

# Lab 2

*Rolf Sievert (rolsi701)*

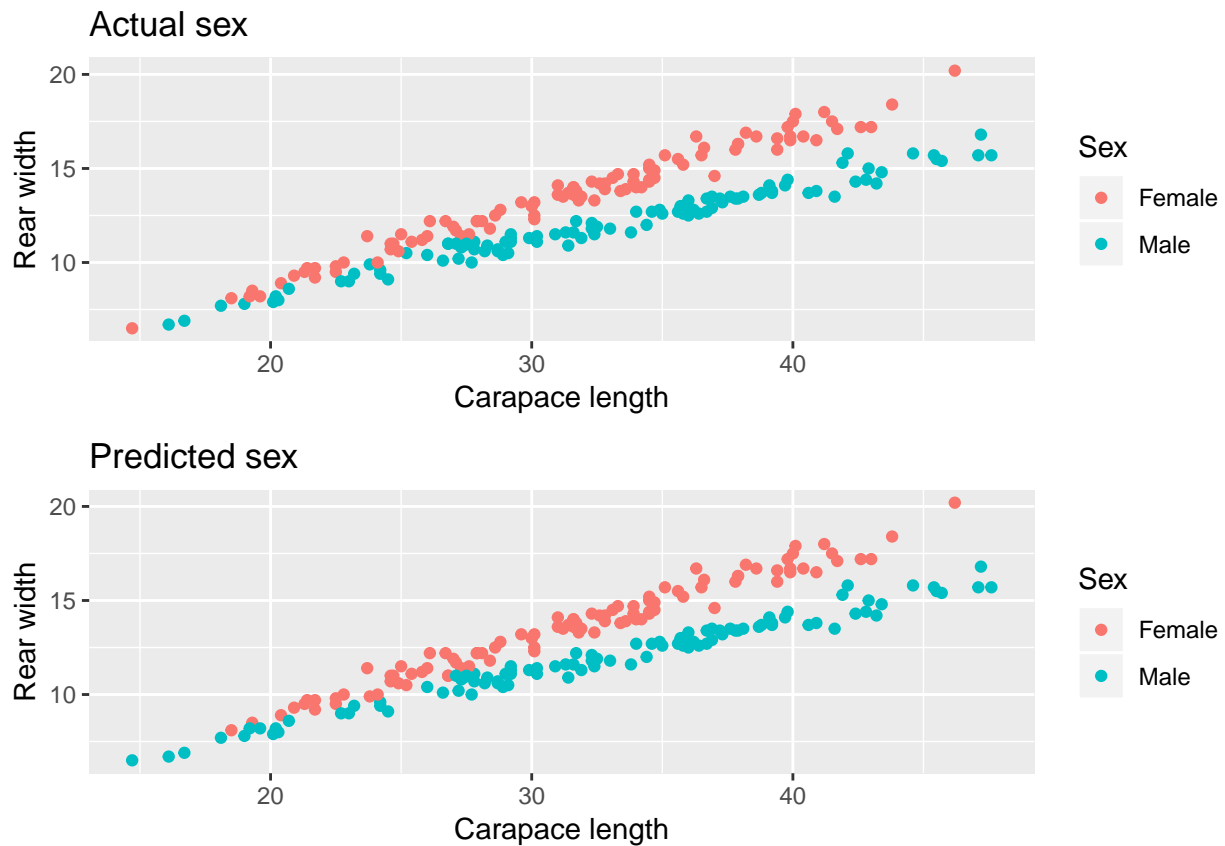
*2018-12-10*

## Contents

|                        |           |
|------------------------|-----------|
| <b>Assignment 1</b>    | <b>2</b>  |
| Step 1 & 2 . . . . .   | 2         |
| Step 3 . . . . .       | 2         |
| Step 4 . . . . .       | 3         |
| <b>Assignment 2</b>    | <b>4</b>  |
| Step 1 . . . . .       | 4         |
| Step 2 . . . . .       | 4         |
| Step 3 . . . . .       | 5         |
| Step 4 . . . . .       | 5         |
| Step 5 . . . . .       | 6         |
| Step 6 . . . . .       | 6         |
| <b>Assignment 4</b>    | <b>7</b>  |
| Step 1 . . . . .       | 7         |
| Step 2 . . . . .       | 9         |
| Step 3 . . . . .       | 10        |
| <b>Appendix</b>        | <b>12</b> |
| Assignment 1 . . . . . | 12        |
| Assignment 2 . . . . . | 14        |
| Assignment 4 . . . . . | 18        |

# Assignment 1

## Step 1 & 2



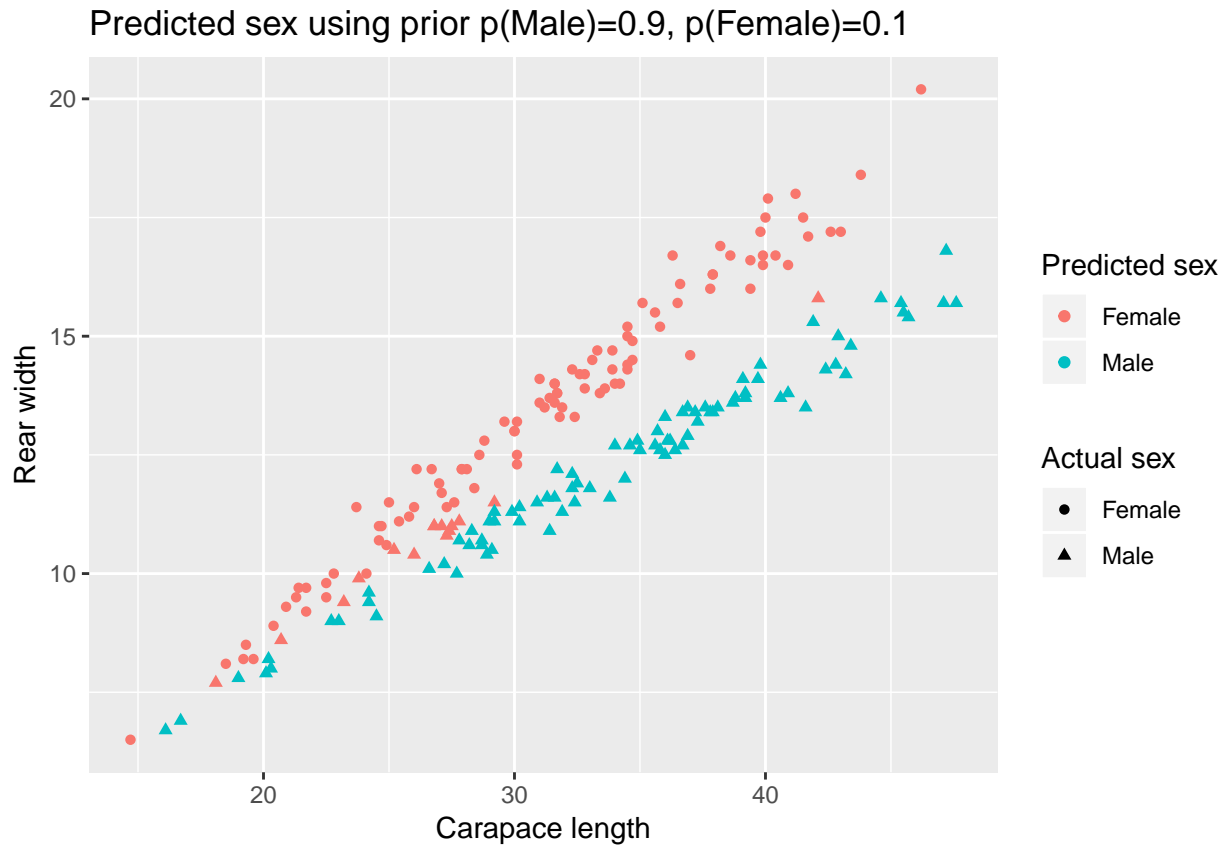
## Misclassification rate for lda: 0.035

Since the sexes seems to follow two separate lines, they are separatable. Therefore it's appropriate to classify this data with a linear discriminant analysis. The only problem is where these two lines intersect and the sexes are impossible to differ. Some data may also be crossing this line and get a false classification.

The misclassification rate is really good, which strengthens the argument that the classification method is appropriate.

## Step 3

This time, the same prediction is made, but with the priors  $p(\text{Male}) = 0.9$  and  $p(\text{Female}) = 0.1$ . See plot below.



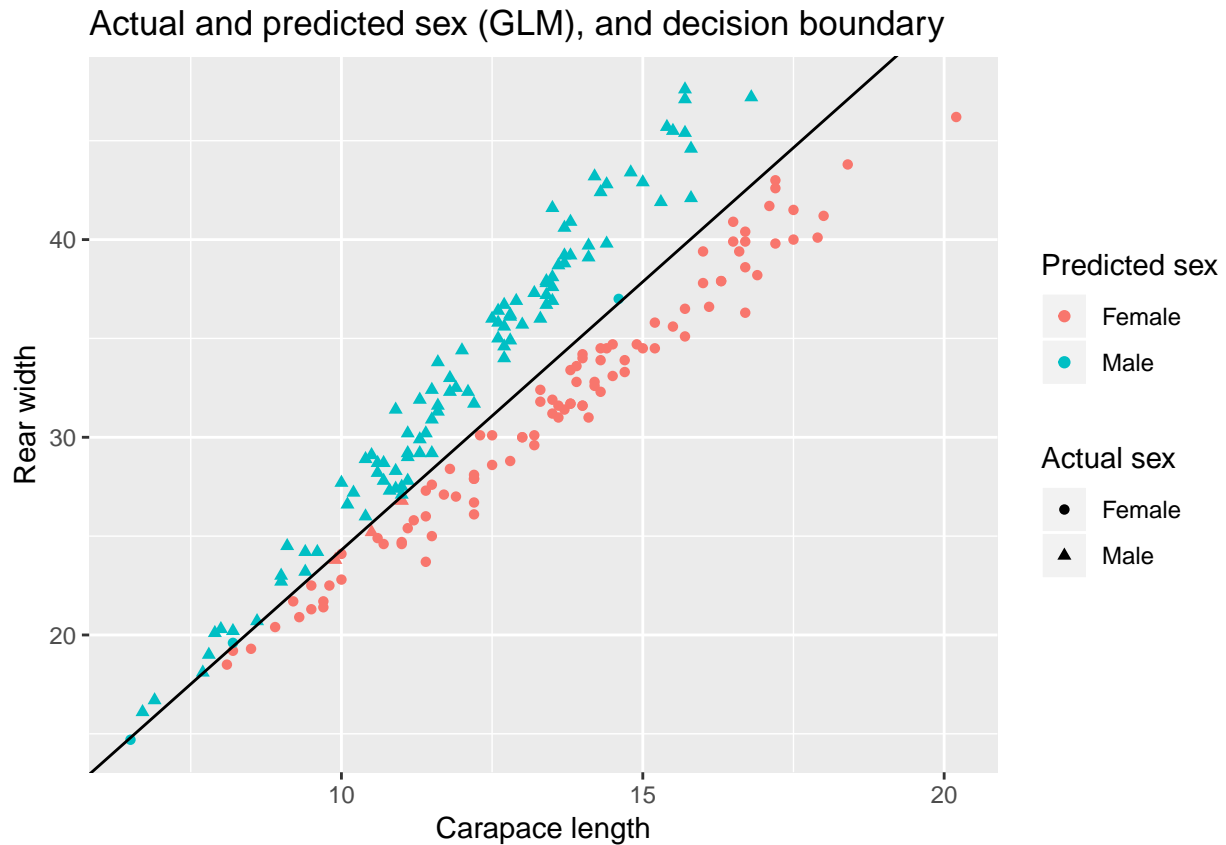
```
## Misclassification rate for lda with p(Male=0.9), p(Female)=0.1: 0.075
```

This time, the misclassification is greater. That is because the priors are not according to the actual distribution of sexes.

#### Step 4

Here the equation of the decision boundary is plotted together with classification using GLM.

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



```
## Misclassification rate for GLM: 0.035
```

The misclassification rate is the same as in step 2.

## Assignment 2

### Step 1

Firstly, the data was separated into training, validation and test data.

### Step 2

```
## Misclassification on deviance with test: 0.268
```

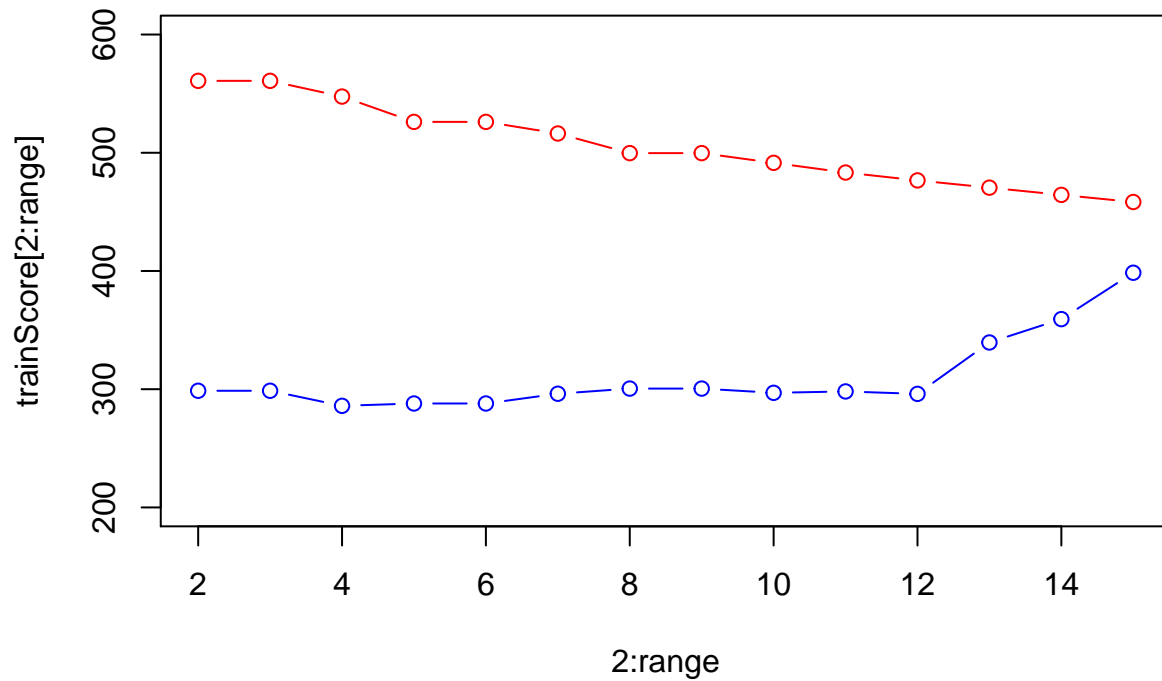
```
## Misclassification on deviance with train: 0.212
```

```
## Misclassification on gini with test: 0.368
```

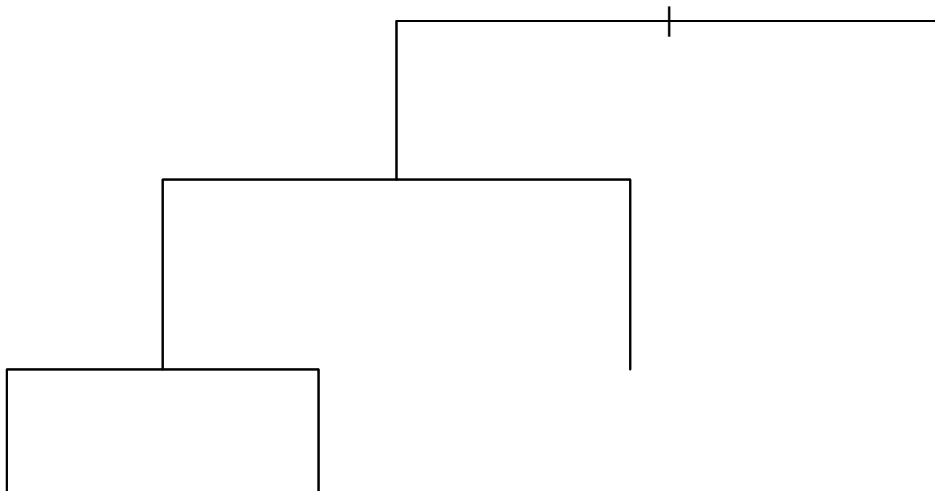
```
## Misclassification on gini with train: 0.24
```

Here, misclassification rate is lower when using deviance as measure of impurity.

### Step 3



```
## Misclassification on optimal tree: 0.256
## Tree depth is 3 as can be seen in the plot.
```



```
## Used variables in optimal tree:
## 'savings' 'duration' 'history'
```

### Step 4

```
## Confusion table of naïve bayes:
##           Actual
## Predicted bad good
##      bad   46   49
##      good  30  125
```

```
## Misclassification with naïve bayes: 0.316
```

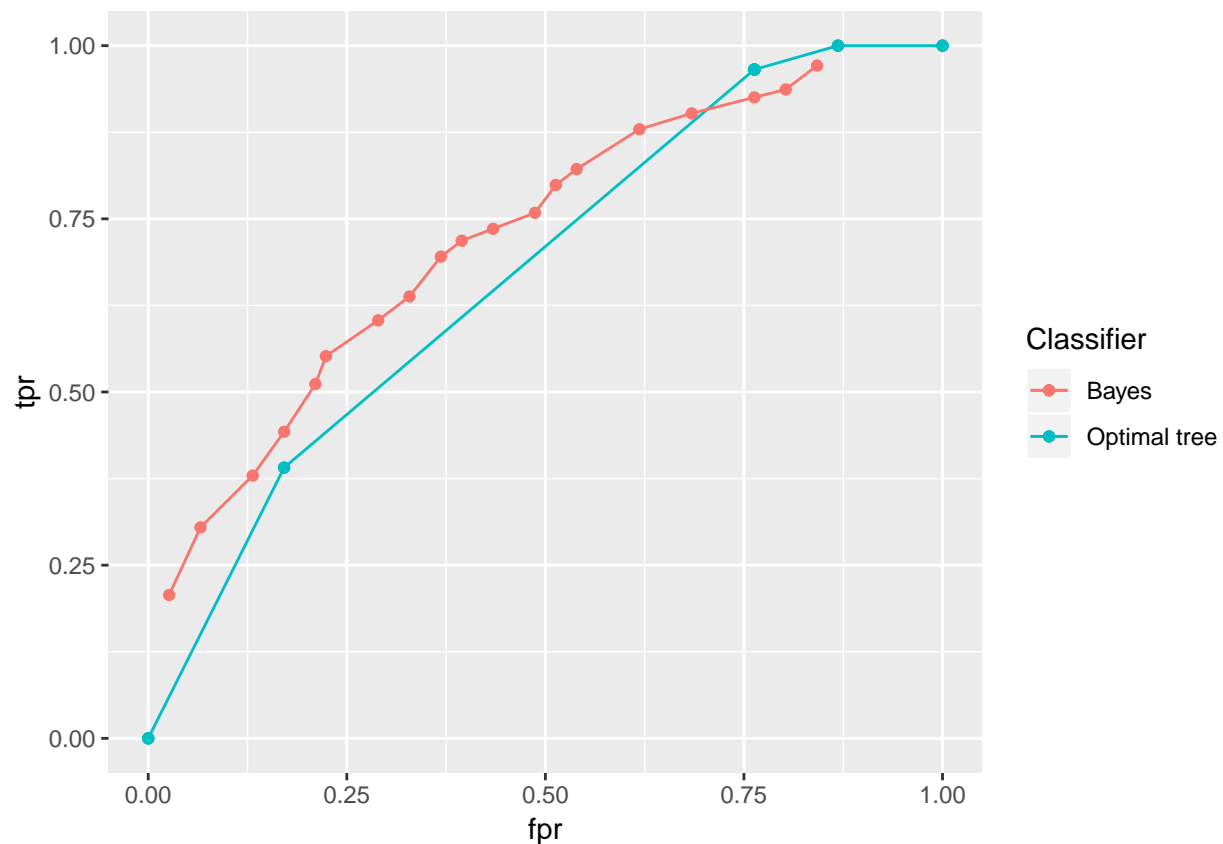
```
## Confusion table of naïve bayes:
```

```
##           Actual
## Predicted bad good
##      bad   95   98
##      good  52  255
```

```
## Misclassification with naïve bayes: 0.3
```

Naïve Bayes has much better result than in step 3.

## Step 5



Naïve Bayes has better ratio between TPR and FPR. The only exception is around  $\pi = 0.75$ , as can be seen in the graph.

## Step 6

Using loss matrix with naïve bayes.

```
## Confusion table of naïve bayes (using test data):
```

```
##           Actual
## Predicted bad good
##      bad   71  122
##      good   5   52
```

```
## Misclassification with naïve bayes (using test data): 0.508
```

```
## Confusion table of naïve bayes (using train data):
```

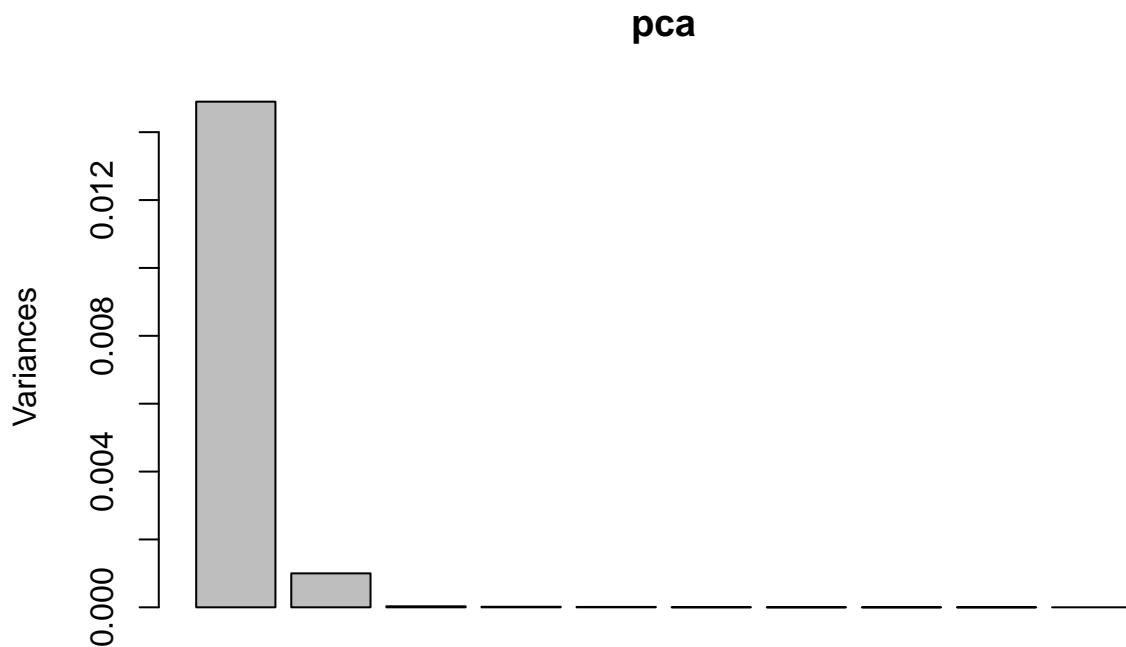
```
##           Actual
## Predicted bad good
##      bad 137 263
##      good  10  90
```

```
## Misclassification with naïve bayes (using train data): 0.546
```

The misclassification is much greater. But the confusion matrix is more favorable from an economic point of view for a company since less are predicted good that are actually bad.

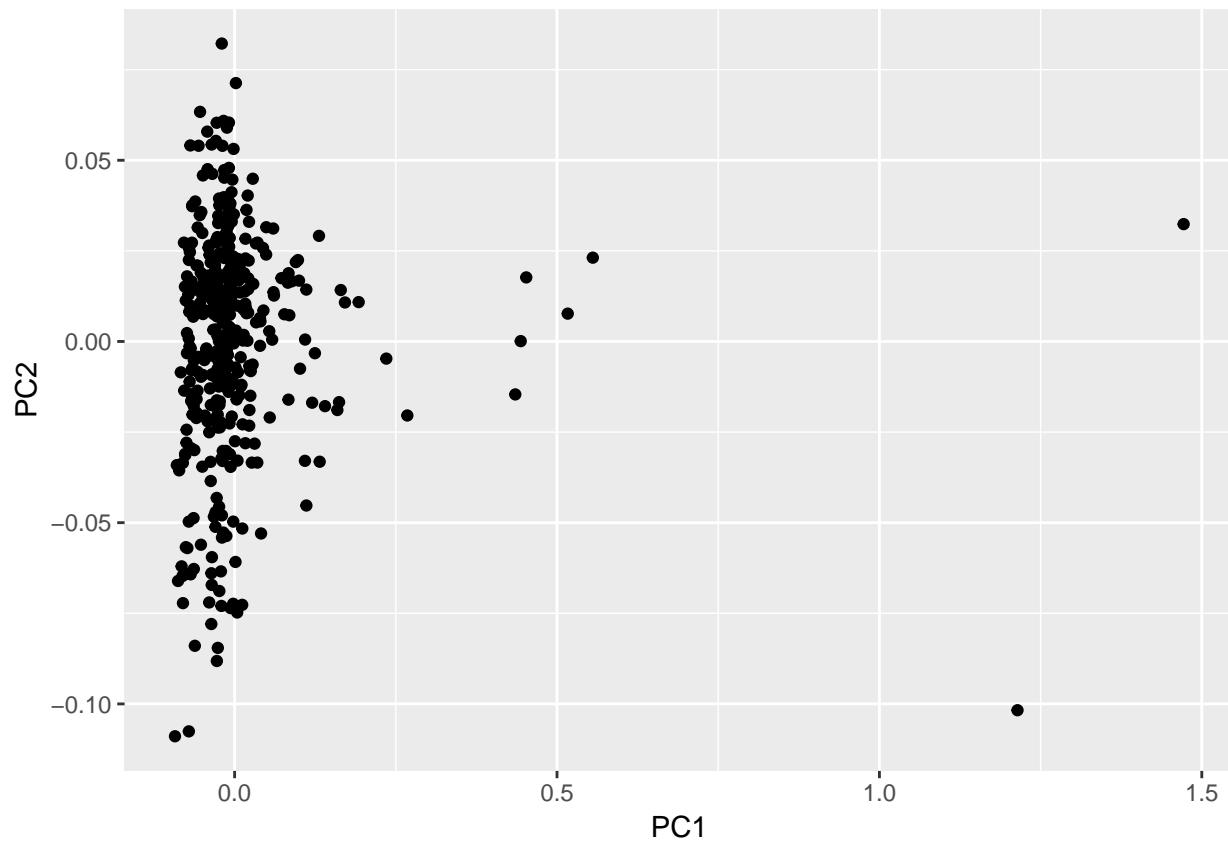
## Assignment 4

### Step 1



```
## [1] "93.332" "6.263" "0.185" "0.101" "0.068" "0.025" "0.009"
## [8] "0.003" "0.003" "0.002" "0.001" "0.001" "0.001" "0.001"
## [15] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## [22] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## [29] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## [36] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## [43] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## [50] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## [57] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## [64] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## [71] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## [78] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## [85] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## [92] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## [99] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
```

```
## [106] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## [113] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## [120] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
## 99% of the variance can be explained by first two PC.
```

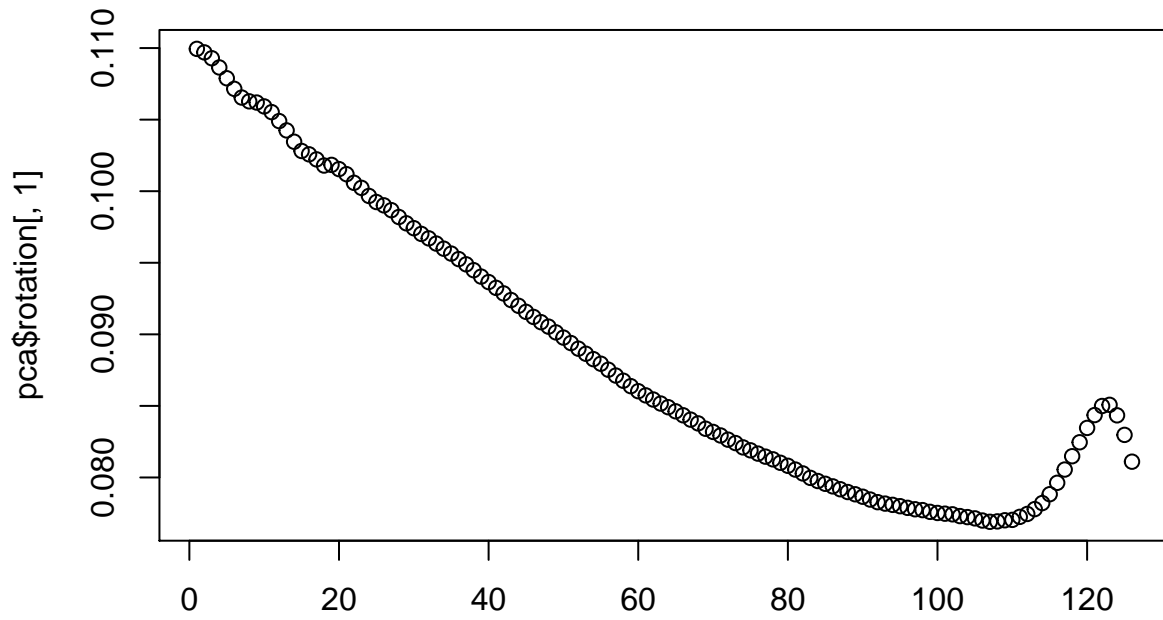


Some diesel fuels depend a lot on PC1, compared to other fuels, I would say that these are unusual.

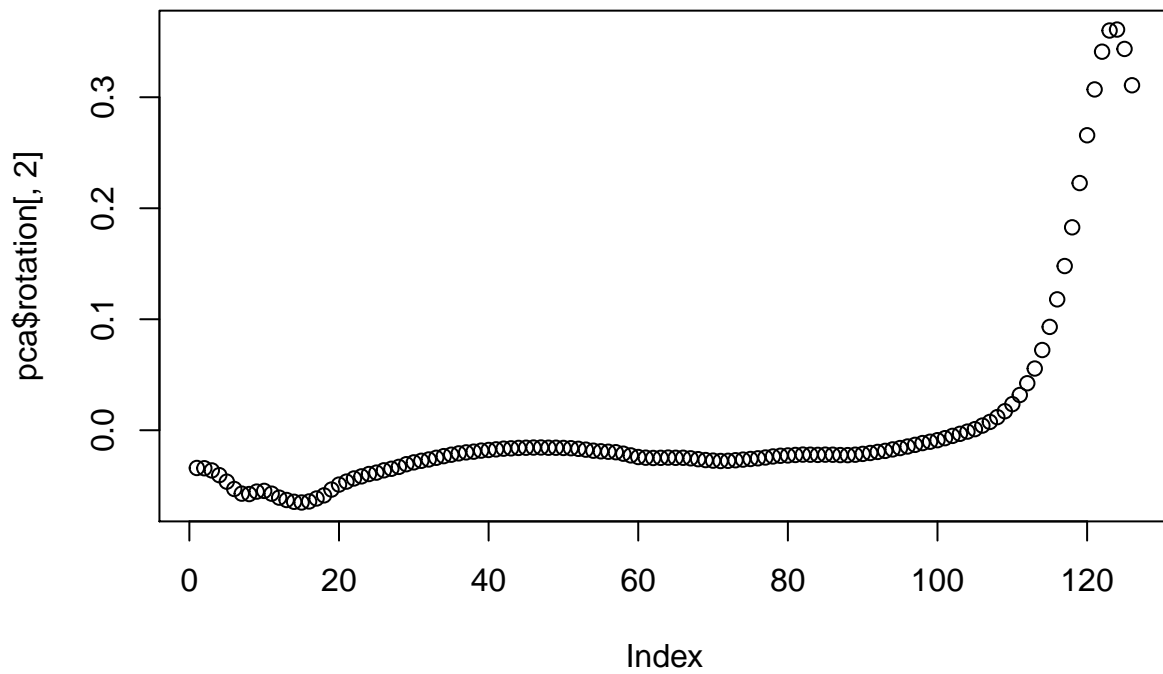


Step 2

**Traceplot, PC1**



**Traceplot, PC2**



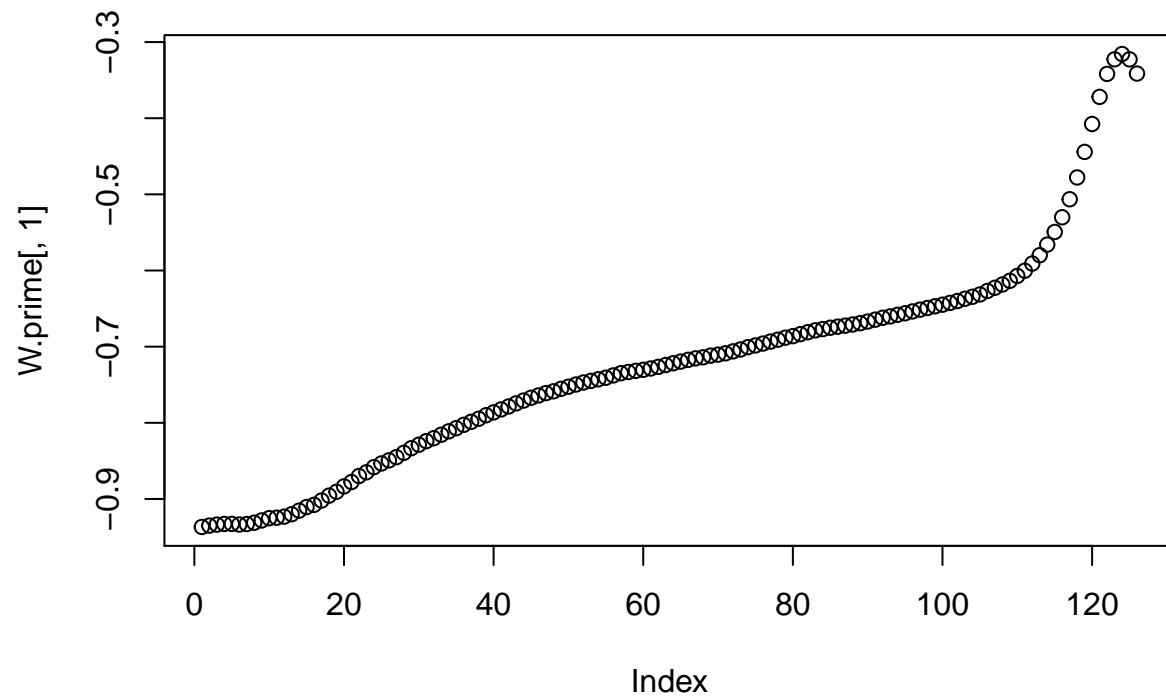
QUESTION??

### Step 3

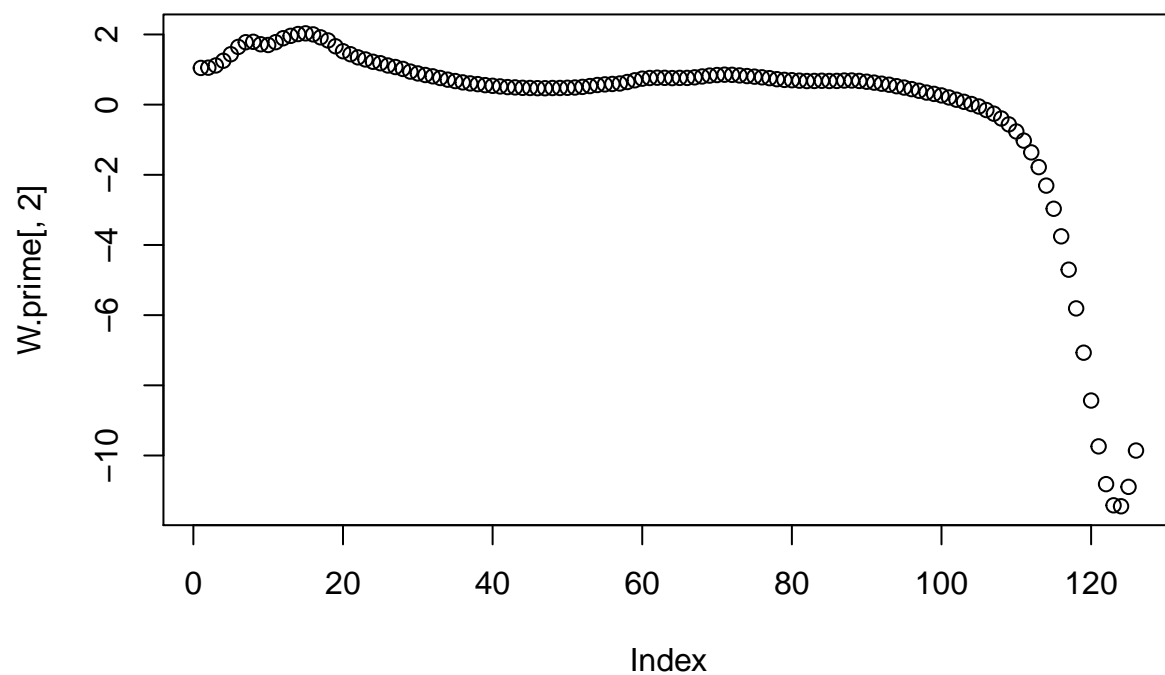
a

```
## 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
```

**Traceplot, PC1**

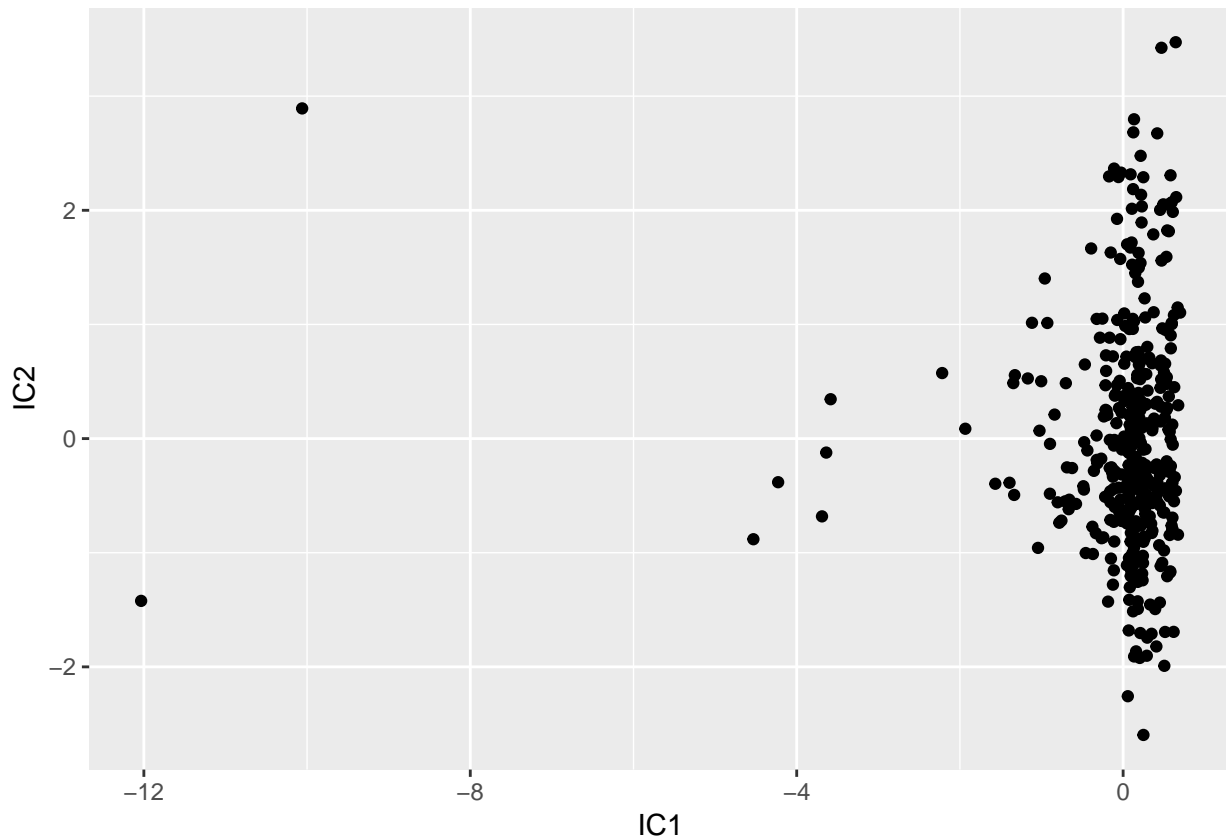


Traceplot, PC2



What kind of measure with W prime?

b



This plot is very similar to the one in step 1, but it is flipped and scaled.

## Appendix

### Assignment 1

```
cat ass1.R

## # Add libraries
## library("gridExtra")
## library("ggplot2")
## library("MASS")
##
## # Set working directory
## setwd("~/courses/tddde01/lab2")
##
## # Read data
## crabs = read.csv("australian-crabs.csv")
##
## crabs.plot = ggplot(data = crabs,
##                      mapping = aes(x=CL,
##                                    y=RW,
##                                    color=sex)) +
```

```

## geom_point() +
## ggtitle("Actual sex") +
## scale_x_continuous(name = "Carapace length") +
## scale_y_continuous(name = "Rear width") +
## scale_color_discrete(name = "Sex")
##
## lda.model = lda(formula = sex~CL+RW, data = crabs)
## lda.predict = predict(lda.model, crabs)
## crabs.predict = ggplot(data = crabs,
##                         mapping = aes(x=CL,
##                                       y=RW,
##                                       color=lda.predict$class)) +
## geom_point() +
## ggtitle("Predicted sex") +
## scale_x_continuous(name = "Carapace length") +
## scale_y_continuous(name = "Rear width") +
## scale_color_discrete(name = "Sex")
##
## grid.arrange(crabs.plot, crabs.predict)
##
## # Calculate misclassification
## crabs.misclass = mean(crabs$sex != lda.predict$class)
## cat("Misclassification rate for lda: ", crabs.misclass)
##
## # Step 3
## prior.male = 0.9
## prior.female = 0.1
## crabs.predict.prior = predict(lda.model,
##                               crabs,
##                               prior=c(Male=prior.male,
##                                       Female=prior.female)
##                               )
## crabs.plot.prior = ggplot(data=crabs,
##                           mapping=aes(x=CL,
##                                       y=RW,
##                                       shape=sex,
##                                       color=crabs.predict.prior$class)) +
## geom_point() +
## ggtitle("Predicted sex using prior p(Male)=0.9, p(Female)=0.1") +
## scale_x_continuous(name = "Carapace length") +
## scale_y_continuous(name = "Rear width") +
## scale_color_discrete(name = "Predicted sex") +
## scale_shape_discrete(name = "Actual sex")
## # Plot
## grid.arrange(crabs.plot.prior)
## # Calculate missclassification
## crabs.prior.misclass = mean(crabs$sex != crabs.predict.prior$class)
## cat("Misclassification rate for lda with p(Male=0.9), p(Female)=0.1: ",
##     crabs.prior.misclass)
##
## # Step 4
## glm.model = glm(formula = sex~RW+CL,
##                  data = crabs,
##                  family = 'binomial')

```

```

## glm.predict = predict(object = glm.model,
##                        newdata = crabs,
##                        type = 'response')
## glm.predicted.sex = ifelse(glm.predict > 0.5, "Male", "Female")
##
## # Calculate line from model
## intercept = coef(glm.model)[1]
## rw = coef(glm.model)[2]
## cl = coef(glm.model)[3]
## border = 0.5
## k = -rw/cl
## m = -(intercept - border)/cl
## # Print plot and line
## glm.plot = ggplot(data = crabs,
##                   mapping = aes(x=RW,
##                                 y=CL,
##                                 color=glm.predicted.sex,
##                                 shape=crabs$sex)) +
## geom_point() +
## ggtitle("Actual and predicted sex (GLM), and decision boundary") +
## scale_x_continuous(name = "Carapace length") +
## scale_y_continuous(name = "Rear width") +
## scale_color_discrete(name = "Predicted sex") +
## scale_shape_discrete(name = "Actual sex") +
## geom_abline(intercept = m, slope = k)
## grid.arrange(glm.plot)
##
## # Print misclassification
## glm.misclass = mean(crabs$sex != glm.predicted.sex)
## cat("Misclassification rate for GLM: ",
##     glm.misclass)

```

## Assignment 2

```
cat ass2.R
```

```

## # Step 1
## # Add libraries
## library("gridExtra")
## library("ggplot2")
## library("tree")
## library("MASS")
## library("e1071")
##
## # Set working directory
## setwd("~/courses/tdde01/lab2")
##
## # Read data
## scores = read.csv2("creditscoring.csv")
## # Read as strings
## scores$good_bad = as.factor(scores$good_bad)
##
## # Split data into train/val/test

```

```

## n=dim(scores)[1]
## set.seed(12345)
## id=sample(1:n, floor(n*0.5))
## train=scores[id,]
## id1=setdiff(1:n, id)
## set.seed(12345)
## id2=sample(id1, floor(n*0.25))
## val=scores[id2,]
## id3=setdiff(id1,id2)
## test=scores[id3,]
##
## # Step 2
## # Create tree
## tree.deviance = tree(formula = good_bad~.,
##                       data = train,
##                       split="deviance")
## tree.gini = tree(formula = good_bad~.,
##                  data = train,
##                  split="gini")
##
## # Make predictions
## predict.deviance.test = predict(tree.deviance, test, type = "class")
## predict.deviance.train = predict(tree.deviance, train, type = "class")
## predict.gini.test = predict(tree.gini, test, type = "class")
## predict.gini.train = predict(tree.gini, train, type = "class")
##
## # Calculate misclassification rates
## misclass = function(predicted, true) {
##   return(mean(predicted != true))
## }
##
## deviance.test.misclass = misclass(predict.deviance.test, test$good_bad)
## deviance.train.misclass = misclass(predict.deviance.train, train$good_bad)
## gini.test.misclass = misclass(predict.gini.test, test$good_bad)
## gini.train.misclass = misclass(predict.gini.train, train$good_bad)
##
## # Print misclassification rates
## cat("Misclassification on deviance with test: ",
##     deviance.test.misclass)
## cat("\nMisclassification on deviance with train: ",
##     deviance.train.misclass)
## cat("\nMisclassification on gini with test: ",
##     gini.test.misclass)
## cat("\nMisclassification on gini with train: ",
##     gini.train.misclass, "\n")
##
## # Step 3
## # Lists of scores
## range = 15
## pruned=rep(0, range)
## trainScore=rep(0, range)
## testScore=rep(0, range)
##
## test.tree = tree(formula=good_bad~., data=train)

```

```

## for(i in 2:range) {
##   # Prune the tree
##   prunedTree=prune.tree(test.tree, best=i)
##   # Make trediction on validation data
##   pred=predict(prunedTree, newdata=val, type="tree")
##   # Append scores
##   trainScore[i]=deviance(prunedTree)
##   testScore[i]=deviance(pred)
## }
##
## # Plot the scores
## plot(2:range, trainScore[2:range], type="b", col="red", ylim=c(200, 600))
## points(2:range, testScore[2:range], type="b", col="blue")
##
## optLeaves = 4
##
## # Info on optimal tree
## optTree = prune.tree(tree.deviance, best=optLeaves)
## summary(optTree)
## optTree.predict = predict(optTree, newdata=test)
## optTree.predict.string = ifelse(optTree.predict[2] > 0.5, "good", "bad")
## optTree.misclass = misclass(optTree.predict.string, test$good_bad)
## cat("\nMisclassification on optimal tree: ",
##     optTree.misclass)
##
## cat("\nTree depth is 5 as can be seen in the plot. \n")
## plot(optTree)
## cat("\n\nUsed variables in optimal tree: \n'savings' 'duration' 'history' 'age' 'purpose'\n\n")
##
## # Step 4
## # Create model and classify
## model.bayes = naiveBayes(formula = good_bad~., data=train)
## predict.bayes.test = predict(model.bayes, newdata=test, type="class")
## predict.bayes.train = predict(model.bayes, newdata=train, type="class")
##
## # Print info on classification with prediction on test
## table.bayes = table(predict.bayes.test, test$good_bad)
## cat("Confusion table of naïve bayes:")
## print(table.bayes)
##
## misclass.bayes = mean(predict.bayes.test != test$good_bad)
## cat("Misclassification with naïve bayes: ", misclass.bayes, "\n\n")
##
## # Print info on classification with prediction on test
## table.bayes = table(predict.bayes.train, train$good_bad)
## cat("Confusion table of naïve bayes:")
## print(table.bayes)
##
## misclass.bayes = mean(predict.bayes.train != train$good_bad)
## cat("Misclassification with naïve bayes: ", misclass.bayes, "\n\n")
##
## # Step 5
## # Calculate TPR and FPR of optimal tree and bayes
## getROC = function(pred, pi) {

```



```

##     tpr = c()
##     fpr = c()
##     for (p in pi) {
##         # Change probabilities to strings
##         tmp = ifelse(pred[, 'good'] > p, "good", "bad")
##         # Get confusion matrix
##         cm = table(predicted=tmp, actual=test$good_bad)
##         if('good' %in% rownames(cm)) {
##             # Calculate TPR, first dim of cm is predicted
##             t = cm['good', 'good'] / sum(cm[, 'good'])
##             # Calculate FPR
##             f = cm['good', 'bad'] / sum(cm[, 'bad'])
##             # Append to list of values
##             tpr = c(tpr, ifelse(is.finite(t), t, 0))
##             fpr = c(fpr, ifelse(is.finite(f), f, 0))
##         } else {
##             tpr = c(tpr, 0)
##             fpr = c(fpr, 0)
##         }
##     }
##     df = data.frame(tpr, fpr)
##     return(df)
## }
##
## pi = seq(0.05, 0.95, 0.05)
## pred.optTree = predict(optTree, newdata=test)
## pred.bayes = predict(model.bayes, newdata=test, type='raw')
## opt.roc = getROC(pred.optTree, pi)
## bayes.roc = getROC(pred.bayes, pi)
##
## #Plot
## roc.plot = ggplot(mapping=aes(col=Classifier)) +
##     geom_point(data=opt.roc, aes(x=fpr, y=tpr, col="Optimal tree")) +
##     geom_line(data=opt.roc, aes(x=fpr, y=tpr, col="Optimal tree")) +
##     geom_point(data=bayes.roc, aes(x=fpr, y=tpr, col="Bayes")) +
##     geom_line(data=bayes.roc, aes(x=fpr, y=tpr, col="Bayes"))
##
## grid.arrange(roc.plot)
##
## # Step 6
## # Create model and classify
## model.bayes = naiveBayes(formula = good_bad~., data=train)
## predict.bayes.test = predict(model.bayes, newdata=test, type="raw")
## predict.bayes.train = predict(model.bayes, newdata=train, type="raw")
##
## # Print info on classification with prediction on test
## loss = 10/1
## predict.bayes.test.loss = ifelse(predict.bayes.test[, 'good'] /
##     predict.bayes.test[, 'bad'] > loss,
##     'good',
##     'bad')
##
## table.bayes = table(Predict=predict.bayes.test.loss, Actual=test$good_bad)
## cat("Confusion table of naïve bayes (using test data):")
## print(table.bayes)

```

```
##
## misclass.bayes = mean(predict.bayes.test.loss != test$good_bad)
## cat("Misclassification with naïve bayes (using test data): ", misclass.bayes, "\n")
##
## # Print info on classification with prediction on train
## predict.bayes.train.loss = ifelse(predict.bayes.train[, 'good'] /
##                                predict.bayes.train[, 'bad'] > loss,
##                                'good',
##                                'bad')
## table.bayes = table(Predict=predict.bayes.train.loss, Actual=train$good_bad)
## cat("Confusion table of naïve bayes (using train data): ")
## print(table.bayes)
##
## misclass.bayes = mean(predict.bayes.train.loss != train$good_bad)
## cat("Misclassification with naïve bayes (using train data): ", misclass.bayes, "\n")
```

## Assignment 4

```
cat ass4.R
```

```
## # Step 1
## # Add libraries
## library("gridExtra")
## library("ggplot2")
## library("MASS")
## library("stats")
## library("fastICA")
##
## # Set working directory
## setwd("~/courses/tddde01/lab2")
##
## # Read data
## spectra = data.frame(read.csv2("NIRSpectra.csv"))
## spectra = subset(spectra, select=-Viscosity)
##
## # Calculate principal components
## pca = prcomp(spectra)
##
## # Get eigen values
## lambda = pca$sdev^2
## # Plot variance of each component
## screeplot(pca)
##
## # Print proportion of variation
## sprintf("%.2.3f", lambda/sum(lambda)*100)
## cat("99% of the variance can be explained by first two PC.\n\n")
##
## # Extract principal components
## PC1 = pca$x[,1]
## PC2 = pca$x[,2]
##
## # Plot the data projected onto PC 1 and 2
## plot.projection = ggplot() +
```

```

##      geom_point(data=spectra, mapping=aes(x=PC1, y=PC2))
## grid.arrange(plot.projection)
##
## # Step 2
## plot(pca$rotation[,1], main="Traceplot, PC1")
## plot(pca$rotation[,2], main="Traceplot, PC2")
##
## # Step 3a
## set.seed(12345)
## res = fastICA(X=spectra,
##               n.comp=2,
##               alg.typ= "parallel",
##               fun = "logcosh",
##               alpha = 1,
##               method = "R",
##               row.norm = FALSE,
##               maxit= 200,
##               tol = 0.0001,
##               verbose = TRUE)
##
## W.prime = res$K %*% res$W
## # Traceplots of fastICA
## plot(W.prime[,1], main="Traceplot, PC1")
## plot(W.prime[,2], main="Traceplot, PC2")
##
## # Step 3b
## # Extract ICs
## IC1 = res$S[,1]
## IC2 = res$S[,2]
##
## # Plot the data projected onto PC 1 and 2
## plot.projection = ggplot() +
##   geom_point(data=spectra, mapping=aes(x=IC1, y=IC2))
## grid.arrange(plot.projection)

```