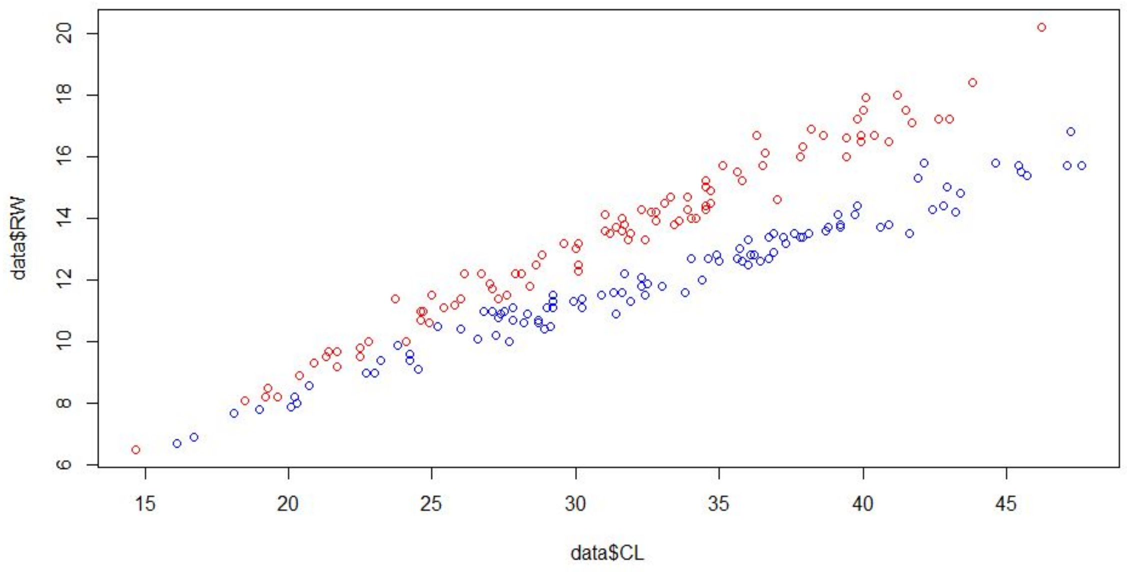
Oskar Hidén- oskhi827 Samuel Persson- sampe028 Oscar Moberg- oscmo462

Assignment 1 - Oscar

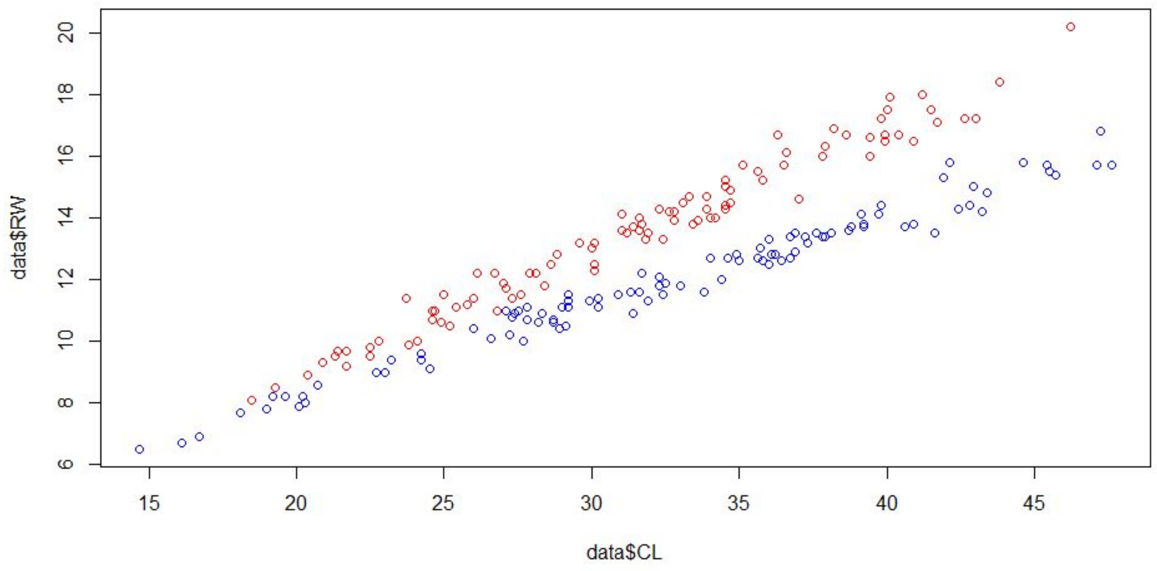
Step 1

Scatterplot of the data provided in this exercise(Male = blue dots, Female = red dots):



Comment: Yes we think LDA could be used to create a good model, the data sets are clearly divided into two groups, however, to the left of the plot the groups are not that clearly divided and that could be a problem using a linear classifier. We think that a linear classifier could be a good choice but we have to try it and look at the result to be sure!

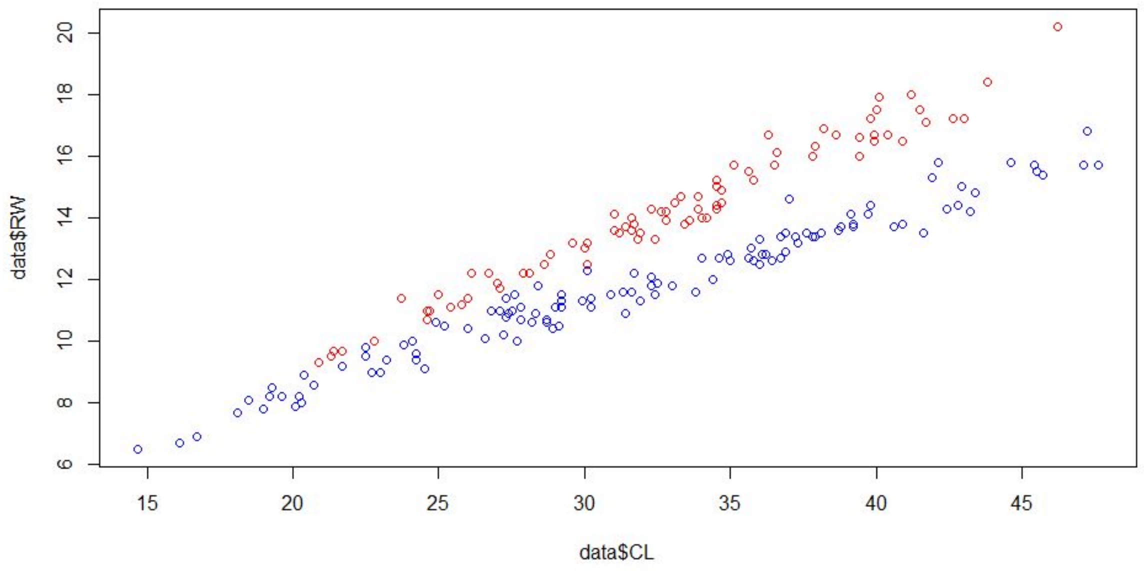
Created a model using LDA. The result can be seen below, as before the blue dots = Males and red dot = Females.



Comment: As you can see there are some errors in this model (7 points). This gives us a misclassification rate of 0.035 which is a very good result.

Step 2

Same as before but with the use of priors: p(Male) = 0.9 and thus p(Female) 0.1:

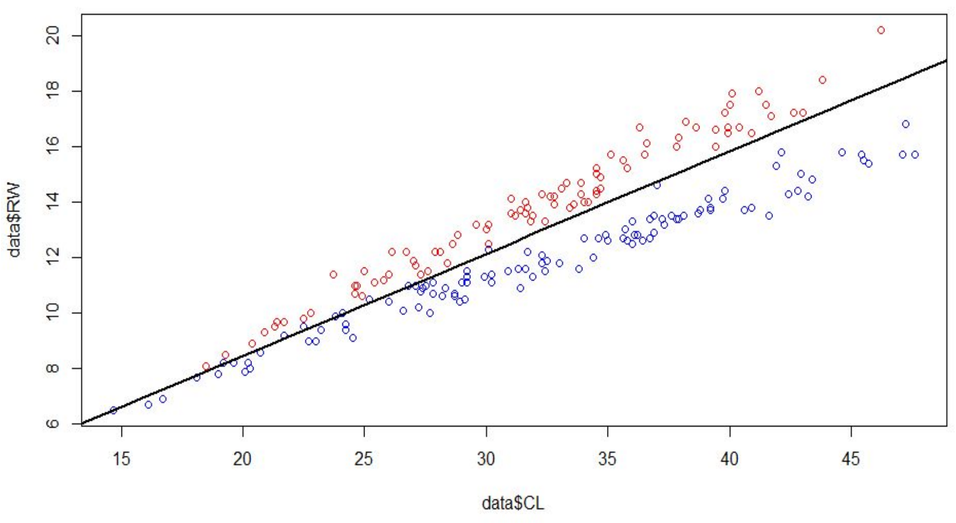


The classification changed to that alot more males were predicted from the data which is obvious considering that we told our model that the probability of Male was higher, 90%.

This time the misclassification rate were 0.08 which is worse than our last model. Clearly the priors had a part in how the data got classified since our model didn't classify any female crabs as a male this time.

Step 3

Logistic regression:
When trying to classify the data using a logistic regression we got the following result.



This time we got a misclassification rate of 0.035 which is very good, in fact the same result as the LDA model. This is due to the fact that if you use LDA for only 2 classes you divide the classes by a decision boundary which, with 2 classes becomes a straight line (same as linear regression)

Equation of the decision boundary:
Y = k*x(transpose) + m
m = coef(GLMmodel)[1]/(-coef(GLMmodel)[3]) = 1.08379
k = coef(GLMmodel)[2]/(-coef(GLMmodel)[3]) = 0.3685758
where:

- GLMmodel[1] is the intercept. (= 13.617)
- GLMmodel[2] is the CL Coefficient. (=4.631)
- GLMmodel[3] is the RW Coefficient. (=-12.564)

Oskar Hidén- oskhi827 Samuel Persson- sampe028 Oscar Moberg- oscmo462

Assignment 2 - Samuel

Step 2

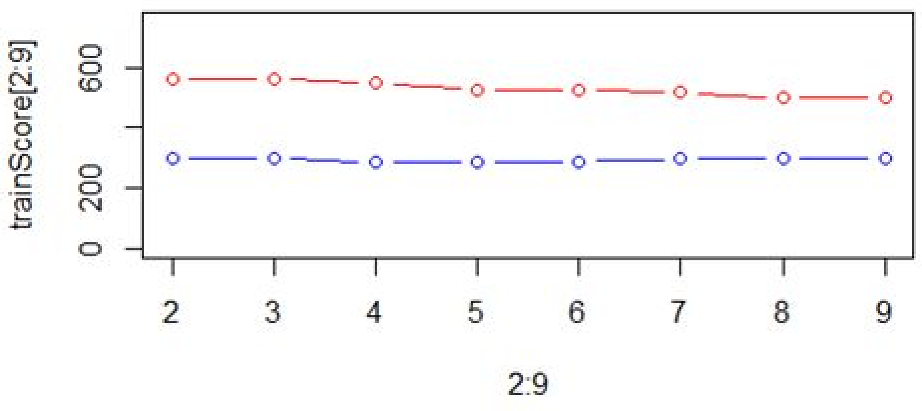
Misclassification rates:

	Train	Test
Deviance	0.212	0.268
Gini index	0.24	0.368

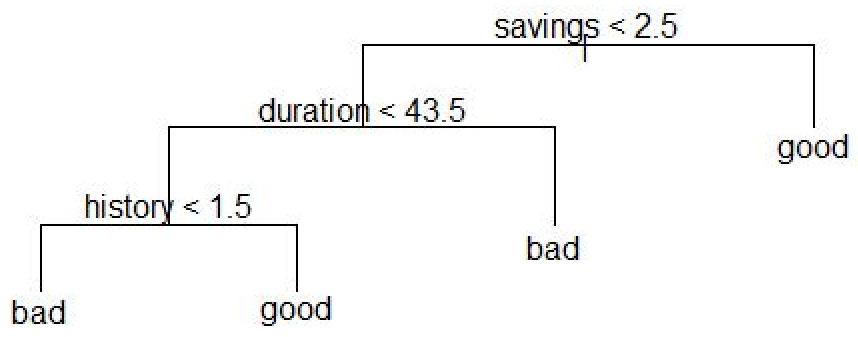
We chose Deviance as measurement because it performs better.

Step 3

The red line is the training data and the blue line the validation data and the x-axis represents the number of leafs.



The optimal depth of the tree is 3 with 4 leaves. See variables in picture below.



If savings are high enough the customers often handle their loans good. Otherwise it firstly depends on the duration and then the history.

The misclassification rate for the test data is 0.256.

Step 4

Here are the confusion matrices and the misclassification rates:

Confusion matrix train		Prediction	
		Bad	Good
Real Values	Bad	95	52
	Good	98	255

Misclassification rate train: 0.3

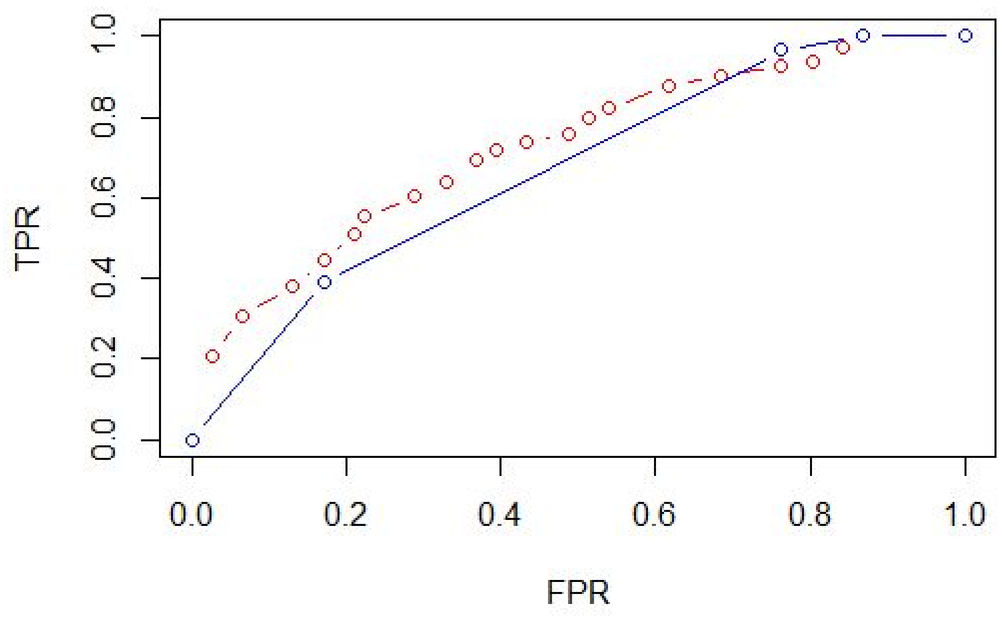
Confusion matrix test		Prediction	
		Bad	Good
Real Values	Bad	46	30
	Good	49	125

Misclassification rate test: 0.316

We see that the misclassification rate for test is higher when we use naïve Bayes model than when we use the decision tree model. In Naïve Bayes all variables are used to predict data. When we previously used all variables in our decision tree it generated a misclassification rate for test data of 0.268. Which is closer to the misclassification rate for Naïve Bayes.

Step 5

In the picture below the red line is the naïve Bayesian model and the blue line is the “optimal” decision tree found earlier. 1 represents good and 0 represents bad.



We can see that the integral of the naïve Bayesian model is higher which should make it in general a better model. However when our classifier classifies a lot of the data as good the decision tree could be better.

The confusion matrices follow:

Confusion matrix train		Prediction	
		Bad	Good
Real Values	Bad	137	10
	Good	263	90

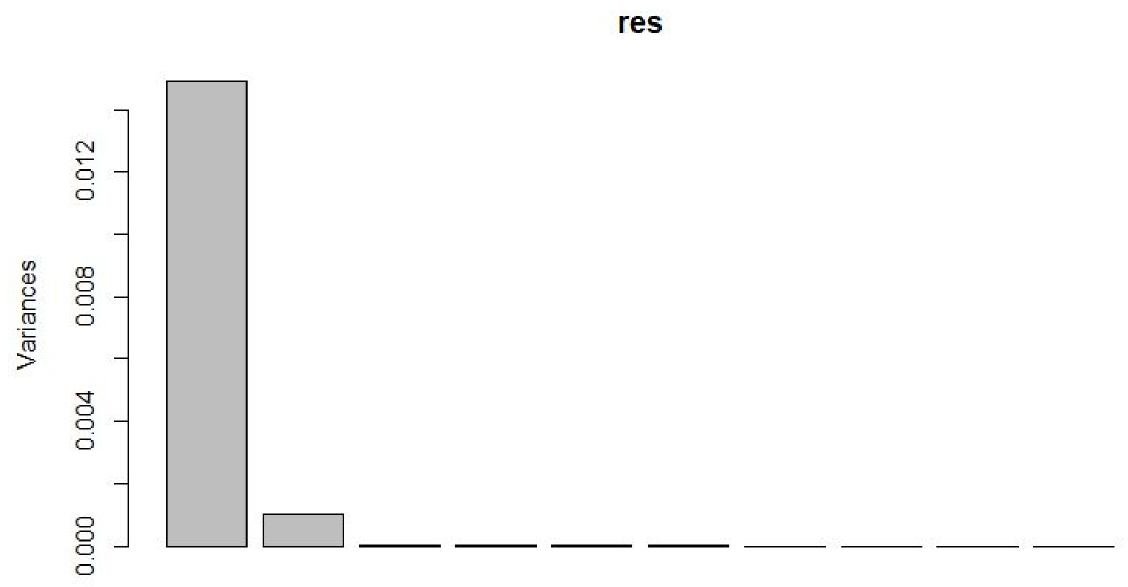
Confusion matrix test		Prediction	
		Bad	Good
Real Values	Bad	71	5
	Good	122	52

We can see that we predict a lot more as bad because we penalize harder when we wrongly predict an enterprise as good. This is because if we predict something wrong here it's penalized 10 times more.

Assignment 4 - Oskar

Step 1

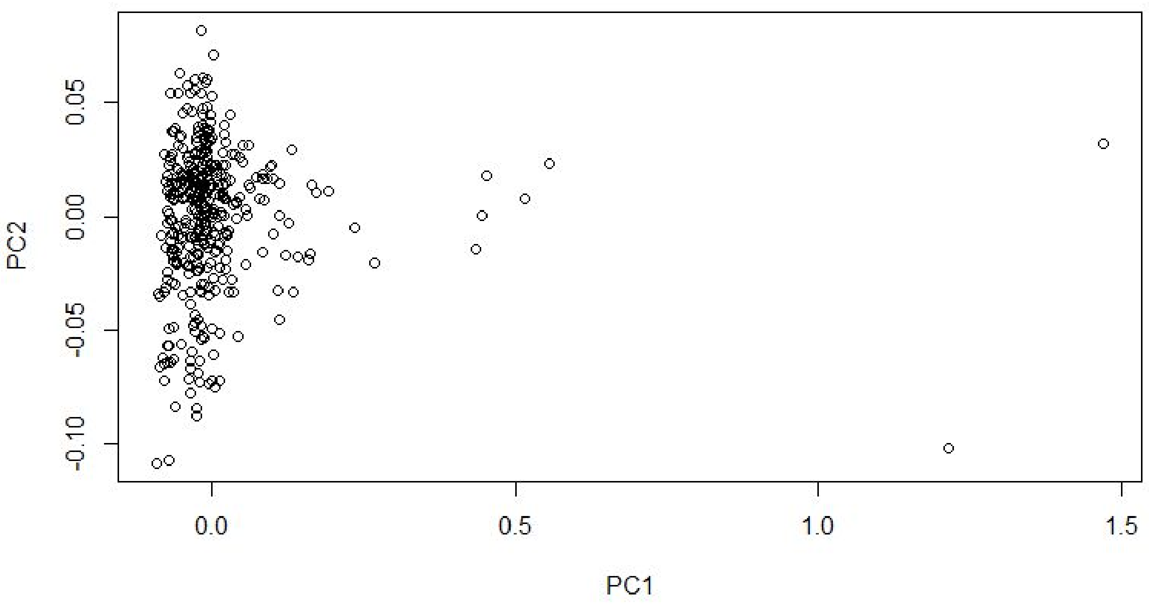
Explained variance from each component:



To get above 99% variance explanation we look at how much variance each component explains:

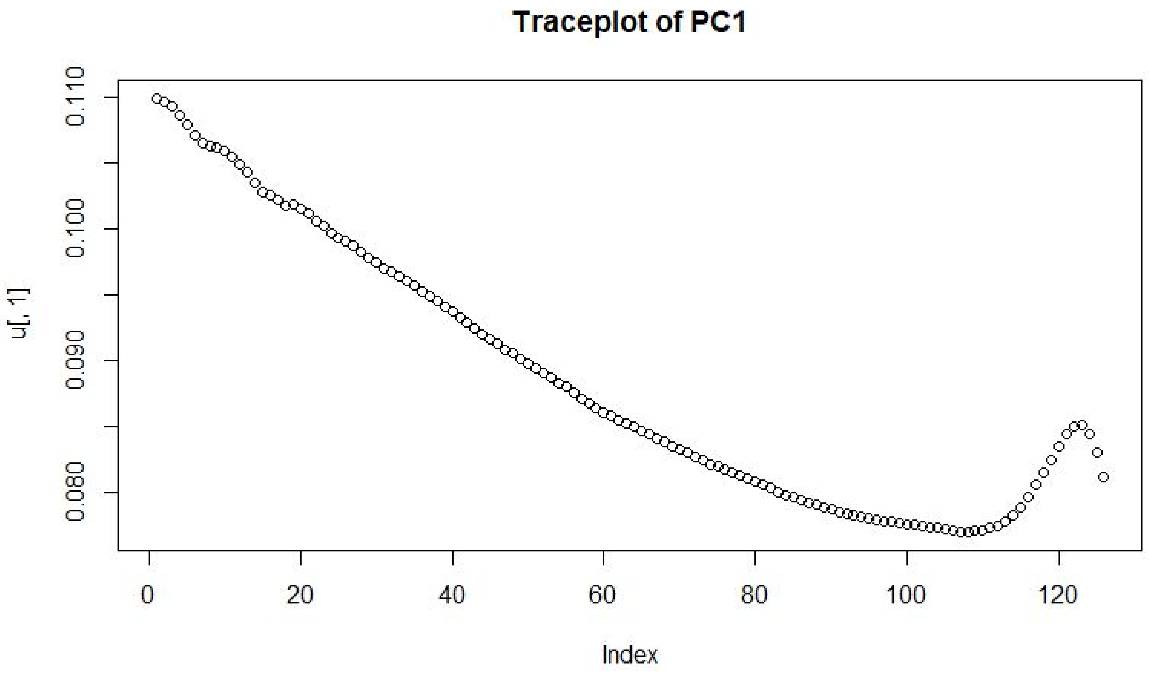
- PC1: 93.332 %
- PC2: 6.263 %
- PC3: 0.185 %
- PC4: 0.101 %
- …
- PC1 and PC2 is chosen to get above 99% explanation.

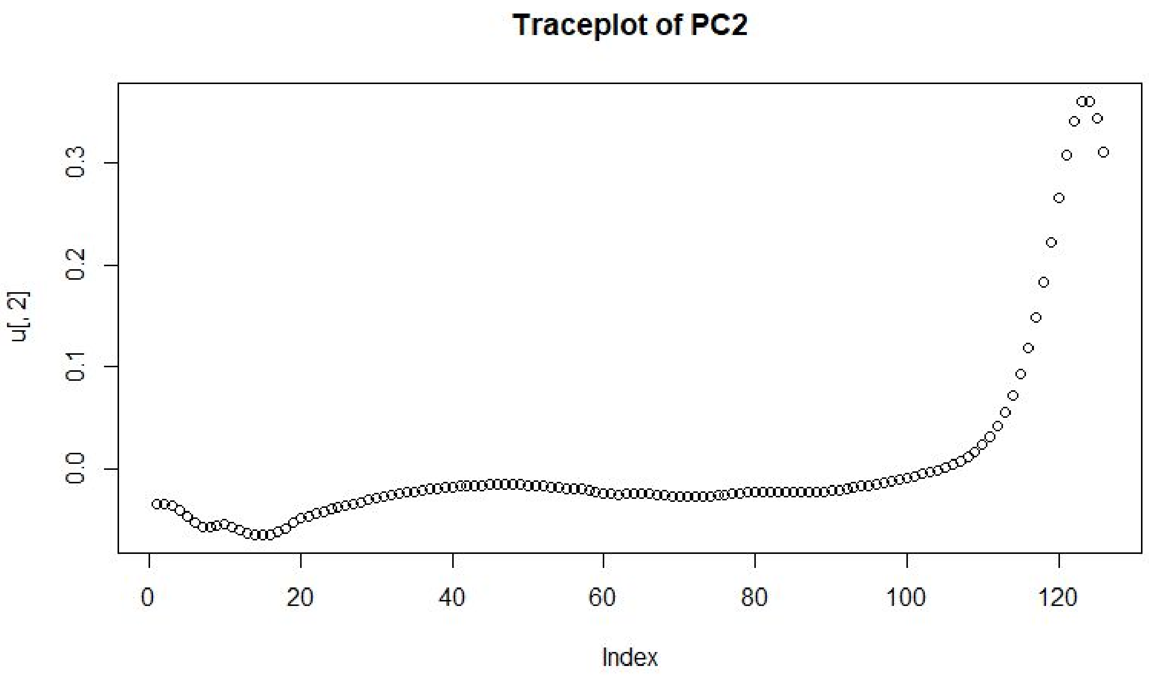
Scores in coordinates (PC1, PC2).



There are some unusual Diesel fuels according to this plot. Two outliers to the far right and some values that deviates from the rest of the data.

Step 2

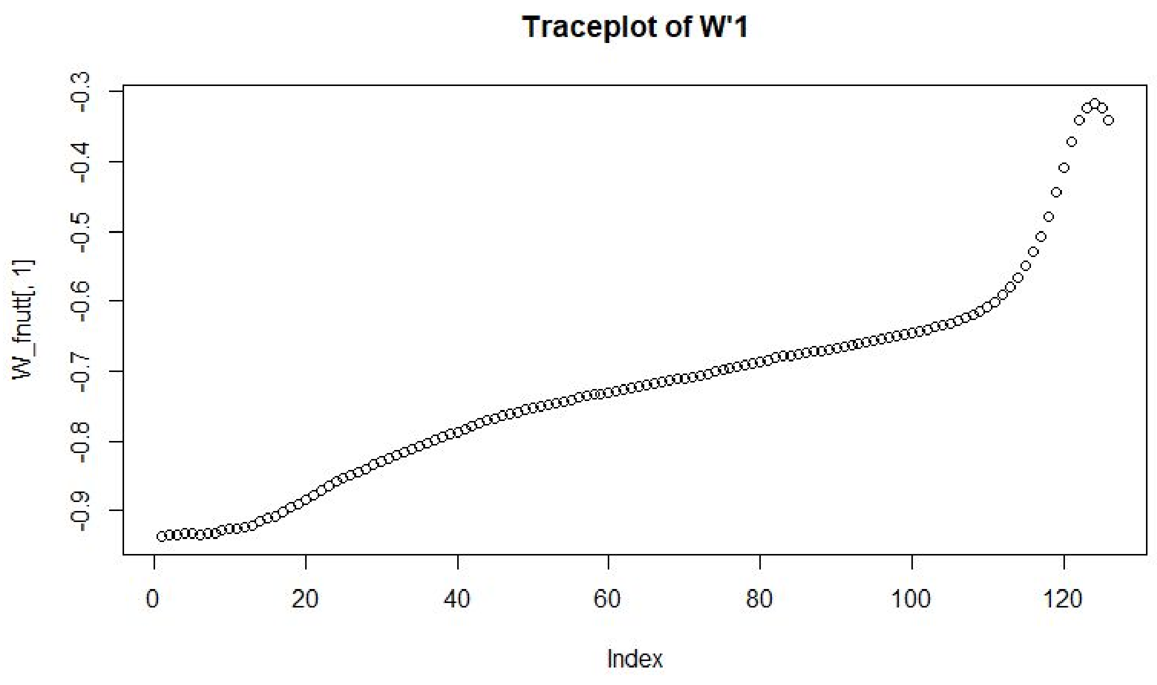


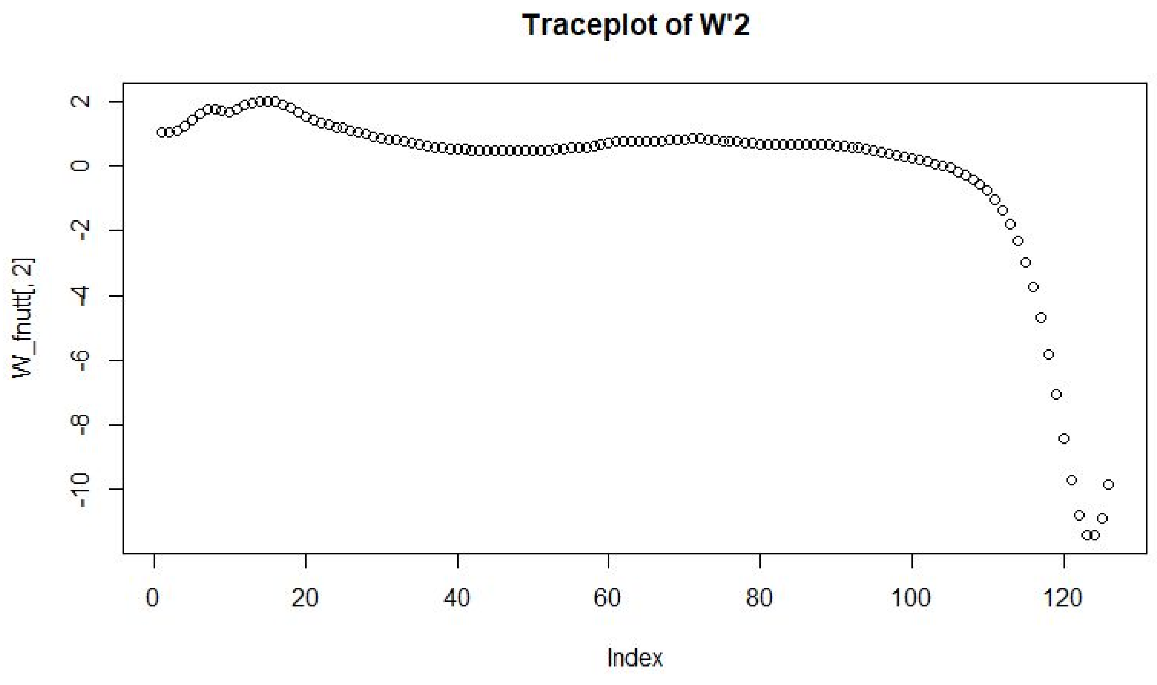


PC2 seems possible to explain with a few original features. Those which have high indexes, higher spectra. PC1, however need around half of our original features to be explained.

Step 3

A

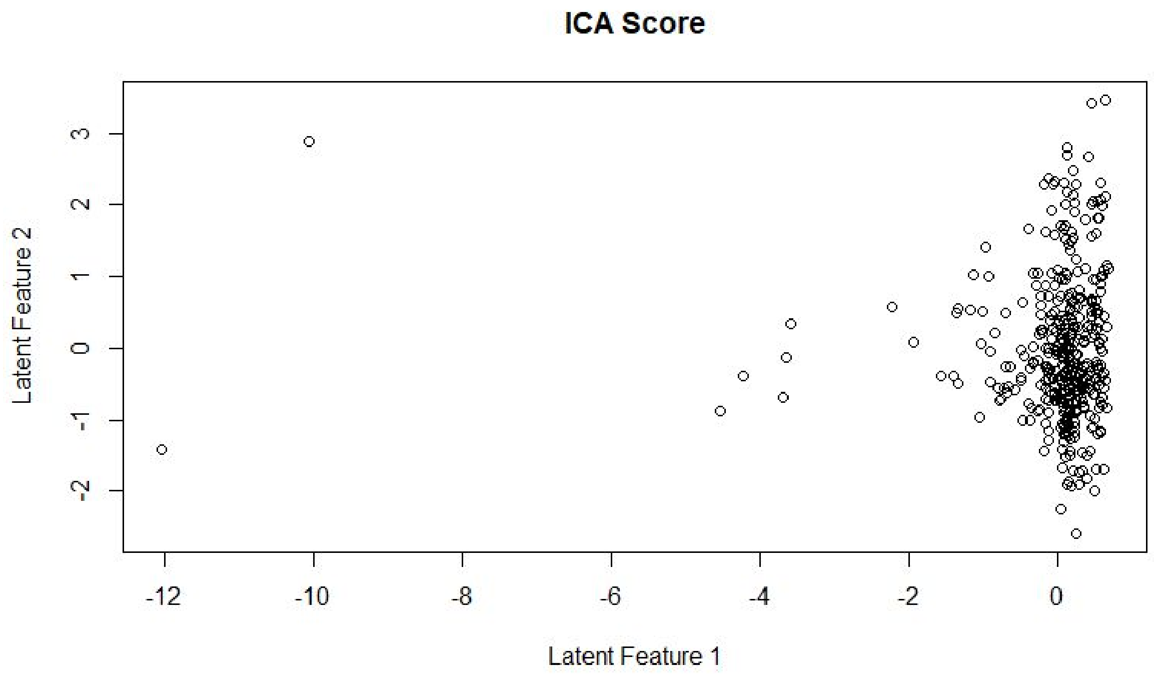

Traceplot of W'1
W_fnutt[, 1]
-0.3
-0.4
-0.5
-0.6
-0.7
-0.8
-0.9
0
20
40
60
80
100
120
Index


Traceplot of W'2
W_fnutt[, 2]
2
0
-2
-4
-6
-8
-10
0
20
40
60
80
100
120
Index

Trace plot of W'2 is similar to *Trace plot of PC2* if we invert the y-axis. *Trace plot of W'1* have some similarity with PC1.

W' = K*W and in fastICA the ICA Scores(S), is generated from X*K*W = S. Therefore, W' is the projection matrix that projects our data(X), onto ICA Scores(S). Likewise, the Principal Components (PC1 and PC2) are the eigenvectors, the projection from X to our new projected data(named scores in step 1).

B



The ICA scores are similar to scores in step 1, but rotated. That's because W' and PC, our projections, had some similarities, but the rotation is probably due to the difference in increase/decrease in our trace plots.

Appendix

Assignment 1

```
# FUNCTIONS
missclass=function(X,X1){
  n=length(X)
  return(1-sum(diag(table(X,X1)))/n)
}
# Assignment 1 - LDA and logistic regression

data = read.csv("australian-crabs.csv", header = TRUE)

#Use australian-crabs.csv and make a scatterplot of carapace length (CL) versus rear width
(RW)
#where observations are colored by Sex. Do you think that this data is easy to classify by
linear
#discriminant analysis? Motivate your answer.
plot(data$CL, data$RW, col = ifelse(data$sex == 'Male', 'blue', 'red'))
# Do you think that this data is easy to classify by linear discriminant analysis?
# Yes, the data sets are clearly divided into two groups.Thus, a linear classifier could be a
good choise.

# Excercise 2
par(mfrow=c(1,1))
library(MASS)
library(glmnet)
LDAmodel = lda(sex ~ CL + RW, data)
print(LDAmodel)
predictedSex = predict(LDAmodel, newdata = data, type = "response")
plot(data$CL, data$RW, col = ifelse(predictedSex$class == 'Male', 'blue', 'red'))

table(data$sex, predictedSex$class)
missclass(data$sex, predictedSex$class)

# Comment: A missclassification rate of 0.035 is a very good fit, it seems like LDA did a good
job
# classifying the data.

#Excercise 3
LDAmodel2 = lda(sex ~ CL + RW, data, prior = c(0.1, 0.9))
print(LDAmodel2)
predictedSex2 = predict(LDAmodel2, newdata = data, type = "response")
plot(data$CL, data$RW, col = ifelse(predictedSex2$class == 'Male', 'blue', 'red'))
```

```
table(data$sex, predictedSex2$class)
missclass(data$sex, predictedSex2$class)

# This time the misclassification rate were 0,8 which is worse than our last model.
# However we did use the priors of P(Male) = 0,9 and P(female) = 0,1. It clearly
# had a part in how the data got classified since our model didn't classify any
# female crabs as a male.

#Excercise 4
GLMmodel = glm(sex ~ CL + RW, family = binomial() , data = data)
print(GLMmodel)
predictedSexGlm = predict(GLMmodel, type = "response")
predictedSexGlm <- ifelse(predictedSexGlm > 0.5, "Male", "Female")

plot(data$CL, data$RW, col = ifelse(predictedSexGlm == 'Female', 'red', 'blue'))

table(data$sex, predictedSexGlm)
missclass(data$sex, predictedSexGlm)
#We got a missclassification rate of 0,04 which is very good, however, not as good as the
LDA method.

par(new=TRUE)
# y = k*x + m
m = coef(GLMmodel)[1]/(-coef(GLMmodel)[3])
k = coef(GLMmodel)[2]/(-coef(GLMmodel)[3])
x = seq(10, 50, 1)
y= k*x + m
lines(x, y, lwd = 2)
```

Assignment 2

```
missClassErr= function(true, predicted){
  n=length(true)
  return(1-sum(diag(table(true,predicted)))/n)
}

TPR123 = function(predictions,true){
  n=sum(true)
  sum=0
  size=length(predictions)
  for(i in 1:size){
    if(predictions[[i]]==1){
      if(predictions[[i]]==true[[i]]){
        sum=sum+1
      }
    }
  }
  print(sum/n)
  return(sum/n)
}


FPR123 =function(predictions,true){
  n=length(true)-sum(true)
  sum=0
  size=length(predictions)
  for(i in 1:size){
    if(predictions[[i]]==1){
      if(predictions[[i]]!=true[i]){
        sum=sum+1
      }
    }
  }
  return(sum/n)
}


naiveBayesClassify = function(pi, data, themodel){
 # predicted=predict(themodel,data=data, "raw")
  return(pi)
  #(ifelse(predicted>pi,1,0))
}
RNGversion('3.5.1')
# Question 1
data =read.csv2("creditscoring.csv")
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
trainInput=subset(train, select=-c(good_bad))
trainOutput = subset(train, select=c(good_bad))

set.seed(12345)
id1=setdiff(1:n, id)
id2=sample(id1, floor(n*0.25))
valid=data[id2,]
validInput=subset(valid, select=-c(good_bad))
validOutput = subset(valid, select=c(good_bad))
print(length(valid$duration))
```

```
id3=setdiff(id1,id2)
test=data[id3,]
print(length(test$duration))
testInput=subset(test, select=-c(good_bad))
testOutput = subset(test, select=c(good_bad))
#arrayTest=ifelse(testOutput=="good",1,0)
#print(sum(arrayTest)/length(testOutput$good_bad))
#print(sum(arrayTest))
#print(length(testOutput$good_bad))
#Question 2
library(tree)
fitDev = tree(good_bad~., data=train,split= "deviance")
fitGini = tree(good_bad~., data=train,split= "gini")

# Test data
predictionDevTest=predict(fitDev, newdata=testInput, type="class")
predictionGiniTest=predict(fitGini, newdata=testInput, type="class")
missClassDevTest=missClassErr(testOutput$good_bad,predictionDevTest)
missClassGiniTest=missClassErr(testOutput$good_bad,predictionGiniTest)

# Train data
predictionDevTrain=predict(fitDev, newdata=trainInput, type="class")
predictionGiniTrain=predict(fitGini, newdata=trainInput, type="class")
missClassDevTrain=missClassErr(trainOutput$good_bad,predictionDevTrain)
missClassGiniTrain=missClassErr(trainOutput$good_bad,predictionGiniTrain)

# Print
# Gini
print(missClassGiniTrain)
print(missClassGiniTest)
# Dev
print(missClassDevTrain)
print(missClassDevTest)

summary(fitDev)
plot(fitDev)
text(fitDev, pretty=0)

# Question 3
validScore=rep(0,9)
trainScore=rep(0,9)
x=prune.tree(fitDev,best=6)
summary(x)
plot(x)
```

```
for(i in 2:9) {
  prunedTree=prune.tree(fitDev,best=i)
  predValid=predict(prunedTree, newdata=valid, type="tree")
  trainScore[i]=deviance(prunedTree)
  validScore[i]=deviance(predValid)
}
plot(2:9, trainScore[2:9], type="b", col="red",ylim=c(0,750))
points(2:9, validScore[2:9], type="b", col="blue")

print(validScore)
print(validScore[which.min(validScore[2:9])+1])

model=prune.tree(fitDev,best=which.min(validScore[2:9])+1)
summary(model)
plot(model)
text(model, pretty=0)

predictTest3 = predict(model, newdata = testInput, type="class")
missClassErrTest3 = missClassErr(predictTest3, testOutput$good_bad)
print(missClassErrTest3)

# Question 4
library(MASS)
library(e1071)
set.seed(12345)
naiveFit <- naiveBayes(good_bad~., data=train)
naivePredictTrain=predict(naiveFit,newdata=trainInput)
naivePredictTest=predict(naiveFit,newdata=testInput)

confusionMatrixTrain=table(train [,"good_bad"], naivePredictTrain)
print(confusionMatrixTrain)
missClassNaiveTrain=missClassErr(trainOutput$good_bad,naivePredictTrain)
print(missClassNaiveTrain)

confusionMatrixTest=table(test [,"good_bad"], naivePredictTest)
print(confusionMatrixTest)
missClassNaiveTest=missClassErr(testOutput$good_bad,naivePredictTest)
print(missClassNaiveTest)

# Question 5
#for(i in 1:19){
testOutput5=ifelse(testOutput$good_bad=="good",1,0)
Nplus = sum(testOutput5)
Nminus= length(testOutput5)-Nplus
```

```
probNaive5=predict(naiveFit,newdata=testInput, "raw")
#Good
probNaive5=probNaive5[,2]
FPR=list()
TPR = list()

for (i in 1:19) {
  s=0.05*i
  prediction=ifelse(probNaive5>s,1,0)
  FPR[[i]]= FPR123(prediction,testOutput5)
  TPR[[i]]= TPR123(prediction,testOutput5)
}
plot(FPR,TPR,xlim=c(0,1),ylim=c(0,1),type="b", col="red")
print(FPR)
print(TPR)
probTree5=predict(model,newdata=testInput)
probTree5=probTree5[,2]
print(probTree5)
FPR1=list()
TPR1 = list()


for (i in 1:19) {
  s=0.05*i
  prediction=ifelse(probTree5>s,1,0)
  FPR1[[i]]= FPR123(prediction,testOutput5)
  TPR1[[i]]= TPR123(prediction,testOutput5)
}
print(FPR1)
print(TPR1)
points(FPR1,TPR1,xlim=c(0,1),ylim=c(0,1),type="b", col="blue")

# Question 6

probNaiveTest6 <- predict(naiveFit,newdata=testInput, "raw")
probNaiveTest6=probNaiveTest6[,2]
predictionTest6=ifelse(probNaiveTest6>10*(1-probNaiveTest6),"good","bad")


probNaiveTrain6 <- predict(naiveFit,newdata=trainInput, "raw")
probNaiveTrain6=probNaiveTrain6[,2]
predictionTrain6=ifelse(probNaiveTrain6>10*(1-probNaiveTrain6),"good","bad")

confusionMatrixTest6=table(test[,"good_bad"],predictionTest6)
confusionMatrixTrain6=table(train [,"good_bad"], predictionTrain6)
```

```
print(confusionMatrixTrain6)
print(confusionMatrixTest6)
```

Assignment 4 - Oskar Hidén

```
NIR_spectra = read.csv2("C:/Users/oskar/OneDrive/Universitet/Linköping
Universitet/År4/Machine learning/Lab 2/NIRSpectra.csv")

#--------Step 1------------
data1 = NIR_spectra
data1$Viscosity = c()
res = prcomp(data1)

#squaring sdev to get values that are (proportional to) eigenvalues
lambda = res$sdev^2
X = res$x

#hom much variance is explained in each component
sprintf("%2.3f",lambda/sum(lambda)*100)

#histogram of explained variance
screeplot(res)

# extract 2 components to get 99 explenation of total variance. PC1, PC2.
plot(res$x[,1], res$x[,2], xlab ="PC1", ylab="PC2")

#----------Step 2---------------
plot(res$rotation[,1], main="Traceplot of PC1")
plot(res$rotation[,2], main="Traceplot of PC2")

#----------Step 3-------------
library(fastICA)
set.seed(12345)
ica = fastICA(data1, 2)
W_fnutt = ica$K %*% ica$W
plot(W_fnutt[,1], main="Traceplot of W'1")
plot(W_fnutt[,2], main="Traceplot of W'2")

#Plot of scores for the two latent features
plot(ica$S, main="ICA Score", xlab="Latent Feature 1", ylab="Latent Feature 2")
```