TDDE01: Lab 3

A report by

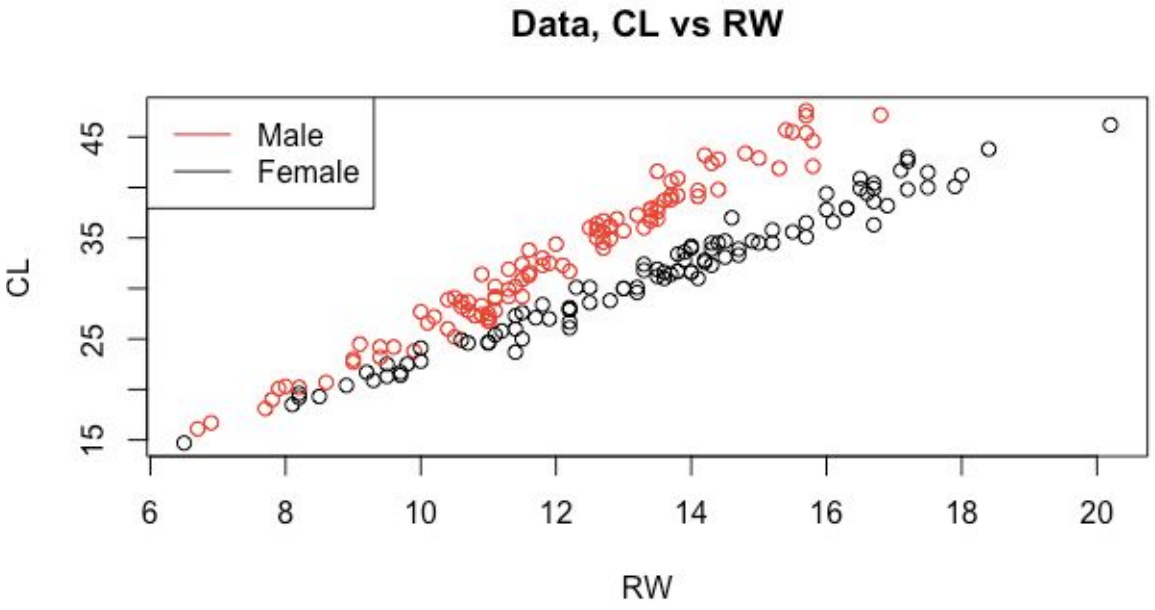
Adam Nyberg
adany869

Assigment 1

In this assignment I was supposed to classify the sex of crabs. To do that I got a dataset containing the sex, carapace length (CL) and rear width (RW).

1.1

First thing I did was to plot the CL versus RW and look at the plot.



By looking at the data I'd argue that the data is easily classified by using linear discriminant analysis because theres a clear linear line dividing the males from the females.

1.2

The goal of this task is to draw a decision boundary and classify the data by implementing IDA with proportional priors. First I divided the data into two sets, one with all the females and one with all males. I then computed the discriminant function coefficients (w_1, w_0) separatly with the following formulas:

$$w_{0i} = -\frac{1}{2}\boldsymbol{\mu}_i^T \Sigma^{-1} \boldsymbol{\mu}_i + \log \pi_i$$

$$w_i = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i$$

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{N_c} \sum_{i:y_i=c} \mathbf{x}_i$$

$$\hat{\boldsymbol{\Sigma}}_c = \frac{1}{N_c} \sum_{i:y_i=c} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_c)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_c)^T$$

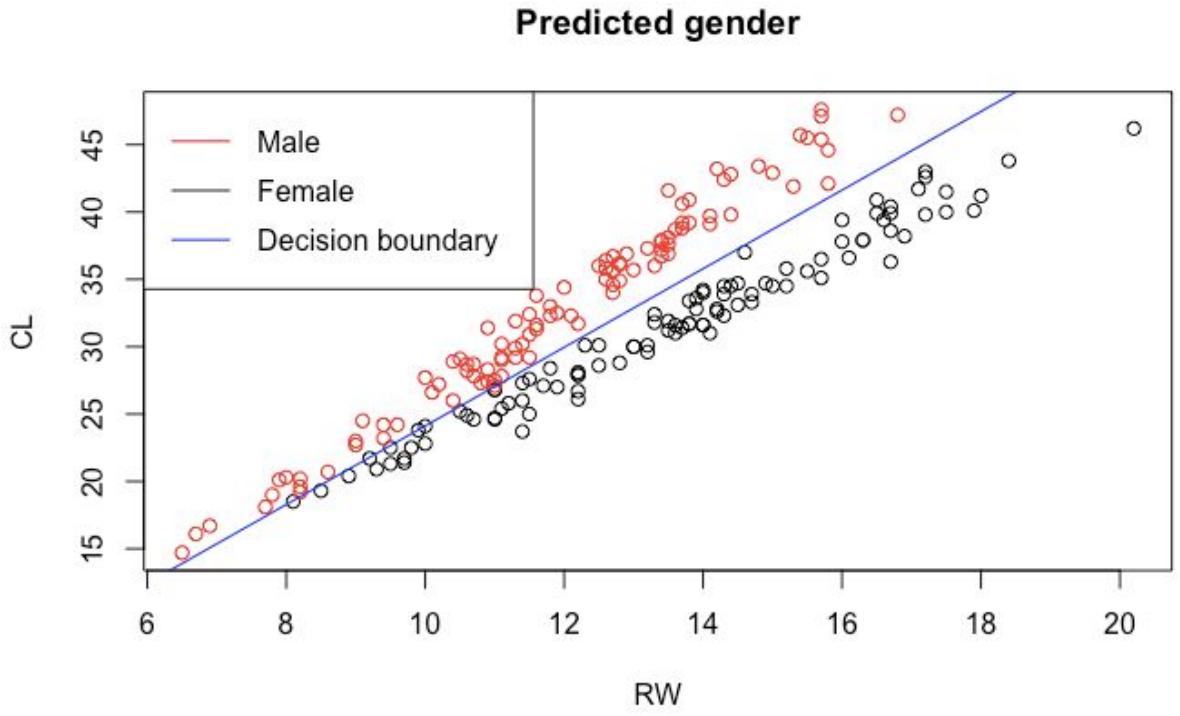
$$\hat{\Sigma} = \frac{1}{N} \sum_{c=1}^{k} N_c \, \hat{\Sigma}_c$$

$$\hat{\pi}_c = \frac{N_c}{N}$$

When I have w_1 and w_0 for both females and males I take the difference between them and now I have the coefficients for both females and males. That gives me $\omega_1 = (-5.682847,\ 1.947504)$ and $\omega_0 = 9.865352$. The equation for the desicion boundary is $CL = -\frac{\omega_1[1]}{\omega_1[2]} * BW - \frac{\omega_0}{\omega_1[2]}$. I can now classify the data by looking on what side of the line every datapoint is.

1.3

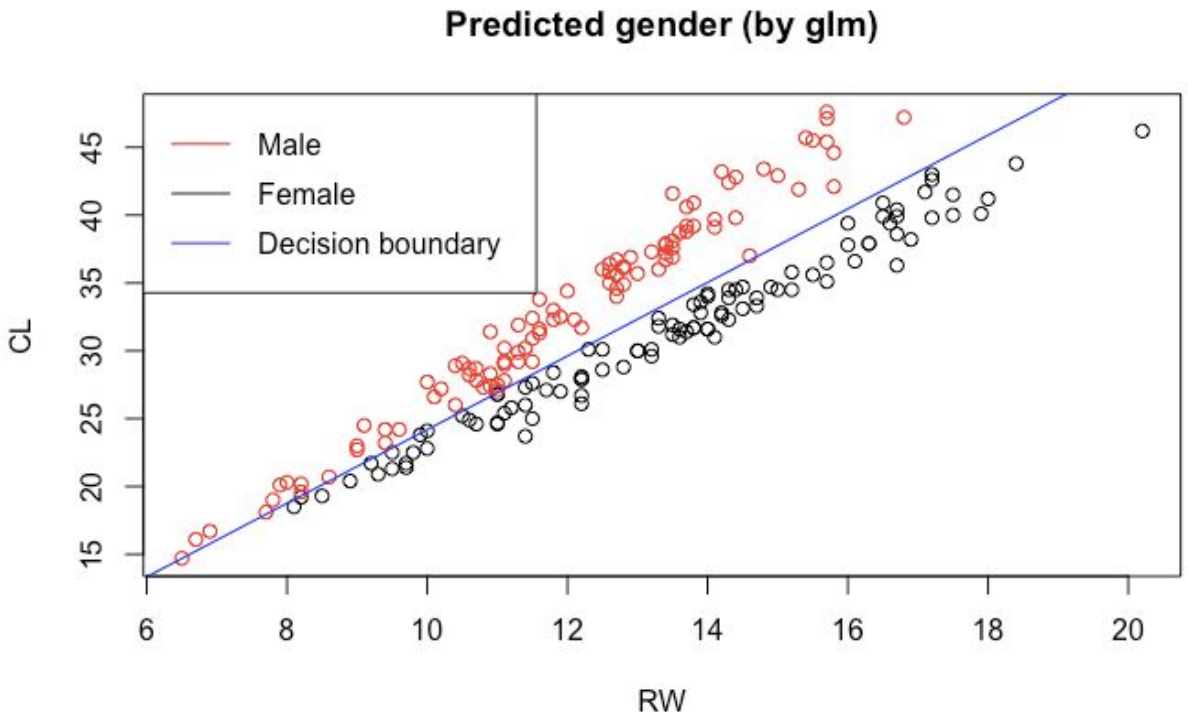
I now add the line calculated in 1.2 to the plot in 1.1.



By looking at that plot you can conclude that the fit of the line is very good and only a few crabs have been misclassified.

1.4

In this part I used the glm() function to classify and draw a decision boundary. That gave me the following plot.



As we can see the line have a slightly different angle when using glm() compared to LDA used in 1.3. LDA performed better than glm() because it fitted the line better to the actual data.

Assigment 2

In this assignment I got a data file with information about customers and their credit scoring. Every row contained information about one customer and a column (good_bad) that decided if they had a good or bad credit score. I imported the data and splitted it into train (50%), test (25%) and validation (25%).

2.2

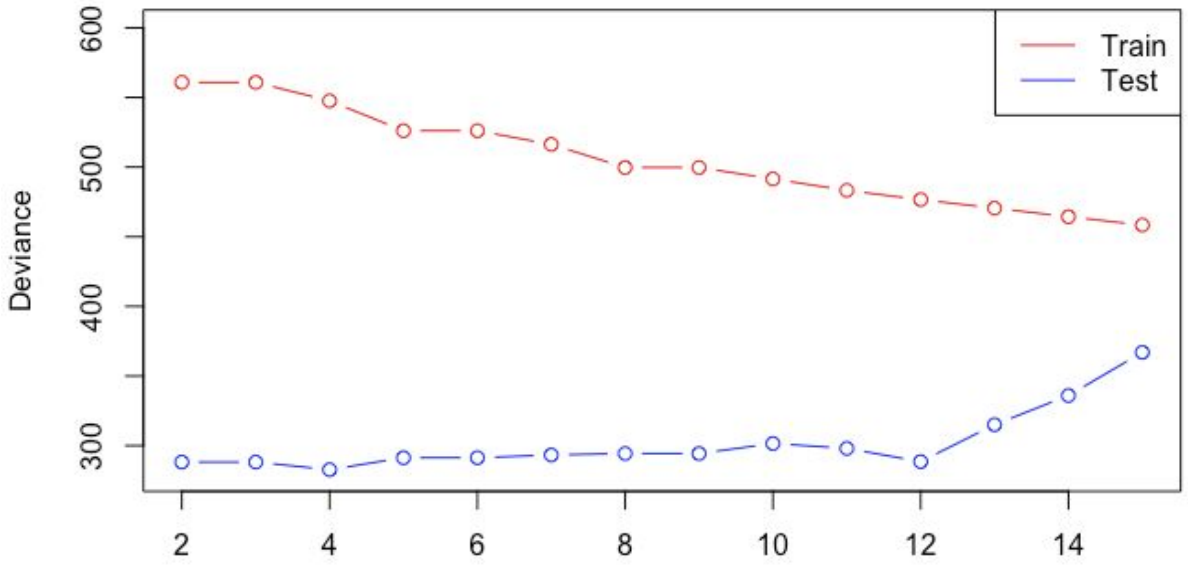
In this task I fitted a decision tree to the training data by using both deviance and gini index. That gave me the following table.

	Misclassification rate train	Misclassification rate test
Deviance	0.212	0.284
Gini index	0.236	0.352

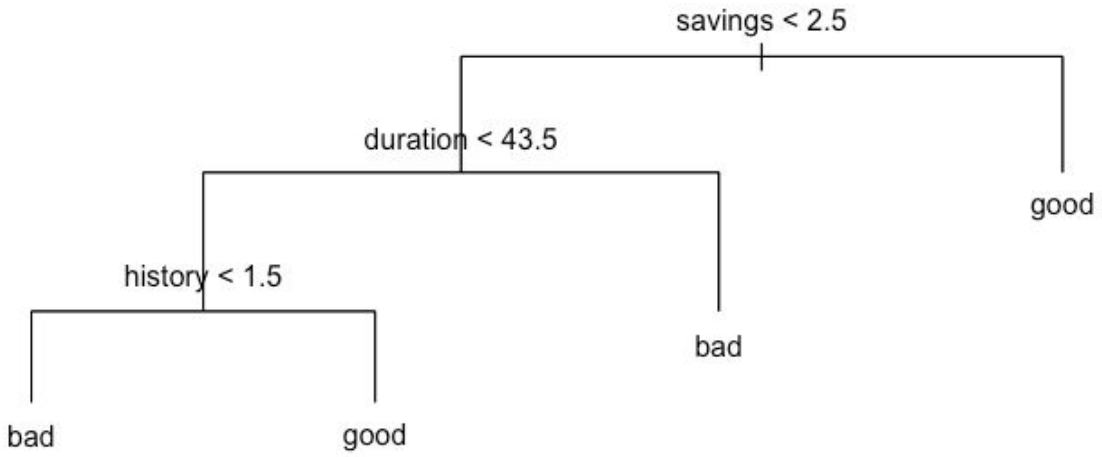
As we can see deviance performed better so I'll continue to use that as a messure of impurity.

2.3

This task was about choosing the optimal tree depth and plot the dependence of deviance for traing and test data based on number of leaves.



In the plot above we can see that the lowest deviance is when the number of leaves are 4. The misclassification rate for 4 leaves is 0.26. Below is the optimal tree plotted.



2.4

In this task I used Naive Bayes to classify test and training data using training data to train. I then got the following confusion matrix and misclassification rates.

Misclassification rate train	Misclassification rate test
0.3	0.32

Confusion matrix train	Confusion matrix test
bayes_fitted_train bad good bad 95 98 good 52 255	bayes_fitted_test bad good bad 47 49 good 31 123

By looking at the misclassification rates for Naive bayes it's clear that for this data our tree model in 2.2 worked better.

2.5

In this task I repeated the classification from 2.4 but using the following loss matrix.

$$L = \begin{matrix} \\ Observed \\ \\ \end{matrix} \begin{matrix} Predicted \\ good \\ bad \end{matrix} \begin{pmatrix} 0 & 1 \\ 10 & 0 \end{pmatrix}$$

That gave us the following misclassification rates and confusion matrix.

Misclassification rate train	Misclassification rate test
0.546	0.556

Confusion matrix train	Confusion matrix test
b_weight_train bad good bad 137 263 good 10 90	b_weight_test bad good bad 70 131 good 8 41

By using this loss matrix we reduced the number of wrong good and increased the number wrong bad instead. This could be the expected behavior in this case of credit scoring when you dont want to give out loans to people who cant pay back.

```
data = read.csv("lab3/australian-crabs.csv")
data$Color = "black"
data$Color[data$sex == "Male"] = "red"
cols = c("red", "black", "blue")
leg = c("Male", "Female", "Decision boundary")

plot(data$RW, data$CL, ylab = "CL", xlab="RW", col=data$Color, main = "Data, CL vs RW")
legend("topleft", lty=c(1,1), col=c("red", "black"), legend = c("Male", "Female"))

X = data[5:6]#ASSIGNMENT 1.2-3
Y = data$sex
disc_fun=function(label, S){
  X1=X[Y==label,]
  #MISSING: compute LDA parameters w1 (vector with 2 values) and w0 (denoted here as b1)
  Nc = length(X1[,1])
  N = length(X[,1])
  u = colSums(X1)/Nc

  b1 = -(1/2)*t(u)%*%solve(S)%*%u + log(Nc/N)
  w1 = solve(S)%*%(u)

  return(c(w1[1], w1[2], b1[1,1]))
}

X1=X[Y=="Male",]
X2=X[Y=="Female",]

S=cov(X1)*dim(X1)[1]+cov(X2)*dim(X2)[1]
S=S/dim(X)[1]

res1=disc_fun("Male",S)#discriminant function coefficients
res2=disc_fun("Female",S)
res = res1 - res2# MISSING: use these to derive  decision boundary coefficients 'res'

d=res[1]*X[,1]+res[2]*X[,2]+res[3]# classification
Yfit=(d>0)
plot(X[,1], X[,2], col=Yfit+1, xlab="RW", ylab="CL", main="Predicted gender")

abline(-res[3]/res[2], -res[1]/res[2], col="blue")#MISSING: use 'res' to plot decision
boundary.
legend("topleft", lty=c(1,1,1), col=cols, legend = leg)

glm_m = glm(sex~RW+CL, family="binomial", data=data)#ASSIGNMENT 1.4
preds = predict(glm_m)
Yfit2=(preds>0)
plot(X[,1], X[,2], col=Yfit2+1, xlab="RW", ylab="CL", main = "Predicted gender (by glm)")
gcof = glm_m$coefficients
abline(-gcof[1]/gcof[3], -gcof[2]/gcof[3], col="blue")
legend("topleft", lty=c(1,1,1), col=cols, legend=leg)

library(tree) #ASSIGNEMNT 2
library(MASS)
library(e1071)
```

```
set.seed(12345)#ASSIGNMENT 2.1
n=nrow(data)
id=sample(1:n, floor(n*0.5))
train=data[id,]
test=data[-id,]
id=sample(1:(n/2), floor((n/2)*0.5))
validation = test[id,]
test = test[-id,]

mis_rate = function(conf_matrix) {#ASSIGNMENT 2.2
  return(1 - sum(diag(conf_matrix))/sum(conf_matrix))
}

dev_m = tree(good_bad~., data=train, split = c("deviance"))# DEVIANCE

dev_fitted_test = predict(dev_m, newdata=test, type="class")
dev_conf_test = table(test$good_bad, dev_fitted_test)
dev_mis_test = mis_rate(dev_conf_test)

dev_fitted_train = predict(dev_m, newdata=train, type="class")
dev_mis_train = mis_rate(table(train$good_bad, dev_fitted_train))

gini_m = tree(good_bad~., data=train, split = c("gini"))# GINI

gini_fitted_test = predict(gini_m, newdata=test, type="class")
gini_conf_test = table(test$good_bad, gini_fitted_test)
gini_mis_test = mis_rate(gini_conf_test)

gini_fitted_train = predict(gini_m, newdata=train, type="class")
gini_mis_train = mis_rate(table(train$good_bad, gini_fitted_train))

tree_m = tree(good_bad~., data=train, split = c("deviance"))#ASSIGNMENT 2.3
l = 15
trainScore=rep(0,l)
testScore=rep(0,l)
misRate = c()

for(i in 2:l) {
  prunedTree=prune.tree(tree_m, best=i)
  pred=predict(prunedTree, newdata=validation, type="tree")
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)

  pred_test=predict(prunedTree, newdata=test, type="class")
  misRate[i]=mis_rate(table(test$good_bad, pred_test))
}

plot(2:l, trainScore[2:l], type="b", col="red", ylim=c(280,600), ylab = "Deviance", xlab =
"")
points(2:l, testScore[2:l], type="b", col="blue")
legend("topright", lty=c(1,1,1), col=c("red", "blue"), legend=c("Train", "Test"))

optimal_tree = prune.tree(tree_m, best=4)
plot(optimal_tree)
```

```
text(optimal_tree)
bayes_m = naiveBayes(good_bad~., data=train, type=c("class"))#ASSIGNMENT 2.4

bayes_fitted_train = predict(bayes_m, newdata = train)
bayes_conf_train = table(bayes_fitted_train, train$good_bad)
bayes_mis_train = mis_rate(bayes_conf_train)

bayes_fitted_test = predict(bayes_m, newdata = test)
bayes_conf_test = table(bayes_fitted_test, test$good_bad)
bayes_mis_test = mis_rate(bayes_conf_test)

classify = function(data) {#ASSIGNMENT 2.5
  newData = replace(data, data[,1]*10 > data[,2], 'bad')
  newData = replace(newData, data[,1]*10 <= data[,2], 'good')
  return(newData[,1])
}

b_weight_test = predict(bayes_m, newdata=test, type=c("raw"))# Test prediction
b_weight_test = classify(b_weight_test)
b_weight_conf_test = table(b_weight_test, test$good_bad)
b_weight_mis_test = mis_rate(b_weight_conf_test)

b_weight_train = predict(bayes_m, newdata=train, type=c("raw"))# Training prediciton
b_weight_train = classify(b_weight_train)
b_weight_conf_train = table(b_weight_train, train$good_bad)
b_weight_mis_train = mis_rate(b_weight_conf_train)
```