "orange") + labs(title = "Predicted values for the LDA() with a decision boundary")

myLogit <- glm(sex ~ RW + CL, family = binomial(link = "logit"), data = crabs)

myDecLog <- coef(myLogit)[1:2]/-coef(myLogit)[3]

ggplot(data = results$myClass) + geom_point(aes(x = RW, y = CL, col = ifelse(myLogit$fitted.values >
    0.5, 1, 0)), show.legend = FALSE) + geom_abline(intercept = myDecLog[1],
    slope = myDecLog[2])


cat("For the Logistic regression \n")

t(table(Predicted = ifelse(myLogit$fitted.values > 0.5, "Male", "Female"), Observed = crabs$sex))
cat("\n")
cat("For the LDA:\n")
t(table(Predicted = results$myClass[, 4], Observed = crabs$sex))
CS <- read.csv2("creditscoring.csv")

# Suffle the rows
set.seed(12345)
CS <- CS[sample(nrow(CS)), ]

# Divide them up in different sets
csTrain <- CS[1:(nrow(CS) * 0.5), ]
csValid <- CS[((nrow(CS) * 0.5) + 1):floor(nrow(CS) * 0.75), ]
csTest <- CS[((nrow(CS) * 0.75) + 1):nrow(CS), ]
library(tree)
library(partykit)

myreturn <- matrix(ncol = 2, nrow = 2)
colnames(myreturn) <- c("Train", "Test")
rownames(myreturn) <- c("Deviance", "Gini")


# For the deviance
tDev <- tree(good_bad ~ ., data = csTrain, split = "deviance")

predVals <- predict(tDev, newdata = csTrain)
trainTable <- table(predicted = ifelse(predVals[, 1] > predVals[, 2], "bad",
    "good"), Train = csTrain$good_bad)

myreturn[1, 1] <- (1 - sum(diag(trainTable))/nrow(csTrain))

predValsT <- predict(tDev, newdata = csTest)
testTable <- table(predicted = ifelse(predValsT[, 1] > predValsT[, 2], "bad",
    "good"), Test = csTest$good_bad)
```

```r
myreturn[1, 2] <- (1 - sum(diag(testTable))/nrow(csTest))

# For the Gini

tGin <- tree(good_bad ~ ., data = csTrain, split = "gini")

predValsGin <- predict(tGin, newdata = csTrain)
giniTab <- table(Predicted = ifelse(predValsGin[, 1] > predValsGin[, 2], "bad",
    "good"), Test = csTrain$good_bad)


myreturn[2, 1] <- (1 - sum(diag(giniTab))/nrow(csTrain))

predValsGinT <- predict(tGin, newdata = csTest)
giniTabT <- table(Predicted = ifelse(predValsGinT[, 1] > predValsGinT[, 2],
    "bad", "good"), Test = csTest$good_bad)

myreturn[2, 2] <- (1 - sum(diag(giniTabT))/nrow(csTest))

knitr::kable(myreturn)

valid23 <- data.frame(trainS = 1, testS = 1)
tDev2 <- tree(good_bad ~ ., data = csTrain, split = "deviance")
rad <- 1

for (i in 2:15) {
    tDev22 <- prune.tree(tDev2, best = i)
    valid23[rad, 1] <- deviance(tDev22)
    valid23[rad, 2] <- 2 * deviance(predict(tDev22, newdata = csValid, type = "tree"))
    rad <- rad + 1
}

valid23$best <- 2:15
ggplot(data = valid23, aes(x = best)) + geom_line(aes(y = trainS), col = "red") +
    geom_line(aes(y = testS), col = "blue") + labs(title = "The optimal number of leafs",
    x = "Number of leaf nodes", y = "Deviance")



best <- prune.tree(tDev2, best = 4)
summary(best)
predTest <- predict(best, csTest)
tT <- table(ifelse(predTest[, 1] > predTest[, 2], "bad", "good"), csTest$good_bad)
paste("Missclassification rate for test data", 1 - (sum(diag(tT))/nrow(csTest)))
plot(best)
text(best)
library(e1071)
# ??e1071
cat("Training data")
cat("\n")
baybay <- naiveBayes(formula = good_bad ~ ., data = csTrain)
bBay <- table(Predicted = predict(baybay, newdata = csTrain), Observed = csTrain$good_bad)
print(bBay/nrow(csTrain))
```

```
1 - sum(diag(bBay))/nrow(csTrain)
cat("\n Test data")
cat("\n")
bBayTest <- table(Predicted = predict(baybay, newdata = csTest), Observed = csTest$good_bad)
print(bBayTest/nrow(csTest))
1 - sum(diag(bBayTest))/nrow(csTest)
rawprobs <- predict(baybay, csTrain, type = "raw")

rawBayes <- rawprobs[, 2]/rawprobs[, 1]  #good/bad

lossMat <- matrix(c(0, 10, 1, 0), ncol = 2)


lossBayBay <- table(Predicted = ifelse(rawBayes > lossMat[2, 1]/lossMat[1, 2],
    "Good", "Bad"), Observed = csTrain$good_bad)
# table(Predicted=ifelse(rawBayes > lossMat[1,2]/lossMat[2,1],'Good','Bad'
# ),Observed=csTrain$good_bad)
print(lossBayBay)
1 - sum(diag(lossBayBay))/nrow(csTrain)

rawprobsTest <- predict(baybay, csTest, type = "raw")

rawBayesTest <- rawprobsTest[, 2]/rawprobsTest[, 1]  #good/bad
lossTable <- table(Predicted = ifelse(rawBayesTest > lossMat[2, 1]/lossMat[1,
    2], "Good", "Bad"), Observed = csTest$good_bad)
print(lossTable)
1 - sum(diag(lossTable))/nrow(csTest)
```