# Introduction to Machine Learning

## TDDE01 – Lab2

Author:
Christoffer Eduards - chred146

### I. Introduction

This report is part 2 of the laborations in the course TDDE01 – Introduction to Machine Learning. This report consists of three mandatory exercises. The code required for solving these assignments is attached to this rapport and can be viewed under Appendix.

### II. Assignment 1

**LDA and logistic regression**
This exercise is based around the dataset "australian-crabs". The dataset contains multiple attributes for crabs which can be used to perform classifications.

**1.1**
In the first exercise the task is to do a scatterplot of carapace length versus rear width. This plot can be seen in figure 1 below:

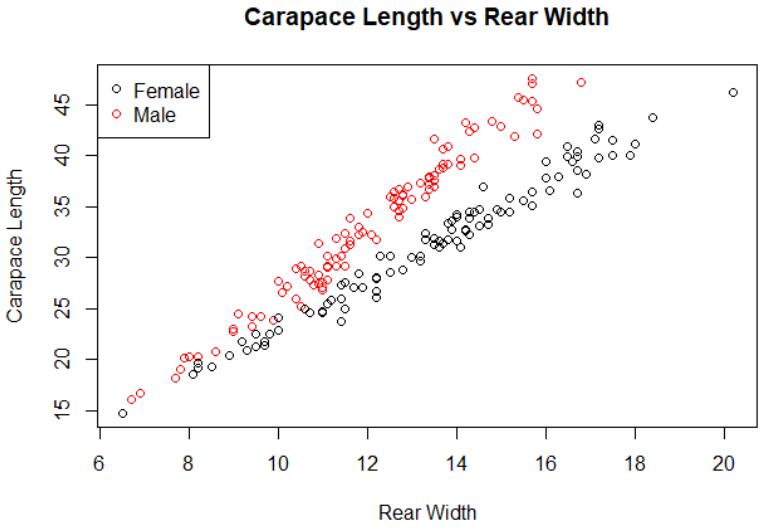


*Figure 1: Carapace length versus rear width*

I would say that a linear discriminant analysis would be very well suited for this data. The classes show clear separation which can be modelled with the use of LDA.

**1.2**
In the second task a LDA analysis is performed with target Sex and features CL and RW with a proportional prior. The plot that illustrates this result is shown in figure 2.

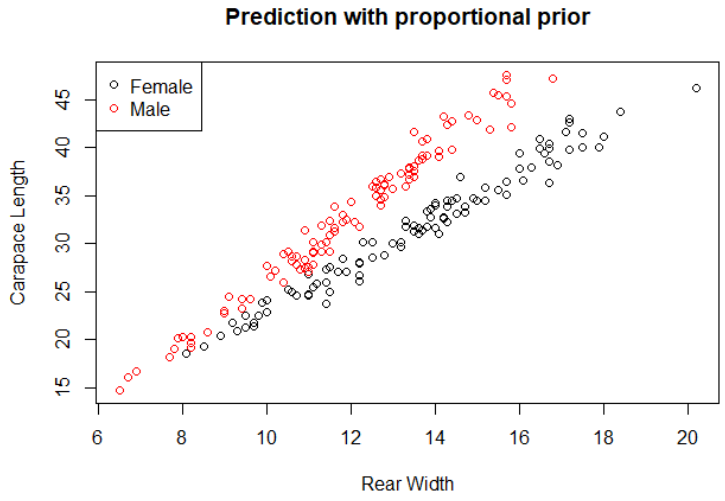


*Figure 2: LDA with proportional prior*

The result has great similarities to the result from the previous task. The sex of some of the smaller crabs are classified wrongly, but the classification seems to be accurate. The misclassification error is 0.035, which further supports the visual interpretation that the quality of fit is high.

**1.3**

The third task is the same exercise as the previous one except for the prior used in the LDA. The priors are now $p(male) = 0.9$ and $p(female) = 0.1$. Since these priors are wrong since the probability of female and male are equal, this should worsen our prediction.

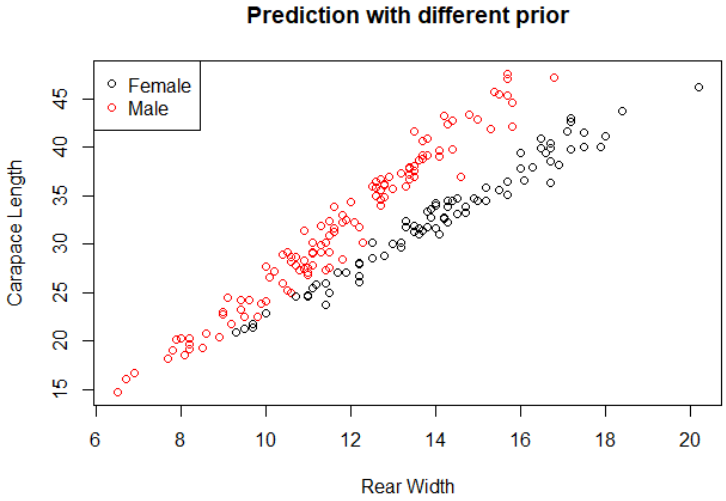


*Figure 3: Prediciton with priors p(male) = 0.9 and p(female) = 0.1*

The misclassification error is 0.08 which is higher than for the previous model. Therefore, the prediction is as expected worse than the previous one with a proportional prior.

**1.4**

In the last task of the exercise the objective is to perform a classification as in the previous tasks but now with logistic regression. In addition to this, a decision boundary is computed for the linear regression. How the decision boundary has been calculated can be seen below:

$$0 = -4.630747y + 12.563893x + 13.616628 <=> y = 2.71x + 2.94$$

where y = Carapace length and x = Rear width.
The classification and the decision boundary can be seen in Figure 4.

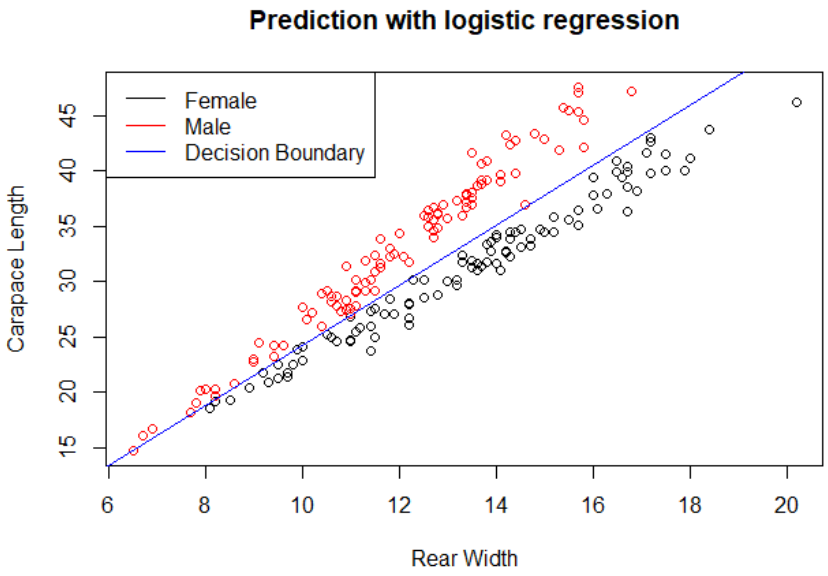


*Figure 4: glm() prediction and decision boundary*

The misclassification rate is 0.035, the same as in task 1.2 when using LDA.

## III. Assignment 2

**Analysis of credit scoring**

The second exercise is based around the dataset "creditscoring" which contains information about how customers have managed their loans based on several variables. The goal of the exercise is to create a model to predict how a new customer will manage their loan, i.e. if they are able to pay back the loan.

**2.1**

The first task was just to import the data to R. This was done by the code specified on lecture 1e.

**2.2**

In the second task the goal was to fit a decision tree by using two different measures of impurity, deviance and gini index.

When using deviance, I got the misclassification rate of 0.268 and with gini index I got the misclassification rate of 0.364. Since the misclassification rate is lower for deviance I choose that measure for the following steps.

**2.3**

In the third exercise the objective was to use training and validation data to choose the optimal tree depth. Figure 5 illustrates the dependence of deviances for training and validation data on number of leaves.

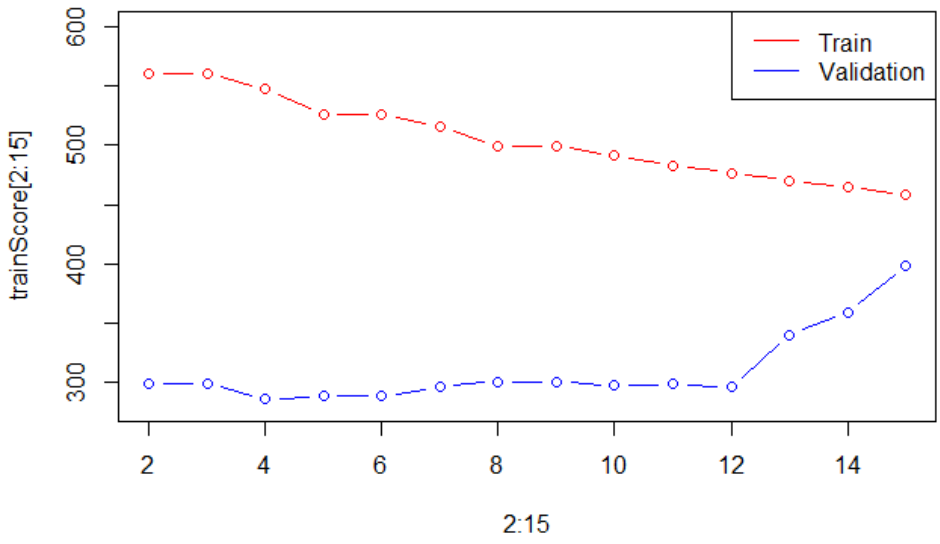


*Figure 5: Optimal tree depth*

From the figure we see that the optimal number of leaves is four since it has the lowest amount of deviance for the validation data. It is also possible to see that for the training data the model gets more and more overfitted when the number of leaves increases. The next task was to report the optimal tree. Since we know that the optimal number of leaves is four, we can plot the optimal tree which can be seen in Figure 6.

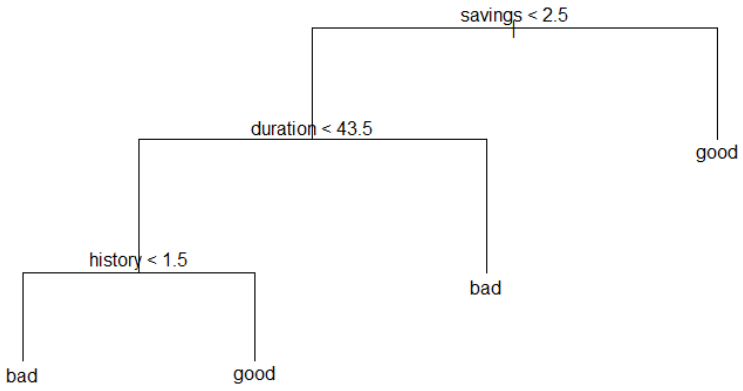


*Figure 6: Optimal tree*

The figure show that the variables used are savings, duration and history. The tree also show that the optimal depth is three. The tree illustrates how a prediction regarding the ability to pay back loan for a new customer will be performed with these three variables. If savings is > 2.5 the customer is predicted to have good ability. Otherwise we move down one level in the tree structure. If duration is > 43.5 the customer is predicted to have bad ability to pay back their loans. If not, we move down to the next level. If the history variable is > 1.5, the prediction is good, otherwise it is bad.

The misclassification rate for test data is 0.38.

**2.4**

In the fourth task training data is used to perform classification using Naïve Bayes. The confusion matrix and misclassification rate for **train data** is presented below:

```
          Truth
Prediction bad good
      bad   95   98
      good  52  255
Misclassification rate = 0.3
```

Confusion matrix and misclassification rate for **test data**:

```
          Truth
Prediction bad good
      bad   46   49
      good  30  125
Misclassification rate = 0.316
```

The misclassification rate using Naïve Bayes classification is lower than when using the tree. With Naïve Bayes it is 0.316 and when using the tree classification it is 0.38. Therefore, with this dataset Naïve Bayes is a better classifier.

**2.5**

In this task two classification is performed, one with the optimal tree and one with Naïve Bayes. The classification principle used is:

$$\hat{Y} = 1 \; if \; p(Y =' \; good'|X) > \pi, otherwise \; \hat{Y} = 0$$

where $\pi = 0.05, 0.1, 0.15, \ldots, 0.9, 0.95$. The ROC curves for the two methods is plotted in Figure 7.

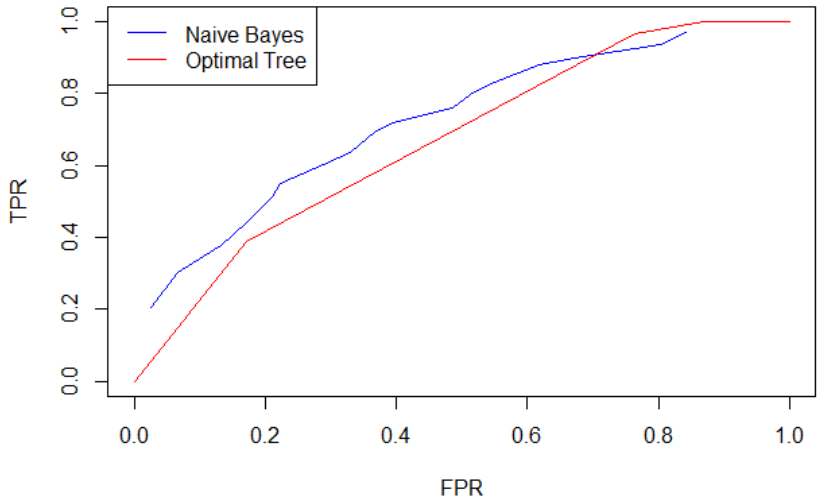


*Figure 7: ROC curves*

The best classifier is the one with the greatest area under curve which we can see from the plot is the Naïve Bayes. This is the result that could be expected since the misclassification rates from the previous tasks showed that Naïve Bayes is better.

**2.6**

In this task the classification from 2.4 is repeated but now the following loss matrix is used:

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

The confusion matrix and misclassification rate for **train data** is:

```
      Prediction
Truth  bad good
  bad  137   10
  good 263   90
Misclassification rate = 0.546
```

The confusion matrix and misclassification rate for **test data** is:

```
      Prediction
Truth  bad good
  bad   71    5
  good 122   52
Misclassification rate = 0.508
```

With this loss matrix the number of values classified as bad increases with about 150 %. If the private enterprise would be very risk avert this could be a strategy to take on a low amount of risk. This is further shown by the low amount of values that are classified as good when it in fact is bad. As an example, in task four for the training data 38 % of the customers that were predicted to being able to pay back their loan were in fact not able to do so. With the use of the loss matrix this amount is now 11 %.

An interesting thing to point out with the result is that the misclassification rate is lower for the test data than for train data which is uncommon.

## IV. Assignment 4

**Principal components**

The objective in this assignment is to investigate how a near-infrared spectra can be used to predict viscosity levels. This is done with the help of the dataset "NIRspectra".

**1.1**

In the first task a standard PCA is conducted. A plot that shows how much variation is explained by each feature is provided in Figure 8.

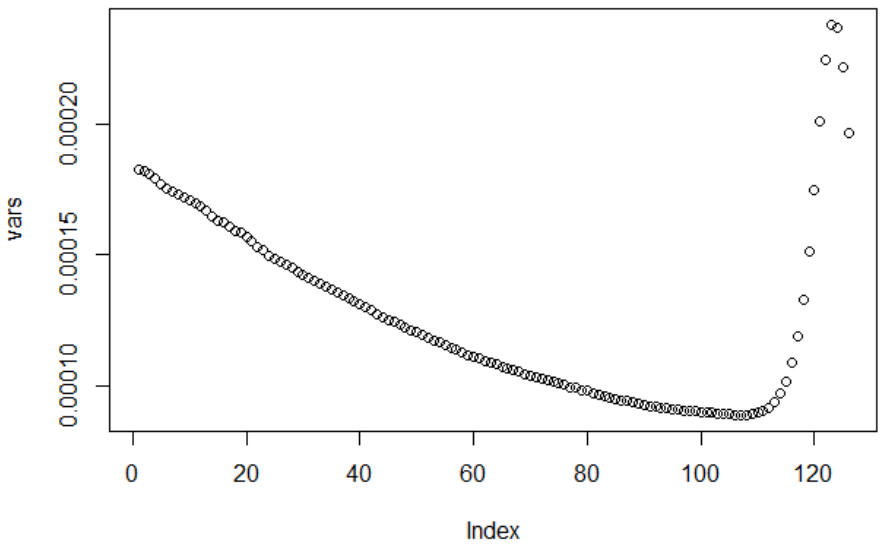


*Figure 8: Variation from features*

This plot does not show how many PC that should be extracted. Instead of plotting the variances from each feature, the variances from the PC components can be plotted which is shown in Figure 9.

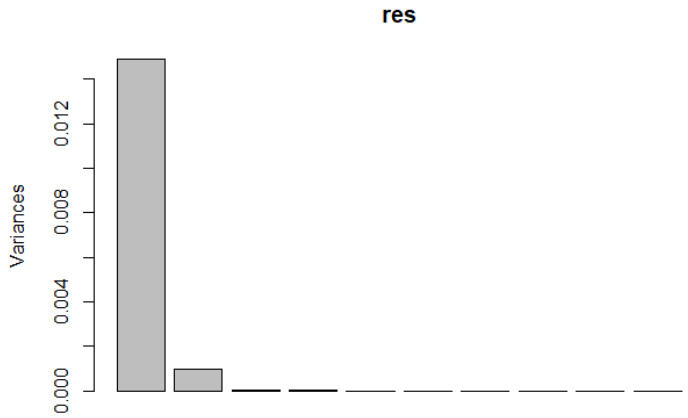


*Figure 9: Variances from PC components*

The plot show that the absolute majority of the variance come from the first PC. With the use of the lambdas we can chose PC1 and PC2 which explains 93.332 + 6.263 = 99.595 % of the total variance. The plot below shows the scores in the coordinates PC1 and PC2.

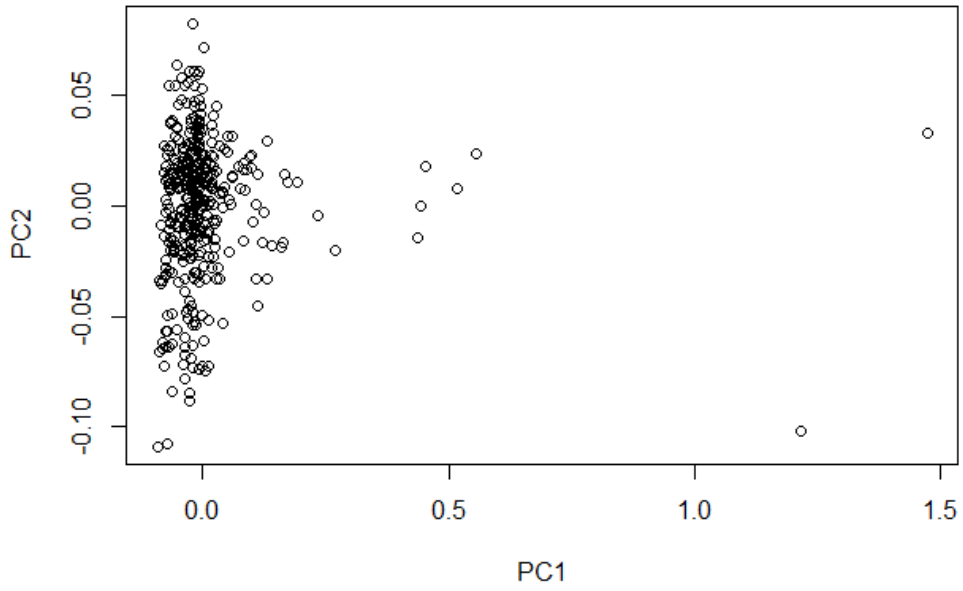


*Figure 10: Scores in PC1 and PC2*

From the plot it is possible to see that there are two measures of diesel fuels that differs severely from the other data points. They are located to the right in Figure 10.

**2.2**

In this task trace plots are created of the loadings of the components that were selected in the previous task (PC1, PC2).

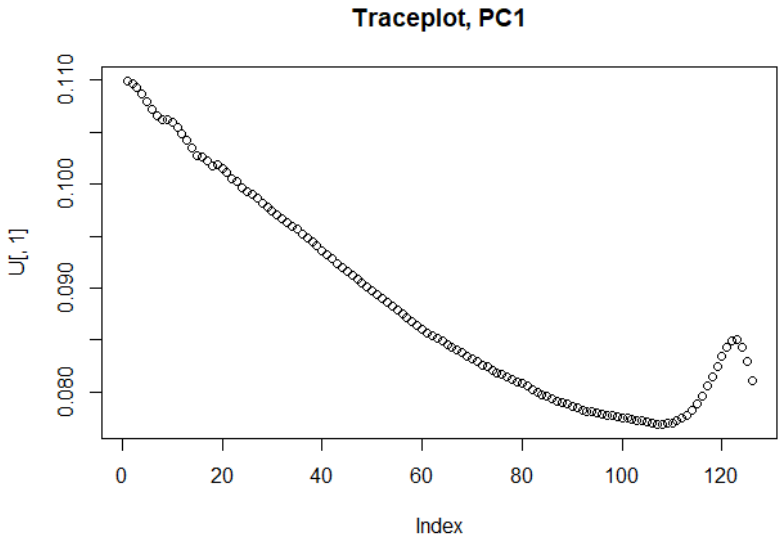


*Figure 11: Traceplot of PC1*

From the traceplot of PC1 we can see that it is described by most of the features.

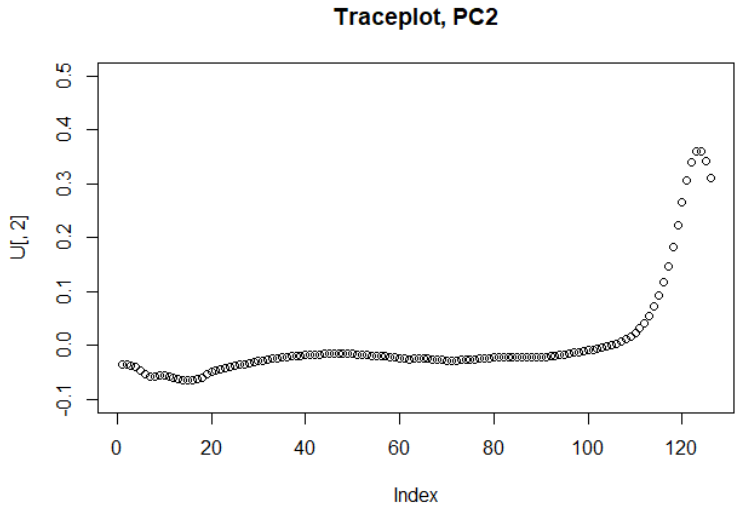


*Figure 12: Traceplot of PC2*

Different from PC1, PC2 is described mainly by a few features with index ≈ 120.

**2.3**

In this task Independent Component Analysis is performed using fastICA with the two components from the previous exercise.

**a)**

In this task $W^{´} = KW$ is computed. K is the pre-whitening matrix which is used to whiten our data. In our case K is used to project our data onto the two first principal components. W is calculated by $XKW = S$, where W is chosen to maximize the negentropy approximation. So $W^{´} = KW <=> XW^{´} = S$. This can be interpreted as $W^{´}$ is both used to pre-whiten the data and project it onto the independent components.

Two traceplots are presented which represents the components of $W^{´}$.

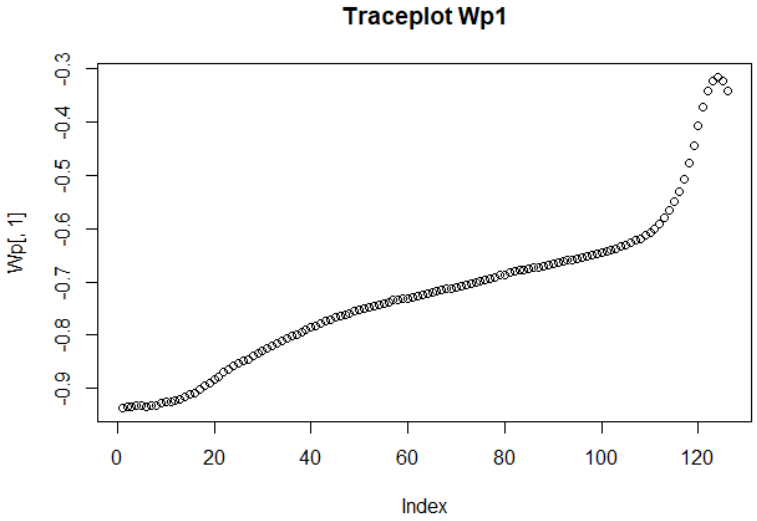


*Figure 13: Traceplot $W_1^{´}$*

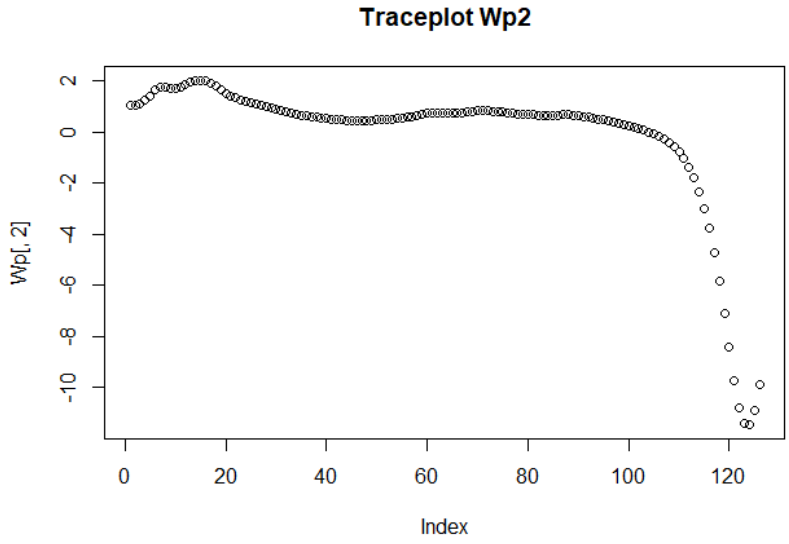


*Figure 14: Traceplot $W_2^{'}$*

These traceplots have similarities to Figure 11 and 12. PC2 and Wp2 follow each other but are inverted in such way that as Wp2 becomes smaller, PC2 grows, Wp2 grows. PC1 and Wp1 are not as similar as Wp2 and PC2.

**b)**

In this task a plot is presented of the scores of the first two latent features.

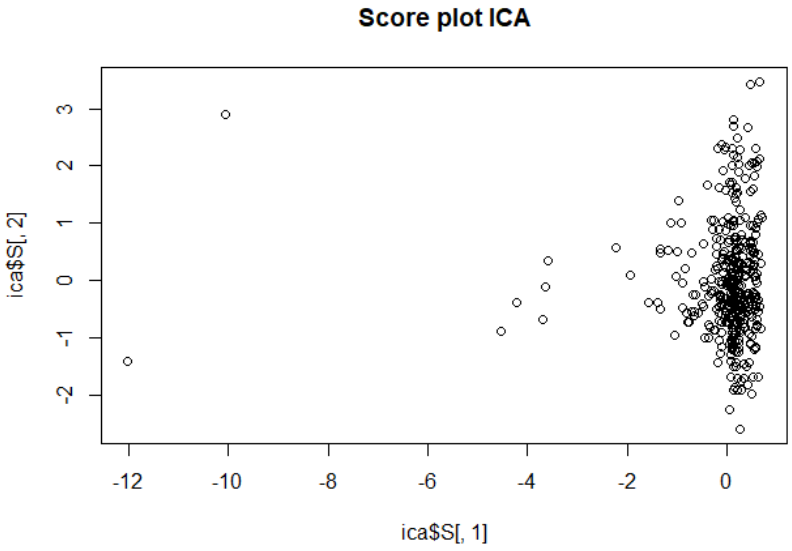


*Figure 15: Scores of latent features*

Compared to Figure 10 with the scores from the PCA, it is possible to see a pattern between the scores of PCA and ICA. It is scaled in a different way and rotated, both vertically and horizontally.

## V. Appendix

**Assignment 1**

```
setwd("C:/Users/Christoffer Eduards/OneDrive/R/Lab2/Assignment 1")
library("MASS")
#Misclassification function
misclass = function(preds, truth){
  return(mean(preds != truth))
}
#Classification function used in 1.4
classify = function(pred, const){
  classified = c()
  for(i in 1:length(pred)){
    classified[i] = if(pred[i] > const) "Male" else "Female"
  }
  return(classified)
}
#1.1
#Load data and set color
data=read.csv("australian-crabs.csv")
data$Color = "black"
data$Color[data$sex == "Male"] = "red"
#Plot CL vs RW
plot(data$RW, data$CL, xlab = "Rear Width", ylab = "Carapace Length", col=data$Color,
main="Carapace Length vs Rear Width")
legend("topleft", pch=c(1,1), legend = c("Female", "Male"), col = c("black", "red"))
#1.2
#Fit LDA model to data and perform prediction
fit_lda = lda(data$sex~ data$CL + data$RW, data = data)
pred_proportional = predict(fit_lda, data)
#Plot the LDA classification
plot(data$RW, data$CL, xlab = "Rear Width", ylab = "Carapace Length",
col=as.numeric(pred_proportional$class), main="Prediction with proportional prior")
```

```
legend("topleft", pch=c(1,1), legend = c("Female", "Male"), col = c("black", "red"))

#Calculate misclasification rate

misclass_lda = misclass(pred_proportional$class, data$sex)

print(misclass_lda)

#1.3

#LDA model fitting and prediction with new priors

fit0.9_lda = lda(data$sex~ data$CL + data$RW, data = data, prior = c(0.1, 0.9))

pred0.9 = predict(fit0.9_lda, data)

plot(data$RW, data$CL, xlab = "Rear Width", ylab = "Carapace Length",
col=as.numeric(pred0.9$class), main="Prediction with different prior")

legend("topleft", pch=c(1,1), legend = c("Female", "Male"), col = c("black", "red"))

misclass_lda0.9 = misclass(pred0.9$class, data$sex)

print(misclass_lda0.9)

#1.4

#Logistic regression model fitting and prediction

fit_glm = glm(data$sex~ data$CL + data$RW, data=data, family=binomial)

pred_glm = predict(fit_glm, data, type="response")

classified = round(pred_glm)

plot(data$RW, data$CL, xlab = "Rear Width", ylab = "Carapace Length", col=as.numeric(classified)+1,
main="Prediction with logistic regression")

legend("topleft", pch=c(1,1), legend = c("Female", "Male"), col = c("black", "red"))

classify_sex = (classify(pred_glm, 0.5))

misclass_glm = misclass(classify_sex, data$sex)

print(misclass_glm)

#Coefficients for decision boundary

glmCof = fit_glm$coefficients

abline(-glmCof[1]/glmCof[2], glmCof[3]/-glmCof[2], col = "blue")

legend("topleft", lty=c(1,1,1), col=c("black", "red", "blue"), legend=c("Female", "Male", "Decision
Boundary"))
```

**Assignment 2**

```
setwd("C:/Users/Christoffer Eduards/OneDrive/R/Lab2/Assignment 2")

library(e1071)
```

```
library(tree)
library(MASS)
#Misclassification function
misclass = function(preds, truth) {
  return(mean(preds != truth))
}
#Used to create a matrix with probabilities in 2.5
probMat = function(pi, input) {
  m = length(pi)
  probMat = matrix(nrow = dim(input)[1], ncol = length(pi))
  for (i in 1:dim(input)[1])
    for (j in 1:m) {
      probMat[i, j] = ifelse(input[i, 2] > pi[j], 1, 0)
    }
  return (probMat)
}
#ROC function used in 2.5
ROC = function(p, input) {
  m = length(p)
  TPR = numeric(m)
  FPR = numeric(m)

  for (i in 1:m) {
    t = table(test$good_bad, as.numeric(input[, i]))
    if (dim(t)[2] == 1 & colnames(t) == "1") {
      TP = t[2, 1]
      FP = t[1, 1]
      nPlus = TP
      nMinus = FP
      TPR[i] = TP / nPlus
      FPR[i] = FP / nMinus
```

```
    } else if (dim(t)[2] == 1 & colnames(t) == "0") {
      nPlus = t[2, 1]
      nMinus = t[1, 1]
      TP = 0
      FP = 0
      TPR[i] = TP / nPlus
      FPR[i] = FP / nMinus
    } else{
      TP = t[2, 2]
      FP = t[1, 2]
      nPlus = TP + t[2, 1]
      nMinus = FP + t[1, 1]
      TPR[i] = TP / nPlus
      FPR[i] = FP / nMinus
    }
  }
  return(list(TPR = TPR, FPR = FPR))
}
#2.1
#Load data and create train, validation and test sets
data = read.csv2("creditscoring.csv")
n = dim(data)[1]
set.seed(12345)
id = sample(1:n, floor(n * 0.5))
train = data[id, ]
id1 = setdiff(1:n, id)
set.seed(12345)
id2 = sample(id1, floor(n * 0.25))
valid = data[id2, ]
id3 = setdiff(id1, id2)
test = data[id3, ]
```

```
#2.2a
#Fit and predict with tree and deviance measure
fit_devTree = tree(good_bad ~ ., data = train, split =  c("deviance"))
plot(fit_devTree)
text(fit_devTree)
print(summary(fit_devTree))
pred_test = predict(fit_devTree, newdata = test, type = "class")
misclass_devTest = misclass(pred_test, test$good_bad)
print(misclass_devTest)
#2.2b
#Fit and predict with tree and gini measure
fit_ginTree = tree(good_bad ~ ., train, split =  c("gini"))
plot(fit_ginTree)
text(fit_ginTree)
print(summary(fit_ginTree))
pred_test = predict(fit_ginTree, newdata = test, type = "class")
misclass_ginTest = misclass(pred_test, test$good_bad)
print(misclass_ginTest)
#2.3
#Finds optimal tree depth
trainScore = rep(0, 15)
testScore = rep(0, 15)
for (i in 2:15) {
  prunedTree = prune.tree(fit_devTree, best = i)
  pred = predict(prunedTree, newdata = valid,
                 type = "tree")
  trainScore[i] = deviance(prunedTree)
  testScore[i] = deviance(pred)
}
plot(
  2:15,
```

```
  trainScore[2:15],
  type = "b",
  col = "red",
  ylim = c(280, 600)
)
points(2:15, testScore[2:15], type = "b", col = "blue")
legend(
  "topright",
  lty = c(1, 1, 1),
  col = c("red", "blue"),
  legend = c("Train", "Validation")
)
best_Tree = prune.tree(fit_devTree, best = 4)
pred_best = predict(best_Tree, newdata = valid, type = "class")
plot(best_Tree)
text(best_Tree)
misclass_best = misclass(pred_best, test$good_bad)
print(misclass_best)
#2.4
#Fitting and prediction with Naive Bayes
fit_bayes = naiveBayes(good_bad ~ ., data = train, type = c("class"))
pred_bayesTrain = predict(fit_bayes, newdata = train)
conf_bayesTrain = table(Prediction = pred_bayesTrain, Truth = train$good_bad)
misclass_bayesTrain = misclass(pred_bayesTrain, train$good_bad)
print(conf_bayesTrain)
print(misclass_bayesTrain)
pred_bayesTest = predict(fit_bayes, newdata = test)
conf_bayesTest = table(Prediction = pred_bayesTest, Truth = test$good_bad)
misclass_bayesTest = misclass(pred_bayesTest, test$good_bad)
print(conf_bayesTest)
print(misclass_bayesTest)
```

```
#2.5
#Create ROC curves with Naive Bayes and tree
fit_bayes = naiveBayes(good_bad ~ ., data = train)
pred_bayesTest = predict(fit_bayes, newdata = test, type = "raw")
fit_devTree = tree(good_bad ~ ., data = train, split =  c("deviance"))
best_Tree = prune.tree(fit_devTree, best = 4)
pred_best = predict(best_Tree, newdata = test, class = "raw")
pi = seq(from = 0.05, to = 0.95, by = 0.05)
probMatBayes = probMat(pi, pred_bayesTest)
rocBayes = ROC(pi, probMatBayes)
probMatTree = probMat(pi, pred_best)
rocTree = ROC(pi, probMatTree)
plot(
  rocBayes$FPR,
  rocBayes$TPR,
  xlab = "FPR",
  ylab = "TPR",
  type = "l",
  col = "blue",
  xlim = c(0, 1),
  ylim = c(0, 1)
)
points(rocTree$FPR, rocTree$TPR, col = "red", type = "l")
legend(
  "topleft",
  lty = c(1, 1),
  legend = c("Naive Bayes", "Optimal Tree"),
  col = c("blue", "red")
)
#2.6
#Create confusion matrix and calculate misclassification rate with Naive Bayes and loss matrix
```

```
loss = matrix(c(0, 1, 10, 0), 2, 2)

fit_bayes = naiveBayes(good_bad ~ ., data = train)

pred_lossTrain = predict(fit_bayes, train, type = "raw")

classify_lossTrain = c()

classify_lossTrain = ifelse(pred_lossTrain[, 1] / pred_lossTrain[, 2] > loss[2, 1] / loss[1, 2], "bad",
"good")

misclass_lossTrain = misclass(classify_lossTrain, train$good_bad)

conf_lossTrain = table(Truth = train$good_bad, Prediction = classify_lossTrain)

print(misclass_lossTrain)

print(conf_lossTrain)

pred_lossTest = predict(fit_bayes, test, type = "raw")

classify_lossTest = c()

classify_lossTest = ifelse(pred_lossTest[, 1] / pred_lossTest[, 2] > loss[2, 1] / loss[1, 2], "bad", "good")

misclass_lossTest = misclass(classify_lossTest, test$good_bad)

conf_lossTest = table(Truth = test$good_bad, Prediction = classify_lossTest)

print(misclass_lossTest)

print(conf_lossTest)
```

**Assignment 4**

```
setwd("C:/Users/Christoffer Eduards/OneDrive/R/Lab2/Assignment 4")

library(fastICA)

data=read.csv2("NIRSpectra.csv")

set.seed(12345)

data1=data

data1$Viscosity=c()

#4.1

#Conduct a standard PCA

lambda=res$sdev^2

res=prcomp(data1)

#eigenvalues

lambda

#proportion of variation

sprintf("%2.3f",lambda/sum(lambda)*100)
```

```
screeplot(res)
plot(res$x[,1], res$x[,2], xlab="PC1", ylab="PC2")
vars=apply(data1,2,var)
plot(vars)
#4.2
#Trace plots of loadings of the components from 4.1
U = res$rotation
plot(U[,1], main = "Traceplot, PC1")
plot(U[,2], main = "Traceplot, PC2", ylim = c(-0.1,0.5))
#4.3
#Independent component analysis with fastICA
ica = fastICA(data1, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
        method = "R", row.norm = FALSE, maxit = 200, tol = 0.0001, verbose = TRUE)
Wp = ica$K %*% ica$W
plot(Wp[,1], main = "Traceplot Wp1")
plot(Wp[,2], main = "Traceplot Wp2")
plot(ica$S[,1], ica$S[,2], main = "Score plot ICA")
```