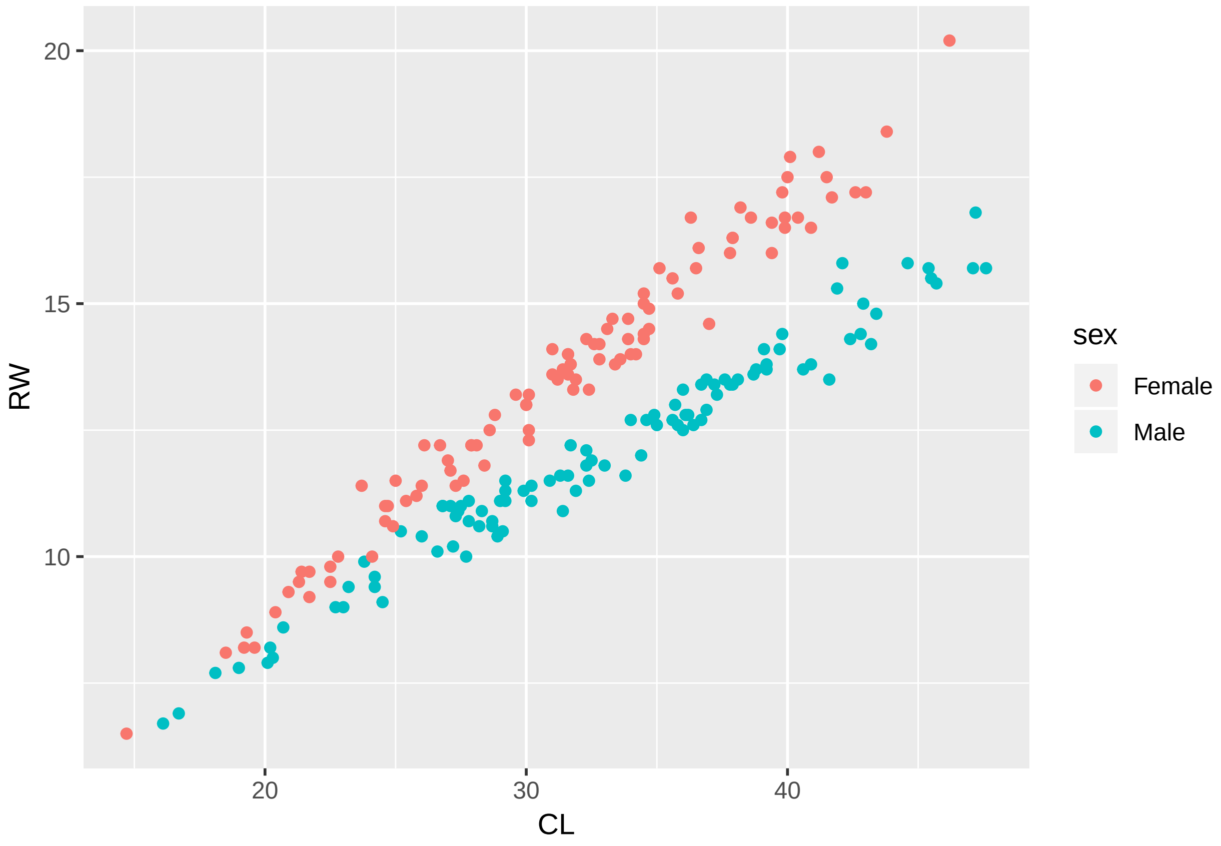
lab1

Erik Tedhamre

8 December 2018

Assignment 1.1

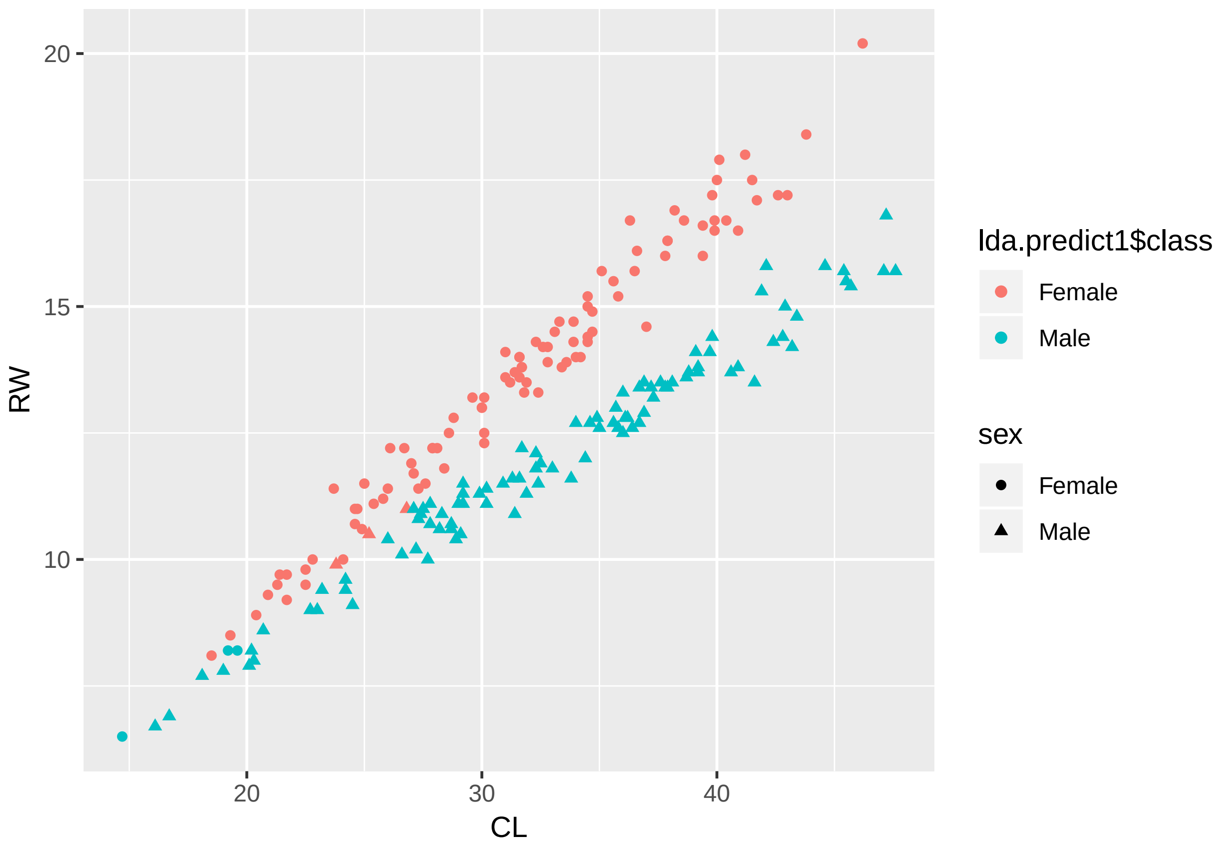
```r
library("ggplot2")
library("MASS")
data <- data.frame(read.csv("australian-crabs.csv"))
ggplot(data = data, mapping = aes(CL, RW, color = sex)) + geom_point()
```



A plot of carapace length versus rear width where the observations are colored by sex. Looking at the graph the data seems reasonably easy to classify by linear discriminant analysis. Because there seems to be a line between the two sexes. There's also a linear relationship betweem the two variables.

Assignment 1.2

```r
set.seed(12345)
lda.model1 <- lda(sex ~ CL + RW, data = data)
lda.predict1 <- predict(lda.model1, data)
ggplot.0.5 <- ggplot(data = data, mapping = aes(CL, RW, color = lda.predict1$class, shape = sex )) + ge
ggplot.0.5
```

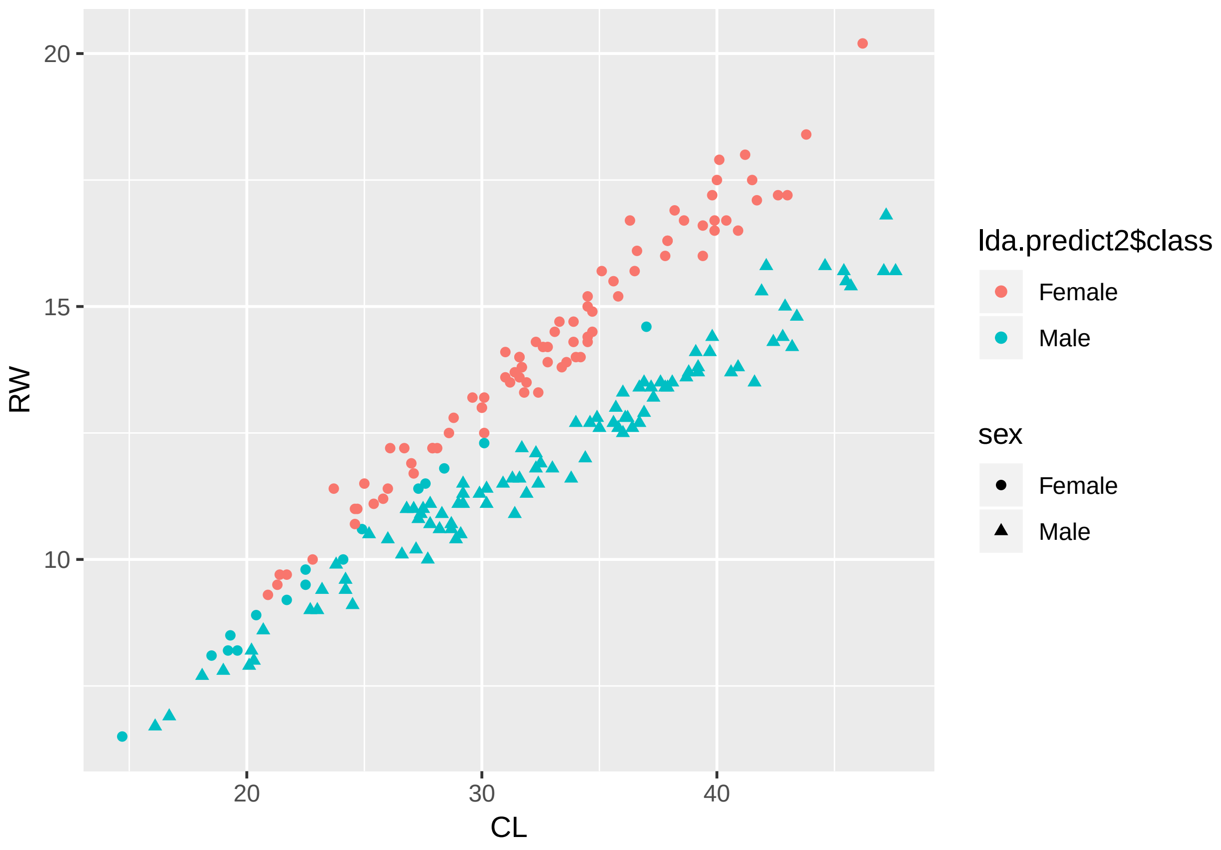


```
mcr.0.5 <- mean(lda.predict1$class != data$sex)
```

The missclassification rate for the linear discriminant analysis is **0.035**. This is pretty reasonable considering we saw on the original graph that there was one area with a bit of an overlap. If this is good enough for actual use is hard to say, it mostly depends on how much we would lose on an incorrect classification.

Assignment 1.3

```
set.seed(12345)
lda.model2 <- lda(sex ~ CL + RW, data = data, prior = c(Female = 0.1, Male = 0.9))
lda.predict2 <- predict(lda.model2, data)
ggplot(data = data, mapping = aes(CL, RW, color = lda.predict2$class, shape = sex )) + geom_point()
```



The number of males increased since we are assuming a wheighted distribution. Especially the areas containing both types of observations are now classified as only males instead of both.

```
mcr.0.9 <- mean(lda.predict2$class != data$sex)
```

The missclassification rate for the weighted linear discriminant analysis is **0.08**.

Assignment 1.4

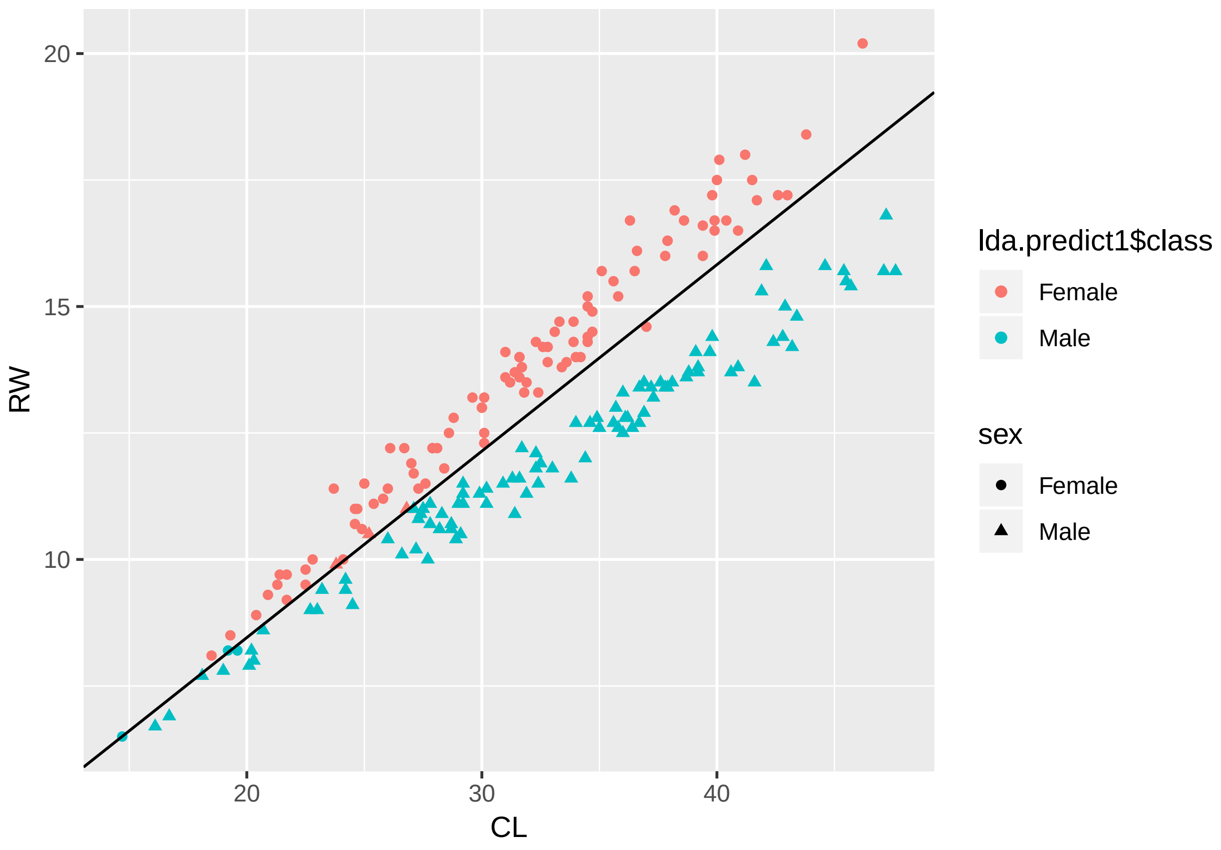
```
set.seed(12345)
glm.model <- glm(as.factor(sex) ~ CL + RW, family = binomial, data = data)
```

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

```
glm.predict <- predict(glm.model, data, type = 'response')
mcr.glm <- mean(glm.predict != data$sex)
```

The missclassification rate is **0.035** which is the same as the original linear discriminant analysis.

```
glm.predict.0.5 <- ifelse(glm.predict > 0.5, "Male", "Female")
glm.slope <- coef(glm.model)[2]/(-coef(glm.model)[3])
glm.intercept <- coef(glm.model)[1]/(-coef(glm.model)[3])
ggplot.0.5 + geom_abline(slope = glm.slope, intercept = glm.intercept)
```



The decision line is drawn in the graph. It's described by $\mathbf{RW = 0.6CL + 1.08379}$.

Assignment 2

Splitting the data into partitions

```
library("e1071")
library("MASS")
library("tree")
library("ggplot2")
setwd("~/TDDE01/lab2")
data.credit <- data.frame(read.csv("creditscoring.csv"))
n=dim(data.credit)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data.credit[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=data.credit[id2,]
id3=setdiff(id1,id2)
test=data.credit[id3,]
```

The models used in calculating the following confusion matrices.

```
set.seed(12345)
tree.fit.dev <- tree(good_bad ~., data = train, split = "deviance")
tree.fit.gini <- tree(good_bad ~., data = train, split = "gini")
```

```r
set.seed(12345)
pred.dev.train <- predict(tree.fit.dev, newdata = train, type="class")
mean(train$good_bad != pred.dev.train)
```

```
## [1] 0.212
```

Missclassification rate for deviance on train data.

```r
set.seed(12345)
pred.dev.test <- predict(tree.fit.dev, newdata = test, type="class")
mean(test$good_bad != pred.dev.test)
```

```
## [1] 0.268
```

Missclassification rate for deviance on test data.

```r
set.seed(12345)
pred.gini.train <- predict(tree.fit.gini, newdata = train, type="class")
mean(train$good_bad != pred.gini.train)
```

```
## [1] 0.236
```

Missclassification rate for gini on train data.

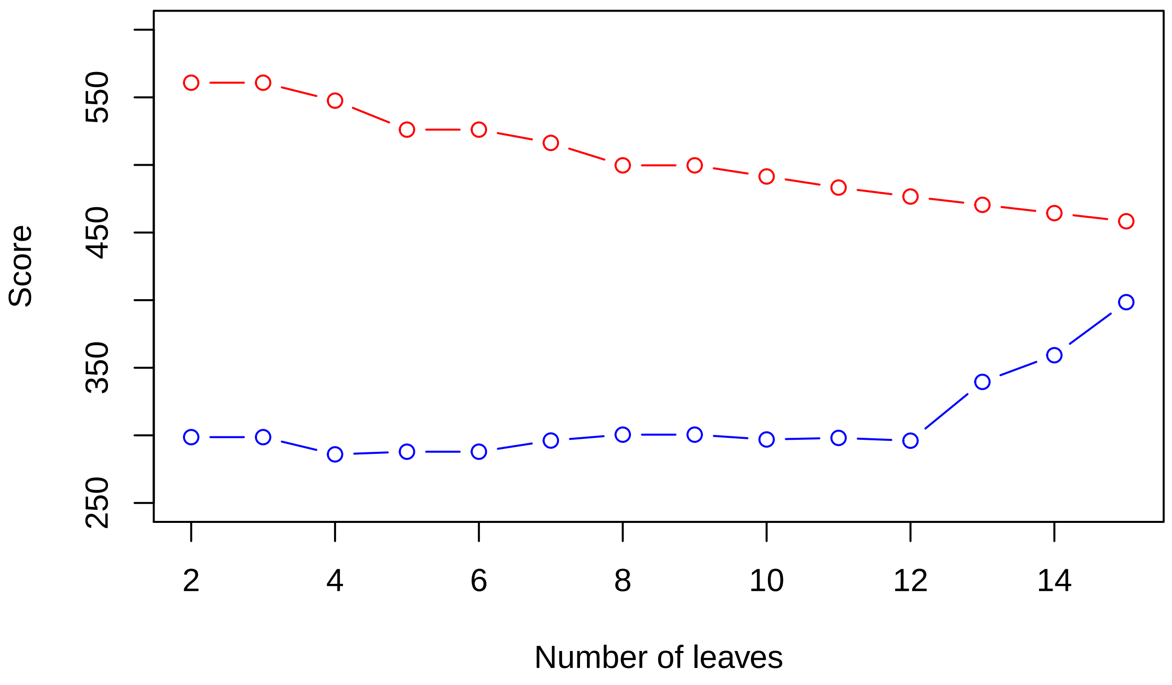
```r
set.seed(12345)
pred.gini.test <- predict(tree.fit.gini, newdata = test, type="class")
mean(test$good_bad != pred.gini.test)
```

```
## [1] 0.364
```

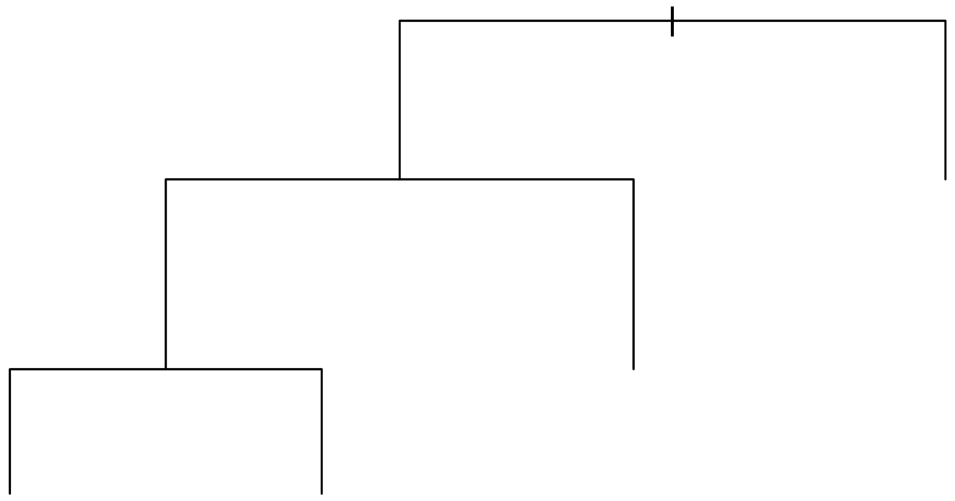
Missclassification rate for gini on test data.

The confusion matrix is the best for deviance compared to gini based on the number of correct predictions for the test data.

Assignment 2.3



Looking at the graph we see a minimum value for 4.



The optimal depth of the tree is three which can be seen in the graph.

```
##       predicted
## actual bad good
##   bad   23   54
##   good  12  161
```

Confusion matrix for the validation data for the tree data

```
##
## Classification tree:
## snip.tree(tree = fit, 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
```

As seen above the variables used in the tree are “savings”, “duration” and “history”.

```
## [1] 0.264
```

Missclassification rate is **0.264**.

Assignment 2.4

```
##       predicted
## actual bad good
##   bad   95   98
##   good  52  255
```

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

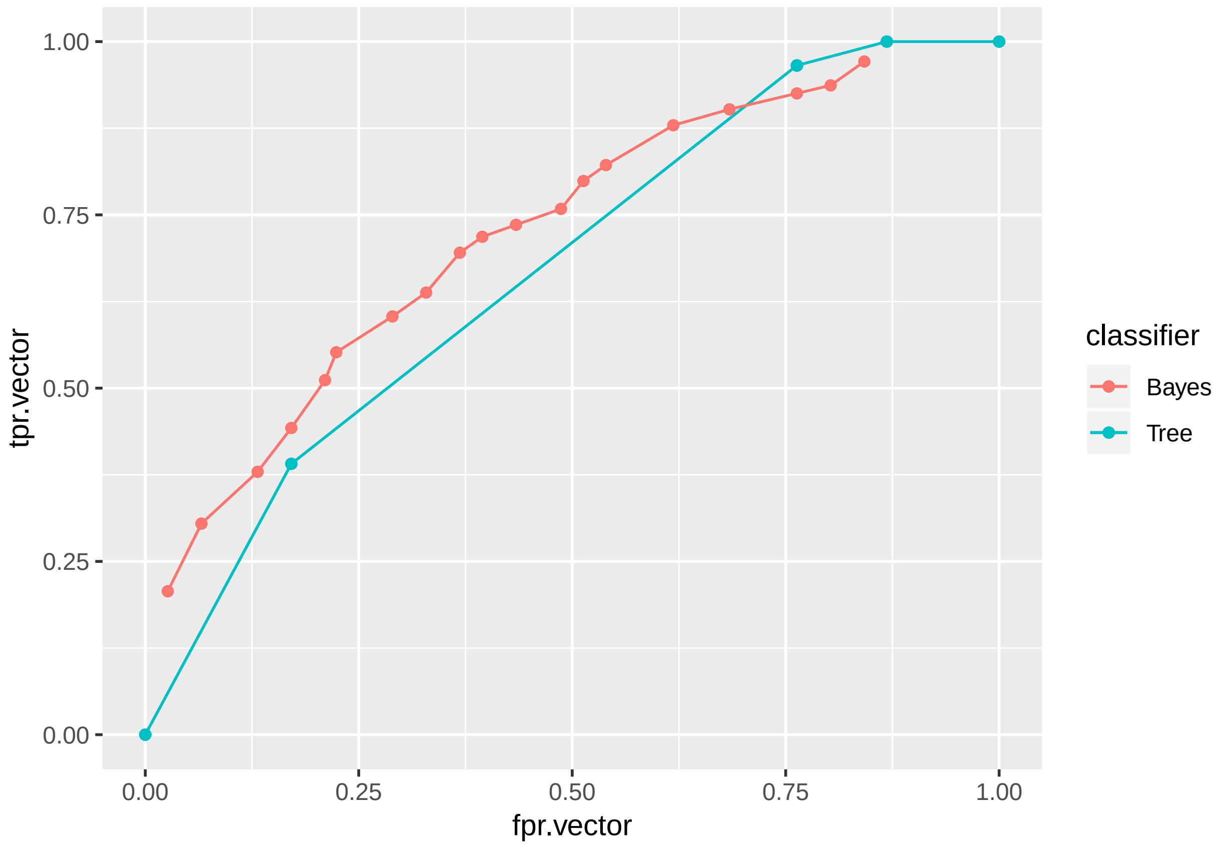
Missclassification rate for bayes on train data.

```
##       predicted
## actual bad good
##   bad   46   49
##   good  30  125
```

```
## [1] 0.316
```

Missclassification rate for bayes on test data.

The tree prediction is a bit better than the bayesian prediction.



Graph of radius of convergence

```
##       predict
## actual bad good
##   bad   71    5
##   good 122   52
```

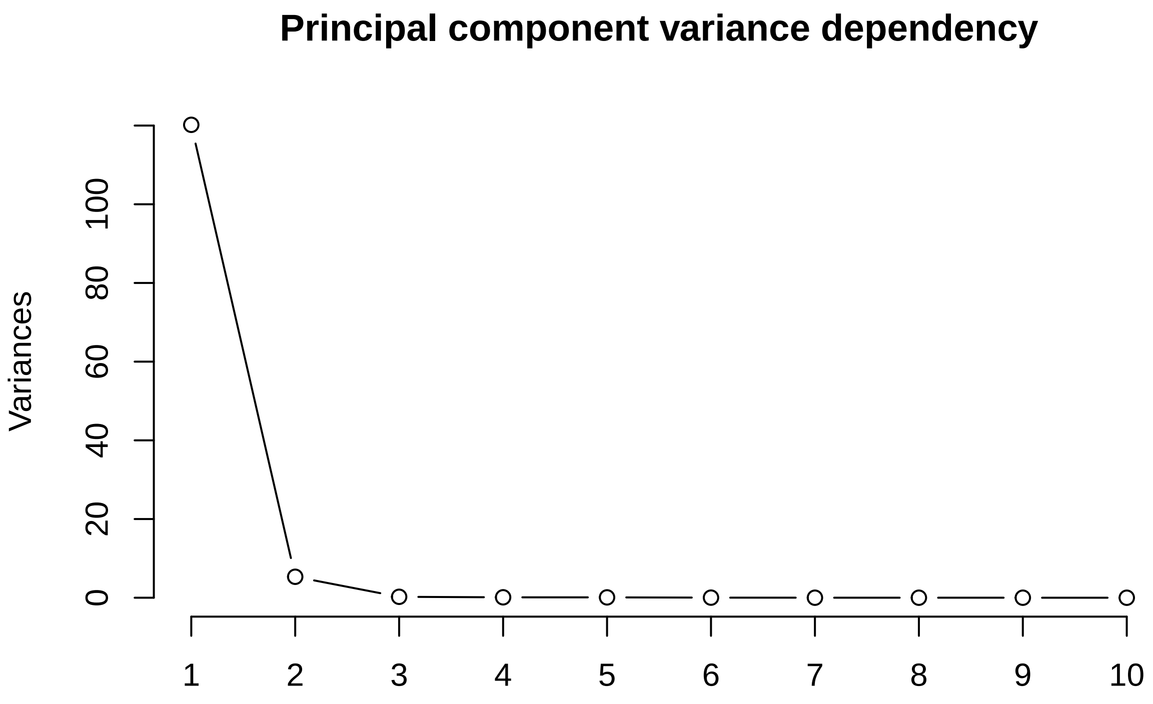
Confusion matrix for bayes with loss matrix on test data.

```
##       predict
## actual bad good
##   bad  137   10
##   good 263   90
```

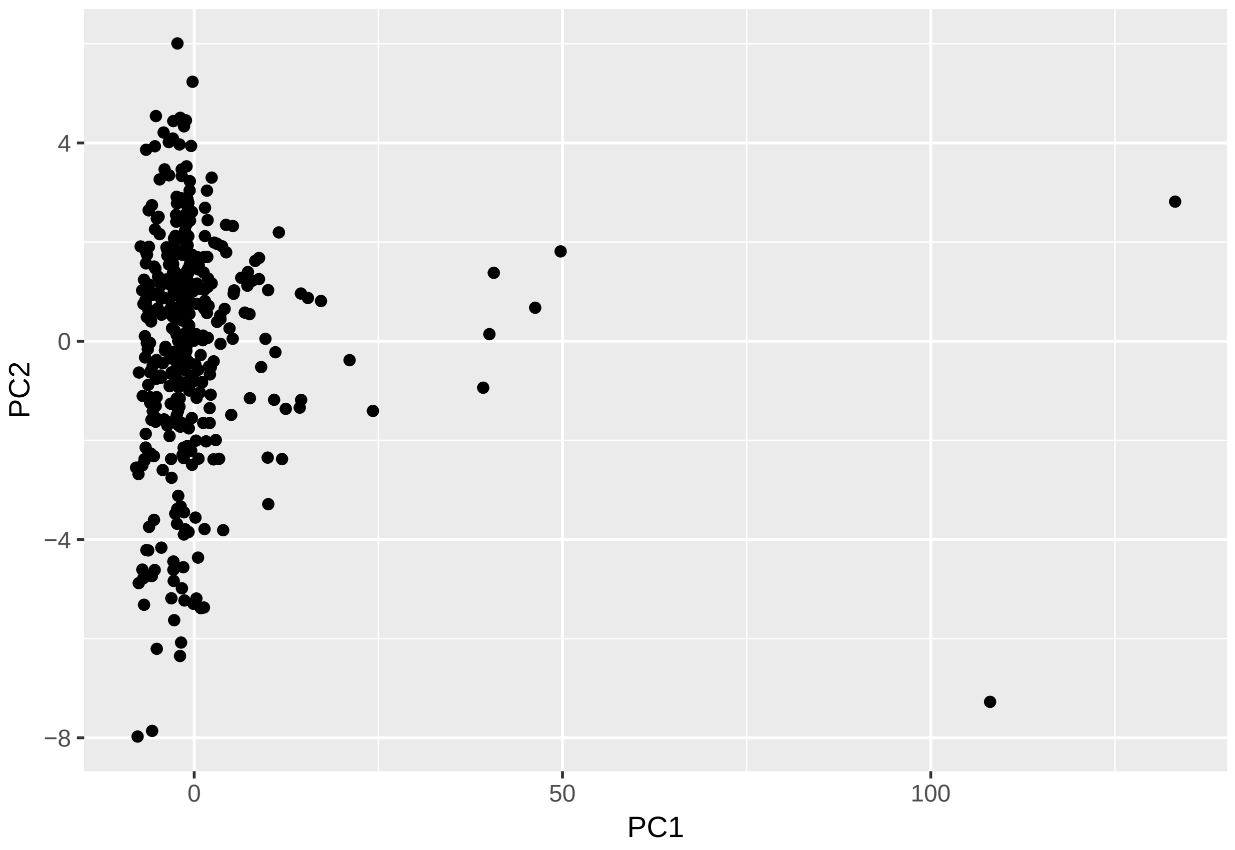
Confusion matrix for bayes with loss matrix on train data.

It's a lot more expensive to have a false positive than it is to have a false negative according to a loss matrix. Meaning that we will be more cautious with the "good" classification. This means that we should have a much higher frequency of data being classified as "bad".

Assignment 4.1

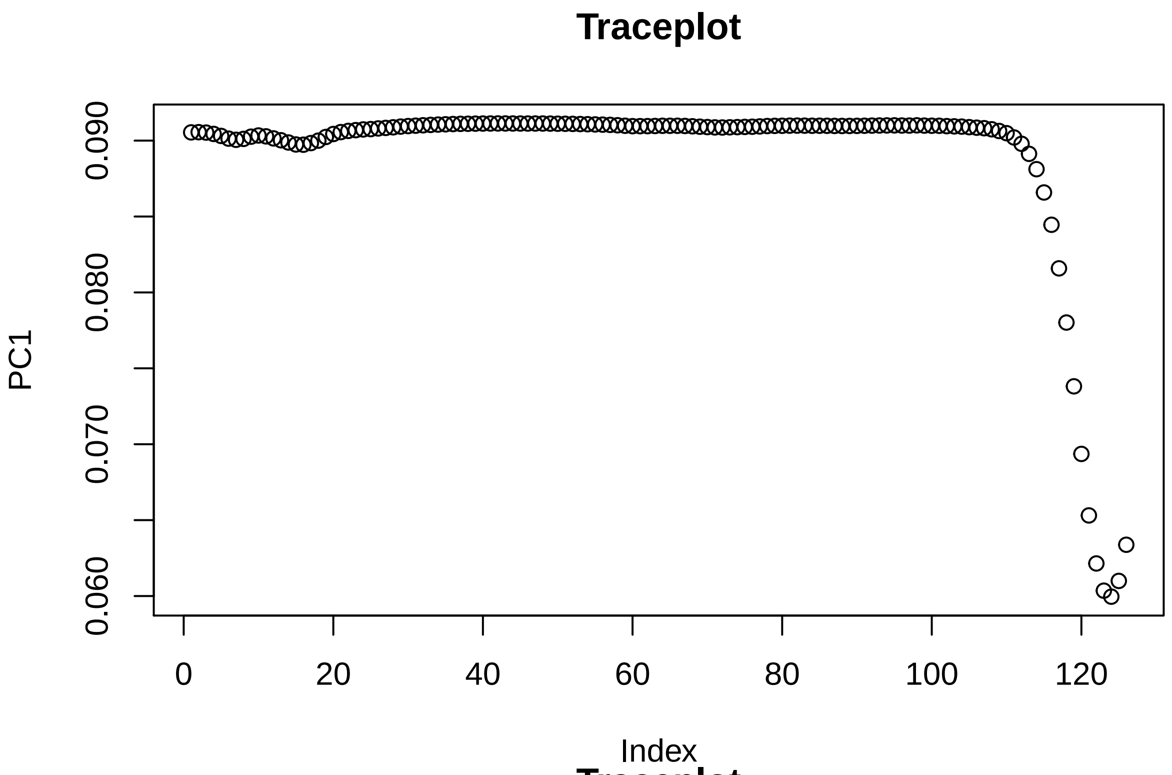


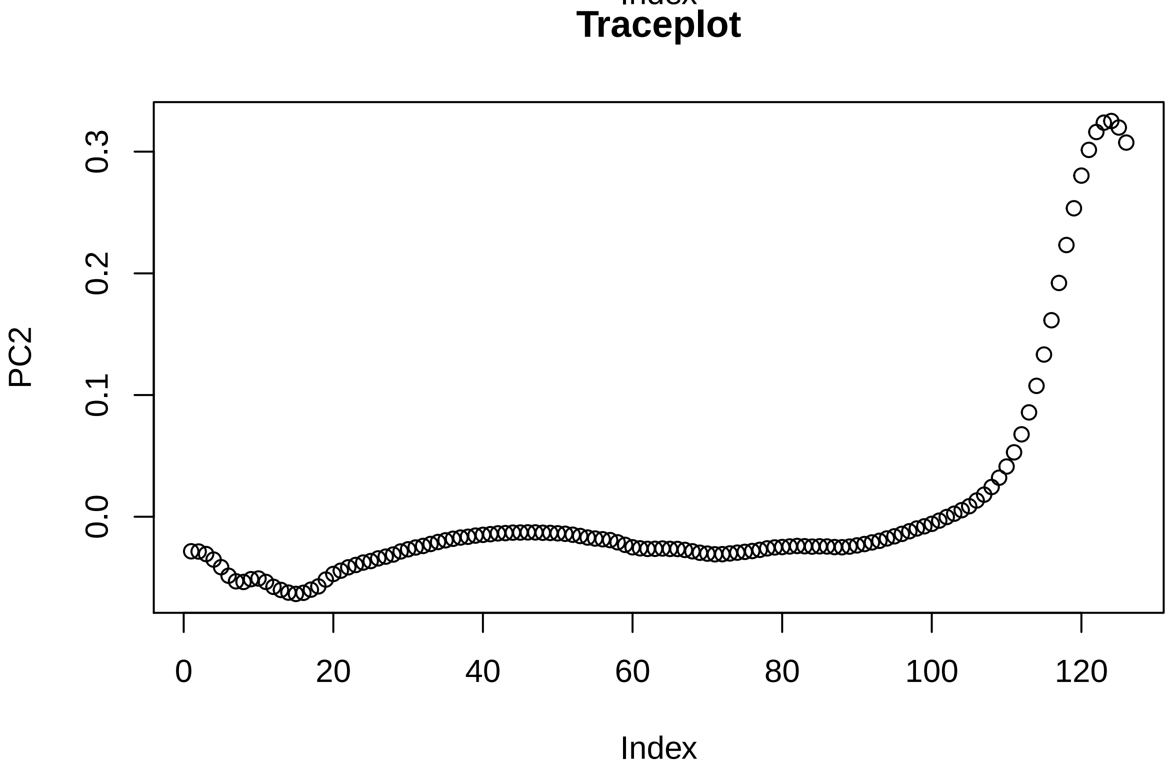
By looking at summary(pca.fit), can see that PC1 and PC2 cumulatively explains 99% of the variance. The summary(pca.fit) is not printed here since it's output is very large.



Plot of the scores in the in the (PC1, PC2) coordinates. There seems to be two fuels that differ greatly from the other, in that they have a much higher coefficent for PC1, while a lot of the other observations seem to lie close to zero.

Assignment 4.2

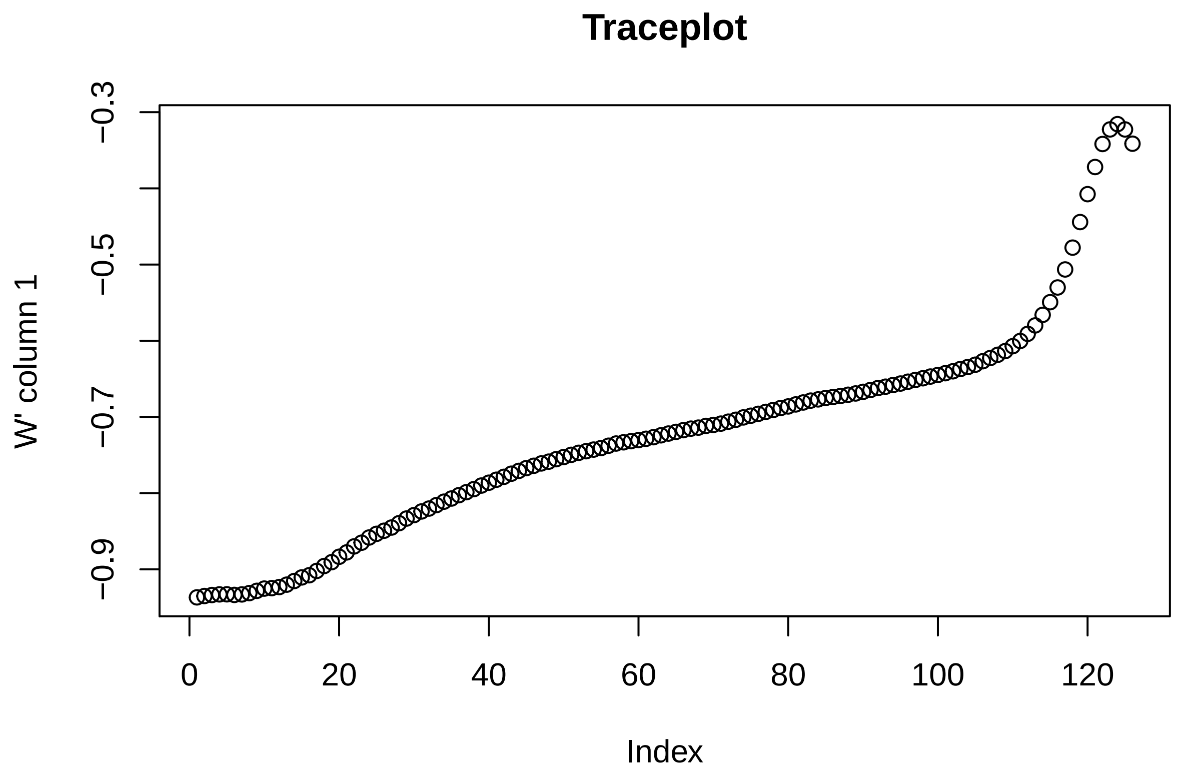


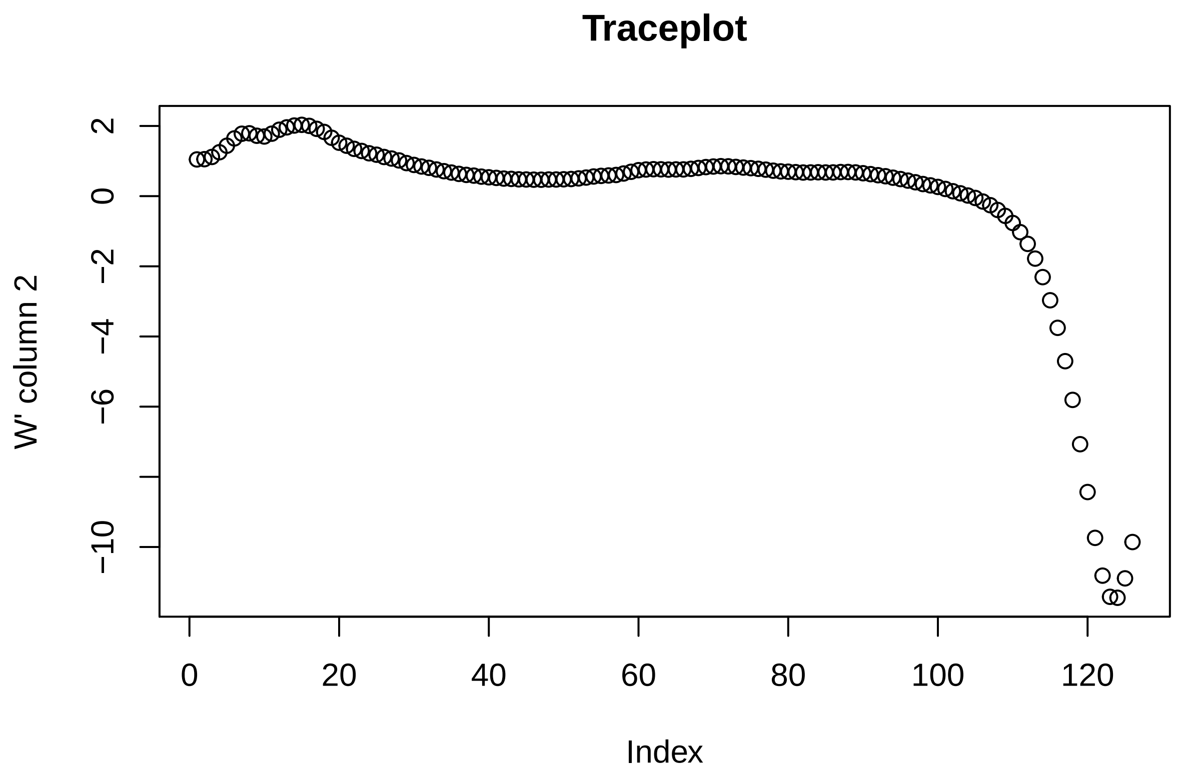


PC1 is mostly represented in equal amount by the different factors, while PC2 is mostly represented by **X864 to X880**.

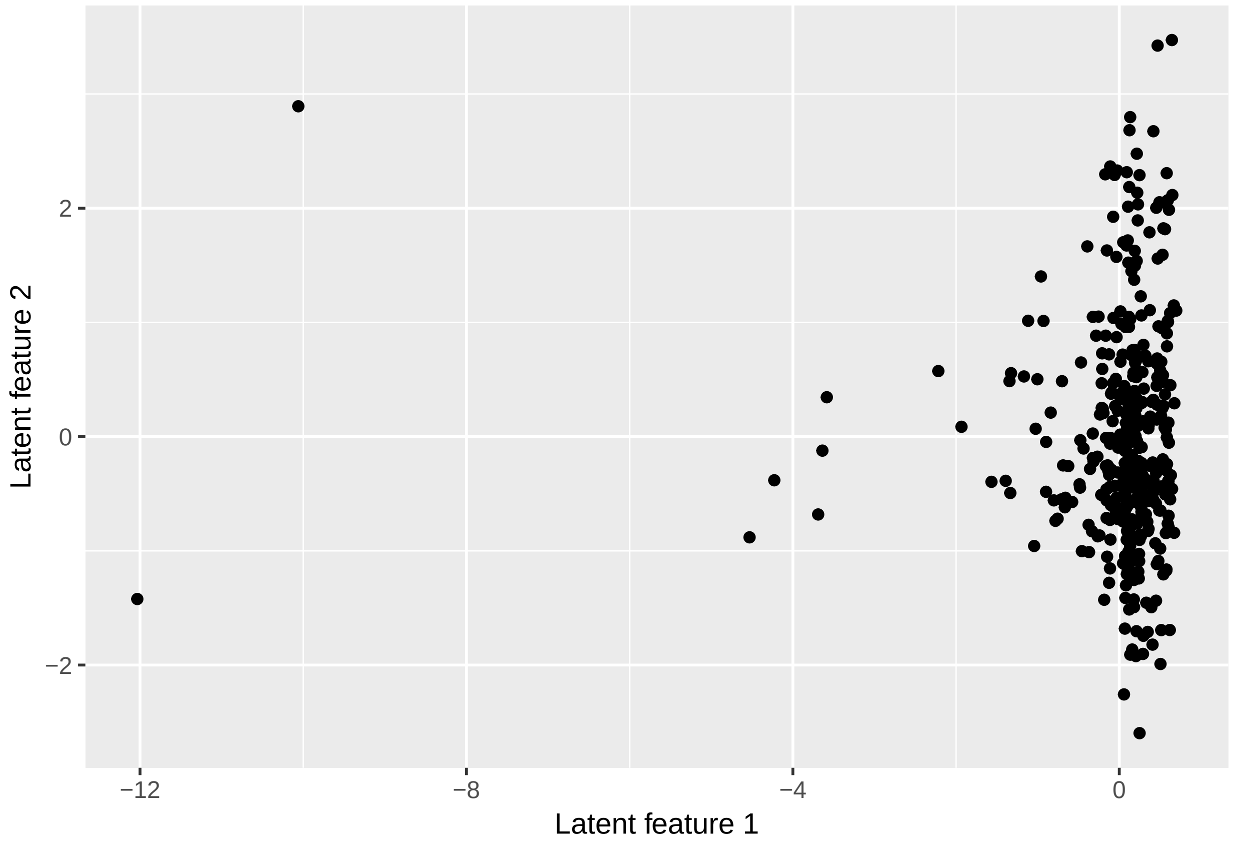
Assignment 4.3

```
## Centering

## Whitening

## Symmetric FastICA using logcosh approx. to neg-entropy function

## Iteration 1 tol = 0.01930239

## Iteration 2 tol = 0.01303959

## Iteration 3 tol = 0.002393582

## Iteration 4 tol = 0.0006708454

## Iteration 5 tol = 0.0001661602

## Iteration 6 tol = 3.521604e-05
```





They have similar apperance but the axes are different so I would think it's mostly a coincidence.



This is a mirroring of assignment 4.1.

Appendix Code

Assignment 1

```
library("ggplot2")
library("MASS")
# 1.1
setwd("~/TDDE01/lab2")
data <- data.frame(read.csv("australian-crabs.csv"))
ggplot(data = data, mapping = aes(CL, RW, color = sex)) + geom_point()
# Yes, it seems to be two pretty distinct data sets, perhaps the beginning could be tricky
# 1.2
lda.model1 <- lda(sex ~ CL + RW, data = data)
lda.predict1 <- predict(lda.model1, data)
ggplot.0.5 <- ggplot(data = data, mapping = aes(CL, RW, color = lda.predict1$class, shape = sex )) + ge
mcr.0.5 <- mean(lda.predict1$class != data$sex)
# 1.3
lda.model2 <- lda(sex ~ CL + RW, data = data, prior = c(0.1, 0.9))
lda.predict2 <- predict(lda.model2, data)
ggplot.0.9 <- ggplot(data = data, mapping = aes(CL, RW, color = lda.predict2$class, shape = sex )) + ge
mcr.0.9 <- mean(lda.predict2$class != data$sex)
# It got worse since the distribution of the data is 50/50 and not 90/10
#1.4
glm.model <- glm(as.factor(sex) ~ CL + RW, family = binomial, data = data)
glm.predict <- predict(glm.model, data, type = 'response')
glm.predict.0.5 <- ifelse(glm.predict > 0.5, "Male", "Female")
mcr.glm <- mean(glm.predict.0.5 != data$sex)
summary(glm.model)
glm.slope <- coef(glm.model)[2]/(-coef(glm.model)[3])
glm.intercept <- coef(glm.model)[1]/(-coef(glm.model)[3])
ggplot.0.5 + geom_abline(slope = glm.slope, intercept = glm.intercept)
```

Assignment 2

```
library("e1071")
library("MASS")
library("tree")
library("ggplot2")
setwd("~/TDDE01/lab2")
data.credit <- data.frame(read.csv("creditscoring.csv"))
n=dim(data.credit)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data.credit[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=data.credit[id2,]
id3=setdiff(id1,id2)
test=data.credit[id3,]
```

```r
#1.2
tree.fit.dev <- tree(good_bad ~., data = train, split = "deviance")
tree.fit.gini <- tree(good_bad ~., data = train, split = "gini")

pred.dev.train <- predict(tree.fit.dev, newdata = train, type="class")
mean(train$good_bad != pred.dev.train)

pred.dev.test <- predict(tree.fit.dev, newdata = test, type="class")
mean(test$good_bad != pred.dev.test)

pred.gini.train <- predict(tree.fit.gini, newdata = train, type="class")
mean(train$good_bad != pred.gini.train)

pred.gini.test <- predict(tree.fit.gini, newdata = test, type="class")
mean(test$good_bad != pred.gini.test)

#Deviance provides better result based on the diagonal of the confusion matrices


fit=tree(good_bad~., data=train)
set.seed(12345)
max.size <- 15
trainScore=rep(0,max.size)
testScore=rep(0,max.size)
for(i in 2:max.size) {
  prunedTree=prune.tree(fit,best=i)
  pred=predict(prunedTree, newdata=valid, type="tree")
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)
}
plot(2:max.size, trainScore[2:max.size], type="b", col="red", ylim=c(250,600))
points(2:max.size, testScore[2:max.size], type="b", col="blue")

# 4 is very nice thank you sir

finalTree=prune.tree(fit, best=4)
# Depth 3 from plot(finalTree)
Yfit=predict(finalTree, newdata=valid, type = "class")
table(valid$good_bad,Yfit)
mean(valid$good_bad != Yfit)
#Variables actually used in tree construction:
#[1] "savings"  "duration" "history"


fit.bayes=naiveBayes(good_bad~., data=train)
Yfit.bayes.train=predict(fit.bayes, newdata=train)
Yfit.bayes.test=predict(fit.bayes, newdata=test)
table(Yfit.bayes.train, train$good_bad)
table(Yfit.bayes.test, test$good_bad)

# Tree is a bit better, very nice sir

createROCmatrix <- function(pred, pi.vector){
  tpr.vector = numeric()
  fpr.vector = double()

  for (pi.value in pi.vector) {
    predict.pi <- ifelse(pred[,2] > pi.value, "good", "bad")
    TP <- length(which(predict.pi == "good" & test$good_bad == "good"))
    TN <- length(which(predict.pi == "bad" & test$good_bad == "bad"))
    FP <- length(which(predict.pi == "good" & test$good_bad == "bad"))
    FN <- length(which(predict.pi == "bad" & test$good_bad == "good"))

    tpr.vector <- append(tpr.vector, TP/(TP+FN))
    fpr.vector <- append(fpr.vector, FP/(FP+TN))
  }
  return(data.frame(pi.vector, tpr.vector, fpr.vector))
}

Yfit.tree.test=predict(finalTree, newdata=test)
pi.vector <- seq(0.05, 0.95, 0.05)

tree.predict <- predict(finalTree, newdata = test)
bayes.predict <- predict(fit.bayes, newdata = test, type = "raw")

tree.ROC.matrix <- createROCmatrix(tree.predict, pi.vector)
bayes.ROC.matrix <- createROCmatrix(bayes.predict, pi.vector)

ggplot(data = NULL, aes(col = classifier)) +
  geom_point(data = tree.ROC.matrix, aes(x = fpr.vector, y = tpr.vector, col="Tree")) +
  geom_line(data = tree.ROC.matrix, aes(x = fpr.vector, y = tpr.vector, col="Tree")) +
  geom_point(data = bayes.ROC.matrix, aes(x = fpr.vector, y = tpr.vector, col="Bayes")) +
  geom_line(data = bayes.ROC.matrix, aes(x = fpr.vector, y = tpr.vector, col="Bayes"))

bayes.predict.train <- predict(fit.bayes, newdata = train, type = "raw")
loss = matrix(c(0,1,10,0), nrow = 2, ncol = 2)
bayes.loss.predict <-ifelse((bayes.predict[,2]/bayes.predict[,1]) > (loss[2,1]/loss[1,2]), "good", "bad
bayes.loss.predict.train <- ifelse(bayes.predict.train[,2]/bayes.predict.train[,1] > loss_matrix[3]/los
table(test$good_bad, bayes.loss.predict)
table(train$good_bad, bayes.loss.predict.train)
```

Assignment 4

```
library("ggplot2")
library("fastICA")
setwd("~/TDDE01/lab2")
set.seed(12345)
data <- data.frame(read.csv2("NIRSpectra.csv"))
data.features <- subset(data, select = -c(Viscosity))
pca.fit <-prcomp(x = data.features, center = TRUE, scale. = TRUE)
print(pca.fit)
plot(pca.fit, type="l")
# Proportion of variance for PC1 & PC2 = 0.99614
qplot(x=pca.fit$x[,1], y=pca.fit$x[,2], data=pca.fit$scores)
# 2.2
```

```
plot(pca.fit$rotation[,1], main = "Traceplot, PC1")
plot(pca.fit$rotation[,2], main = "Traceplot, PC2")

# 2.3
set.seed(12345)
a <- fastICA(X = data, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1, method = "R", row.norm = FA
W.prim = a$K %*% a$W
plot(W.prim[,1], main = "Traceplot, W.prim 1")
plot(W.prim[,2], main = "Traceplot, W.prim 2")
qplot(x = a$S[,1], y = a$S[,2], data = data.features, xlab = "Latent feature 1", ylab = "Latent feature
```