TDDE01 - Computer lab 2
Group report

Emil Frid (emifr995)
Emir Alkazhami (emial637)
Nigel Cole (nigco547)

December 2019

Abstract

Each member contributed an assignment each to this report. Nigel contributed with assignment 1, Emir contributed with assignment 2 and Emil contributed with assignment 4. We were all acitve with solving the lab individually and shared ideas orally.

1 Assignment 1 - LDA and logistic regression

1.1 Task 1

Yes, the data for Male and Female are nicely split and the LDA boundary can be placed between them. The distributions have similiar shapes but with different angles.

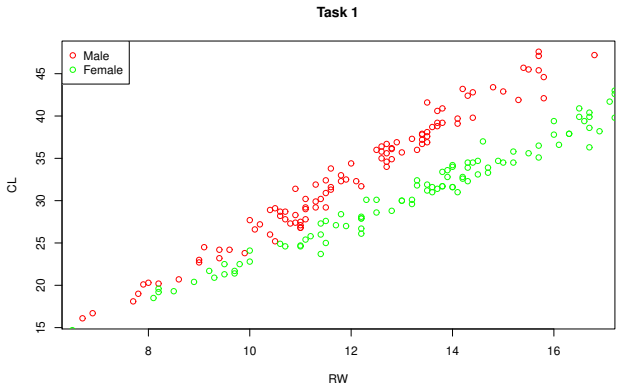

Figure 1: The male and female data groups for task 1.

1.2 Task 2

With LDA we see that a few of the data points shift sex. The points that shifted are located where the groups are connected. The fit is good. The misclassification error rate is 0.035. Confusion matrix:

$$\begin{array}{cc} & \begin{array}{cc} \multicolumn{2}{c}{\text{True}} \\ 0 & 1 \end{array} \\ \text{Prediction} \begin{array}{c} 0 \\ 1 \end{array} & \begin{pmatrix} 97 & 4 \\ 3 & 96 \end{pmatrix} \end{array} \tag{1}$$

1.3 Task 3

With the priors $p(Male) = 0.9, p(Female) = 0.1$ we instead have a lot more points that shift sex from Female to Male with the misclassification error rate of 0.08. The prior probabilities are the probabilities of an observation coming from a particular group. Because the prior probability for Male is higher than the prior probability for Female, the function will make it more likely that an

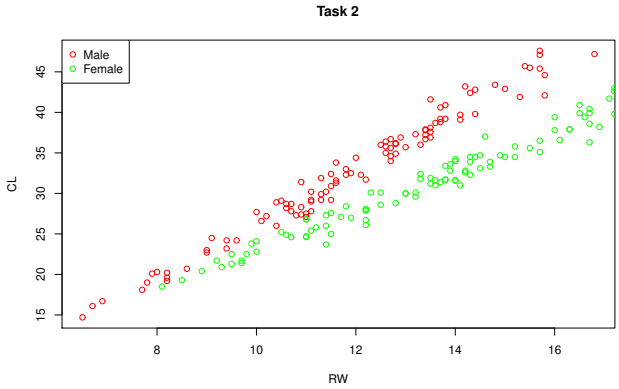

Figure 2: LDA implemented, a few points changed sex for task 2.

observation will be classified as a Male. The confusion matrix is:

$$\begin{array}{cc} & \text{True} \\ & \begin{array}{cc} 0 & 1 \end{array} \\ \text{Prediction} \begin{array}{c} 0 \\ 1 \end{array} & \begin{pmatrix} 84 & 0 \\ 16 & 100 \end{pmatrix} \end{array} \tag{2}$$

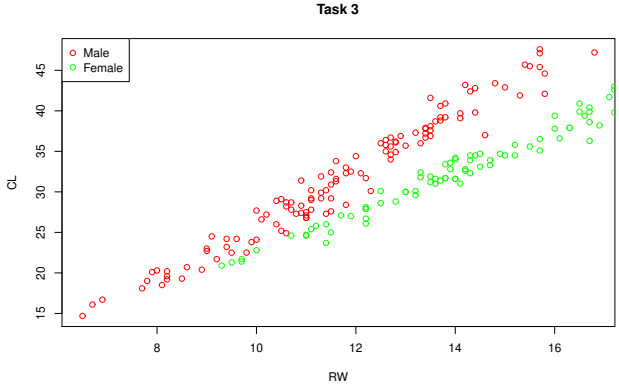

Figure 3: LDA with added priors, many points shift sex for task 3.

1.4 Task 4

With the *glm*() function we get a misclassification error rate of 0.035 which is the same as for task 2. Certain points have, compared to task 3 with LDA,

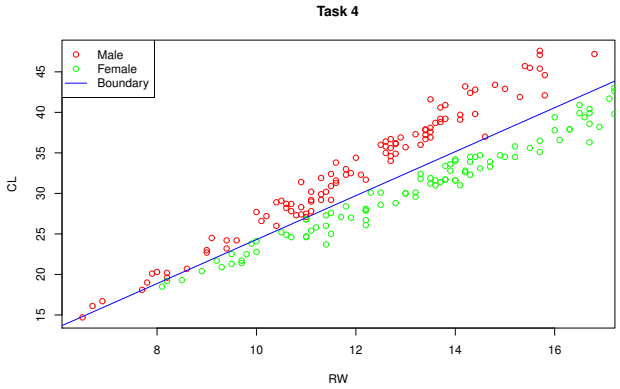

Figure 4: Using the $glm()$ function instead and also adding the decision boundary for task 4.

shifted from Male to Female. The actual points that are Male and Female are not the same as in task 2, even though the misclassification error rate is the same. The equation used for the decision boundary is

$$boundary = RW * x + CL * y + intercept \tag{3}$$

where y in R is

$$y = (boundary - m\_coef[3] * x - m\_coef[1])/m\_coef[2] \tag{4}$$

where the elements in m_coef are the coefficients of the $glm()$ model. The confusion matrix is:

$$\begin{array}{cc} & \text{True} \\ & \begin{array}{cc} 0 & 1 \end{array} \\ \text{Prediction} \begin{array}{c} 0 \\ 1 \end{array} & \begin{pmatrix} 97 & 4 \\ 3 & 96 \end{pmatrix} \end{array} \tag{5}$$

2 Assignment 2 - Analysis of credit scoring

2.1 Task 2

		Train	Test
Misclassification rates:	Gini	23.8%	37.2%
	Deviance	21.2%	26.8%

The deviance measure gives the best result.

2.2 Task 3

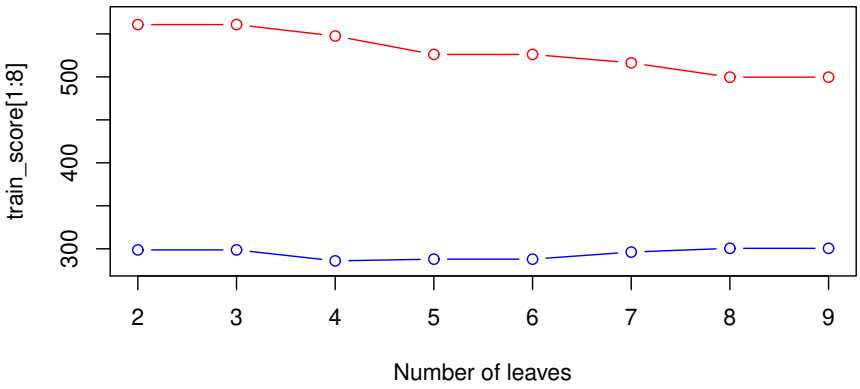

Figure 5: Red: train, Blue: test

Four leaves seem to give the best results.

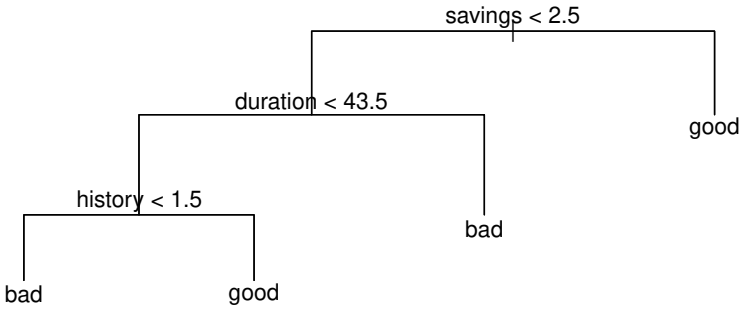

Figure 6: The resulting best pruned tree

The final tree have depth 3 and uses the variables: savings, duration and history. The variable names are not self explanatory, but it seems like a person with large savings who is taking a loan over a short time span is more likely to pay the

debt back.
The misclassification rate on the test data is: 25.6%
Confusion matrix for test data:

$$\begin{array}{cc} & \begin{matrix}\text{True} \\ Bad & Good\end{matrix} \\ \begin{matrix}\text{Prediction} \\ Bad \\ Good\end{matrix} & \begin{pmatrix} 18 & 6 \\ 58 & 168 \end{pmatrix} \end{array} \tag{6}$$

2.3 Task 4

Misclassification rate on training data: 30.0%
Misclassification rate on testing data: 31.6%

Confusion matrix for training data:

$$\begin{array}{cc} & \begin{matrix}\text{True} \\ Bad & Good\end{matrix} \\ \begin{matrix}\text{Prediction} \\ Bad \\ Good\end{matrix} & \begin{pmatrix} 95 & 98 \\ 52 & 225 \end{pmatrix} \end{array} \tag{7}$$

Confusion matrix for test data:

$$\begin{array}{cc} & \begin{matrix}\text{True} \\ Bad & Good\end{matrix} \\ \begin{matrix}\text{Prediction} \\ Bad \\ Good\end{matrix} & \begin{pmatrix} 46 & 49 \\ 30 & 125 \end{pmatrix} \end{array} \tag{8}$$

The decision tree gets a better misclassification rate than the Naive Bayes classifier. This is because the decision tree is better at classifying the 'Good' class than the 'Bad' class, and there is many more samples of class 'Good'.

This does not have to mean that the tree is a better classifier though. It seems biased towards the 'Good' class since it's the larger one, and it's possible to argue that the the Naive Bayes solution generalizes better for new data because the error rate is about the same for the two classes.

We try to explain this by the fact that the training data is unbalanced. There are 147 samples of class 'Bad' and 323 samples of class 'Good'. This will affect the tree classification since the trees use majority voting when assigning class labels to its leaves. Therefore the tree will be more prone the predict a sample as 'Good'. This is shown in confusion matrix (6) with many misclassifications of class 'Bad'.

The Naive Bayes classifier on the other hand, is based on Bayes theorem and takes the class-proportions in to account.

$$P(Y = y|X = x) \propto P(X = x|Y = y)P(Y = y)$$

We assume that the class distributions are Gaussian and that we can estimate $P(X = x|Y = y)$ from the data. $P(Y = y)$ are the class-proportions.

2.4 Task 5

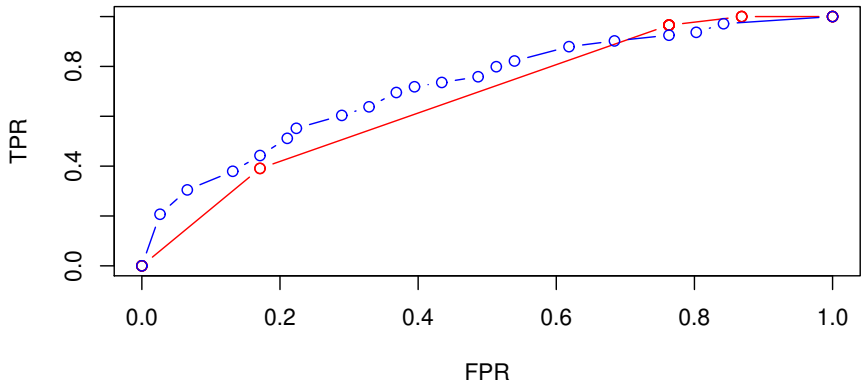

Figure 7: ROC curve. Red: Decision tree, Blue: Naive Bayes

The Naive Bayes classifier have more area under the curve and is therefore better for this data.

2.5 Task 6

Misclassification rate for the training data: 54.6%
Misclassification rate for the test data: 50.8%

Confusion matrix for training data:

$$\begin{array}{cc} & \begin{array}{cc} \text{True} & \\ \textit{Bad} & \textit{Good} \end{array} \\ \begin{array}{c} \text{Prediction} \\ \\ \end{array} \begin{array}{c} \textit{Bad} \\ \textit{Good} \end{array} & \begin{pmatrix} 137 & 263 \\ 10 & 90 \end{pmatrix} \end{array} \tag{9}$$

Confusion matrix for test data:

$$\begin{array}{cc} & \begin{array}{cc} \text{True} & \\ \textit{Bad} & \textit{Good} \end{array} \\ \begin{array}{c} \text{Prediction} \\ \\ \end{array} \begin{array}{c} \textit{Bad} \\ \textit{Good} \end{array} & \begin{pmatrix} 71 & 122 \\ 5 & 52 \end{pmatrix} \end{array} \tag{10}$$

In task 6 we have a worse misclassification rate than in task 4. This is because we are misclassifying a lot more samples of class 'Good' than in task 4. This happens because we are weighting the loss-values for classifying a sample of type 'Bad' by 10, basically shifting the our prediction boundary towards the 'Good' class.

In other words, we say that classifying a sample of class 'Bad' wrong is worse than misclassifying a sample of class 'Good'. Therefore we take a safe approach and say that we have to be really certain in our prediction to classify it as 'Good'.

3 Assignment 4 - Principal components

3.1 Task 1

A standard PCA was conducted using the feature space. The PCA showing how much variation is explained by each feature is presented in Figure 8.

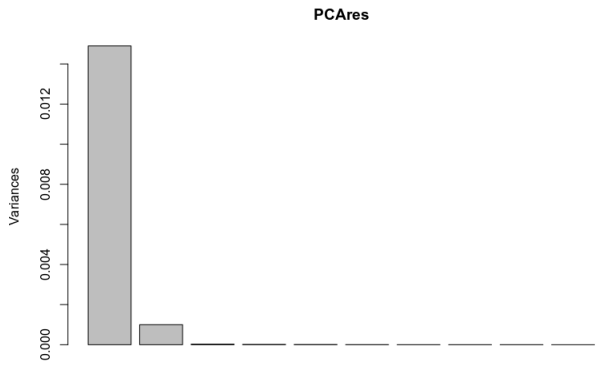

Figure 8: The PCA showing how much variation is explained by each feature.

In Figure 8 it is clear that the first two features stand for the majority of the variations, but not all variations and in some situations those small percentages are important. So it is not necessarily clear how many PC that should be extracted.

The first two components are enough to explain at least 99 % of the total variance.

Figure 9 presents the scores in the coordinates (PC1,PC2). In the figure it is seen that there are some outliers.

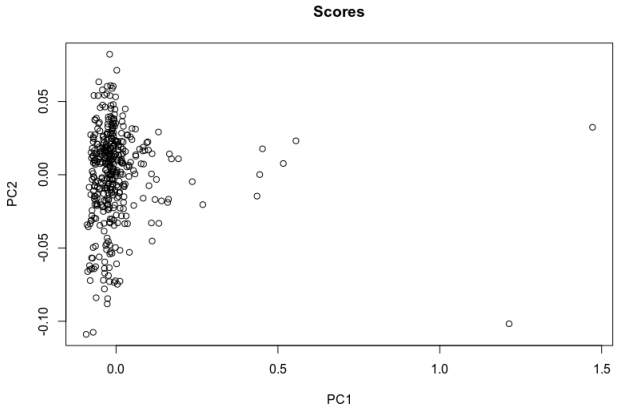

Figure 9: The first two scores.

3.2 Task 2

Figures 10 and 11 presents the trace plots of the loadings of the components selected in Task 1.

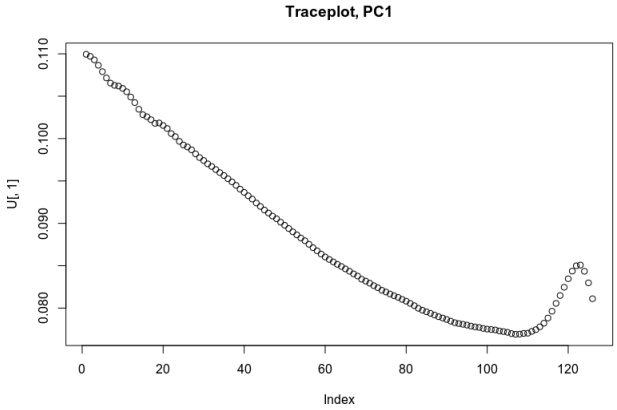

Figure 10: The trace plot for PC1.

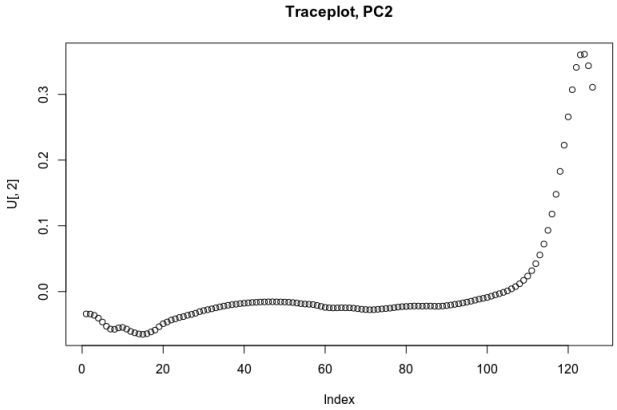

Figure 11: The trace plot for PC2.

The larger absolute value in the trace plots the more those features explain the PC. Therefore, it is seen in Figure 11 that PC2 is mostly explained by the last 14, or so, features.

3.3 Task 3

An Independent Component Analysis (ICA) was performed with the number of components selected in Task 1. The $W' = KW$ was computed and the columns of W' are presented in form of trace plots in Figures 12 and 13.

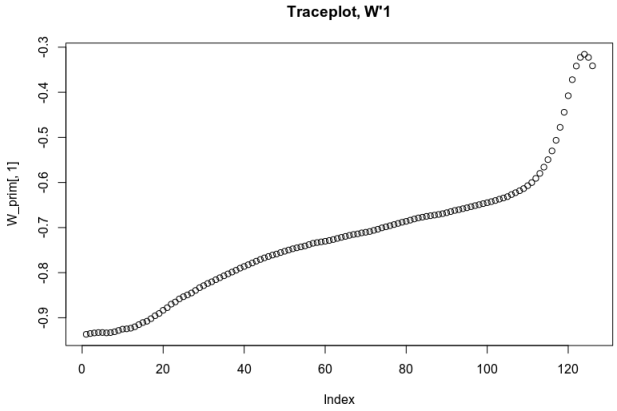

Figure 12: The trace plot for the first column in W'.

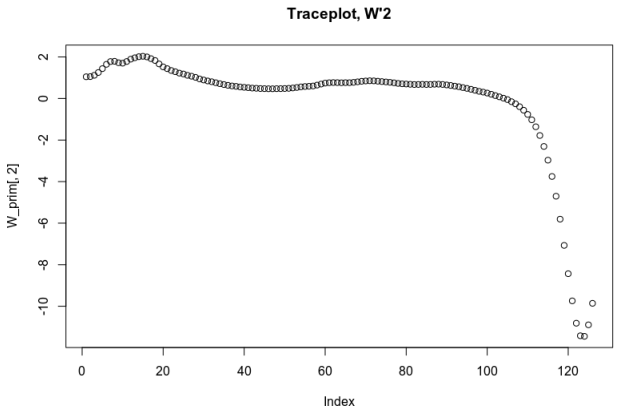

Figure 13: The trace plot for the first column in W'.

Compared to the trace plots in Task 2 these seem to be inverted and scaled larger. W' is the complete transformation and selection matrix for the ICA, the U_{ICA}.

Figure 14 presents the scores from ICA.

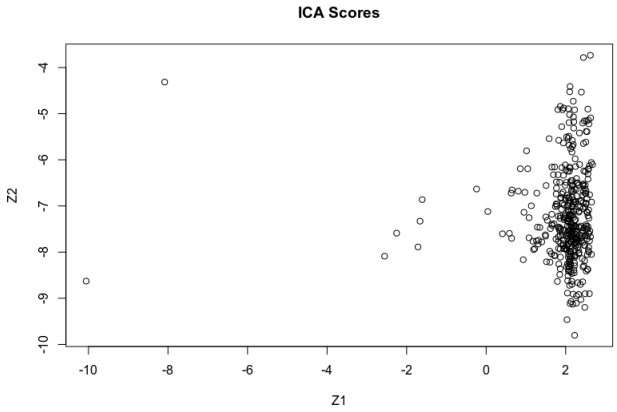

Figure 14: The ICA scores.

Just as for the trace plots the scores are inverted and scaled larger compared to the scores from PCA in Task 1.

A R code

Here is the code presented.

A.1 Assignment 1

```
RNGversion('3.5.1')
setwd("/home/nigel/Documents/Liu/TDDE01/Lab2")

# Task 1
data = read.csv("australian-crabs.csv")
CL = data$CL
RW = data$RW
sex = data$sex
sexMale = which(sex %in% "Male")
sexFemale = which(sex %in% "Female")

plot(RW[sexMale], CL[sexMale], col = "red", xlab = "RW", ylab = "CL", main = "Task 1")
points(RW[sexFemale], CL[sexFemale], col = "green")
legend("topleft", pch = 1, col = c("red", "green"), legend = c("Male", "Female"))
# Yes, data groups are nicely split and the LDA boundary can be placed nicely.

# Task 2
library(MASS)
model = lda(formula = sex ~ CL + RW, data = data)
pred_model = predict(object = model, newdata = data, type = "response")

pred_male = pred_model$class == "Male"
pred_female = pred_model$class == "Female"

plot(RW[pred_male], CL[pred_male], col = "red", xlab = "RW", ylab = "CL", main = "Task 2")
points(RW[pred_female], CL[pred_female], col = "green")
legend("topleft", pch = 1, col = c("red", "green"), legend = c("Male", "Female"))

# Misclassification
misclass=function(X,X1){n=length(X)
return(1-sum(diag(table(X,X1)))/n)
}

# Confusion matrix and misclass. error rate = 0.035
compare = 1:length(sex)
compare[pred_male] = "Male"
compare[pred_female] = "Female"
misclass(compare, sex)
table(compare, sex)

# Task 3
library(MASS)
model = lda(formula = sex ~ CL + RW, data = data, prior = c(0.1, 0.9))
```

```
pred_model = predict(object = model, newdata = data, type = "
    response")

pred_male = pred_model$class == "Male"
pred_female = pred_model$class == "Female"

plot(RW[pred_male], CL[pred_male], col = "red", xlab = "RW", ylab =
     "CL", main = "Task 3")
points(RW[pred_female], CL[pred_female], col = "green")
legend("topleft", pch = 1, col = c("red", "green"), legend = c("
    Male", "Female"))

# Confusion matrix and misclass. error rate = 0.08
compare = 1:length(sex)
compare[pred_male] = "Male"
compare[pred_female] = "Female"
misclass(compare, sex)
table(compare, sex)

# Task 4
model = glm(formula = sex ~ CL + RW, family = "binomial", data =
    data)
pred_model = predict(object = model, newdata = data, type = "
    response")

pred_male = pred_model > 0.5
pred_female = pred_model <= 0.5

plot(RW[pred_male], CL[pred_male], col = "red", xlab = "RW", ylab =
     "CL", main = "Task 4")
points(RW[pred_female], CL[pred_female], col = "green")
legend("topleft", pch = c(1, 1, NA), lty = c(NA, NA, 1), col = c("
    red", "green", "blue"), legend = c("Male", "Female", "Boundary"
    ))

# boundary = RW*x + CL*y + intercept
boundary = 0.5
m_coef = model$coefficients
x = c(6, 18)
y = (boundary - m_coef[3] * x - m_coef[1]) / m_coef[2]
lines(x, y, col = "blue")

# Confusion matrix and misclass. error rate = 0.035
compare = 1:length(sex)
compare[pred_male] = "Male"
compare[pred_female] = "Female"
misclass(compare, sex)
table(compare, sex)
```

Code/ass1.R

A.2 Assignment 2

```
RNGversion('3.5.1')
cat(rep("\n", 30))
```

```r
rm(list = ls())

##############
#---Task 1---
##############
library(readxl)
source("lab1/missclass.R")

data = read_excel("lab2/creditscoring.xls")
data[["good_bad"]] = as.factor(data[["good_bad"]])

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,]

##############
#---Task 2---
##############
library(tree)
model = tree(good_bad~., data=train, split="deviance")
preds_train = predict(model, newdata = train, type="class")
preds_test = predict(model, newdata = test, type="class")
plot(model)
text(model, pretty=0)

deviance_train = missclass(train$good_bad, preds_train)
deviance_test  = missclass(test$good_bad,  preds_test)


model = tree(good_bad~., data=train, split="gini")
preds_train = predict(model, newdata = train, type="class")
preds_test = predict(model, newdata = test, type="class")
plot(model)
text(model, pretty=0)

gini_train = missclass(train$good_bad, preds_train)
gini_test  = missclass(test$good_bad,  preds_test)

##############
#---Task 3---
##############
model = tree(good_bad~., data=train, split="deviance")

train_score=rep(0,8)
test_score=rep(0,8)

for(i in 2:9) {
  pruned_tree = prune.tree(model, best=i)
  pred = predict(pruned_tree, newdata=valid, type="tree")
  train_score[i-1] = deviance(pruned_tree)
```

```
  test_score[i-1]  = deviance(pred)
}

plot(2:9, train_score[1:8], type="b", col="red", ylim=c(280,570),
    xlab="Number of leaves")
points(2:9, test_score[1:8], type="b", col="blue")

BEST_INDEX = 4
final_tree = prune.tree(model, best=BEST_INDEX)
plot(final_tree)
text(final_tree, pretty=0)

pred_test  = predict(final_tree, newdata=test, type="class")
miss_test  = missclass(test$good_bad, pred_test)
test_table = table(pred_test, test$good_bad)

##############
#---Task 4---
##############
library(e1071)
model = naiveBayes(good_bad~., data=train)

preds_train = predict(model, newdata=train)
preds_test  = predict(model, newdata=test)

train_table = table(preds_train, train$good_bad)
train_rate  = missclass(train$good_bad, preds_train)

test_table = table(preds_test, test$good_bad)
test_rate  = missclass(test$good_bad, preds_test)
print(test_table)

##############
#---Task 5---
##############

#Tree model
model = tree(good_bad~., data=train, split="deviance")
tree_model  = prune.tree(model, best=BEST_INDEX)

pred = predict(final_tree, newdata=test)

PI = seq(0.00, 1.00, 0.05)
tpr_vec_tree = rep(0, length(PI))
fpr_vec_tree = rep(0, length(PI))

for(ind in 1:length(PI)){
  pi = PI[ind]
  new_pred = rep("bad", length(pred[,2]))
  new_pred[pred[,2] > rep(pi, length(pred[,2]))] = "good"

  TP = sum(new_pred=="good" & test$good_bad=="good")
  FP = sum(new_pred=="good" & test$good_bad=="bad")
  TN = sum(new_pred=="bad" & test$good_bad=="bad")
  FN = sum(new_pred=="bad" & test$good_bad=="good")

  tpr_vec_tree[ind] = TP/(TP+FN)
```

```
  fpr_vec_tree[ind] = FP/(FP+TN)
}

#Naive bayes model
bayes_model = naiveBayes(good_bad~., data=train)
pred = predict(bayes_model, newdata=test, type="raw")

tpr_vec_bayes = rep(0, length(PI))
fpr_vec_bayes = rep(0, length(PI))

for(ind in 1:length(PI)){
  pi = PI[ind]
  new_pred = rep("bad", length(pred[,2]))
  new_pred[pred[,2] > rep(pi, length(pred[,2]))] = "good"

  TP = sum(new_pred=="good" & test$good_bad=="good")
  FP = sum(new_pred=="good" & test$good_bad=="bad")
  TN = sum(new_pred=="bad" & test$good_bad=="bad")
  FN = sum(new_pred=="bad" & test$good_bad=="good")

  tpr_vec_bayes[ind] = TP/(TP+FN)
  fpr_vec_bayes[ind] = FP/(FP+TN)
}

plot(fpr_vec_tree, tpr_vec_tree, type="b", col="red", xlab="FPR",
    ylab="TPR")
points(fpr_vec_bayes, tpr_vec_bayes, type="b", col="blue")

##############
#---Task 6---
##############
model = naiveBayes(good_bad~., data=train)

good_scale = 1
bad_scale  = 10

preds_train = predict(model, newdata=train, type="raw")
preds_train_temp = rep("bad", length(preds_train[,1]))
preds_train_temp[bad_scale*preds_train[,1] < good_scale*preds_train
    [,2]] = "good"
preds_train = preds_train_temp

preds_test  = predict(model, newdata=test, type="raw")
preds_test_temp = rep("bad", length(preds_test[,1]))
preds_test_temp[bad_scale*preds_test[,1] < good_scale*preds_test
    [,2]] = "good"
preds_test = preds_test_temp


train_table = table(preds_train, train$good_bad)
train_rate  = missclass(train$good_bad, preds_train)

test_table = table(preds_test, test$good_bad)
test_rate  = missclass(test$good_bad, preds_test)
print(test_table)
```

Code/lab2–2.R

A.3 Assignment 4

```r
#setwd("./lab2")
library(readxl)
library(kknn)
library(MASS)
require(ggplot2)
require(scales)
require(gridExtra)
library(fastICA)
source('~/Documents/University/TDDE01/tdde01/lab1/missclass.R')

## Task 1 #################
data = read.csv2("NIRSpectra.csv")

PCAres=prcomp(formula=~.-Viscosity, data = data)
# The eigenvalues of the covariance matrix
lambda=PCAres$sdev^2
#Proportion of variation. First 2 is needed for >99 %
sprintf("%2.3f",lambda/sum(lambda)*100)

screeplot(PCAres)

#Score plot
plot(PCAres$x[,1], PCAres$x[,2],main = "Scores",xlab = "PC1",ylab =
     "PC2")

## Task 2 #################
U=PCAres$rotation
#head(U)

#The larger abs value in the trace plots the more important those
#variabels are.
plot(U[,1], main="Traceplot, PC1")
plot(U[,2], main="Traceplot, PC2")

## Task 3 #################
# a)
set.seed(12345)

data2 = data
data2$Viscosity = c()

ICAres$K

ICAres = fastICA(data2,2)
# Down here is really a matrix multiplication
# but latex couldn't handle the strange syntax.
W_prim = ICAres$K*ICAres$W
plot(W_prim[,1], main="Traceplot, W'1")
plot(W_prim[,2], main="Traceplot, W'2")

# b) score
X = as.matrix(data2)

Z=ICAres$S

```

```r
plot(Z[,1], Z[,2],xlab = "Z1",ylab="Z2",main="ICA Scores")

# U_ICA = U*V <=> W= K*W
# where U=K from PCA and V=W from ICA.

# Z = S = X*U_ICA
```

Code/assignment4.R