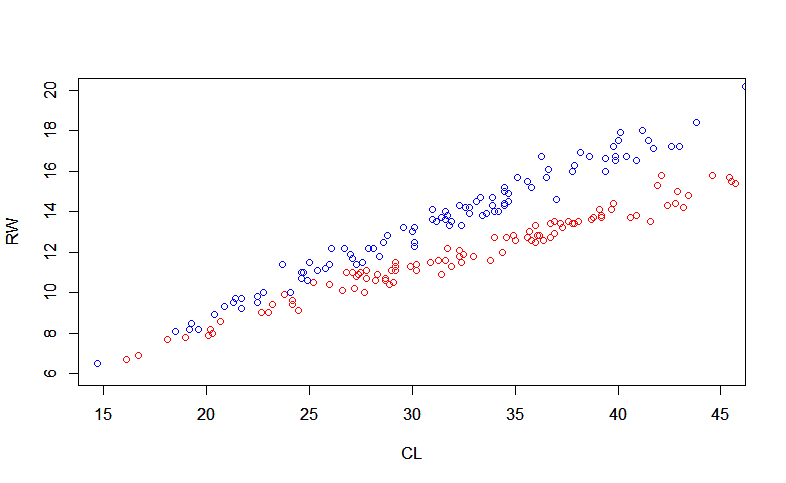
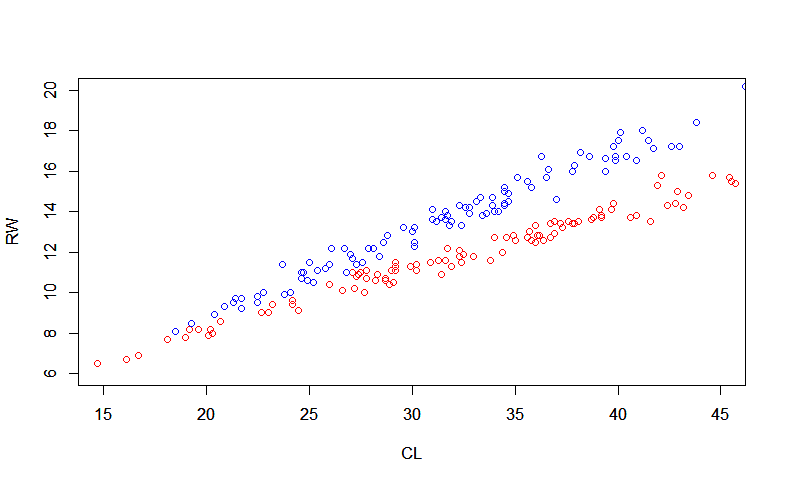
# Assignment 1

## Step 1



Male = red and Female = blue. For low values our data is mixed, but for high values it is separated. Therefor linear discriminant analysis will work but it will be better on classifying the right part of our data, were the differences in RW and CL is greater for each sex.

## Step 2



Misclassification error = 0.035

The two plots are similar and since the data is distributed with some space in y for higher x-values the misclassification rate should be quite low. Which it is. For low x-value the data is somewhat mixed, and probability for misclass is higher. By comparing the two plots we can see that all misclassification is happening in the leftmost part of the plot.

## Step

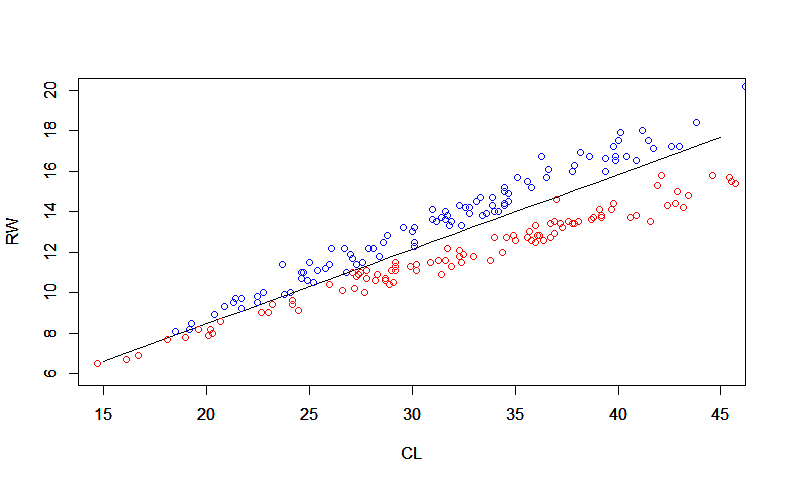
Misclassification error = 0.08

We predict more crabs as Males since the prop of being male is higher. That result in us classifying correctly for all males, but we wrongly classify more females as males. Therefor the misclassification rate increase.

## Step 4

Misclassification error of logistic regression: 0.035

The misclassification error is the same as for LDA.



The decision boundary is given by the function: y(RW) = 0.3685758x(CL) + 1.08379

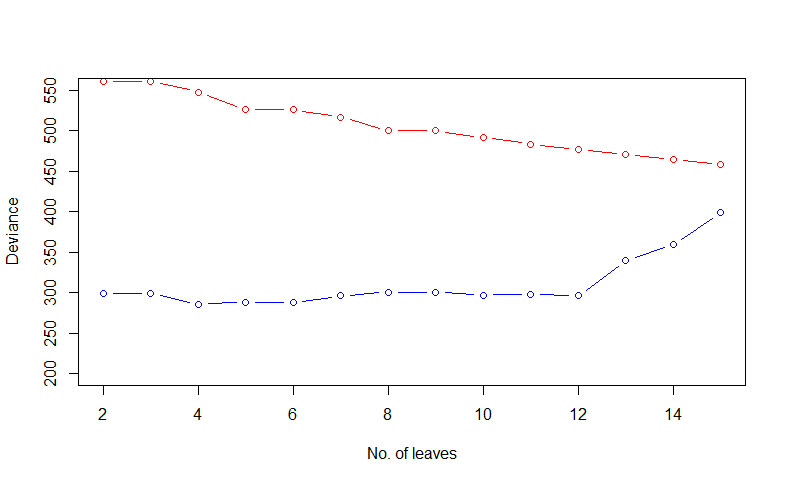
# Assignment 2

## Step 2

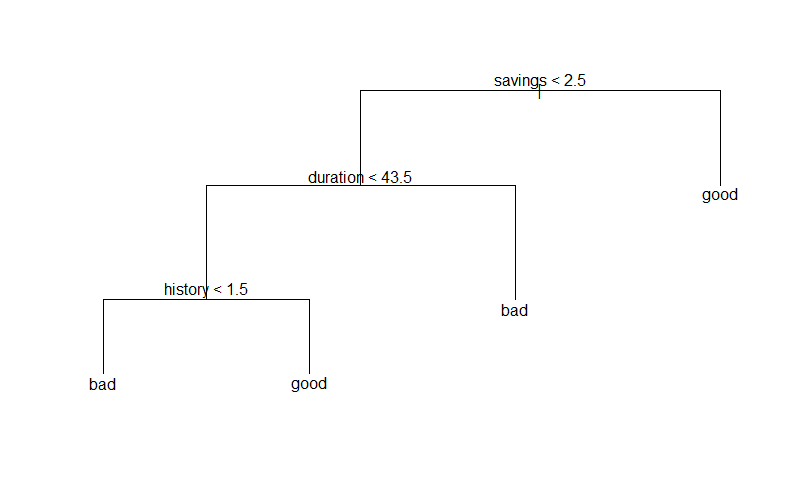
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Misclassification error** | | |
| **Training data** | **Test Data** |
| **Deviance** | 0.2105 | 0.268 | |
| **Gini** | 0.2368 | 0.368 | |

Deviance provides a lower Misclassification rate for test data. Therefore, Deviance model is chosen.

## Step 3



Optimal tree with 4 leaves:



Depth of tree= 3.

Variables used by tree:

* Savings
* Duration
* History

First we classify all enterprices with savings above(or equal) 2.5 as good. Then we look at the remaining part and we classify all with duration above(or equal) 43.5 as bad. Of the remaining part we classify all with a history above(or equal) 1.5 as good. The rest is classified as bad.

Misclassification rate for test data: 0.256

## Step 4

Naive Bayes:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Training Data** | | | **Prediction** | | | |
| **Bad** | | **Good** | |
| **Actual** | **Bad** | 95 | | 52 | |
| **Good** | 98 | | 255 | |

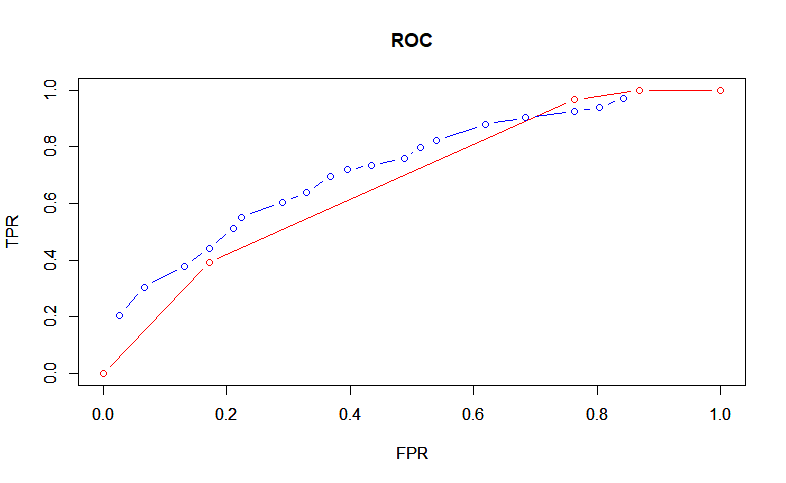
Misclassification rate for training data: 0.3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Data** | | | **Prediction** | | | |
| **Bad** | | **Good** | |
| **Actual** | **Bad** | 46 | | 30 | |
| **Good** | 49 | | 125 | |

Misclassification rate for test data: 0.316

Naïve Bayes has a higher misclassification rate than the optimal tree. 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



Conclusions: Naïve Bayes has a higher AUC, area under the curve. Which indicates that it is a preferred model.

## Step 6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Training Data** | | | **Prediction** | | |
| **Bad** | **Good** | |
| **Actual** | **Bad** | 137 | | 10 |
| **Good** | 263 | | 90 |

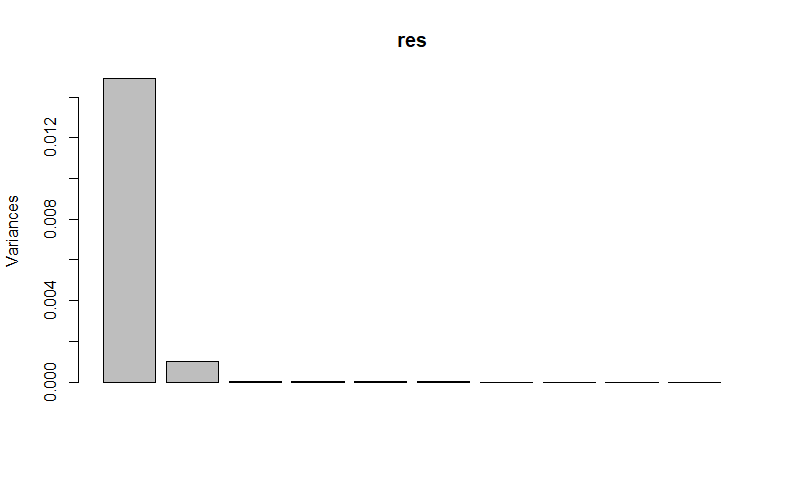
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Data** | | | **Prediction** | | |
| **Bad** | **Good** | |
| **Actual** | **Bad** | 71 | | 5 |
| **Good** | 122 | | 52 |

We predict a lot more credit score as Bad, that makes sense because we have a higher loss when classifying as Good.

# Assignment 4

## Step 1

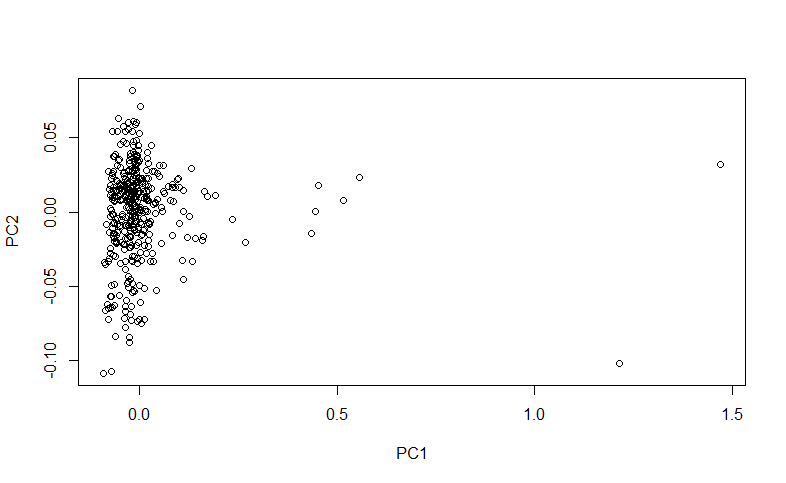
Explained variance from each component:



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

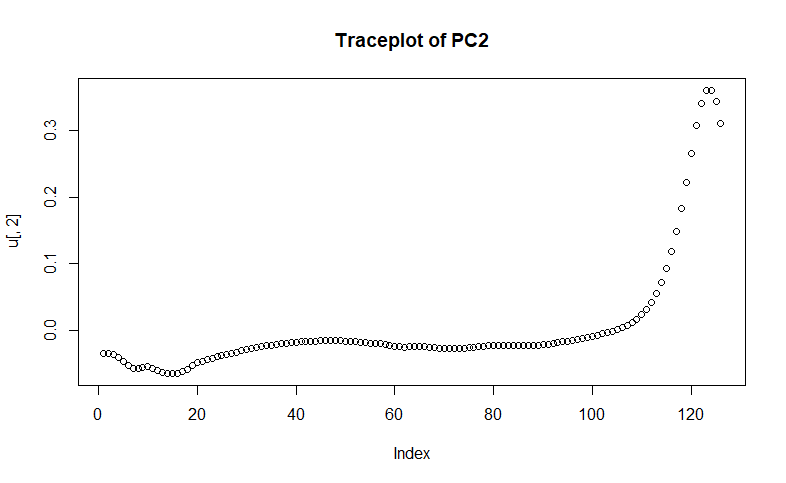
PC1: 93.332 %  
PC2: 6.263 %  
PC3: 0.185 %  
PC4: 0.101 %  
…

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.

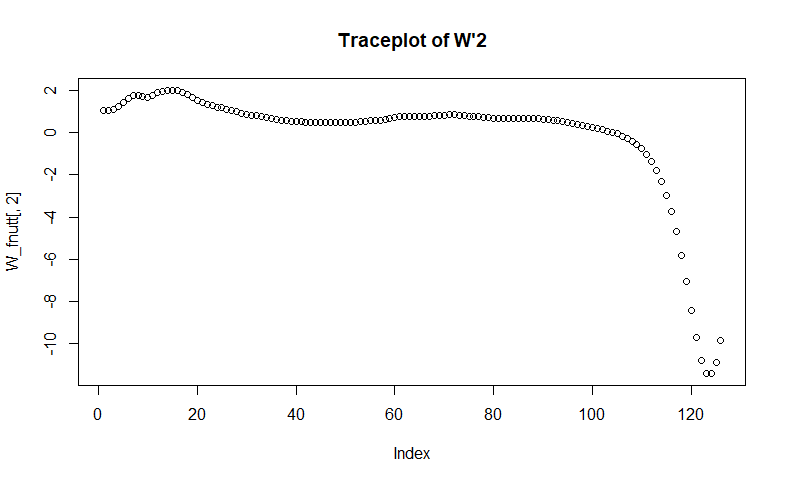
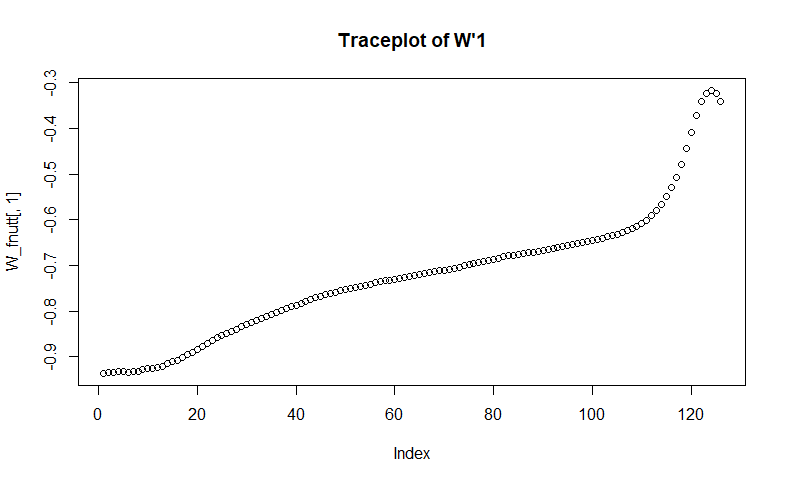
## Step 2



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

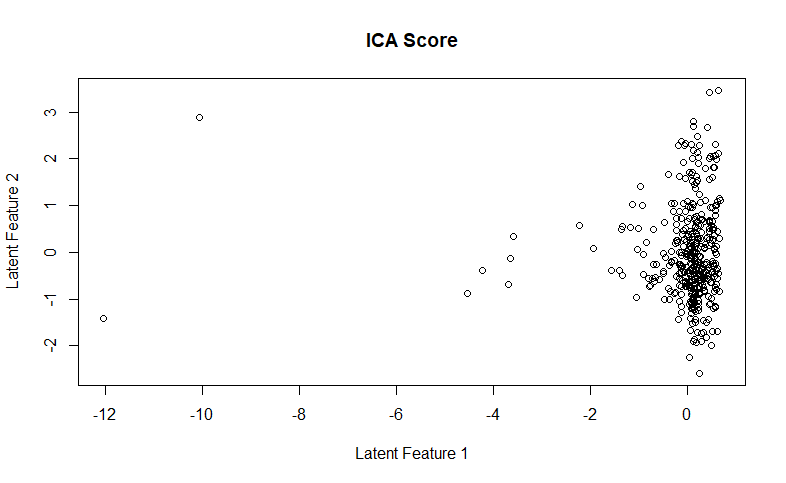
### A



*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

RNGversion('3.5.1')

library(readr)

set.seed(12345)

#Assignment1

australian\_crabs = read.csv("C:/Users/oskar/OneDrive/Universitet/Linköping Universitet/År4/Machine learning/Lab 2/australian-crabs.csv")

#--------step1----------

australian\_crabs\_males = subset(australian\_crabs, sex=="Male")

australian\_crabs\_females = subset(australian\_crabs, sex=="Female")

plot(australian\_crabs\_males[['CL']], australian\_crabs\_males[['RW']], ylim=c(6,20), xlim=c(15,45), col="red", ylab="RW", xlab="CL")

par(new=TRUE)

plot(australian\_crabs\_females[['CL']], australian\_crabs\_females[['RW']], ylim=c(6,20), xlim=c(15,45), col="blue", ylab="RW", xlab="CL")

#---------Step2----------

library(MASS)

lda\_pred = lda(sex~CL + RW, data=australian\_crabs)

print(lda\_pred)

pred = predict(lda\_pred, australian\_crabs)

table(australian\_crabs[['sex']], pred$class)

predicted\_dataset = data.frame(pred$class, australian\_crabs[['CL']], australian\_crabs[['RW']])

names(predicted\_dataset) = c('sex', 'CL', 'RW')

plot(subset(predicted\_dataset, sex=="Male")[['CL']], subset(predicted\_dataset, sex=="Male")[['RW']], ylim=c(6,20), xlim=c(15,45), col="red", ylab="RW", xlab="CL")

par(new=TRUE)

plot(subset(predicted\_dataset, sex=="Female")[['CL']], subset(predicted\_dataset, sex=="Female")[['RW']], ylim=c(6,20), xlim=c(15,45), col="blue", ylab="RW", xlab="CL")

# Misclassification function

misclass=function(X, Xfit){

n=length(X)

return (1-sum(diag(table(X, Xfit)))/n)

}

lda\_pred\_misclassification = misclass(australian\_crabs[['sex']], pred$class)

print(lda\_pred\_misclassification)

#-------step3-------

lda\_pred\_wprior = lda(sex~CL + RW, data=australian\_crabs, prior = c(0.1, 0.9))

print(lda\_pred\_wprior)

pred\_wprior = predict(lda\_pred\_wprior, australian\_crabs)

table(australian\_crabs[['sex']], pred\_wprior$class)

predicted\_dataset\_wprior = data.frame(pred\_wprior$class, australian\_crabs[['CL']], australian\_crabs[['RW']])

names(predicted\_dataset\_wprior) = c('sex', 'CL', 'RW')

plot(subset(predicted\_dataset\_wprior, sex=="Male")[['CL']], subset(predicted\_dataset\_wprior, sex=="Male")[['RW']], ylim=c(6,20), xlim=c(15,45), col="red", ylab="RW", xlab="CL")

par(new=TRUE)

plot(subset(predicted\_dataset\_wprior, sex=="Female")[['CL']], subset(predicted\_dataset\_wprior, sex=="Female")[['RW']], ylim=c(6,20), xlim=c(15,45), col="blue", ylab="RW", xlab="CL")

lda\_pred\_misclassification\_wprior = misclass(australian\_crabs[['sex']], pred\_wprior$class)

print(lda\_pred\_misclassification\_wprior)

#--------step 4----------

#check if sex is as factor

str(australian\_crabs)

logistic\_regression = glm(as.factor(sex) ~ CL + RW, data=australian\_crabs, family = binomial)

print(logistic\_regression)

#dessission boundary

intercept = coef(logistic\_regression)[1]/(-coef(logistic\_regression)[3])

slope = coef(logistic\_regression)[2]/(-coef(logistic\_regression)[3])

x = seq(15,45, by=1)

y = slope\*x + intercept

prediction\_LR = predict(logistic\_regression, australian\_crabs, type="response")

predicted\_sex = prediction\_LR

predicted\_sex[prediction\_LR<0.5] = 'Female' #prediction\_LR will return all rows that are under threashold!

predicted\_sex[prediction\_LR>=0.5] = 'Male'

predicted\_sex

australian\_crabs[["sex"]]

misclass(australian\_crabs[["sex"]], predicted\_sex)

table(australian\_crabs[["sex"]], predicted\_sex)

predicted\_dataset\_LR = data.frame(predicted\_sex, australian\_crabs[['CL']], australian\_crabs[['RW']])

names(predicted\_dataset\_LR) = c('sex', 'CL', 'RW')

plot(x, y, type="l", ylim=c(6,20), xlim=c(15,45), ylab="RW", xlab="CL")

par(new=TRUE)

plot(subset(predicted\_dataset\_LR, sex=="Male")[['CL']], subset(predicted\_dataset\_LR, sex=="Male")[['RW']], ylim=c(6,20), xlim=c(15,45), col="red", ylab="RW", xlab="CL")

par(new=TRUE)

plot(subset(predicted\_dataset\_LR, sex=="Female")[['CL']], subset(predicted\_dataset\_LR, sex=="Female")[['RW']], ylim=c(6,20), xlim=c(15,45), col="blue", ylab="RW", xlab="CL")

#------------Assignment 2------------

#--------------Step 1-------------

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

RNGversion('3.5.1')

n=dim(creditscoring)[1]

set.seed(12345)

id=sample(1:n, floor(n\*0.5))

train=creditscoring[id,]

id1=setdiff(1:n, id)

set.seed(12345)

id2=sample(id1, floor(n\*0.25))

valid=creditscoring[id2,]

id3=setdiff(id1,id2)

test=creditscoring[id3,]

library(tree)

#or:

library(rpart)

#Training data to fit model

fit\_deviance = tree(good\_bad~. , split = "deviance", data = train)

fit\_gini = tree(good\_bad~. , split = "gini", data = train)

summary(fit\_deviance)

summary(fit\_gini)

#Predict using test data.

predict\_deviance = predict(fit\_deviance, newdata = test, type = "class")

#table(test[["good\_bad"]], predict\_deviance)

misclass\_deviance = misclass(test[["good\_bad"]], predict\_deviance)

print(misclass\_deviance)

predict\_gini = predict(fit\_gini, newdata = test, type = "class")

#table(test[["good\_bad"]], predict\_gini)

misclass\_gini = misclass(test[["good\_bad"]], predict\_gini)

print(misclass\_gini)

#------------Step 3---------------

#Deviance is chosen due to lower misclassification rate for test data.

summary(fit\_deviance)

train\_score = rep(0,15)

test\_score = rep(0,15)

for(i in 2:15) {

pruned\_tree = prune.tree(fit\_deviance, best = i)

pred = predict(pruned\_tree, newdata=valid, type="tree")

train\_score[i] = deviance(pruned\_tree)

test\_score[i] = deviance(pred)

}

plot(2:15, train\_score[2:15], type="b", col="red", ylim=c(200,550), ylab="Deviance", xlab="No. of leaves")

points(2:15, test\_score[2:15], type="b", col="blue")

test\_score[1] = 5000

which.min(test\_score)

## Min when best=4

test\_score[4]

pruned\_tree = prune.tree(fit\_deviance, best = 4)

summary(pruned\_tree)

plot(pruned\_tree)

text(pruned\_tree, pretty = 0)

#Misclass for test

prediction\_test = predict(pruned\_tree, newdata = test, type = "class")

table(test[["good\_bad"]], prediction\_test)

misclass(test[["good\_bad"]], prediction\_test)

#-------------Step 4 -------------

library(MASS)

library(e1071)

fit\_naive\_bayes =naiveBayes(good\_bad~., data=train)

summary(fit\_naive\_bayes)

#train data

predict\_naive\_bayes\_train = predict(fit\_naive\_bayes, newdata = train)

table(train[["good\_bad"]], predict\_naive\_bayes\_train)

misclass(train[["good\_bad"]], predict\_naive\_bayes\_train)

#test data

predict\_naive\_bayes\_test = predict(fit\_naive\_bayes, newdata = test)

table(test[["good\_bad"]], predict\_naive\_bayes\_test)

misclass(test[["good\_bad"]], predict\_naive\_bayes\_test)

# remember: 1-(sum(diag(table))/sum(table))

#----------Step 5-----------

# TPR = true positive rate(y-axis)

# FPR = false positive reate(x-axis)

predict\_naive\_bayes\_test = predict(fit\_naive\_bayes, newdata = test, type= "raw")

predict\_naive\_bayes\_test

pi = seq(from = 0.05, to = 0.95, by = 0.05 )

n = length(pi)

#Naive Bayes

TPR = rep(0,n)

FPR = rep(0,n)

for( i in 1:n){

predict = predict\_naive\_bayes\_test[,2]

predict = ifelse(predict>pi[i], "good", "bad")

table = table(test[["good\_bad"]], predict)

print(table)

TPR[i] = (table[2, 2])/sum(table[2, ])

FPR[i] = (table[1, 2])/sum(table[1, ])

}

# tree ROC

#str(test)

prediction\_test = predict(pruned\_tree, newdata = test, type = "vector")

n = length(pi)

TPR\_tree = rep(0,n)

FPR\_tree = rep(0,n)

for( i in 1:n){

pred = as.vector(prediction\_test[,2])

pred = ifelse(pred>pi[i], "good", "bad")

if ( sum(pred=="bad")==0) {

FPR\_tree[i] = 1

TPR\_tree[i] = 1

} else if ( sum(pred=="good")==0) {

TPR\_tree[i] = 0

FPR\_tree[i] = 0

} else {

table = table(test[["good\_bad"]], pred)

print(table)

TPR\_tree[i] = (table[2, 2])/sum(table[2, ])

FPR\_tree[i] = (table[1, 2])/sum(table[1, ])

}

}

plot(FPR\_tree, TPR\_tree, xlim = (0:1), ylim= (0:1), type="b", col="red", xlab="FPR", ylab="TPR", main="ROC")

par(new=TRUE)

plot(FPR, TPR, xlim = (0:1), ylim= (0:1), type="b", col="blue", xlab="FPR", ylab="TPR")

#-------Step 6-------------

fit\_naive\_bayes =naiveBayes(good\_bad~., data=train)

summary(fit\_naive\_bayes)

#train data

naive\_bayes\_train = predict(fit\_naive\_bayes, newdata = train, type="raw")

predict\_train = ifelse(naive\_bayes\_train[,2]/naive\_bayes\_train[,1]>10, "good", "bad")

table(train[["good\_bad"]], predict\_train)

misclass(train[["good\_bad"]], predict\_train)

#test data

naive\_bayes\_test = predict(fit\_naive\_bayes, newdata = test, type="raw")

predict\_test = ifelse(naive\_bayes\_test[,2]/naive\_bayes\_test[,1]>10, "good", "bad")

misclass(test[["good\_bad"]], predict\_test)

table = table(test[["good\_bad"]], predict\_test)

print(table)

#Assignment 4

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")